Optimising Partial-Order Plans Via Action Reinstatement

Max Waters, Lin Padgham and Sebastian Sardina
RMIT University, Melbourne, Victoria 3000, Australia
{max.waters, lin.padgham, sebastian.sardina}@rmit.edu.au

Abstract
This work investigates the problem of optimising a partial-order plan’s (POP) flexibility through the simultaneous transformation of its action ordering and variable binding constraints. While the former has been extensively studied through the notions of deordering and reordering, the latter has received much less attention. We show that a plan’s variable bindings are often related to resource usage and their reinstatiation can yield more flexible plans. To do so, we extend existing POP optimality criteria to support variable reinstatiation, and prove that checking if a plan can be optimised further is NP-complete. We also propose a MAXSAT-based technique for increasing plan flexibility and provide a thorough experimental evaluation that suggests that there are benefits in action reinstatiation.

1 Introduction
This work investigates the problem of optimising a partial-order plan’s (POP) flexibility through the simultaneous transformation of its action ordering and variable binding constraints. While the former has been extensively studied through the notions of deordering and plan reordering, both from a theoretical [Bäckström, 1998; Aghighi and Bäckström, 2017] and practical [Kambhampati and Kedar, 1994; Muise et al., 2016; Siddiqui and Haslum, 2012; Say et al., 2016] perspective, the optimisation of resources (or actions’ domain objects) that are used in the course of executing a plan has received much less attention.

Ultimately, the aim is to achieve a least commitment approach to planning [Weld, 1994] that maximises execution-time flexibility by delaying decisions regarding action ordering and resource utilisation for as long as possible. Consider a planning instance from (a reduced version of) the IPC rovers domain, in which a fleet of rovers must navigate the surface of a planet, collecting soil and rock samples. The domain objects comprise two rovers (R1 and R2) and three waypoints (W1, W2, and W3). There are soil and rock samples at waypoints W2 and W3, resp., and both rovers begin at W1. The goal is to gather both samples. Plan P1 in Figure 1a is an optimal solution: R1 navigates from W1 to W2 and collects the soil sample, then navigates to W3 and collects the rock sample. An examination of the causal structure of P1 shows that

1. move1(R1, W1, W2)
2. get-soil(R1, W2)
3. move2(R1, W2, W3)
4. get-rock(R1, W3)

(a) Plan P1.

1. move1(R1, W1, W2)
2. get-soil(R1, W2)
3. move2(R2, W1, W3)
4. get-rock(R2, W3)

(b) Plan P2.

\[ O = \{ move_1(x_1, x_2, x_3) \} \]

\[ \theta = \{ x_1 \backslash W_1, x_2 \backslash W_2, x_3 \backslash W_3 \} \]

\[ move_2(x_4, x_7, x_8) \]

\[ get-soil(x_4, x_7) \]

\[ get-rock(x_9, x_{10}) \]

\[ \leftarrow \{ move_1 \prec get-soil, move_2 \prec get-rock \} \]

(c) Partial-order plan P3.


Figure 1: Three plans from the rovers domain. Subscripts have been used to better distinguish between actions of the same type.

the order of its actions cannot be altered without compromizing its validity (e.g., step 3 threatens the causal link between steps 1 and 2, as it moves R1 away from W2).

Plan P2 in Figure 1b is a modification of P1 that instead uses rover R2 to collect the rock sample. This modification allows the plan’s ordering constraints to be relaxed: the actions can be executed in any order so long as 1 \preceq 2 and 3 \preceq 4.

This simple example shows that an over-commitment to a particular set of resources (in this case, R1) can reduce a plan’s possible orderings. Thus, this paper introduces the notions of reinstatiated deorderings and reinstatiated reordering, namely, transformations of a plan under which ordering constraints can be removed or arbitrarily modified, resp., and the plan’s actions’ variable bindings can be changed. The idea is to seek alternative variable bindings supporting greater minimisation of the plan’s ordering constraints than is possible if the bindings remained unchanged.

More concretely, we define optimality criteria for partial-order plans (POP) that take variable bindings into consideration (Section 3) and propose a technique for computing optimally reinstatied POPs by encoding the problem into a MAXSAT instance (Section 4). Finally, we experimentally assess the benefits of reinstatiation by comparing it with existing de/reordering techniques (Section 5). Results show that optimising both bindings and orderings incurs a much greater computational cost than optimising orderings alone. However, when (even non-optimal) reinstatied de/reorderings can be found, they are significantly more flexible than plans produced by standard optimisation methods.
2 Preliminaries

2.1 Logical Preliminaries

The logical structures will be expressed in a function-free first-order language \( \mathcal{L} \) with finitely many variable, constant, and predicate symbols, and the usual logical connectives, including equality and precedence [Brachman and Levesque, 2004, Chapter 2]. The letters \( x \), \( y \) and \( z \) indicate variables, \( c \) constants and \( t \), \( u \) and \( v \) terms (all possibly with annotations). The notation \( \bar{t} \) denotes ordered lists of possibly non-unique terms, with \( t[i] \) indicating the \( i \)-th element of the list and \( t_1 \circ t_2 \) being shorthand for \( |t_1| = |t_2| \wedge t_1[1] = t_2[1] \wedge \cdots \wedge t_1[n] = t_2[n] \). For any structure \( \eta \) expressed in \( \mathcal{L} \), \( \text{vars}(\eta) \) and \( \text{consts}(\eta) \) denote the variables and constants appearing in \( \eta \), resp.

A substitution \( \theta \) is a mapping from variables to terms, for example \( \theta = \{ x_1/t_1, \ldots, x_n/t_n \} \) maps each variable \( x_i \) to term \( t_i \), for \( 1 \leq i \leq n \), and every other variable to itself. We use \( \text{domain}(\theta) \) to denote the set of variables explicitly mapped by substitution \( \theta \), and \( \theta(x) \) to denote the term corresponding to variable \( x \) under \( \theta \). The result of applying a substitution \( \theta \) to a logical structure \( \eta \) is written \( \eta^\theta \), and means to simultaneously replace every variable \( x \) in \( \eta \) with \( \theta(x) \). A substitution \( \theta \) is ground if every variable in its domain is mapped to a ground term, and is complete w.r.t. \( \eta \) iff \( \text{vars}(\eta) \subseteq \text{domain}(\theta) \).

2.2 Classical Planning Formalism

Classical planning tasks will be expressed in a standard first-order STRIPS formalism [Lifschitz, 1987].

Classical Planning Tasks

A classical planning task is a tuple \( \Pi = \langle \mathcal{L}, \mathcal{O}, s_f, s_G \rangle \), where \( \mathcal{L} \) is a set of constants, \( \mathcal{O} \) is a set of operators, and \( s_f \) and \( s_G \) are sets of ground literals describing the initial state and goal, respectively. An operator is a tuple \( o = \langle \text{name}(o), \text{vars}(o), \text{pre}(o), \text{post}(o) \rangle \), where \( \text{name}(o) \) is the name, or type of the operator, \( \text{vars}(o) \) is a list of variables and \( \text{pre}(o) \) and \( \text{post}(o) \) are finite sets of (ground or non-ground) literals with variables taken from \( \text{vars}(o) \). It will be assumed that for all operators, \( \text{name}(o) = o \), and so operators will be referred to by name. When distinguishing between different operators of the same name, the notation \( o(\bar{x}) \) is used, where \( \text{vars}(o) = \bar{x} \). An action is a ground operator \( a = \langle \text{pre}(a), \text{post}(a) \rangle \), where \( \text{pre}(a) \) and \( \text{post}(a) \) are finite sets of ground literals. If \( o \) is an operator and substitution \( \theta \) is ground and complete w.r.t. \( \text{vars}(o) \), then \( \text{pre}(o) \theta \) and \( \text{post}(o) \theta \) are the action resulting from instantiating \( o \) with \( \theta \).

Classical and Partial-Order Plans

A classical plan \( \bar{a} \) is a finite sequence of actions. Assuming that no variable appears in more than one operator, a plan can also be represented as \( \bar{o} \theta \), where \( \bar{o} \) is a list of operators and \( \theta \) is a ground substitution that is complete w.r.t. \( \bar{o} \). A planning task can be “embedded” into a plan by including the distinguished actions \( a_f \) and \( a_G \) whose pre/postconditions are the task’s initial state and goal, resp. In this form, a classical plan is valid iff it is executable (i.e., the actions can be applied in sequence without violating their precondition requirements).

A partial-order plan (POP) is a generalised classical plan that need not completely define the order of its operators:

Definition 1. A partial-order plan (POP) is a tuple \( P = \langle O, \theta, \prec \rangle \) where \( O \) is a set of operators, \( \theta \) is a ground substitution which is complete w.r.t. to \( O \), and \( \prec \) is a strict, transitively closed partial order over \( O \).

As with classical plans, a planning task is “embedded” into a POP via the inclusion of the distinguished parameter-free operators \( o_f \) (which yields the initial state) and \( o_G \) (which checks the goals). A POP is a compact representation of a set of classical plans, known as its linearisations:

Definition 2. A classical plan \( \langle o_1, \ldots, o_n \rangle \theta \) is a linearisation of POP \( P = \langle O, \theta, \prec \rangle \) iff \( O = \{ o_1, \ldots, o_n \} \) and \( \prec \subseteq \{ a_i \prec a_j : 1 \leq i < j \leq n \} \).

A POP’s validity is defined w.r.t. its linearisations:

Definition 3. A POP is valid iff all of its linearisations are executable.

Classical plans are special cases of POPs, and any classical plan \( \langle o_1, \ldots, o_n \rangle \theta \) can be expressed as an equivalent partial-order plan \( \langle o_1, \ldots, o_n \rangle \theta \) [Weld, 1994], not the validity of its linearisations. A partial-order causal link planning (POCL) [Bäckström, 1998] is typically used to describe a POP’s causal structure by identifying which actions produce, consume or threaten which ground literals. Here, it describes how operators produce, consume or threaten (possibly non-ground) literals:

Definition 4. Let \( o \) be an operator and \( q(\bar{t}) \) a literal. Then:

\[
\begin{align*}
\text{prods}(o, q(\bar{t})) & \defeq q(\bar{t}) \in \text{post}(o). \\
\text{cons}(o, q(\bar{t})) & \defeq q(\bar{t}) \in \text{pre}(o). \\
\text{thrtns}(o, q(\bar{t})) & \defeq \neg q(\bar{t}) \in \text{post}(o).
\end{align*}
\]

In the field of partial-order causal link planning (POCL), a PCT-based notion of POP validity is used that derives from the implied causal dependencies between a POP’s operators [Weld, 1994], not the validity of its linearisations. A causal link associates a consumer, (i.e., an operator precondition), with a producer (i.e., an operator postcondition):

Definition 5. A causal link is a 4-tuple \( \langle o_p, q(\bar{t}), o_c, q(\bar{u}) \rangle \) s.t. \( \text{prods}(o_p, q(\bar{t})) \) and \( \text{cons}(o_c, q(\bar{u})) \).

The consumer is supported by the causal link iff it is preceded by, and codesigned with, the producer (i.e., \( \theta(\bar{t}) = \theta(\bar{u}) \)). The causal link is threatened iff a codesignated threat (i.e., \( o_t, q(\bar{v}) \) s.t. \( \text{thrtns}(o_t, q(\bar{v})) \) and \( \theta(\bar{v}) = \theta(\bar{u}) \)) can be ordered between the producer and consumer. A POP’s implicit unthreatened causal links are denoted \( L_P \), and defined as follows:

Definition 6. The unthreatened causal links of a POP \( P = \langle O, \theta, \prec \rangle \) is the set \( L_P \) s.t. \( \langle o_p, q(\bar{t}), o_c, q(\bar{u}) \rangle \in L_P \) iff:

1. \( o_p \prec o_c \).
2. \( \theta(\bar{t}) = \theta(\bar{u}) \), and
3. for all \( o_t, q(\bar{v}) \) s.t. \( \text{thrtns}(o_t, q(\bar{v})) \), either \( \theta(\bar{u}) \neq \theta(\bar{v}) \), \( o_t \prec o_p \) or \( o_c \prec o_t \).
A POP \( P \) is \textit{POCL-valid} iff every consumer is supported by an unthreatened causal link:

\[ \text{Definition 7. A POP } P = \langle O, \theta, \prec \rangle \text{ is POCL-valid iff for all } o_c, q(u) \in O \text{ s.t. } \text{cons}(o_c, q(u)), \text{ there exist } o_p \in O \text{ and } q(\tilde{f}) \text{ s.t. } (o_p, q(\tilde{f}), o_c, q(u)) \in LP. } \]

This notion of validity is stronger than that in Definition 3, meaning that a POP might not be POCL-valid despite its linearisations being executable [Kambhampati and Nau, 1996]. However, any such POP can always be made POCL-valid by adding ordering constraints [Bercher and Olz, 2020].

2.3 Partial Weighted MAXSAT

The partial weighted maximum satisfiability (partial weighted MAXSAT) problem is a generalised optimisation variant of SAT that distinguishes between soft clauses, which are assigned a numeric weight, and hard clauses, which are unweighted. The aim of the partial weighted MAXSAT problem is to find an interpretation that satisfies all hard clauses and maximises the total weight of the satisfied soft clauses. The syntax \( \phi \) indicates that clause \( \phi \) has weight \( k \), and no weight marking indicates a hard clause. For example, the formula \( (p_1 \lor p_2) \land \neg p_1 \land \neg p_2 \) is trivially unsatisfiable, but the interpretation \( \{ p_1 / T, p_2 / \perp \} \) is an optimal solution to the partial weighted MAXSAT problem.

3 Optimality Criteria for Partial-Order Plans

Bäckström [1998]’s seminal work on action ordering and plan flexibility considers two types of modification: deordering, in which constraints can be removed but not added, and reordering, which allows both. Both must result in a valid POP:

\[ \text{Definition 8. Let } P = \langle O, \theta, \prec \rangle \text{ and } Q = \langle O, \theta, \prec' \rangle \text{ be two POPs. Then:} \]

- \( Q \) is a reordering of \( P \) iff they are both valid.
- \( Q \) is a deordering of \( P \) iff it is a reordering and \( \prec' \subseteq \prec \).
- \( Q \) is a strict deordering of \( P \) iff it is a deordering and \( \prec' < \prec \).

Bäckström defines the relative optimality of two POPs by comparing their ordering constraints. A minimal deordering of a POP cannot be relaxed any further without rendering the POP invalid, and a minimum deordering is the smallest of all valid reorderings of a POP. A minimum reordering has the fewest possible ordering constraints while remaining valid:

\[ \text{Definition 9. Let } P = \langle O, \theta, \prec \rangle \text{ and } Q = \langle O, \theta, \prec' \rangle \text{ be two POPs. Then:} \]

- \( Q \) is a minimal deordering of \( P \) iff \( Q \) is a deordering of \( P \) and there is no POP \( R \) such that \( R \) is a strict deordering of \( Q \).
- \( Q \) is a minimum deordering of \( P \) iff \( Q \) is a deordering of \( P \) and there is no POP \( R = \langle O, \theta', \prec'' \rangle \) such that \( R \) is a deordering of \( P \) and \( |\prec''| < |\prec'| \).
- \( Q \) is a minimum reordering of \( P \) iff \( Q \) is a reordering of \( P \) and there is no POP \( R = \langle O, \theta', \prec'' \rangle \) such that \( R \) is a reordering of \( P \) and \( |\prec''| < |\prec'| \).

While a minimal deordering of a given POP can be found in polynomial time, deciding whether there exists a de/reordering with fewer than \( k \) ordering constraints is NP-complete, and finding a minimum de/reordering is NP-hard and cannot be approximated within a constant factor.

A key limitation of the optimality criteria above is the assumption that the POP’s variable bindings remain static. We thus define more generalised criteria that accommodate modifications to a POP’s variable bindings. In all cases, the resulting POP must remain both ground (i.e., its variable bindings must be complete and ground w.r.t. \( O \)) and valid:

\[ \text{Definition 10. Let } P = \langle O, \theta, \prec \rangle \text{ and } Q = \langle O, \theta', \prec' \rangle \text{ be two POPs. Then:} \]

- \( Q \) is a reinstatiated reordering of \( P \) iff both \( P \) and \( Q \) are valid.
- \( Q \) is a reinstatiated deordering of \( P \) iff \( Q \) is a reinstatiated deordering of \( P \) and \( \prec' \subseteq \prec \).
- \( Q \) is a reinstatiated strict deordering of \( P \) iff \( Q \) is a reinstatiated deordering of \( P \) and \( \prec' < \prec \).

For example, plans \( P_1,P_2 \) in Figure 1 use the same operators, and so they are reinstatiated reorderings of each other, whereas \( P_3 \) is a reinstatiated strict deordering of \( P_1 \) and \( P_2 \).

The notion of a “least-constrained” POP can now be generalised to allow for changes in variable bindings. A \textit{minimal reinstatiated deordering} of a POP is a reinstatiated deordering that cannot be relaxed any further: no modification to the POP’s variable bindings will allow for the removal of any ordering constraints. A \textit{minimum reinstatiated deordering} of a POP has the fewest ordering constraints of all of its reinstatiated reorderings. Of all the possible modifications to a POP’s ordering and binding, a \textit{minimum reinstatiated reordering} contains the fewest ordering constraints:

\[ \text{Definition 11. Let } P = \langle O, \theta, \prec \rangle \text{ and } Q = \langle O, \theta', \prec' \rangle \text{ be two POPs. Then:} \]

- \( Q \) is a minimal reinstatiated deordering of \( P \) iff \( Q \) is a reinstatiated deordering of \( P \) and there is no POP \( R \) such that \( R \) is a reinstatiated strict deordering of \( Q \).
- \( Q \) is a minimum reinstatiated deordering of \( P \) iff \( Q \) is a reinstatiated deordering of \( P \) and there is no POP \( R = \langle O, \theta'', \prec'' \rangle \) such that \( R \) is a reinstatiated deordering of \( P \) and \( |\prec''| < |\prec'| \).
- \( Q \) is a minimum reinstatiated reordering of \( P \) iff \( Q \) is a reinstatiated reordering of \( P \) and there is no POP \( R = \langle O, \theta'', \prec'' \rangle \) such that \( R \) is a reinstatiated reordering of \( P \) and \( |\prec''| < |\prec'| \).

For example, plan \( P_3 \) has two ordering constraints, and as there is no reinstatiation of \( P_1 \) which allows for fewer than this, \( P_3 \) is a minimum reinstatiated reordering of \( P_1 \).

Complexity Results

Deciding whether a POP has a reinstatiated de/reordering with fewer than \( k \) ordering constraints is NP-complete, and the optimisation problem of finding a minimum reinstatiated de/reordering cannot be approximated within a constant factor (proofs for minimum reinstatiated deordering are provided, those for reordering are a trivial modification):
Theorem 1. Given a POP $P = \langle O, \theta, \prec \rangle$ and an integer $k > 0$, determining whether there exists a POP $Q = \langle O, \theta', \prec' \rangle$ s.t. $Q$ is a reinstatiated deordering (or reordering) of $P$ and $|\prec'| < k$ is NP-complete.

Proof. For membership, guess a POP $Q = \langle O, \theta', \prec' \rangle$ and verify in P-time that $Q$ is valid, $\prec' \subseteq \prec$ and $|\prec'| < k$. Hardness is by reduction from the NP-complete [Bäckström, 1998] decision problem of minimum deordering, which asks whether $P$ has a deorder with $< k$ ordering constraints. Construct $P'' = \langle O', \theta, \prec \rangle$ where $o \in O$ iff there exists a $o' \in O'$ s.t. $\text{vars}(o') = \text{vars}(o)$, $\text{post}(o') = \text{post}(o)$ and $\text{pre}(o') = \text{pre}(o) \cup \{ x = \theta(x) : x \in \text{vars}(o') \}$. As the operators in $O'$ have preconditions that “fix” the variable bindings, $P$ has a deorder with $< k$ ordering constraints iff $P''$ has a reinstatiated deorder with $< k$ ordering constraints. □

Theorem 2. The problem of finding a minimum reinstatiated deordering (or reordering) of a POP is not in APX unless $\text{NP} \subseteq \text{DTIME}(\text{poly log } n)$.

Proof. Proof is by reduction. Assume that minimum reinstatiated deordering is in APX. Then there must be a function $A$, that approximates it within a constant factor. As the reduction in the Theorem 1 proof preserves solutions, $A$ also approximates minimum deordering within a constant factor, which is impossible unless $\text{NP} \subseteq \text{DTIME}(\text{poly log } n)$ [Bäckström, 1998]. □

Optimality Under POCL-validity

Under the additional requirement that the input and optimised POPs be POCL-valid, minimum de/reordering remain NP-complete (a trivial corollary of Theorem 4.8 in [Bäckström, 1998]). However, due to the stronger requirements of POCL-validity there exist minimum POCL-valid de/reorderings that can be further optimised while remaining valid under Definition 3 [Kambhampati and Nau, 1996]. These complexity and completeness results can be trivially extended to minimum reinstatiated POCL-valid de/reordering.

4 Partial Weighted MAXSAT Encoding

The problem of optimising a POP’s ordering can be naturally expressed as an instance of the partial weighted MAXSAT problem (Section 2.3). This approach was introduced by Muijse et al. [2016], whose MD and MR encodings transform a POP into MAXSAT instances with optimal solutions corresponding to minimum POCL-valid de/reorderings, resp.

This section introduces MRD and MRR: generalised encodings that allow the POP to be reinstatiated. Their optimal solutions therefore correspond to minimum POCL-valid reinstatiated de/reorderings, resp. They also optimise MD and MR by breaking symmetries and removing propositions and clauses that are not required to enforce plan validity.

We note due to the difficulty of expressing Definition 3 into MAXSAT, all four encodings instead preserve plan validity with the POCL requirements in Definition 7. While easier to encode, this raises the possibility that the resulting optimised POPs contain unnecessary ordering constraints.

4.1 The MD and MR Encodings

MD and MR use two “types” of propositional variables. If $P = \langle O, \prec, \theta \rangle$ is the input POP, then for each pair of actions $a_1, a_2 \in O\theta$, a proposition $p_{a_1 \prec a_2}$ is introduced to indicate that $a_1$ must precede $a_2$ in the resulting POP. For all $a_c, a_p$ and $q$ such that $a_c$ consumes $q$ and $a_p$ produces $q$, a proposition $p_{a_p, q, a_c}$ is introduced to encode the requirement that in the final POP, $a_p$ be causally linked to $a_c$ with respect to $q$.

Formulæ 1–5 ensure that the output POP is acyclic and transistively closed with all actions ordered between the initial state and goal:

\[ \bigwedge_{a} \neg p_{a \prec a}. \] (1)
\[ \bigwedge_{a_1, a_2, a_3} p_{a_1 \prec a_2} \wedge p_{a_2 \prec a_3} \rightarrow p_{a_1 \prec a_3}. \] (2)
\[ \bigwedge_{a \notin \{a_1, a_2\}} p_{a \prec a}. \] (3)

Formulæ 4 and 5 encode POCL-validity as in Definition 7:

\[ \bigwedge_{a_p, a_c, q, a_x} p_{a_p, q, a_x} \rightarrow \bigwedge_{a_p} p_{a_p, a_x} \lor p_{a_x, a_1}. \] (4)
\[ \bigwedge_{a_x, q, a_x} \bigvee_{a_p} p_{a_p, a_x} \land p_{a_p, q, a_x}. \] (5)

Formula 6 adds a soft unit clause with the negation of each ordering proposition, meaning that higher weight solutions will have fewer ordering constraints:

\[ \bigwedge_{a_1, a_2} \neg p_{a_1 \prec a_2}. \] (6)

Definition 12. MR encodes a POP into a partial weighted MAXSAT instance through Formulæ 1–6.

The MD encoding extends MR with clauses that disallow any ordering constraints that were not in the input plan, thus forcing the resulting POP to be deordering of the input:

\[ \bigwedge_{(a_1, a_2) \notin \prec} \neg p_{a_1 \prec a_2}. \] (7)

Definition 13. MD encodes a POP into a partial weighted MAXSAT instance through Formulæ 1–7.

4.2 The MRD and MRR Encodings

MRD and MRR allow the input POP to be reinstatiated, and so use different variable “types” to express the allowable rebindings. Three types are used, as per the following sets:

- $\mathcal{P}_<$ contains propositions of the form $p_{a_1 \prec a_2}$, which indicate that $a_1 \prec a_2$ in the resulting POP.
- $\mathcal{P}_\theta$ contains propositions of the form $p_{\theta(t) = \theta(u)}$, which encode a requirement that in the final POP $\theta(t) = \theta(u)$. 
\[ \begin{align*}
& p(x_1, x_2, x_3, x_4) \\
& a_1(x_1) \quad q(x_1) \\
& a_2(x_2) \quad q(x_2) \\
& a_3(x_3, x_4) \quad b(x_3, x_4) \\
& p \quad q(x_3)
\end{align*} \]

Figure 2: Plan \( P_4 \) from a synthetic domain. Pre/postconditions are to the left/right of operators, resp., and arrows indicate causal links.

- \( \mathcal{P}_C \) contains propositions of the form \( p_{o_p, \bar{c}_o, \bar{c}_c} \) indicating that there must be some \( q \) s.t. causal link \( \langle o_p, q(\bar{t}), o_c, q(\bar{u}) \rangle \) is unthreatened in the final POP.

Miu\'se et al. [2016] observed that the cubic number of clauses generated by Formula 2 make MD and MR infeasible for large plans. As this is exacerbated in MRD and MRR by the need to also close the variable equality relation, a number of optimisations are introduced.

The first removes any ordering or binding constraints (and propositions that encode them) that are not required to preserve causal links. For example, consider Plan \( P_4 \) in Figure 2. Operator \( a_1 \) cannot be used to produce any of \( o_2 \)'s preconditions, and nor does it threaten any of its postconditions (nor vice-versa). Therefore, no minimum de/reordering of \( P_4 \) will require that \( a_1 \prec o_2 \) or \( o_2 \prec a_1 \) or that \( x_1 \) and \( x_2 \) be (non-) codified. MRD and MRR therefore exclude the propositions \( p_{a_1, o_2, o_1, p_{x_1, x_2}} \) and \( p_{x_1, x_2} \).

The second optimisation prevents the encoding of possible causal links if the required ordering constraints are known to be unsatisfiable. For example, Formula 7 in MD enforces de/reordering with negated unit clauses. Rather than include these, Pops can contain multiple operators of the same type.

Definition 14. \( o_1, o_2, q(\bar{t}) \) and \( q(\bar{u}) \) are causally related w.r.t. \( \prec_A \) if \( o_1 \prec_A o_2 \) and either:

1. \( \text{prods}(o_1, q(\bar{t})) \) and \( \text{cos}(o_2, q(\bar{u})) \),
2. \( \text{thtrns}(o_1, q(\bar{t})) \) and \( \text{prods}(o_2, q(\bar{u})) \), or
3. \( \text{cos}(o_1, q(\bar{t})) \) and \( \text{thtrns}(o_2, q(\bar{u})) \).

The sets of propositional variables can now be defined:

Definition 15. Let \( P = \langle O, \Theta, \prec \rangle \) be a POP and \( \prec_A \subseteq O^2 \) be a set of allowable orderings. Then \( \mathcal{P}_<_A \), \( \mathcal{P}_0 \), and \( \mathcal{P}_C \) are the smallest sets s.t.:

- \( p_{o_1 \prec o_2} \in \mathcal{P}_<_A \) iff either (i) there exists a \( q(\bar{t}) \) and \( q(\bar{u}) \) s.t. \( o_1, o_2, q(\bar{t}) \) and \( q(\bar{u}) \) are causally related w.r.t. \( \prec_A \), or (ii) there exists an \( o_3 \) s.t. \( p_{o_1 \prec o_3, p_{o_3 \prec o_2}} \in \mathcal{P}_<_A \).

\[ \mathcal{P}_0 \] iff either (i) there exists a \( o_1, o_2, q(\bar{t}) \) and \( q(\bar{u}) \) that are causally related w.r.t. \( \prec_A \) and some \( i \) s.t. \( \bar{t}[i] = t, \bar{u}[i] = u \), or (ii) there exists a \( v \) s.t. \( p_{t=v, p_{v=u}} \in \mathcal{P}_0 \).

- \( p_{o_p, \bar{c}_o, \bar{c}_c} \in \mathcal{P}_C \) iff there exists a \( q \) s.t. \( \text{prods}(o_p, q(\bar{t})), \text{cons}(o_c, q(\bar{u})) \) and \( o_p \prec_A o_c \).

MRD and MRR encode a POP \( P = \langle O, \Theta, \prec \rangle \) and a set of allowable orderings \( \prec_A \subseteq O^2 \) through the following sets of clauses. Formulæ 8–10 ensure that any solution represents an acyclic, transitively closed POP where all operators precede the goal and are preceded by the initial state. They derive from Formulæ 1–3 in MR, but have removed any propositions and constraints not required to preserve validity:

\[ \bigwedge_{p_{t=v}} p_{t=v} \quad \bigwedge_{p_{u=t}} p_{u=t} \quad \bigwedge_{p_{t=v}} p_{t=v} \quad \bigwedge_{p_{u=t}} p_{u=t} \]

Formulæ 11 and 12 ensures that the equality relation over variables and constants is symmetric and transitive, and Formulæ 13 ensures that each variable is bound to exactly one object. They are extensions to MD and MR that ensure the consistency of the POP’s variable bindings:

\[ \bigwedge_{p_{t=v}} p_{t=v} \quad \bigwedge_{p_{u=t}} p_{u=t} \]

Formulæ 14 and 15 encode POCL-validity. Formulæ 14 requires that each consumer be causally linked to at least one producer, and Formulæ 15 defines the ordering, binding and threat protection constraints required for a causal link to hold. These derive from Formulæ 4 and 5 in MD and MR, but have been optimised to remove redundant propositions, and generalised to allow the POP’s variable bindings to change:

\[ \bigwedge_{o_c \prec q(\bar{u}): \text{cons}(o_c, q(\bar{u}))} \bigwedge_{p_{o_p, \bar{c}_o, \bar{c}_c} \in \mathcal{P}_C} \bigwedge_{p_{o_p, \bar{c}_o, \bar{c}_c} \in \mathcal{P}_C} \]

\[ \bigwedge_{p_{o_p, \bar{c}_o, \bar{c}_c} \in \mathcal{P}_C} \bigwedge_{1 \leq i \leq |\bar{u}|} p_{t[i]=v[i]} \quad \bigwedge_{p_{o_p, \bar{c}_o, \bar{c}_c} \in \mathcal{P}_C} \]

1. Implemented code merges all \( p_{t=v} \) and \( p_{u=t} \) and updates Formulæ 11 and 12 accordingly.
Formula 16, an optimised version of Formula 6 in MD and MR, adds soft clauses that minimise the ordering constraints:

$$\bigwedge_{p_0 < o_2} \neg p_{o_1, o_2}.$$ \hfill (16)

The MRD and MRR encodings can now be defined. The allowed orderings for MRD are those from the input plan, forcing the resulting POP to be a deordering of the input:

**Definition 16.** MRD encodes a POP $P = (O, \theta, \prec)$ into a partial weighted MAXSAT instance through Formulae 8–16, with the allowable orderings $\prec_A = \prec$.

The MRR encoding only disallows orderings for the purposes of symmetry breaking:

**Definition 17.** If $P = (O, \theta, \prec)$ is a POP and $\prec'$ is an arbitrary linearisation of $P$, then MRR encodes $P$ into a partial-weighted MAXSAT instance through Formulae 8–16, with the allowable orderings $\prec_A = O^2 \setminus \{(o_2, o_1) : \text{name}(o_1) = \text{name}(o_2), o_1 \prec' o_2\}.

5 Experimental Evaluation

While a minimum reinstated de/reorder of a POP can never be less flexible than a minimum de/reorder (Definitions 9 and 11), the question remains whether, in practice, reinstatiation provides any significant increase in flexibility. The search for a minimum de/reorder often times out before an optimal (or indeed any) solution has been found [Muise et al., 2016], and allowing variable bindings creates a harder problem with an exponentially larger search space. This evaluation will thus address the question of whether reinstated de/reordering can provide more flexibility than standard de/reordering under the same resource constraints.

5.1 Experimental Setup

We answer the question above by comparing optimised plans produced by the Loandra MAXSAT solver [Berg et al., 2019] using the MD, MR, MRD and MRR encodings.

We also compare the MAXSAT-based techniques with the explanation-based order generalisation (EOG) deordering technique of Kambhampati and Kedar [1994]. First, a validation structure is constructed: a subset of $L_P$ (Definition 6) that links each consumer to its earliest producer:

**Definition 18.** If $P = (O, \theta, \prec)$ is a POP then $V_P$ is a validation structure of $P$ iff $V_P \subseteq L_P$ and there exists a total order $\prec \subseteq \prec'$ s.t. if $(o_p, q(i), o_c, q(i')) \in V_P$ and $(o_p', q(i'), o_c, q(i)) \in L_P$ then $o_p \prec' o_p'$.

This validation structure serves as a deordering heuristic:

**Definition 19.** If $P$ is a POP with validation structure $V_P$, then the explanation-based order generalisation of $P$ w.r.t. $V_P$ is the POP $P' = (O, \theta, \prec')$ where $o_1 \prec' o_2$ iff there exists a $q(i), q(i')$ s.t. either:

1. $(o_1, q(i), o_2, q(i')) \in V_P$,
2. there exists an $o_3$ and $q(i)\), s.t. $(o_2, q(i), o_3, q(i')) \in V_P$, $o_1 \prec o_2$ and $\text{thrtns}(o_3, q(i'))$.

Results are summarised in Table 1. Domains where no plan was improved by any technique (agricola, pegsol, snake, organic-synth-split, sokoban, visitall) have been excluded.
Table 1: For each domain, \( f_{EOG} \) is the mean flex produced by EOG, and for each encoder, \( C \) is the \% of plans for which a (satisficing or optimal) solution was found, \( T \) is the mean run time in minutes, and \( \Delta_{EOG} \) is the mean \% flex increase over EOG. All flex values are computed from plans for which every encoder found a solution, and all \( \Delta_{EOG} \) values are statistically significant with \( p < 0.01 \) as calculated from a single-tailed, paired t-test. Empty cells indicate no data, or no significant difference.

**Effectiveness of EOG**

Our results confirm those of Muise et al. [2016] that EOG always finds a minimum deorder of the input plan, despite a lack of optimality guarantees. Also, EOG takes < 3s in 88% of plans, with nearly 100% coverage. We thus exclude MD from the rest of the discussion.

As well as consistently finding minimum deorders, EOG found a minimum reorder in 69% of plans that were solved optimally by MR, a minimum reinstated deorder in 53% of the plans solved optimally by MRD, and a minimum reinstated reorder in 45% of the plans solved optimally by MRR. As EOG can neither reorder actions nor rebind variables, this means that in a significant number of plans, no reordering or reinstatement was necessary in order to find an optimally ordered POP.

**Benefit of Reinstatement**

To assess the practical benefit of reinstatement, we compare the coverage, run time and final flex of MRD and MRR with EOG and MR. To allow a meaningful comparison, mean flex values are computed from plans for which a solution was found by all encoders.

Overall, MRR provides a 20\% flex increase over EOG.\(^3\) This contrasts with the 3\% flex increase provided by MR, meaning that reinstatement has yielded a further flex increase of 17\%. This improvement is consistent: MRR times out before finding as flexible a solution as EOG in < 1\% of the 1260 plans solved by both, with similar results for MR.

MRR improves on EOG in 24 domains. The largest differences are in freecell and pipesworld, where MRR provides flex increases of 94\% and 96\%, resp., while MR provides 2\%, and mprime and transport, where MRR provides increases of 60\% and 46\%, resp., and MRR provides no significant benefit.

However, this additional flexibility comes at significant computational cost. MRR has a mean coverage and run time of 52\% and 22.43m, resp., with coverage dropping significantly in floorile (5\%), parking (5\%) and terms (0\%).

The less general MRD approach provides less additional flex at less computational cost. With a mean coverage and run time of 74\% and 13.52m, resp., it provides an overall flex increase of 13\% over EOG and statistically significant increases in 21 domains, pipesworld being the highest (74\%).

Interestingly, the decrease in coverage is not simply due to larger MAXSAT formulae. Table 2 shows that, as expected, MRD and MRR generate the largest encodings, and coverage always decreases with encoding size. However, within each size range, the encoders’ coverages differ significantly (e.g., MRD and MR always have more coverage than MRR), suggesting that the problem is the formulae’s structure, not size. A likely cause is so-called object symmetries: combinations of differently labelled but functionally equivalent domain objects (e.g., rovers R1 and R2 in Figure 1). Their presence can result in a small increase in formula size but an exponential number of equivalent (non-) solutions.

**Influence of Planner**

Finally, we note that the source of the input plan influences plan flexibility. Madagascar produces POPs, more specifically parallel plans where each step is a set of unordered actions. Interestingly, EOG can still increase their initial mean flex of 0.08 to 0.5 by removing unnecessary ordering constraints produced by the division of the plan into steps. MRR further improves on this by 15\%. Over the sequential (\( flex = 0 \)) plans generated by Dual-BFWS and LAMA, EOG gives a mean flex of 0.38 and 0.36, resp., which is improved upon by MRR by 30\% and 14\%, resp. Thus, while Dual-BFWS benefits most from reinstatement, optimised Madagascar plans are the most flexible.

\(^3\)All stated flex differences are statistically significant with \( p < 0.01 \) as calculated from a single-tailed, paired t-test.

Table 2: Plan count (\( n_P \)) and coverage (\( C \)) by encoding size (in number of clauses) for MR, MRD and MRR. Note that the encoding size is a log scale, each size range being twice that of the previous.

For each domain, \( f_{EOG} \) is the mean flex produced by EOG, and for each encoder, \( C \) is the \% of plans for which a (satisficing or optimal) solution was found, \( T \) is the mean run time in minutes, and \( \Delta_{EOG} \) is the mean \% flex increase over EOG. All flex values are computed from plans for which every encoder found a solution, and all \( \Delta_{EOG} \) values are statistically significant with \( p < 0.01 \) as calculated from a single-tailed, paired t-test. Empty cells indicate no data, or no significant difference.

**Effectiveness of EOG**

Our results confirm those of Muise et al. [2016] that EOG always finds a minimum deorder of the input plan, despite a lack of optimality guarantees. Also, EOG takes < 3s in 88\% of plans, with nearly 100\% coverage. We thus exclude MD from the rest of the discussion.

As well as consistently finding minimum deorders, EOG found a minimum reorder in 69\% of plans that were solved optimally by MR, a minimum reinstated deorder in 53\% of the plans solved optimally by MRD, and a minimum reinstated reorder in 45\% of the plans solved optimally by MRR. As EOG can neither reorder actions nor rebind variables, this means that in a significant number of plans, no reordering or reinstatement was necessary in order to find an optimally ordered POP.

**Benefit of Reinstatement**

To assess the practical benefit of reinstatement, we compare the coverage, run time and final flex of MRD and MRR with EOG and MR. To allow a meaningful comparison, mean flex values are computed from plans for which a solution was found by all encoders.

Overall, MRR provides a 20\% flex increase over EOG.\(^3\) This contrasts with the 3\% flex increase provided by MR, meaning that reinstatement has yielded a further flex increase of 17\%. This improvement is consistent: MRR times out before finding as flexible a solution as EOG in < 1\% of the 1260 plans solved by both, with similar results for MR.

MRR improves on EOG in 24 domains. The largest differences are in freecell and pipesworld, where MRR provides flex increases of 94\% and 96\%, resp., while MR provides 2\%, and mprime and transport, where MRR provides increases of 60\% and 46\%, resp., and MRR provides no significant benefit.

However, this additional flexibility comes at significant computational cost. MRR has a mean coverage and run time of 52\% and 22.43m, resp., with coverage dropping significantly in floorile (5\%), parking (5\%) and terms (0\%).

The less general MRD approach provides less additional flex at less computational cost. With a mean coverage and run time of 74\% and 13.52m, resp., it provides an overall flex increase of 13\% over EOG and statistically significant increases in 21 domains, pipesworld being the highest (74\%).

Interestingly, the decrease in coverage is not simply due to larger MAXSAT formulae. Table 2 shows that, as expected, MRD and MRR generate the largest encodings, and coverage always decreases with encoding size. However, within each size range, the encoders’ coverages differ significantly (e.g., MRD and MR always have more coverage than MRR), suggesting that the problem is the formulae’s structure, not size. A likely cause is so-called object symmetries: combinations of differently labelled but functionally equivalent domain objects (e.g., rovers R1 and R2 in Figure 1). Their presence can result in a small increase in formula size but an exponential number of equivalent (non-) solutions.

**Influence of Planner**

Finally, we note that the source of the input plan influences plan flexibility. Madagascar produces POPs, more specifically parallel plans where each step is a set of unordered actions. Interestingly, EOG can still increase their initial mean flex of 0.08 to 0.5 by removing unnecessary ordering constraints produced by the division of the plan into steps. MRR further improves on this by 15\%. Over the sequential (\( flex = 0 \)) plans generated by Dual-BFWS and LAMA, EOG gives a mean flex of 0.38 and 0.36, resp., which is improved upon by MRR by 30\% and 14\%, resp. Thus, while Dual-BFWS benefits most from reinstatement, optimised Madagascar plans are the most flexible.

\(^3\)All stated flex differences are statistically significant with \( p < 0.01 \) as calculated from a single-tailed, paired t-test.
6 Discussion

This paper presented a practical technique for plan optimisation through the modification of both ordering and variable binding constraints. Results show that in 52% of cases, MRR provides a flex increase of 20% over the EOG baseline in 22m, with results varying by domain. However, as reflected by the low coverage of MRR, this comes with considerable computational cost. Additionally, in a significant number of cases, the additional computation simply reveals that the original bindings were already optimal.

Whether MRR is preferable to less costly methods such as EOG depends on the application. If offline preprocessing time is available and execution-time flexibility is paramount, or the application domain is one where reinstatement can quickly and consistently improve plan flexibility (e.g., mystery), then reinstatement is clearly worthwhile.

This work generalises the POP optimality definitions of Bäckström [1998] and optimisation techniques of Muise et al. [2016] to allow changes in variable bindings. Other POP optimisation techniques have been studied. Muise et al. present extensions to MD and MR that also minimise the POP’s size by removing any actions without dependent consumers (a problem shown to be NP-complete by Olz and Bercher [2019]), and Bercher and Olz [2020] study the theoretical aspects of modifying a POP’s orderings in order to minimise its makespan. Siddiqui and Haslum [2012]’s block decomposition partitions a POP’s actions into macro-operator blocks. The implicit requirement that blocks not be interleaved means that this approach can sometimes reorder plans that cannot be reordered by standard methods. Say et al. [2016] present MILP models for optimising POP size and flexibility. They avoid the cubic size of Formula 2 by using variables to represent action start and end times, resulting in significant efficiency gains over MD and MR.

Unlike the work presented here, the above techniques do not allow variable bindings to change. Thus, further work could investigate whether generalising these techniques by allowing reinstatement results in more flexible block decompositions or further reductions in plan size or makespan, and whether a MILP encoding of MRD and MRR can provide an improvement in execution time.

More general optimisation approaches relax a plan into a partial plan, which need not completely specify orderings or bindings. While a POP’s linearisations are different orderings of the same set of actions, a partial plan defines a set of classical plans that differ in both their ordering and variable bindings. Kambhampati and Kedar [1994] generalise EOG, and keep only the ordering and binding constraints needed to enforce POCL validity. Waters et al. [2018] show that partial plan constraints are intractable, limiting their execution-time use, and so present an algorithm that searches for partial plans that can be instantiated in polynomial time.

In Section 5, we suggest that object symmetries prevent effective reinstatement. As there is much work on symmetry in planning Joslin and Roy, 1997; Riddle et al., 2016; Fox and Long, 1999; Rintanen, 2003; Pochter et al., 2011, symmetry breaking in the context of plan optimisation is an interesting area for future work.

References


