Optimization of Partial-Order Plans via MaxSAT

Christian Muise and **Sheila A. McIlraith**
Dept. of Computer Science
University of Toronto
Toronto, Canada. M5S 3G4
{cjmuise,sheila}@cs.toronto.edu

J. Christopher Beck
Dept. of Mechanical & Industrial Engineering
University of Toronto
Toronto, Canada. M5S 3G8
jcb@mie.utoronto.ca

Abstract

Partial-order plans (POPs) are attractive because of their least commitment nature, providing enhanced plan flexibility at execution time relative to sequential plans. Despite the appeal of POPs, most of the recent research on automated plan generation has focused on sequential plans. In this paper we examine the task of POP generation by relaxing or modifying the action orderings of a sequential plan to optimize for plan criteria that promote flexibility in the POP. Our approach relies on a novel partial weighted MaxSAT encoding of a sequential plan that supports the minimization of deordering or reordering of actions. We further extend the classical least commitment criterion for a POP to consider the number of actions in a solution, and provide an encoding to achieve least commitment plans with respect to this criterion. Our partial weighted MaxSAT encoding gives us an effective means of computing a POP from a sequential plan. We compare the efficiency of our approach to a previous approach for POP generation via sequential-plan relaxation. Our results show that while the previous approach is proficient at producing the optimal *deordering* of a sequential plan, our approach gains greater flexibility with the optimal *reordering*.

1 Introduction

Partial-order planning reflects a least commitment strategy (Weld 1994). Unlike a sequential plan that specifies a set of actions and a total order over those actions, a partial-order plan (POP) only specifies those action orderings necessary to achieve the goal from the initial state. In doing so, a POP embodies a family of sequential plans – a set of linearizations all sharing the same actions, but differing with respect to the order of the actions.

The flexibility afforded by POPs makes them attractive for real-time execution, multi-agent taskability, and a range of other applications that can benefit from their least commitment nature (Veloso, Pollack, and Cox 1998; Weld 1994). However, in recent years research on plan generation has shifted away from partial-order planning towards sequential planning, primarily due to the success of heuristic-based forward-search planners. To regain the least commitment nature of POPs while leveraging fast sequential plan generation, it is compeling to examine the computation of POPs via sequential planning technology. Indeed this approach has been realized in the planner POPF (Coles et al. 2010), which generated a POP by searching in a heuristic-based forward-chaining manner.

Another possibility for leveraging the strengths of sequential planning is to generate a sequential plan with a state-of-the-art planner, and subsequently relax the plan to a POP. Removing ordering constraints from the sequential plan, referred to as a *deordering*, or allowing changes in the ordering constraints, referred to as a *reordering*, are approaches that have been theoretically investigated (Bäckström 1998). Unfortunately, finding the optimal deordering or reordering is NP-hard to solve, and difficult to approximate within a constant factor. Nevertheless, with the advent of powerful optimization techniques (such as MaxSAT), we can effectively solve many problems in practice.

In this paper we focus on the optimization problem of computing the minimum deordering and minimum reordering of a sequential plan treated as a POP. The minimum deordering of a POP minimizes the number of ordering constraints between actions, so long as the POP remains valid and no two actions have their order reversed. Similarly, a minimum reordering minimizes the number of ordering constraints, but has no restriction on which ordering constraints are forbidden. These two notions cover a natural aspect of least commitment planning – minimizing the ordering constraints placed on a POP. We extend this characterization to consider the number of actions in a plan. In the spirit of least commitment planning, we argue that a POP should first commit to as few actions as possible before committing to as few ordering constraints as possible. With the various notions of least commitment planning in mind (deordering, reordering, etc), we propose a set of criteria against which we evaluate our work. These include the number of actions and ordering constraints found in the transitive closure of a POP, and the number of linearizations the POP represents. The criteria serve as a measure of the flexibility of a POP.

Our approach is to use a family of novel encodings for partial weighted MaxSAT whose solution corresponds to an optimal least commitment plan. Unlike typical SAT-based planning techniques, we represent an action occurrence once, giving us a succinct representation for use with a modern MaxSAT solver. We compare our approach to an existing algorithm for relaxing a sequential plan, due to Kambhampati and Kedar (Kambhampati and Kedar 1994), and evaluate our approach empirically. We demonstrate the

efficiency of using our MaxSAT encoding to relax a sequential plan optimally, and demonstrate the strength of Kambhampati and Kedar's algorithm in computing a deording of a sequential plan.

We find that the existing algorithm is extremely proficient at computing the minimum deordering, matching the optimal solution in every problem tested. However, we find that the minimum reordering is usually far more flexible than the minimum deordering (having fewer ordering constraints and far more linearizations). Our approach to encoding the problem gives us the first technique, to the best of our knowledge, for computing the optimal reordering of a POP. We further see a benefit in the flexibility of a POP when we use the proposed extension to least commitment planning that considers the number of actions.

In the next section we provide background on the notation we use throughout the paper. We follow with a description of least commitment planning in Section 3, and then describe our encoding for the various forms of optimization criteria in Section 4. In Section 5 we describe the prior approach to relaxing a plan, and we present the experimental results in Section 6. We finish with a brief discussion and conclusion in Sections 7 and 8.

2 Background

Propositional Planning

For the purposes of this document, we restrict ourselves to STRIPS planning problems. In STRIPS, a planning problem is a tuple $\Pi = \langle F, O, I, G \rangle$ where F is a finite set of fluents, O is the set of operators, $I \subseteq F$ is the initial state, and $G \subseteq F$ is the goal state. We refer to a *complete state* (or just *state*) as a subset of F. We interpret fluents that do not appear in a complete state as being false in that state. We characterize an operator $o \in O$ by three sets: $PRE(o)$, the fluents that must be true in order for o to be executable; $ADD(o)$, the fluents that operator o adds to the state; and $DEL(o)$, the fluents that operator o deletes from the state. An *action* refers to a specific instance of an operator, and it inherits the PRE, ADD, and DEL sets of the matching operator. We say that an action a is *executable* in state s iff $PRE(a) \subseteq s$. A sequence of actions is a *sequential plan* for the problem Π if the execution of each action in sequence, when starting in state I, causes the goal to hold in the final state.

We will make use of two further items of notation with respect to a set of actions A. We define **adders**(f) to be the set of actions in A that add the fluent f (i.e., $\{a \mid f \in ADD(a)\}$), and we define **deleters**(f) to be the set of actions in A that delete the fluent f (i.e., $\{a \mid f \in DEL(a)\}$).

Partial-Order Plans

For this paper, we adopt the notation typically used by the partial-order planning community. With respect to a planning problem $\Pi = \langle F, O, I, G \rangle$, a partial-order plan (POP) is a tuple $P = \langle \mathcal{A}, \mathcal{O}, \mathcal{C} \rangle$ where $\mathcal{A}$ is the set of actions in the plan (all of which have a corresponding operator in O), $\mathcal{O}$ is a set of orderings between the actions in $\mathcal{A}$ (e.g., $(a_1 \prec a_2) \in \mathcal{O}$), and $\mathcal{C}$ is a set of causal links between the actions in $\mathcal{A}$ (Weld 1994). A causal link is an annotated ordering constraint where the annotation of the link is a fluent from F that represents the reason for that link's existence.

For a causal link $(a_1 \overset{f}{\prec} a_2) \in \mathcal{C}$, we can assume that $f \in ADD(a_1)$ and $f \in PRE(a_2)$. The ordering constraints found in $\mathcal{C}$ will always be a subset of the ordering constraints in $\mathcal{O}$, and we assume that $\mathcal{O}$ is transitively closed. Where convenient, we will ignore the set $\mathcal{C}$ and simply use $\langle \mathcal{A}, \mathcal{O} \rangle$ to represent a POP. We refer to a total ordering of the actions in $\mathcal{A}$ that respects $\mathcal{O}$ as a *linearization* of P. A POP provides a compact representation for multiple linearizations.

Intuitively, a POP is valid for a planning problem if it is able to achieve the goal. There are two related formal notions of what a valid POP consists of. The first, only referring to $\mathcal{A}$ and $\mathcal{O}$, says that a POP P is valid for a planning problem Π iff every linearization of P is a plan for Π. While simple and intuitive, this notion is rarely used to verify the validity of a POP since there may be a prohibitively large number of linearizations represented by the POP.

The second notion is slightly more involved, and refers to open preconditions and threats. An *open precondition* is a precondition p of an action $a \in \mathcal{A}$ that does not have an associated causal link (i.e., $\nexists a' \in \mathcal{A}$ s.t. $(a' \overset{p}{\prec} a) \in \mathcal{C}$). If a precondition is not open, we say that it is *supported*, and we refer to the associated action in the causal link as the *achiever* for the precondition.

A *threat* in a POP refers to an action that can invalidate a causal link due to the ordering constraints (or lack thereof). Formally, if $(a' \overset{p}{\prec} a) \in \mathcal{C}$, we say that an action a'' threatens the causal link if the following is true:

- We can order a'' between a' and a (i.e., $\{(a'' \prec a'), (a \prec a'')\} \cap \mathcal{O} = \emptyset$)
- The action a'' can delete p (i.e., $p \in DEL(a'')$)

The existence of a threat means that a linearization may exist that is not executable because one of the preconditions of an action in the linearization is not satisfied.

We will typically add two special actions to the POP, a_I and a_G, that encode the initial and goal states through their add effects and preconditions (i.e., $PRE(a_G) = G$ and $ADD(a_I) = I$). With this modification, we say that a POP $P = \langle \mathcal{A}, \mathcal{O}, \mathcal{C} \rangle$ is a valid POP for the planning problem Π iff it has no open preconditions and no causal link in the set $\mathcal{C}$ has a threatening action in $\mathcal{A}$. The two notions of POP validity are similar in the sense that if the second notion holds, then the first follows. If the first notion holds for $\mathcal{A}$ and $\mathcal{O}$, then a set of causal links $\mathcal{C}$ exists such that the second notion holds for $P = \langle \mathcal{A}, \mathcal{O}, \mathcal{C} \rangle$. Further details on these notions of validity can be found in (Russell and Norvig 2009).

Partial Weighted MaxSAT

In boolean logic, the problem of Satisfiability (SAT) is to find a True/False setting of boolean variables such that a logical formula referring to those variables evaluates to True (Biere et al. 2009). Typically, we write problems in Conjunctive Normal Form (CNF) which is made up of a conjunction of clauses, where each clause is a disjunction of

literals. A literal is either a boolean variable or its negation. A setting of the variables satisfies a CNF formula iff every clause has at least one literal that the setting satisfies.

The MaxSAT problem is the optimization variant of the SAT problem in which the goal is to maximize the number of satisfied clauses (Biere et al. 2009, Ch. 19). Adding non-uniform weights to each clause allows for a more natural representation of the optimization problem, and we refer to this as the weighted MaxSAT problem. If we wish to force the solver to find a solution that satisfies a particular subset of the clauses, we refer to clauses in this subset as *hard*, while all other clauses in the problem are *soft*. When we have a mix of hard and soft clauses, we have a partial weighted MaxSAT problem (Biere et al. 2009, Ch. 19.6).

In a partial weighted MaxSAT problem, only the soft clauses are given a weight, and a valid solution corresponds to any setting of the variables that satisfies the hard clauses in the CNF. An optimal solution to a partial weighted MaxSAT problem is any valid solution that maximizes the sum of the weights on the satisfied soft clauses.

3 Least Commitment Criteria

The aim of least commitment planning is to find flexible solutions that allow us to defer decisions regarding the execution of the plan. Considering only the ordering constraints of a POP, two appealing notions for least commitment planning are the *deordering* and *reordering* of a POP. Following (Bäckström 1998), we define these formally:

Definition 1. Let $P = \langle \mathcal{A}, \mathcal{O} \rangle$ and $Q = \langle \mathcal{A}', \mathcal{O}' \rangle$ be two POPs, and Π a planning problem. Then,

1. Q is a reordering of P wrt. Π iff P and Q are valid POPs for Π, and $\mathcal{A} = \mathcal{A}'$
2. Q is a deordering of P wrt. Π iff P and Q are valid POPs for Π, $\mathcal{A} = \mathcal{A}'$, and $\mathcal{O}' \subseteq \mathcal{O}$.

Recall that we assume the ordering constraints of a POP are transitively closed. We define the optimal deordering and optimal reordering as follows:

Definition 2. Let $P = \langle \mathcal{A}, \mathcal{O} \rangle$ and $Q = \langle \mathcal{A}', \mathcal{O}' \rangle$ be two POPs, and Π a planning problem. Then,

1. Q is a minimum deordering of P wrt. Π iff
   (a) Q is a deordering of P wrt. Π, and
   (b) there is no deordering $\langle \mathcal{A}'', \mathcal{O}'' \rangle$ of P wrt. Π s.t. $|\mathcal{O}''| < |\mathcal{O}'|$
2. Q is a minimum reordering of P wrt. Π iff
   (a) Q is a reordering of P wrt. Π, and
   (b) there is no reordering $\langle \mathcal{A}'', \mathcal{O}'' \rangle$ of P wrt. Π s.t. $|\mathcal{O}''| < |\mathcal{O}'|$

Note that we use $<$ rather than $\subset$ for 1(b) and 2(b) since the orderings in $\mathcal{O}'$ and $\mathcal{O}''$ may only partially overlap. We will equivalently refer to a minimum deordering (resp. reordering) as an *optimal* deordering (resp. reordering). In both cases, we prefer a POP that has the smallest set of ordering constraints. In other words, no POP exists with the same actions and fewer ordering constraints while remaining valid with respect to Π. The problem of finding the minimum deordering or reordering of a POP is NP-hard, and cannot be approximated within a constant factor unless $NP \in \text{DTIME}(n^{poly\ log\ n})$ (Bäckström 1998).

While the notion of a minimum deordering or reordering of a POP addresses the commitment of ordering constraints, in the spirit of least commitment planning we would like to commit to as few actions as possible. To this end, we provide an extended criterion of what a least commitment POP (LCP) should satisfy:

Definition 3. Let $P = \langle \mathcal{A}, \mathcal{O} \rangle$ and $Q = \langle \mathcal{A}', \mathcal{O}' \rangle$ be two POPs valid for Π. Q is a *least commitment POP* (LCP) of P iff Q is the minimum reordering of itself and there is no valid POP $\langle \mathcal{A}'', \mathcal{O}'' \rangle$ for Π such that $\mathcal{A}'' \subseteq \mathcal{A}$ and $|\mathcal{A}''| < |\mathcal{A}'|$.

Intuitively, we can compute the LCP of an arbitrary POP by first minimizing the number of actions, and then minimizing the number of ordering constraints.

It may turn out that preferring fewer actions causes us to commit to more ordering constraints, simply due to the interaction between the actions we choose. However, in practice we usually place a much greater emphasis on minimizing the number of actions in a POP, as, in the standard interpretation of a POP, every every action must be executed.

Following the above criteria we will evaluate the quality of a POP by the number of actions and ordering constraints it contains, as these metrics give us a direct measure of the least commitment nature of a POP. Another property of interest is a POP's *flexibility*, which provides a measure of the robustness inherent in the POP. We measure the flexibility, whenever computationally feasible, as the number of linearizations a POP represents.

As we have discussed earlier, verifying a POP's validity by way of the linearizations is not always practical. As such, we will not attempt to compute POPs that maximize the number of linearizations, but rather we will compute POPs that adhere to one of the above criteria: minimum deordering, minimum reordering, or LCP.

4 A Partial Weighted MaxSAT Encoding

To generate a POP, we encode the problem of finding a minimum deordering or reordering as a partial weighted MaxSAT problem. Solutions to the default encoding correspond to a LCP. That is, no POP exists with a proper subset of the actions, or with a proper subset of the ordering constraints. We add further clauses to produce encodings that correspond to optimal deorderings or reorderings.

We use two types of propositional variables: action variables and ordering variables. For every action $a \in \mathcal{A}$, the variable x_a indicates whether or not the action a appears in the POP. For every pair of actions $a_1, a_2 \in \mathcal{A}$, the variable $\kappa(a_1, a_2)$ indicates an ordering constraint between action a_1 and a_2 in the POP.

In a partial weighted MaxSAT encoding there must be a distinction between hard and soft clauses. We first present the hard clauses of the encoding as boolean formulae which we subsequently convert to CNF, and later describe the soft clauses with their associated weight. We define the formulae that ensure the POP generated is acyclic, and the ordering constraints produced include the transitive closure (here,

actions are universally quantified, and for formula (4) we assume $a_I \neq a_i \neq a_G$):

$$(\neg\kappa(a,a)) \quad (1)$$

$$(x_{a_I}) \wedge (x_{a_G}) \quad (2)$$

$$\kappa(a_i,a_j) \rightarrow x_{a_i} \wedge x_{a_j} \quad (3)$$

$$x_{a_i} \rightarrow \kappa(a_I,a_i) \wedge \kappa(a_i,a_G) \quad (4)$$

$$\kappa(a_i,a_j) \wedge \kappa(a_j,a_k) \rightarrow \kappa(a_i,a_k) \quad (5)$$

(1) ensures that there are no self-loops; (2) ensures that we include the initial and goal actions; (3) ensures that if we use an ordering variable, then we include both actions in the POP; (4) ensures that an action cannot appear before the initial action (or after the goal); and (5) ensures that a solution satisfies the transitive closure of ordering constraints. Together, (1) and (5) ensure the POP will be acyclic, while the remaining formulae tie the two types of variables together and deal with the initial and goal actions.

In contrast to the typical SAT encoding for planning problems, we do not require the actions to be placed in a particular layer. Instead, we represent each action only once and handle the ordering between actions through the κ variables.

Finally, we include the formulae needed to ensure that every action has its preconditions met, and there are no threats in the solution:

$$\Upsilon(a_j,a_i,p) \equiv \bigwedge_{a_k \in \textbf{deleters}(p)} x_{a_k} \rightarrow \kappa(a_k,a_j) \vee \kappa(a_i,a_k) \quad (6)$$

$$x_{a_i} \rightarrow \bigwedge_{p \in PRE(a_i)} \bigvee_{a_j \in \textbf{adders}(p)} \kappa(a_j,a_i) \wedge \Upsilon(a_j,a_i,p) \quad (7)$$

Intuitively, $\Upsilon(a_j,a_i,p)$ ensures that if a_j is the achiever of precondition p for action a_i, then no deleter of p will be allowed to occur between the actions a_j and a_i. Υ ensures that every causal link remains unthreatened in a satisfying assignment. Formula (7) ensures that if we include action a_i in the POP, then every precondition p of a_i (the conjunction) must be satisfied by at least one achiever a_j (the disjunction). $\kappa(a_j,a_i)$ orders the achiever correctly, while $\Upsilon(a_j,a_i,p)$ removes threats.

In order to achieve a POP that is least commitment, we prefer solutions that first minimize the actions, and then minimize the ordering constraints. We achieve this by adding a soft unit clause for every variable in our encoding. We weight the κ variables with a unit cost and weight the action variables high enough for the solver to focus on satisfying them first:[1]

- $w(\neg\kappa(a_i,a_j)) = 1, \ \ \forall a_i,a_j \in \mathcal{A}$
- $w(\neg x_a) = 1 + |\mathcal{A}|^2, \ \ \forall a \in \mathcal{A} \setminus \{a_I,a_G\}$

Note that the weight of any single action clause is greater than the weight of all ordering constraint clauses. Since the soft clauses are all unit clauses, we are able to use negation and solve the encoding with a MaxSAT procedure. A violation of any one of the unit clauses means that the solution includes the action or ordering constraint corresponding to the variable in the violated clause.

[1] In domains with non-uniform action cost we could replace the weight of 1 in the action clause with the cost of the action, allowing us to minimize the total cost of the actions in the POP.

Proposition 1. Given a planning problem Π and a valid POP $P = \langle \mathcal{A}, \mathcal{O} \rangle$, any variable setting that satisfies the formulae (1)-(7) will correspond to a valid POP for Π where the ordering constraints are transitively closed.

Proof sketch. We have already seen that the POP induced by a solution to the hard clauses will be a acyclic and transitively closed (due to formulae (1)-(5)). We can further see that there will be no open preconditions since we include a_G, and the conjunction of (7) ensures that every precondition will be satisfied when the POP includes an action. Additionally, there are no threats in the final solution because of formula (6), which will be enforced every time a precondition is met by formula (7). Since the POP corresponding to any solution to the hard clauses will have no open preconditions and no threats, the second notion of POP validity allows us to conclude that the POP will be valid for Π. □

Proposition 2. Given a planning problem Π and a valid POP $P = \langle \mathcal{A}, \mathcal{O} \rangle$, any valid POP $Q = \langle \mathcal{A}', \mathcal{O}' \rangle$, where $\mathcal{A}' \subseteq \mathcal{A}$ and $\mathcal{O}'$ is transitively closed, has a corresponding variable setting that satisfies formulae (1)-(7).

Proof sketch. The proposition follows from the direct encoding of the POP Q where $x_a = True$ iff $a \in \mathcal{A}'$ and $\kappa(a_i,a_j) = True$ iff $(a_i \prec a_j) \in \mathcal{O}'$. If Q is a valid POP, then it will be acyclic, include a_I and a_G, have all actions ordered after a_I and before a_G, and be transitively closed (satisfying (1)-(5)). We further can see that (6) and (7) must be satisfied: if (7) did not hold, then there would be an action a in the POP with a precondition p such that *every* potential achiever of p has a threat that could be ordered between the achiever and a. Such a situation is only possible when the POP is invalid, which is a contradiction. □

Proposition 3. Given a planning problem Π and a valid POP $P = \langle \mathcal{A}, \mathcal{O} \rangle$, a partial weighted MaxSAT solver will find a solution to the soft clauses and formulae (1)-(7) that minimizes the number of actions in the corresponding POP, and subsequently minimizes the number of ordering constraints.

Proof sketch. With $|\mathcal{A}|$ actions, there can only be $|\mathcal{A}|^2$ ordering constraints. Since every soft clause that corresponds to an ordering constraint has weight 1, the total sum of satisfying every ordering constraint clause will be $|\mathcal{A}|^2$. Since the weight of satisfying any action clause is greater than $|\mathcal{A}|^2$, the soft clauses corresponding to actions dominate the optimization criteria. As such, there will be no valid POP for Π which has a subset of the actions in P and has fewer actions than a solution that satisfies formulae (1)-(7) while maximizing the weight of the satisfied soft clauses. □

Theorem 1 (Encoding Correctness)**.** Given a planning problem Π, and a valid POP P for Π, a solution to our partial weighted MaxSAT encoding is a LCP for P.

Proof sketch. This follows from propositions 1, 2, and 3. □

Observe that (1)-(7) never use the sequential plan. An optimal solution to the encoding will correspond to a LCP. To enforce solutions that are minimum deorderings or reorderings, we introduce two sets of hard clauses.

All Actions For optimal deorderings and reorderings, we require every action to be a part of the POP. We consider a formula that ensures we use every action (and so the optimization works only on the ordering constraints). To achieve this, we simply need to add each action as a hard clause:

$$(x_a), \quad \forall a \in \mathcal{A} \tag{8}$$

Deordering For a deordering we must forbid any explicit ordering that contradicts the sequential plan. Assume our sequential plan is $[a_0, \cdots, a_k]$. We ensure that the computed solution is a deordering by adding the following family of hard unit clauses:

$$(\neg\kappa(a_j, a_i)), \quad 0 \leq i < j \leq k \tag{9}$$

Intuitively, (9) simply forbids any ordering that contradicts the orderings found in the transitive closure of the sequential plan, thus ensuring the solution is a deordering.

Due to space limitations, we refrain from proving the correctness of the two encoding extensions (8) and (9).

5 Relaxer Algorithm

We investigate the efficiency of an existing algorithm for relaxing a sequential plan to produce a deordering. Originally due to Kambhampati and Kedar (1994), the algorithm operates by removing ordering constraints from a sequential plan in a systematic manner. A heuristic guides the procedure, and as pointed out in (Bäckström 1998), the process does not provide any guarantee that the resulting POP is minimally constrained (that is, we may be able to remove further ordering constraints and the POP remains valid). Despite this lack of theoretical guarantee, we show later that the algorithm produces excellent results.

The intuition behind the algorithm is to remove any ordering $(a_i \prec a_k)$ from the sequential plan where a_i does not contribute to a precondition of a_k and a_i does not threaten a precondition of a_k (and vice versa). For example, consider the case where our sequential plan is $[a_1 \cdots, a_i, \cdots, a_k, \cdots, a_n]$ and $p \in PRE(a_k)$. The algorithm will keep the ordering $(a_i \prec a_k)$ only if leaving it out would create a threat for a precondition of one of the actions, or if a_i is the earliest action in the sequence where the following holds:

1. $p \in ADD(a_i)$: a_i is an achiever for p
2. $\forall a_j, i < j < k, p \notin DEL(a_j)$: p is not threatened.

Algorithm 1, which we will refer to as the *Relaxer Algorithm*, presents this approach formally. We use **index**(a,$\vec{a}$) to refer to the index of action a in the sequence $\vec{a}$.

If $\vec{a}$ is a valid plan, line 11 will evaluate to true before either line 8 evaluates to true or the for-loop at line 6 runs out of actions. That is, we know that an unthreatened achiever exists and the earliest such one is found.

The achiever is then added to the POP as a new causal link (line 14), and the for-loop at line 17 adds all of the necessary ordering constraints so the achiever remains unthreatened. Note that for any deleter found in this for-loop, either line 18 or 20 must evaluate to true.

Algorithm 1: Relaxer Algorithm

```
   Input: Sequential plan, a⃗, including a_I and a_G
   Output: Partial-order plan, ⟨A, O, C⟩
1  A = set(a⃗);
2  O = C = ∅;
3  foreach a ∈ A do
4      foreach f ∈ PRE(a) do
5          ach = Null;
6          for i = (index(a, a⃗) − 1) ··· 0 do
7              // Stop if we find a deleter of f
8              if f ∈ DEL(a⃗[i]) then
9                  break;
10             // See if we have an earlier achiever
11             if f ∈ ADD(a⃗[i]) then
12                 ach = a⃗[i];
13         // Add the appropriate causal link
14         C = C ∪ {(ach -f≺ a)};
15         O = O ∪ {(ach ≺ a)};
16         // Add orderings to avoid threats
17         foreach a′ ∈ deleters(f) \ {a} do
18             if index(a′, a⃗) < index(ach, a⃗) then
19                 O = O ∪ {(a′ ≺ ach)};
20             if index(a′, a⃗) > index(a, a⃗) then
21                 O = O ∪ {(a ≺ a′)};
22 return ⟨A, O, C⟩;
```

After going through the outer loop at line 3, every action in the newly formed POP has an unthreatened causal link for each of its preconditions. We are left with a valid POP, as there are no open preconditions or causal threats.

6 Evaluation

We evaluate the effectiveness of using the partial weighted MaxSAT solver, minimaxsat1.0 (Heras, Larrosa, and Oliveras 2008), to optimally relax a plan using our proposed encoding. To measure the quality of the POPs we generate, we consider the number of actions, ordering constraints, and linearizations (whenever feasible to compute). Further, we investigate the effectiveness of the Relaxer Algorithm to produce a minimally constrained deordering.

For our analysis, we use six domains from the International Planning Competition (IPC)[2] that allow for a partially ordered solution: Depots, Driverlog, Logistics, TPP, Rovers, and Zenotravel. These domains demonstrate both the strengths and weaknesses of our approach. We conducted the experiments on a Linux desktop with an eight-core 3.0GHz processor. Each run was limited to 30 minutes and 2GB of memory.

We generated an initial sequential plan by using the FF planner (Hoffmann and Nebel 2001). The encodings provided in Section 4 were converted to CNF using simple distributive rules so they may be used with the minimaxsat1.0 solver. We investigated the possibility of using a starting solution from the POPF planner (Coles et al. 2010), but

[2] `http://ipc.icaps-conference.org/`

Domain	Num Probs	FF Solved	Successfully Encoded
Depots	22	22	11
Driverlog	20	16	15
Logistics	35	35	30
TPP	30	30	7
Rovers	20	20	19
Zeno	20	20	15
ALL	147	143	97

Table 1: Number of instances successfully encoded. We indicate the number problems per domain, the number of problems for which FF finds a sequential plan, and the number of problems that our approach can encode successfully.

found that POPF failed to solve as many of the problems as FF (albeit POPF handles a far richer set of planning problems). Of the problems that were mutually solved by FF and POPF, we found that the POPs produced by the POPF planner were quite over-constrained, having many more ordering constraints than necessary. Once relaxed (by way of a similar encoding), the optimal deorderings, optimal reorderings, and LCPs of both FF and POPF plans were very similar. That is to say, there is usually little difference between the POPs generated by relaxing a sequential FF plan and the POPs generated by relaxing the POP found by POPF. For this reason, we only present the results for FF.

With two extensions for the encoding (**A**ll **A**ctions and **D**e**O**rdering) we have 4 possible variations for every instance. We will be primarily concerned with the following combinations for the settings:

- $\neg$ **AA**, $\neg$ **DO**: No additional clauses correspond to the default encoding where an optimal solution gives us a least commitment POP. We denote this setting as **LCP**.
- **AA**, $\neg$ **DO**: When we require all of the actions, solutions correspond to a minimum reordering of the sequential plan. We denote this setting as **MR**.
- **AA**, **DO**: When we require all of the actions, and a deordering, solutions correspond to a minimum deordering of the sequential plan. We denote this setting as **MD**.

In the following evaluation, we only report on the problems where FF was able to find a sequential plan. We found that many problems in the TPP and Depots domains are too large to encode in CNF. In these cases, we found that the combinatorial explosion for converting formulae (6) and (7) to CNF caused the encoding size to become too large. In Table 1 we present the number of problems per domain, solved by FF, and successfully encoded. Below we discuss a potential solution to dealing with this drawback.

POP Quality We begin by examining the relative quality of the POPs produced with different optimization criteria (LCP, MR, and MD), as well as the Relaxer Algorithm (abbreviated as RX). We report the number of actions and ordering constraints in the generated POP. Since the number of actions for RX, MR, and MD are equal to the number of actions in the sequential plan, we report the value only for RX and LCP. Table 2 shows the results for all six domains on

	$\lvert\mathcal{A}\rvert$		$\lvert\mathcal{O}\rvert$			
Domain	RX	LCP	RX	MD	MR	LCP
Depots (10)	34.2	30.9	451.8	451.8	407.9	339.1
Driverlog (15)	27.5	26.5	332.6	332.6	326.9	297.3
Logistics (25)	59.8	59.3	906.3	906.3	883.5	894.0
TPP (5)	13.4	13.4	74.8	74.8	74.8	74.8
Rovers (17)	30.6	30.1	214.3	214.3	208.8	200.2
Zeno (15)	19.8	19.8	137.1	137.1	136.6	136.6
ALL (87)	36.0	35.2	439.5	439.5	425.8	414.1

Table 2: Mean number of actions and ordering constraints for the various approaches. Numbers next to the domain indicate the number of instances solved by all methods (and included in the mean).

the problems for which every approach succeeded in finding a solution (87 of the 97 successfully encoded problems).

There are a few items of interest to point out. First, columns 4 and 5 coincide perfectly. It turns out, perhaps surprisingly, that the Relaxer Algorithm is able to produce the optimal deordering in every case, even though it is not guaranteed to do so. Since the algorithm can only produce deorderings, this is the best we could hope for from the algorithm. Second, we see the number of ordering constraints for the LCP approach is greater than those for the MR approach (on average) in the Logistics domain. The reason for this is because POPs in the Logistics domain require more ordering constraints for a solution with slightly fewer actions. For example, in prob15 the MR solution has 100 actions and 2278 ordering constraints, while the LCP solution reduces the number of actions in the POP to 96 at the expense of requiring 2437 ordering constraints.

In general, the LCP has fewer actions and ordering constraints than the optimal reordering, which in turn has fewer ordering constraints than the optimal deordering. If the LCP has the same number of actions as the sequential plan, then the LCP and minimum reordering coincide. We can see this effect in the TPP and Zenotravel where the number of actions and ordering constraints for LCP and MR are equal.

Finally, we note that in 4 problems (1 from Logistics, and 3 from Rovers), we found that the number of actions and constraints for either the LCP or MR POP matched that of the Relaxer POP, but the number of linearizations differed. Further, the differences in linearizations were not in the same direction (some better, and some worse). While the number of ordering constraints in a POP (for a given number of actions) usually gives an indication of the number of linearizations for that POP, these 4 problems indicate that this is not always the case.

Encoding Difficulty To measure the difficulty of solving the encoded problems, we computed the average time minimaxsat1.0 required to find an optimal solution (since the timing results had such a high standard deviation, we include both the mean and median values). It should be noted that an initial solution was consistently produced almost immediately by minimaxsat1.0's pre-processing step (a stochastic local search that satisfies every hard clause and serves as a lower bound on the optimal solution). Table 3 shows the average solving time for each domain given a particular set-

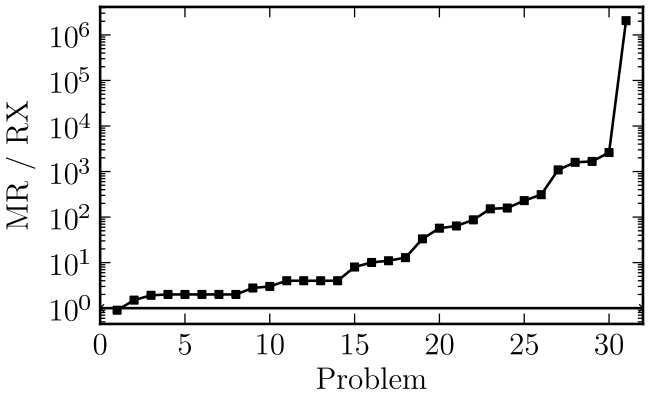

Figure 1: Ratio of Linearizations. The y-axis represents the number of linearizations induced by the POP for the optimal reordering divided by the number of linearizations induced by the POP for the optimal deordering. The x-axis ranges over all problems where the number of linearizations differed (∼40%), and is sorted based on the y-axis value.

ting. The largest solving time recorded was just under 500 seconds for a problem in the Rovers domain with the LCP setting. For comparison, we additionally include the average time the Relaxer Algorithm required to find a deordering.

For at least one of MR and LCP, 10 of the 97 problems successfully encoded proved too difficult for minimaxsat1.0 to solve, causing the solver to time out. We found that MD was on average easier to solve, but overall the majority of the problems were readily handled by minimaxsat1.0: 74% being solved in under 5 seconds. Being a polynomial algorithm, Relaxer consistently found a solution very quickly. The maximum time for a problem was just over 4 seconds.

Reordering Flexibility We have already seen that the Relaxer Algorithm is capable of producing the minimum deordering. To further evaluate the flexibility of the optimal deordering, we compare the number of linearizations induced by the optimal deordering with the number of linearizations induced by the optimal reordering. We found that of the 78 problems we could successfully compute the linearizations for, approximately 40% of the problems exhibited a difference between the optimal deordering and optimal reordering. Figure 1 shows the number of linearizations for the optimal reordering divided by the number of linearizations for the optimal deordering. For readability, we omit the 47 instances where the linearizations matched.

The ratio of linearizations ranges from 0.9 (an anomaly discussed below) to over two million. While the Relaxer Algorithm is proficient at finding the optimal deordering, this result demonstrates that there are still significant gains to be had in terms of flexibility by using the optimal reordering.

7 Discussion

The results paint an overall picture of how the optimization criteria compare to one another. We find that the Relaxer Algorithm is extremely adept at finding the optimal deordering, despite its lack of theoretical guarantee. In contrast, in many of the domains we see gains in terms of flexibility of the POP if we compute the optimal reordering or a least commitment POP. The encoding for the optimal deordering is easier for minimaxsat1.0 to solve compared to the optimal reordering or LCP, but the majority of problems for all optimization criteria were readily handled by minimaxsat1.0.

Whenever possible, we used the number of linearizations a POP represents as a measure of the POP's flexibility (this may not always be possible to compute due to the structure and size of the POP). We found that there are problems where the number of actions and ordering constraints in two POPs are equal, while the number of linearizations is not. Since the objective function of the encoding includes only the number of actions and ordering constraints, there is no guarantee on the number of linearizations that will result from a computed POP. An optimal reordering may even have fewer linearizations than an optimal deordering (which was observed in one case, as seen in Figure 1).

For a concrete example, consider two POPs on four actions $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$. Ignoring causal links, Figure 2 shows the structure of the POPs P_1 and P_2. Both POPs have the same number of actions and ordering constraints, but the number of linearizations differ: P_1 has 6 linearizations while P_2 only has 5. These POPs serve as a basic example of how the LCP criteria does not fully capture the notion of POP flexibility. However, we should point out that fewer ordering constraints usually indicates more linearizations.

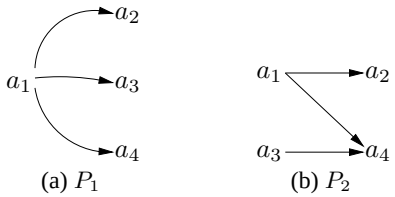

Figure 2: Two POPs with the same number of actions and ordering constraints, but different number of linearizations.

For the Depots and TPP domains, the encoding size became prohibitive. The larger formulae (i.e., (6) and (7)) are not too large in themselves, but when converting to CNF the theory becomes large. In future work, we plan to use the Tseitin encoding to convert the theory (Tseitin 1970). The Tseitin encoding will allow us to avoid the exponential blow-up in theory size, at the expense of introducing more variables. We also hope to investigate versions of partial weighted MaxSAT solvers tailored to problems in which only unit clauses are soft (as is the case with our encoding). There are other optimization techniques we plan on investigating, including constraint programming encodings, mixed-integer programming models, and a restricted form of partial-order causal link (POCL) search.

The encoding technique we have presented differs significantly from the standard SAT-based planning encodings. In particular, we avoid the need to encode an action for every layer in a planning graph by appealing to the fact that we already know the (superset of) actions that will be in the solution. The core of our encoding follows an approach similar

Domain	Mean Time				Median Time				Mean	Mean # Clauses		
	RX	MD	MR	LCP	RX	MD	MR	LCP	# Vars	MD	MR	LCP
Depots	0.54	89.63	55.63	19.72	0.44	1.26	1.55	1.71	1346.55	398205.91	397566.36	397532.64
Driverlog	0.58	1.35	38.36	6.02	0.55	0.77	0.70	0.87	943.87	39082.60	38638.20	38610.67
Logistics	1.16	9.40	127.66	105.02	0.80	3.13	5.69	4.98	7378.00	927265.59	923648.79	923576.59
TPP	0.87	3.58	171.71	171.68	0.61	0.44	0.50	0.50	1024.57	110964.86	110477.00	110452.57
Rovers	1.14	3.88	72.78	81.21	1.00	1.90	1.97	2.34	1428.74	111052.00	110371.11	110337.63
Zeno	0.60	0.86	0.85	0.87	0.46	0.42	0.53	0.43	541.73	18728.53	18477.47	18457.67
ALL	0.89	14.48	77.99	63.65	0.61	1.16	1.28	1.38	2972.67	364842.46	363397.60	363356.12

Table 3: Average time for the MaxSAT encoding to be solved by minimaxsat1.0, average time for the Relaxer Algorithm to compute a deordering, and average number of clauses and variables in the encoding. Mean and median values of run time are given in seconds. The mean is used to compute the average number of variables and clauses.

to Variant-II of Robinson *et al*. (Robinson et al. 2010). We similarly encode the ordering between any pair of actions as a variable ($\kappa(a_i, a_j)$), but rather than encoding a relaxed planning graph, we encode the formulae that need to hold for a valid POP. There are also similarities between our work and that of (Do and Kambhampati 2003). In particular, the optimization criteria for minimizing the number of ordering constraints coincide, as does the optional use of constraints to force a deordering. However, while Do and Kambhampati focus on temporal relaxation in the context of action ordering, we take the orthogonal view to minimize the number of actions required.

It is natural to consider the impact the choice of initial plan has on the final POP. As was mentioned earlier, the choice of initial solution between FF and POPF makes little difference in the quality of the optimally relaxed POP. The question remains open, however, on how to best compute an initial set of actions for our encoding.

8 Conclusion

In this paper we proposed a practical method for computing the optimal deordering and reordering of a sequential plan. Despite the theoretical complexity of computing the optimal deordering or reordering being NP-hard, we are able to compute the optimal solution by leveraging the power of modern MaxSAT solvers. We further propose an extension to the classical least commitment criterion that considers the number of actions in a solution, and demonstrate the added flexibility of a POP that satisfies this criterion.

Our approach uses a family of novel encodings for partial weighted MaxSAT where a solution corresponds to an optimal POP satisfying one of the three criteria we investigate (minimal deordering, minimal reordering, and our proposed least commitment POP). We solve the encoding with a state-of-the-art partial weighted MaxSAT solver, minimaxsat1.0, and find that the majority of problems are readily handled by minimaxsat1.0. We do, however, find that two domains present a problem for the encoding phase of our approach. In TPP and Depots, we find that the encoding size becomes too large to handle, which limits the applicability of our current encoding technique. In the future, we hope to employ the Tseiten encoding to limit this drawback.

We also investigated an existing algorithm for deordering sequential plans, and discovered that it successfully computes the optimal deordering in every problem we tested (despite its lack of theoretical guarantee). Since the algorithm is polynomial, and quite fast in practice, it is well suited for relaxing a POP if we require a deordering. However, if a reordering or least commitment POP is acceptable, then we can produce a far more flexible POP by using one of the proposed encodings.

Acknowledgements

The authors gratefully acknowledge funding from the Ontario Ministry of Innovation and the Natural Sciences and Engineering Research Council of Canada (NSERC). We would also like to thank the anonymous referees for useful feedback on earlier drafts of the paper.

References

Bäckström, C. 1998. Computational aspects of reordering plans. *Journal of Artificial Intelligence Research* 9(1):99–137.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T. 2009. Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Twentieth International Conference on Automated Planning and Scheduling (ICAPS 10)*.

Do, M., and Kambhampati, S. 2003. Improving the temporal flexibility of position constrained metric temporal plans. In *AIPS Workshop on Planning in Temporal Domains (AIPS 03)*.

Heras, F.; Larrosa, J.; and Oliveras, A. 2008. MiniMaxSAT: An efficient weighted Max-SAT solver. *Journal of Artificial Intelligence Research* 31(1):1–32.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14(1):253–302.

Kambhampati, S., and Kedar, S. 1994. A unified framework for explanation-based generalization of partially ordered and partially instantiated plans. *Artificial Intelligence* 67(1):29–70.

Robinson, N.; Gretton, C.; Pham, D. N.; and Sattar, A. 2010. Partial weighted maxsat for optimal planning. In *Proceedings of the 11th Pacific Rim International Conference on Artificial Intelligence, Daegu, Korea, August 30 - September 02, 2010*.

Russell, S., and Norvig, P. 2009. *Artificial intelligence: a modern approach*. Prentice hall.

Tseitin, G. 1970. On the complexity of proofs in propositional logics. In *Seminars in Mathematics*, volume 8, 1967–1970.

Veloso, M.; Pollack, M.; and Cox, M. 1998. Rationale-based monitoring for planning in dynamic environments. In *Artificial Intelligence Planning Systems*, 171–179.

Weld, D. 1994. An introduction to least commitment planning. *AI Magazine* 15(4):27.