Plan Relaxation via Action Debinding and Deordering

Max Waters
RMIT University
Melbourne, Australia

Bernhard Nebel
University of Freiburg
Freiburg, Germany

Lin Padgham
RMIT University
Melbourne, Australia

Sebastian Sardina
RMIT University
Melbourne, Australia

Abstract

While seminal work has studied the problem of relaxing the ordering of a plan's actions, less attention has been given to the problem of relaxing and modifying a plan's variable bindings. This paper studies the problem of relaxing a plan into a *partial plan* which specifies which operators must be executed, but need not completely specify their order or variable bindings. While partial plans can provide an agent with additional flexibility and robustness at execution time, many operations over partial plans are intractable. This paper tackles this problem by proposing and empirically evaluating a fixed-parameter tractable algorithm which searches for tractable, flexible partial plans.

1 Introduction

While seminal work has studied the problems of *deordering* and *reordering* plans (Bäckström 1998), the equally important problems of *deinstantiating* and *reinstantiating* plans, i.e., the relaxation of their variable bindings, has received less attention. This paper studies the problem of generalising a totally-ordered, ground plan by relaxing both the ordering of its actions and the bindings of those actions' parameters.

A *partial plan* is a generalised plan comprising a set of actions to be executed, and a set of constraints defining the allowable action orderings and object values for the actions' parameters. The benefit of partial plans is that they provide flexibility and robustness at execution time, two desirable features under dynamic or partially-observable environments. Robustness follows from the fact that partial plans represent sets of different *valid*, totally-ordered, ground plans, so goal achievability is guaranteed when one such concrete plan is executed to completion. Flexibility arises from a *least-commitment* execution strategy, under which specific orderings of actions or variable bindings are decided only when absolutely necessary.

This work investigates the problem of finding *relaxations* of (partial) plans by "lifting" both their ordering and variable binding constraints. The objective is to maintain the partial plan's validity while increasing its flexibility, i.e., to ensure that any legal orderings and variable bindings produce a ground plan which achieves the intended goal, while allowing the relaxed partial plan to represent as much of the solution space as possible.

Interestingly (and surprisingly), finding a *minimum relaxation* of a partial plan, i.e., one with constraints that are as relaxed as possible while keeping the plan valid, is a polynomial time operation. However, this apparently encouraging result is undermined by the fact that extracting a ground, totally-ordered plan which respects the plan's constraints is computationally infeasible, clearly a serious drawback at execution time. Because of this, the challenge is to maximise a partial plan's flexibility while keeping the cost of instantiating it within the bounds of what is suitable for real-time decision making.

Thus, this paper proposes and empirically evaluates an algorithm of bounded complexity that searches for relaxed partial plans. The algorithm first transforms the input plan into a *constraint formula*, i.e., a formula of first-order logic with models which represent possible orderings and variable bindings. Then, a policy-based search process iteratively relaxes this formula while maintaining plan validity and keeping its "complexity" (as measured by the treewidth of its primal graph) below an input parameter.

Results show that in all test cases, the algorithm can produce a partial plan of quadratic "complexity" which represents all plans found by the standard PRF deordering algorithm (Bäckström 1998). While certain planning domains resist any kind of relaxation, in a majority of cases the algorithm also finds significant numbers of additional plans with different variable bindings, and in some cases finds reorderings not found by PRF, or rebindings where PRF was unable to find any reorderings.

2 A Motivating Example

Consider a modified version of the well-known *Barman* domain[1], in which a barman uses shot glasses and shakers to prepare cocktails out of various ingredients. The objects comprise two hands (LH and RH), one shot glass (SHOT), one shaker (SHKR), and one cocktail (CT) that is created by mixing two ingredients (ING_1 and ING_2). Initially, the shot glass and shaker are both clean, empty and on the table, and both hands are empty. The goal is for the shot glass to con-

[1] Full description at http://www.plg.inf.uc3m.es/ipc2011-deterministic/DomainsSequential.html

tain the cocktail. Shot glasses must be clean before use, and hands must be empty to pick up glasses or shakers.

Plan P_1 in Figure 1a is an optimal solution to this planning problem. The barman first picks up the shot glass with its right hand. Then, in steps 2–4, ING_1 is poured into the glass, and then from the glass into the shaker, and finally the glass is cleaned. The same is done with ingredient ING_2 over steps 5–7. The barman then puts down the shot glass, picks up the shaker, and shakes it to create the cocktail. The cocktail is then poured back into the glass, achieving the goal.

A common technique for providing flexibility at execution time is to *reorder* the plan. For example, relative to the domain, it does not matter whether ING_1 or ING_2 is poured first. Therefore if, at execution time, ING_1 is not available, the barman could pour ING_2 first in the hope that ING_1 will soon become available again. Unfortunately, P_1 does not allow for such a reordering of actions.

When reordering a plan, it is essential the plan remains valid, i.e., that it is legally executable w.r.t. actions' preconditions and achieves the goal. As indicated by the action name, step 6 requires that the shaker not be empty, a condition which is only produced by step 3. Therefore step 3 *must* be executed before step 6, and as a result ING_1 must be poured first. Indeed, the actions in P_1 are so tightly causally linked that P_1 *cannot be reordered at all*.

However, the desired reordering of ingredients can be achieved by modifying instead the parameters of some of the actions, i.e., by *reinstantiating* the plan. As with reordering, reinstantiations must preserve plan validity. In this example, modifying the ingredient order while maintaining the plan's validity is trivial: swap occurrences of ING_1 and ING_2 in steps 2–7, as done in plan P_2 in Figure 1b.

As well as reinstantiating P_1, plan P_2 *now admits a new reordering*. In plan P_1, the action $drop(\mathsf{RH}, \mathsf{SHOT})$ in step 8 empties hand RH, which is a precondition for the action $grasp(\mathsf{RH}, \mathsf{SHKR})$ in step 9: step 8 must precede step 9. However, in plan P_2, steps 8 and 9 can be executed in any order, or in parallel, so long as they are both executed between steps 7 and 10. This shows that reinstantiations can yield new reorderings.

The remainder will be as follows. Section 3 defines some logical concepts, Section 4 introduces *partial plans*, and Section 5 defines their desirable characteristics. Section 6 describes an algorithm that searches for tractable relaxed plans, and Section 7 evaluates it experimentally.

3 Preliminaries

The logical structures in this paper are 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. The notation $\vec{t}$ is used to denote ordered lists of possibly non-unique terms, with $\vec{t}[i]$ indicating the i-th element of the list. The notation $\vec{t} = \vec{s}$ is used as shorthand for $|\vec{t}| = |\vec{s}| \wedge \vec{t}[1] = \vec{s}[1] \wedge \cdots \wedge \vec{t}[|\vec{t}|] = \vec{s}[|\vec{t}|]$. For any structure η expressed in $\mathcal{L}$, the notation $\mathrm{vars}(\eta)$ denotes the variables appearing in η.

The concept of a *substitution* – a mapping from variables to terms – will be used throughout. The notation $\theta = \{x_1 \backslash t_1, \ldots, x_n \backslash t_n\}$ describes a substitution mapping each variable x_i to term t_i, for $1 \leq i \leq n$, and every other variable to itself. The set of variables explicitly mapped by substitution θ is denoted $\mathrm{domain}(\theta)$. The notation is used $\theta(x)$ to denote the term corresponding to variable x under substitution θ, and $\theta(\vec{x})$ is its generalization to list of variables. The result of applying a substitution θ to a formula ϕ is written $\phi\theta$, and means to simultaneously replace every variable x in ϕ with $\theta(x)$. A substitution θ is *ground* if every variable in its domain is mapped to a ground term, and is *complete* w.r.t. formula ϕ if $\mathrm{vars}(\phi) \subseteq \mathrm{domain}(\theta)$.

Planning formalism Planning tasks will be expressed in a standard first-order STRIPS formalism. A *planning task* is a tuple $\Pi = \langle D, A, s_i, s_g \rangle$, where D is a set of constants, A is a set of operators, and s_i and s_g are sets of ground literals describing the initial state and goal, respectively. An *operator* α is a tuple $\langle \mathrm{vars}(\alpha), \mathrm{pre}(\alpha), \mathrm{post}(\alpha) \rangle$, where $\mathrm{vars}(\alpha)$ is a list of variables and $\mathrm{pre}(\alpha)$ and $\mathrm{post}(\alpha)$ are finite sets of (ground or non-ground) literals with variables taken from $\mathrm{vars}(\alpha)$. When referring to an operator, the notation $\alpha(x_1, \ldots, x_n)$ is used, where $\mathrm{vars}(\alpha) = \langle x_1, \ldots, x_n \rangle$. An *action* a is a ground operator $\langle \mathrm{pre}(a), \mathrm{post}(a) \rangle$, where $\mathrm{pre}(a)$ and $\mathrm{post}(a)$ are finite sets of ground literals. If α is an operator and substitution θ is ground and complete w.r.t. $\mathrm{vars}(\alpha)$, then $\langle \mathrm{pre}(\alpha)\theta, \mathrm{post}(\alpha)\theta \rangle$ is the action resulting from *instantiating* α with θ.

A *classical plan* $\vec{a}$ is a finite sequence of actions. With the above definitions of actions and operators in mind, and assuming that no variable appears in more than one operator, a plan can be represented as $\vec{\alpha}\theta$, where $\vec{\alpha}$ is a list of operators and θ is a ground substitution that is complete w.r.t. every operator in $\vec{\alpha}$. Representing a plan in this way separates the abstract definition of the plan's operators from the variable bindings used to create a particular instantiation.

As plans are typically discussed with reference to some planning task, it will be assumed every classical plan is book-ended by the distinguished actions a_i, which has no parameters, no preconditions and postconditions that are the plan's initial state, and a_g, which has no parameters, no postconditions and preconditions that are the plan's goal. The advantage of "embedding" a planning task into a classical plan is that the plan is valid *iff* it is executable.

The producer-consumer-threat formalism It will be helpful to describe the causal links between a partial plan's operators in the context of the *producer-consumer-threat formalism* (PCT) (Bäckström 1998). Typically, the PCT approach describes dependencies between a plan's actions by identifying which actions produce or consume which propositions. However, the approach used here will describe how abstract, non-ground operators produce or consume (either ground or non-ground) literals. An operator α is said to *produce* the literal $q(\vec{t})$ *iff* $q(\vec{t}) \in \mathrm{post}(\alpha)$, and α *consumes* $q(\vec{t})$ *iff* $q(\vec{t}) \in \mathrm{pre}(\alpha)$. Similarly, α is a *threat* to literal $q(\vec{t})$ *iff* $\neg q(\vec{t}) \in \mathrm{post}(\alpha)$. These three conditions are written $\mathrm{cons}(\alpha, q(\vec{t}))$, $\mathrm{prod}(\alpha, q(\vec{t}))$, and $\mathrm{threat}(\alpha, q(\vec{t}))$, resp.

1. $grasp(\mathsf{RH}, \mathsf{SHOT})$
2. $fill(\mathsf{SHOT}, \mathsf{ING}_1, \mathsf{RH}, \mathsf{LH})$
3. $pourToCleanShaker(\mathsf{SHOT}, \mathsf{ING}_1, \mathsf{SHKR}, \mathsf{RH}, \mathsf{LH})$
4. $clean(\mathsf{SHOT}, \mathsf{ING}_1, \mathsf{RH}, \mathsf{LH})$
5. $fill(\mathsf{SHOT}, \mathsf{ING}_2, \mathsf{RH}, \mathsf{LH})$
6. $pourToUsedShaker(\mathsf{SHOT}, \mathsf{ING}_2, \mathsf{SHKR}, \mathsf{RH}, \mathsf{LH})$
7. $clean(\mathsf{SHOT}, \mathsf{ING}_2, \mathsf{RH}, \mathsf{LH})$
8. $drop(\mathsf{RH}, \mathsf{SHOT})$
9. $grasp(\mathsf{RH}, \mathsf{SHKR})$
10. $shake(\mathsf{CT}, \mathsf{ING}_1, \mathsf{ING}_2, \mathsf{SHOT}, \mathsf{SHKR}, \mathsf{RH}, \mathsf{LH})$
11. $pourShakerToShot(\mathsf{CT}, \mathsf{SHOT}, \mathsf{RH}, \mathsf{SHKR})$

(a) Plan P_1: ING_1 first, then ING_2.

1. $grasp(\mathsf{LH}, \mathsf{SHOT})$
2. $fill(\mathsf{SHOT}, \mathsf{ING}_2, \mathsf{LH}, \mathsf{RH})$
3. $pourToCleanShaker(\mathsf{SHOT}, \mathsf{ING}_2, \mathsf{SHKR}, \mathsf{LH}, \mathsf{RH})$
4. $clean(\mathsf{SHOT}, \mathsf{ING}_2, \mathsf{LH}, \mathsf{RH})$
5. $fill(\mathsf{SHOT}, \mathsf{ING}_1, \mathsf{LH}, \mathsf{RH})$
6. $pourToUsedShaker(\mathsf{SHOT}, \mathsf{ING}_1, \mathsf{SHKR}, \mathsf{LH}, \mathsf{RH})$
7. $clean(\mathsf{SHOT}, \mathsf{ING}_1, \mathsf{LH}, \mathsf{RH})$
8. $drop(\mathsf{LH}, \mathsf{SHOT})$
9. $grasp(\mathsf{RH}, \mathsf{SHKR})$
10. $shake(\mathsf{CT}, \mathsf{ING}_1, \mathsf{ING}_2, \mathsf{SHOT}, \mathsf{SHKR}, \mathsf{RH}, \mathsf{LH})$
11. $pourShakerToShot(\mathsf{CT}, \mathsf{SHOT}, \mathsf{RH}, \mathsf{SHKR})$

(b) Plan P_2: ING_2 first, then ING_1; hands are swapped.

Figure 1: An optimal plan (P_1) for the Barman problem and a reinstantiation of it (P_2) which admits a reordering.

4 Partial Plans

A partial plan specifies which operator types must be executed, but need not fully specify their order or even the bindings for their parameters. Instead, a partial plan provides a set of conditions that must be satisfied by the orderings and parameter bindings. Such conditions take the form of a *constraint formula*, an unquantified boolean formula in which each atom is either a codesignation between free variables or between a free variable and a constant, or an ordering constraint over two operators. Formally:

Definition 1. *Let $\mathcal{L} = \langle V, C, P\rangle$ be a first-order language with finitely many variable, constant, and predicate symbols V, C, and P, resp.; let O be a set of operators constructed in $\mathcal{L}$. The* **constraint language** *$\mathcal{L}_C$ is the language generated using the following context-free grammar, where $t_1, t_2 \in V \cup C$ and $\alpha_1, \alpha_2 \in O$:*

$$\phi ::= \top | \bot | t_1 = t_2 | \alpha_1 \preceq \alpha_2 | (\neg\phi) | (\phi \wedge \phi) | (\phi \vee \phi) | (\phi \rightarrow \phi).$$

As customary, the ordering relation $\preceq$ is transitive and reflexive, and $\alpha_1 \prec \alpha_2$ is shorthand for $(\alpha_1 \preceq \alpha_2 \wedge \alpha_2 \not\preceq \alpha_1)$. Co-designation constraints restrict the allowable bindings for variables. For variables x and y, $x = y$ requires x and y to be bound to the same constant, and for variable x and constant c, $x = c$ requires that x be bound to c.

A partial plan is a set of operators and a constraint formula. As with classical plans, the assumption is made that a partial plan has a planning task "embedded" within it through the operators α_i and α_g, which simulate the initial state and goal condition, resp. It is also assumed that no variable appears in more than one operator.[2]

Definition 2. *A* **partial plan** *is a tuple $P = \langle A, \phi\rangle$, where A is a finite set of operators and ϕ is a constraint formula such that:*

- *For all $\alpha_1, \alpha_2 \in A$ such that $\alpha_1 \neq \alpha_2$, $\mathrm{vars}(\alpha_1) \cap \mathrm{vars}(\alpha_2) = \varnothing$.*
- *There exist $\alpha_i, \alpha_g \in A$, where $\mathrm{vars}(\alpha_i) = \mathrm{pre}(\alpha_i) = \mathrm{vars}(\alpha_g) = \mathrm{post}(\alpha_g) = \varnothing$.*
- *$\mathrm{vars}(\phi) \subseteq \mathrm{vars}(A) \cup A$.*

For example, consider a small planning problem in the *barman* domain. The constants are two hands, one shot glass and one ingredient. In the initial state, both hands are empty and the shot glass is clean and on the table. The goal is for the shot to contain the ingredient. A solution to this problem is the partial plan $P = \langle A, \phi_o \wedge \phi_d \wedge \phi_c\rangle$, such that:

$$A = \{\alpha_i(\mathsf{LH}, \mathsf{RH}, \mathsf{SHOT}, \mathsf{ING}), grasp(h_1, s_1) \\ fill(s_2, i_1, h_2, h_3), \alpha_g(\mathsf{SHOT}, \mathsf{ING})\}.$$

$$\phi_o \stackrel{\text{def}}{=} \alpha_i \prec grasp \prec fill \prec \alpha_g.$$

$$\phi_d \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i \leq 3} h_i = \mathsf{LH} \vee h_i = \mathsf{RH}) \wedge s_1 = \mathsf{SHOT} \wedge i_1 = \mathsf{ING}.$$

$$\phi_c \stackrel{\text{def}}{=} h_1 = h_2 \wedge h_2 \neq h_3 \wedge s_1 = s_2.$$

Here, constraint formula ϕ_o specifies a total order over the operators, ϕ_d defines which constants can be bound to which variables, and ϕ_c defines the causal links, e.g., $h_1 = h_2$ ensures that the same hand is used to hold the shot glass in *fill* as was used to pick it up in *grasp*.

Ground instantiations of partial plans A partial plan compactly represents the set of classical plans that satisfy its constraints, i.e., the partial plan's *ground instantiations*.

Definition 3. *Let $P = \langle A, \phi\rangle$ be a partial plan, $\vec{\alpha}\theta$ be a classical plan of length n, and $C(\vec{\alpha}\theta)$ be the constraint formula defined as follows:*

$$C(\vec{\alpha}\theta) \stackrel{\text{def}}{=} \big(\bigwedge_{x \in \mathrm{vars}(\vec{\alpha})} x = \theta(x)\big) \wedge \big(\bigwedge_{1 \leq i < j \leq n} \vec{\alpha}[i] \prec \vec{\alpha}[j]\big).^3$$

Then, $\vec{\alpha}\theta$ is a **ground instantiation** *of P* iff *$C(\vec{\alpha}\theta) \models \phi$ and $A = \{\alpha : \alpha \in \vec{\alpha}\}$. Plan P is* **satisfiable** iff *there exists some classical plan that is a ground instantiation of P.*

Theorem 1. PARTIAL PLAN SATISFIABILITY. *Determining the satisfiability of a partial plan is NP-complete.*

Proof sketch. To prove membership in NP, let $P = \langle A, \phi\rangle$ be a partial plan and guess a classical plan $\vec{\alpha}\theta$. Construct (in polynomial time) the conjunction of positive constraint literals $C(\vec{\alpha}\theta)$, and then the formula ρ by replacing every literal which appears in ϕ with *true* if it appears in $C(\vec{\alpha}\theta)$, or *false*

[2] Instances of a 0-arity operator must be distinguished with subscripts.

[3] The expression $x = \theta(x)$ indicates a substitution where the left and right sides are the values before and after substitution, resp.

if not. $\vec{\alpha}\theta$ is a ground instantiation of P *iff* ρ evaluates to *true* and $A = \{\alpha \in \vec{\alpha}\}$. NP-hardness is proved by reduction from SAT. From input propositional formula φ, construct a partial plan $P = \langle A, \psi \rangle$ such that ψ is satisfiable *iff* φ is, and no operators in A have any preconditions or postconditions. P is valid *iff* φ is satisfiable. □

Validity of partial plans The concept of plan validity can be generalised to cover partial plans. For a partial plan to be valid, all of its ground instantiations must be valid (i.e., executable), and to prevent an unsatisfiable plan from being trivially valid, it must admit at least one ground instantiation:

Definition 4. *Let $P = \langle A, \phi \rangle$ be a partial plan. Then:*

1. *P is* **sound** iff *every ground instantiation of P is executable.*
2. *P is* **valid** iff *it is satisfiable and sound.*

Theorem 2. PARTIAL PLAN SOUNDNESS. *Determining the soundness of a partial plan is co-NP-complete.*

Theorem 3. PARTIAL PLAN VALIDITY. *Determining the validity of a partial plan is DP-complete.*

Proof sketch. As DP is the set of decision problems that are the conjunction of an NP and a co-NP problem, it follows trivially from Definition 4 and Theorems 1 and 2. □

5 Flexible and Tractable Partial Plans

This section will discuss two criteria to measure the degree to which a given partial plan provides flexibility and robustness. Each ground instantiation of a partial plan represents a different sequence of actions for realizing the goal. Therefore, the first criterion is simply how many different concrete "options" a partial plan provides, that is, the number of ground instantiations of the plan in question. Obviously, no matter how many ground instantiations a partial plan admits, it cannot provide much flexibility if the executing agent cannot determine in a reasonable amount of time which actions are compatible with it. So, the second criterion measures a partial plan's tractability, i.e., the complexity of the problem of finding a ground instantiation of the plan. Theorem 1 shows that finding an instantiation of a partial plan is, in general, intractable. However, this section shows that islands of tractability do exist within the space of partial plans, and that partial plans can be classified by the cost of instantiating them.

Minimally constrained partial plans

The relative "constrained-ness" of two partial plans can be measured by comparing the relative strengths of their constraint formulae. If $P = \langle A, \phi \rangle$ and $Q = \langle A, \psi \rangle$ are two valid partial plans, then Q is a *relaxation* of P *iff* every model of ϕ is also a model of ψ, and Q is a *minimum relaxation* of P *iff* it has the weakest possible constraints while remaining valid. Formally:

Definition 5. *Let $P = \langle A, \phi \rangle$ be a valid partial plan and $Q = \langle A, \psi \rangle$ be a partial plan. Then:*

1. *Q is a* **relaxation** *of P* iff *Q is valid and $\phi \models \psi$.*
2. *Q is a* **proper relaxation** *of P* iff *it is a relaxation of P and $\psi \not\models \phi$.*
3. *Q is a* **minimum relaxation** *of P* iff *it is a relaxation of P and there are no proper relaxations of Q.*

The minimum relaxation of any valid partial plan can be directly defined with reference to the Modal Truth Criterion (MTC) (Chapman 1987). The MTC determines the validity of a classical plan by requiring that it be *necessarily* true that the preconditions of all actions in the plan hold at the point when that action is executed. A precondition will necessarily hold if there is some previous action with an effect that produces the required condition, and no intermediate action with an effect that undoes it. This can be generalised to cover partial plans: a partial plan meets the MTC *iff* all of its ground instantiations meet the MTC. This can be expressed as a constraint formula as follows:

Definition 6. *The* **modal truth criterion** *for a partial plan $P = \langle A, \phi \rangle$ requires that $\phi \models \mathrm{MTC}(A)$, where:*

$$\mathrm{MTC}(A) \stackrel{\text{def}}{=} \bigwedge_{\{\alpha_c, q(\vec{t}):\mathrm{cons}(\alpha_c, q(\vec{t})), \alpha_c \in A\}} \mathrm{MTC}_c(A, \alpha_c, q(\vec{t})), \text{ where}$$

$$\mathrm{MTC}_c(A, \alpha_c, q(\vec{t})) \stackrel{\text{def}}{=} \bigvee_{\{\alpha_p, \vec{u}:\mathrm{prod}(\alpha_p, q(\vec{u})), \alpha_p \in A\}} \Big[\vec{t} = \vec{u} \wedge \alpha_p \prec \alpha_c \wedge \bigwedge_{\{\alpha_t, \vec{v}:\mathrm{threat}(\alpha_t, q(\vec{v})), \alpha_t \in A\}} (\vec{t} \neq \vec{v} \vee \alpha_t \prec \alpha_p \vee \alpha_c \preceq \alpha_t)\Big].$$

The definition MTC_c applies the MTC to a single consumer, and requires that if $\mathrm{cons}(\alpha_c, q(\vec{t}))$, then the list of symbols (variables or constants) $\vec{t}$ must be codesignated with some $\vec{u}$ such that $\mathrm{prod}(\alpha_p, q(\vec{u}))$ and $\alpha_p \prec \alpha_c$. Furthermore, it requires that any operator α_t such that $\mathrm{threat}(\alpha_t, q(\vec{v}))$ applies not be ordered between α_p and α_c, or that $\vec{v}$ not be codesignated with $\vec{t}$. The definition MTC applies this to every consumer appearing in the plan. The following two results follow from the above definition:

Theorem 4. MTC SOUNDNESS. *A partial plan $P = \langle A, \phi \rangle$ is sound* iff $\phi \models \mathrm{MTC}(A)$.

Theorem 5. MINIMUM RELAXATION. *A minimum relaxation of a valid partial plan can be found in polynomial time.*

If $P = \langle A, \phi \rangle$ is a valid partial plan, then the partial plan $Q = \langle A, \mathrm{MTC}(A) \rangle$ is a minimum relaxation of P, and can be constructed in polynomial time.

Comparison with plan ordering The complexity results and definitions above are quite different to those found in the partial plan de/reordering literature (Bäckström 1998). This difference stems from how partial plans and partial order plans represent constraints – typically, partial order plans define a partial order over a set of ground actions, while a partial plan's constraints are represented with the more expressive constraint language $\mathcal{L}_C$ (Definition 1), a fragment of first-order logic.

Representing ordering constraints as a binary relation results a distinction between *minimal* and *minimum orderings*. A partial plan is a ordering of another if it contains the same

actions, but its ordering relation is a subset of the other's. A ordering is minimal if its ordering relation cannot be reduced while remaining valid, and is a minimum if it has the smallest ordering relation of all possible orderings. However, the expressiveness of $\mathcal{L}_C$ means that there is no equivalent minimal/minimum distinction for partial plans. It follows from Theorems 4 and 5 that for any two partial plans P and Q, the minimum relaxation of P is also the minimum relaxation of Q, i.e., any "minimal" relaxation of a partial plan would also be a minimum relaxation. Additionally, while the problem of finding a minimum ordering of a partial order plan is intractable, the expressivity of $\mathcal{L}_C$ renders the equivalent problem in the context of partial plans trivial – $\mathcal{L}_C$ is expressive enough to directly define a minimum relaxation of a partial plan (Theorem 5).

This expressivity comes with a computational cost. Instantiating or validating a partial order plan takes polynomial time, but are intractable problems for partial plans. Therefore, the remainder of this section will focus on defining islands of tractability within the space of partial plans.

Parameterised complexity of partial plans

The problem of finding tractable but structurally restricted classes of partial plans can be examined in the context of *parameterised complexity*. A parameterised problem associates a parameter with each input instance. This allows for more fine-grained analysis of the problem's complexity, and aids the design of algorithms that are efficient when the parameter is small, even if the size of the input is large. The parameterised hierarchy comprises the set of complexity classes $\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq \text{XP}$. A problem is *fixed-parameter tractable*, or in complexity class FPT, *iff* it can be solved in time $O(f(k) \times n^{O(1)})$, where n is the size of the input and f is a function depending solely on parameter k. When f is exponential, then exponential time is required. However, if k is *fixed*, then the time required scales polynomially with n, meaning that problems in FPT remain feasible so long as k stays small.

Complexity class XP contains all parameterised problems that can be solved in a running time of $O(n^{f(k)})$. As with FPT, if k is fixed, then XP problems scale polynomially with the size of the input. However, as k is in the exponent, large problems may become infeasible.

Interestingly, it has been observed that many computationally hard graph problems are tractable when parameterised with the *treewidth* of the graph. A graph's treewidth (Robertson and Seymour 1986) is a positive integer that, intuitively, measures its "cyclicity," e.g., a tree has a treewidth of 1 and a complete graph of n vertices has a treewidth of $n - 1$. Consequently, a common approach in parameterised complexity analysis is to demonstrate that an otherwise intractable problem is solvable in polynomial time when limited to instances with an underlying structure that can be described as a graph with bounded treewidth.

The underlying structure of a partial plan is its so-called *primal graph*, i.e., an undirected graph where the vertices – the variables and operators appearing in the plan – are joined by an edge *iff* they appear together in some clause of the partial plan's constraint formula. The treewidth of a partial plan is simply the treewidth of its primal graph.

Definition 7. *Let $P = \langle A, \phi \rangle$ be a partial plan where ϕ is of the form $\phi_1 \wedge \cdots \wedge \phi_n$, where each ϕ_i is a disjunction of constraint fomulae. Then:*

1. *The* **primal graph** *of P is a graph $\langle V, E \rangle$ where $V = A \cup \text{vars}(A)$ and $(v_1, v_2) \in E$* iff *there exists $1 \leq i \leq n$ such that both v_1 and v_2 appear in ϕ_i.*
2. *The* **treewidth** *of P, denoted $\text{tw}(P)$, is equal to the treewidth of the primal graph of P.*

The following result states that a partial plan with bounded treewidth can be instantiated in polynomial time:

Theorem 6. PARAMETERISED PARTIAL PLAN SATISFIABILITY. *Determining the satisfiability of a partial plan P is in* XP *and is $W[1]$-hard when parameterised with $\text{tw}(P)$.*

Proof sketch. Let $P = \langle A, \phi_1 \wedge \cdots \wedge \phi_n \rangle$ such that $\text{tw}(P) = k$. As $\text{tw}(P) = k$, no conjunct ϕ_i can contain more than $k+1$ variables. As all assignments of constants to variables that satisfy each clause ϕ_i can be enumerated in time $|P|^{k+1}$, P can converted into an equivalent CSP S of treewidth k in time $|P|^{k+1}$, and then solved in time $|S|^k$ (Freuder 1990). Therefore satisfiability of P can be determined in time $O(|P|^{f(k)})$ and is in XP. Proof of $W[1]$-hardness is by fpt-reduction from K-CLIQUE. From the graph $G = \langle V, E \rangle$ construct, in time $k^2|E|$, a partial plan $P = \langle A, \phi \rangle$, such that no operators in A contain any preconditions or postconditions, and ϕ is defined as follows:

$$\phi = \bigwedge_{i,j \in 1 \ldots k, i \neq j} x_i \neq x_j \wedge (\bigvee_{\langle v,u \rangle \in E} x_i = v \wedge x_j = u).$$

P has a treewidth of $k-1$ and is satisfiable *iff* G has a clique of k vertices. □

While computing a graph's treewidth is intractable, determining if it is bounded by some constant k is in FPT with respect to k (Bodlaender 1993). As the size of the primal graph of a partial plan $P = \langle A, \phi \rangle$ is bounded by $(|\text{vars}(A)| + |A|)^2$, the following holds:

Observation 1. *Determining whether $\text{tw}(P) \leq k$ for some partial plan P and $k > 0$ is in* FPT.

Tractable partial plans

The relative tractability of two partial plans can be determined by comparing their treewidths. Given two partial plans P and Q such that Q is a relaxation of P, plan Q is a *minimal k-treewidth relaxation* of P if it cannot be relaxed any further without its treewidth exceeding k. In turn, Q is a *minimum k-treewidth relaxation* of P if, amongst all possible relaxations of P, Q admits the most models while keeping its treewidth bounded by k. Formally:

Definition 8. *Let P and Q be valid partial plans, let integer $k > 0$ and let $\#P$ denote the number of ground instantiations of partial plan P. Then:*

1. *Q is a* **k-treewidth relaxation** *of P* iff *Q is a relaxation of P and $\text{tw}(Q) \leq k$.*

2. *Q is a* **proper k-treewidth relaxation** *of P* iff *Q is a proper relaxation of P and* $\mathrm{tw}(Q) \leq k$.
3. *Q is a* **minimal k-treewidth relaxation** *of P* iff *it is a k-treewidth relaxation of P and there is no proper k-treewidth relaxation of Q.*
4. *Q is a* **minimum k-treewidth relaxation** *of P* iff *it is a k-treewidth relaxation of P and there is no R such that R is a k-treewidth relaxation of P and $\#P < \#R$.*

The exact complexity of finding minimum and minimal k-treewidth relaxations remains open. However, the decision problem corresponding to the optimisation problem of finding a minimal k-treewidth relaxation is intractable:

Theorem 7. MINIMAL K-TREEWIDTH RELAXATION. *For any partial plan P and integer k such that* $\mathrm{tw}(P) \leq k$*, deciding the existence of a proper k-treewidth relaxation of P is NP-hard and in Σ_2^P.*

6 Restricted Cases

Since finding a tractable relaxation of a plan is an intractable problem, this section will investigate a specialised class of partial plans that express constraints as a *causal structure*. A causal structure defines which producers can be used to bring about the conditions required by each consumer in a partial plan, and turns out to be a convenient representation for manipulating sets of constraints. While causal structures are less expressive than constraint formulae, finding a minimal k-treewidth relaxation is a tractable task.

As discussed above, the Modal Truth Criterion (MTC, Definition 6) requires that when a partial plan is instantiated, each consumer in the plan, i.e., each precondition $q(\vec{t})$ of each operator α_c, be *causally linked* with some producer, i.e., a postcondition $q(\vec{s})$ of some operator α_p. This implies that α_p must precede α_c, $\vec{s}$ must be codesignated with $\vec{t}$, and any threats to this link, i.e., postconditions $\neg q(\vec{u})$ of some operator α_t, cannot be both codesignated with the consumer and ordered between the producer and consumer. Generally speaking, a causal structure strengthens the MTC by requiring that a consumer be causally linked to one of a specified set of potential producers.

Definition 9. *A* **producer-consumer option** *(PC option) is a tuple $\langle \alpha_p, q(\vec{s}), \alpha_c, q(\vec{t}) \rangle$ such that* $\mathrm{prod}(\alpha_p, q(\vec{s}))$, $\mathrm{cons}(\alpha_c, q(\vec{t}))$ *and $\alpha_p \neq \alpha_c$. A* **causal structure** *C is a set of PC options.*

For example, consider the operators α_i and $grasp(h, s)$, where $\mathrm{post}(\alpha_i) = \{empty(\mathsf{LH}), empty(\mathsf{RH}), onTable(\mathsf{SHOT}_1), onTable(\mathsf{SHOT}_2)\}$, $\mathrm{pre}(grasp(h, s)) = \{empty(h), onTable(s)\}$, and C is a simple causal structure defined below:

$$C = \{\langle \alpha_i, empty(\mathsf{LH}), grasp(h, s), empty(h) \rangle, \\ \langle \alpha_i, empty(\mathsf{RH}), grasp(h, s), empty(h) \rangle, \\ \langle \alpha_i, onTable(\mathsf{SHOT}_1), grasp(h, s), onTable(s) \rangle\}.$$

This causal structure requires that h be bound to either LH or RH, that s be bound to SHOT_1 and that $\alpha_i \prec grasp$. Expanding this structure can relax the constraints, e.g., adding $\langle \alpha_i, onTable(\mathsf{SHOT}_2), grasp(h, s), onTable(s) \rangle$ would allow s to be bound to SHOT_2.

A useful special case of causal structures is the one containing all PC options that are implicit in an operator set:

Definition 10. *The* **minimally constrained causal structure** *for operator set A is the set C_A such that for any pc option $L = \langle \alpha_p, q(\vec{s}), \alpha_c, q(\vec{t}) \rangle$, $L \in C_A$* iff *$\alpha_p, \alpha_c \in A$.*

Producer-Consumer Plans

A *producer-consumer plan* (PC plan) is a partial plan that represents constraints as a causal structure, with the condition that every consumer has at least one producer option:

Definition 11. *A* **producer-consumer plan** *is a tuple $P = \langle A, C \rangle$, where A is a finite set of operators and C is a causal structure such that:*

- *For all $\alpha_1, \alpha_2 \in A$ such that $\alpha_1 \neq \alpha_2$,* $\mathrm{vars}(\alpha_1) \cap \mathrm{vars}(\alpha_2) = \varnothing$.
- *There exist $\alpha_i, \alpha_g \in A$, where* $\mathrm{vars}(\alpha_i) = \mathrm{pre}(\alpha_i) = \mathrm{vars}(\alpha_g) = \mathrm{post}(\alpha_g) = \varnothing$.
- $C \subseteq C_A$.
- *If $cons(\alpha_c, q(\vec{t}))$ and $\alpha_c \in A$, then there exists an α_p and $q(\vec{s})$ such that $\langle \alpha_p, q(\vec{s}), \alpha_c, q(\vec{t}) \rangle \in C$.*

A PC plan $P = \langle A, C \rangle$ can be converted into a standard partial plan by translating its causal structure into a constraint formula. The translation, denoted $\Phi(A, C)$, encodes the MTC as in Definition 6, but with the additional requirement that a consumer be causally linked with a producer with which it is connected by a PC option in C:

Definition 12. *For any PC plan $P = \langle A, C \rangle$, $\Phi(A, C)$ encodes A and C into a constraint formula:*

$$\Phi(A, C) \stackrel{\text{def}}{=} \bigwedge_{\{\alpha_c, q(\vec{t}) : \mathrm{cons}(\alpha_c, q(\vec{t})), \alpha_c \in A\}} \Phi_c(A, C, \alpha_c, q(\vec{t})), \text{ where}$$

$$\Phi_c(A, C, \alpha_c, q(\vec{t})) \stackrel{\text{def}}{=} \bigvee_{\{\alpha_p, q(\vec{s}) : \langle \alpha_p, q(\vec{s}), \alpha_c, q(\vec{t}) \rangle \in C\}} \Big[\vec{t} = \vec{s} \wedge \alpha_p \prec \alpha_c \wedge \\ \bigwedge_{\{\alpha_t, q(\vec{u}) : \mathrm{threat}(\alpha_t, q(\vec{u})), \alpha_t \in A\}} (\vec{t} \neq \vec{u} \vee \alpha_t \prec \alpha_p \vee \alpha_c \preceq \alpha_t) \Big].$$

The above encoding can be used to apply any property of partial plans to PC plans. If $P = \langle A, C \rangle$ is a PC plan and $Q = \langle A, \Phi(A, C) \rangle$ is a partial plan, then P is satisfiable, sound or valid *iff* Q is, respectively, and $\mathrm{tw}(P) = \mathrm{tw}(Q)$.

A key property of Definition 12 is that for any partial plan $P = \langle A, C \rangle$, the formula $\Phi(A, C)$ is at least as strong as the modal truth criterion for A, i.e., $\Phi(A, C) \models \mathrm{MTC}(A)$. As a partial plan is sound *iff* its constraint formula entails the MTC (Observation 4), and is valid *iff* it is satisifiable and sound, the following observation follows:

Observation 2. *A PC plan is valid* iff *it is satisfiable.*

It follows from Definition 12 that if $P = \langle A, C \rangle$ and $Q = \langle A, C' \rangle$ are two PC plans such that $C \subseteq C'$, then $\Phi(A, C) \models \Phi(A, C')$. From Observation 2 it follows that if P is valid, $\Phi(A, C)$ is satisfiable, and thus so is $\Phi(A, C')$. Therefore, *any valid PC plan can be relaxed by expanding its causal structure while remaining valid*:

Algorithm 1 Minimal k-Treewidth Relaxation (MKTR)

Input: Valid PC plan $P = \langle A, C\rangle$, integer k.
Result: A minimal k-treewidth relaxation of P.

```
if tw(P) > k then
    return ⊥
end if
Construct C_A.
while ∃L ∈ C_A \ C s.t. tw(⟨A, C∪{L}⟩) ≤ k do
    Add L to C
end while
return P
```

Observation 3. *If $P = \langle A, C\rangle$ and $Q = \langle A, C'\rangle$ are PC plans such that $C \subseteq C'$, then if P is valid so is Q.*

From Definition 12, P's primal graph is a subgraph of Q's primal graph, meaning that P's treewidth is bounded by that of Q (Bodlaender 1998):

Observation 4. *If $P = \langle A, C\rangle$ and $Q = \langle A, C'\rangle$ are two PC plans such that $C \subseteq C'$, then* $\mathrm{tw}(P) \leq \mathrm{tw}(Q)$.

Minimal k-treewidth PC relaxation

The MKTR algorithm (Algorithm 1) finds a minimal k-treewidth PC plan in fixed-parameter tractable time.

Definition 13. *Let $P = \langle A, \phi\rangle$ and $Q = \langle A, \psi\rangle$ be two valid PC plans. Q is a* **minimal k-treewidth PC relaxation** *of P* iff *it is a k-treewidth relaxation of P and there is no PC plan R such that R is a proper k-treewidth relaxation of Q.*

Theorem 8. *Finding a minimal k-treewidth PC relaxation of a PC plan is* FPT *for parameter k.*

Proof sketch. Consider MKTR. From Observations 1 – 4, MKTR computes a k-treewidth PC relaxation of its input. As the loop is executed a polynomial number of times, independent of k, and computing $\mathrm{tw}(P) < k$ is in FPT w.r.t. k, it follows that MKTR is also in FPT w.r.t k. □

The MKTR algorithm can be applied to a valid classical plan $\vec{\alpha}\theta$ by first converting it into the PC plan $P = \langle A, C\rangle$ where $A = \{\alpha : \alpha \in \vec{\alpha}\}$ and $\langle \alpha_p, q(\vec{s}), \alpha_c, q(\vec{t})\rangle \in C$ *iff* $q(\vec{s})$ is the *last* producer in $\vec{\alpha}$ to be causally linked to $q(\vec{t})$.

7 Implementation and Evaluation

MKTR has been implemented[4] and compared with the PRF deordering algorithm (Bäckström 1998) over 80 problems from six IPC domains. Initial tests indicated that the encoding in Definition 12 produces constraint formulae with excessively high treewidths. Thus, MKTR has been optimised by *(i)* using encodings that produce constraint formulae with lower treewidths at the expense of fewer models, and *(ii)* simplifying the resulting formula with a specialised AC-3 algorithm. Neither modification affects Theorem 8.

[4] bitbucket.org/max_waters/mktr

Relaxation policies Because the success of MKTR will be influenced by which PC option is selected at each step, two selection policies have been tested.

Keeping a partial plan's treewidth as low as possible allows more PC options to be added. It can be seen in Definition 12 that the more threats there are to a PC option, the more variables will appear in the clause encoding it. Therefore, the *Minimise Threats* policy (MT) selects the PC option with the fewest threats, i.e., the link $\langle \alpha_p, q(\vec{s}), \alpha_c, q(\vec{t})\rangle \in C \setminus C_A$ with the fewest producers α_t and $q(\vec{u})$, such that $\mathrm{threat}(\alpha_t, q(\vec{u}))$.

Because relaxing a producer's variable bindings can in turn relax those of its consumers, the *Relax Producers* policy (RP) relaxes the bindings of producers with the most consumers. The possible bindings for operator α can be increased by adding more producer options for either α's preconditions, or the preconditions of other operators that are threatened by α's postconditions. The policy uses two measures. If α is an operator and $\vec{a}$ is the input plan, then $n_c(\alpha, \vec{a})$ denotes the number of operators with preconditions that are causally linked to a postcondition of α. And $n_t(\alpha, \vec{a})$ is equal to the highest $n_c(\alpha_t, \vec{a})$ for any α_t that threatens any causal link to any precondition of α. RP selects the PC option $\langle \alpha_p, q(\vec{s}), \alpha_c, q(\vec{t})\rangle$ that maximises $\max(n_c(\alpha_c, \vec{a}), n_t(\alpha_c, \vec{a}))$.

Test set-up All problem instances were taken from either the deterministic track of IPC 2011 or the satisficing or optmising tracks of IPC 2014. For each instance, a satisficing solution was found using the LAMA planner (Richter and Westphal 2010). Ten plans were selected from each domain, with lengths ranging (where possible) from 25 to 150.

Some instances were tested in both their original and a "relaxed" form. In the *barman* and *parking* domains, the planner must make careful use of limited resources, i.e., hands, glasses, and shakers in *barman*, and parking spaces in *parking*. To test if resource constraints influence the effectiveness of MKTR, relaxed versions of instances from these domains were created. For example, *barman* p435-1-2 is the standard instance with two hands, and p435-1-4 is a relaxed version with four. Similarly, *parking* p-12-7 has 12 cars and seven spaces, and p-12-9 is a relaxed version with nine.

Each satisficing plan was relaxed using the above two policies, maximum treewidths of 2 and 5, and a time limit of one hour. It was also deordered using PRF, and the resulting partial order plan was translated into an equivalent partial plan. The ground instantiations of the partial plans produced by MKTR and PRF were counted using the gecode (Gecode Team 2018) constraint solver with a time limit of one hour. In some cases an hour was insufficient, and a lower bound was recorded. However, comparing the constraint formulae of two partial plans can reveal a set-wise comparison of their instantiations. For example, if $P = \langle A, \phi\rangle$ and $Q = \langle A, \psi\rangle$ are partial plans, then P's instantiations are a strict superset of Q's *iff* $\psi \models \phi$ and $\phi \not\models \psi$. The results of PRF and MKTR were compared in this way with gecode (although some cases again timed out).

Domain: barman						
Problem	$\|\vec{a}\|$	PRF	MT, tw = 2	RP, tw = 2	MT, tw = 5	RP, tw = 5
p435-1-2	60	1296	2592	5184	1296	3888
p435-1-4	50	$\geq 1.84\times10^7$	$\geq 3.35\times10^7$	$\geq 3.16\times10^7$	$\geq 1.62\times10^7$	$\geq 1.5\times10^7$
p536-1-2	76	7776	7776	23328	7776	62208
p536-1-4	57	$\geq 1.64\times10^7$	$\geq 1.02\times10^7$	$\geq 2.8\times10^7$	$\geq 1.2\times10^7$	$\geq 1.17\times10^7$
p637-1-2	89	18432	18432	36864	18432	36864
p637-1-4	64	$\geq 1.41\times10^7$	$\geq 2.63\times10^7$	$\geq 2.42\times10^7$	$\geq 1.15\times10^7$	$\geq 1.26\times10^7$
p638-1-2	99	124416	124416	124416	124416	435456
p638-1-4	63	1.4×10^7	$\geq 2.05\times10^7$	$\geq 2.14\times10^7$	$\geq 1.19\times10^7$	$\geq 1.13\times10^7$
p839-1-2	126	$\geq 9.45\times10^6$	$\geq 2.91\times10^7$	$\geq 2.46\times10^7$	$\geq 9.24\times10^6$	$\geq 9.02\times10^6$
p839-1-4	85	$\geq 8.73\times10^6$	$\geq 2.11\times10^7$	$\geq 2.07\times10^7$	$\geq 8.53\times10^6$	$\geq 1.12\times10^7$
p435-2-2	55	648	1296	1296	3456	1296
p435-2-4	42	41580	2×10^6	2×10^6	7.98×10^6	887040
p536-2-2	80	55296	55296	110592	172800	221184
p536-2-4	52	$\geq 1.69\times10^7$	$\geq 1.07\times10^7$	$\geq 1.66\times10^7$	$\geq 1.36\times10^7$	$\geq 1.48\times10^7$
p637-2-2	89	18432	18432	36864	18432	36864
p637-2-4	68	$\geq 1.43\times10^7$	$\geq 1.41\times10^7$	$\geq 1.41\times10^7$	$\geq 9.66\times10^6$	$\geq 1.19\times10^7$
p638-2-2	103	$\geq 1.07\times10^7$	$\geq 1.09\times10^7$	$\geq 1.03\times10^7$	$\geq 1.09\times10^7$	$\geq 9.62\times10^6$
p638-2-4	62	$\geq 1.53\times10^7$	$\geq 1.34\times10^7$	$\geq 1.22\times10^7$	$\geq 1.24\times10^7$	$\geq 1.26\times10^7$
p839-2-2	123	442368	442368	884736	442368	884736
p839-2-4	83	$\geq 1.23\times10^7$	$\geq 1.17\times10^7$	$\geq 1.18\times10^7$	$\geq 1.08\times10^7$	$\geq 1.09\times10^7$

Domain: parking						
Problem	$\|\vec{a}\|$	PRF	MT, tw = 2	RP, tw = 2	MT, tw = 5	RP, tw = 5
p-12-07	40	1	1	1	1	1
p-12-09	21	41	82	82	82	82
p-14-08	32	1	1	1	1	1
p-14-10	28	721	1442	1442	1442	1442
p-16-09	37	1	1	1	1	1
p-16-11	27	4160	8320	8320	8320	8320
p-18-10	58	1	1	1	1	1
p-18-12	35	91800	91800	91800	183600	183600
p-20-11	71	1	1	1	1	1
p-20-13	41	5600	11200	11200	11200	11200
p-28-15	54	1	1	1	1	1
p-28-17	48	5460	5460	10920	5460	10920
p-30-16	64	1	1	1	1	1
p-30-18	48	1820	1820	3640	1820	3640
p-32-17	60	1	1	1	1	1
p-32-19	51	63112	63112	63112	63112	63112
p-34-18	62	1	1	1	1	1
p-34-20	63	$\geq 1.11\times10^7$	$\geq 1.21\times10^7$	$\geq 1.21\times10^7$	$\geq 1.17\times10^7$	$\geq 1.13\times10^7$
p-40-21	106	1	1	1	1	1
p-40-23	61	3654	3654	7308	3654	7308

Domain: child-snack						
Problem	$\|\vec{a}\|$	PRF	MT, tw = 2	RP, tw = 2	MT, tw = 5	RP, tw = 5
p01	32	1×10^6	$\geq 1.69\times10^7$	$\geq 1.58\times10^7$	$\geq 1.38\times10^7$	$\geq 1.93\times10^7$
p04	53	$\geq 1.35\times10^7$	$\geq 1.1\times10^7$	$\geq 1.27\times10^7$	$\geq 9.16\times10^6$	$\geq 1.2\times10^7$
p06	54	$\geq 9.15\times10^6$	$\geq 7.37\times10^6$	$\geq 1.23\times10^7$	$\geq 2.22\times10^6$	$\geq 1.08\times10^7$
p07	59	$\geq 6.43\times10^6$	$\geq 5.15\times10^6$	$\geq 9.98\times10^6$	$\geq 6.56\times10^6$	$\geq 1.12\times10^7$
p08	67	$\geq 9.68\times10^6$	$\geq 9.93\times10^6$	$\geq 1.03\times10^7$	–	$\geq 9.56\times10^6$
p09	72	$\geq 6.64\times10^6$	$\geq 7.04\times10^6$	$\geq 9.53\times10^6$	–	$\geq 7.93\times10^6$
p09-2	77	$\geq 8.92\times10^6$	$\geq 6.96\times10^6$	$\geq 7.74\times10^6$	–	$\geq 3.48\times10^6$
p10	88	$\geq 8.47\times10^6$	$\geq 8.09\times10^6$	$\geq 7.79\times10^6$	$\geq 5.55\times10^6$	$\geq 8.56\times10^6$
p12	85	$\geq 4.12\times10^6$	–	$\geq 5.72\times10^6$	–	$\geq 6.96\times10^6$
p19-2	129	–	–	–	–	$\geq 4.99\times10^6$

Domain: floor-tile						
Problem	$\|\vec{a}\|$	PRF	MT, tw = 2	RP, tw = 2	MT, tw = 5	RP, tw = 5
p01-4-3-2	41	5.39×10^6	1.08×10^7	1.08×10^7	1.08×10^7	1.08×10^7
p01-4-4-2	50	2268	2268	2268	2268	2268
p02-4-4-2	57	$\geq 1.34\times10^7$	$\geq 1.62\times10^7$	$\geq 1.67\times10^7$	$\geq 1.56\times10^7$	$\geq 1.75\times10^7$
p01-5-3-2	46	12096	12096	12096	12096	12096
p02-5-3-2	48	15120	15120	15120	15120	15120
p02-5-4-2	69	$\geq 1.37\times10^7$	$\geq 1.29\times10^7$	$\geq 1.14\times10^7$	$\geq 1.3\times10^7$	$\geq 1.39\times10^7$
p01-5-5-2	92	–	$\geq 4.88\times10^6$	–	$\geq 1.03\times10^7$	$\geq 9.77\times10^6$
p02-5-5-2	92	$\geq 2.39\times10^6$	$\geq 9.25\times10^6$	–	$\geq 1.1\times10^7$	$\geq 3.12\times10^6$
p01-6-5-2	112	–	–	$\geq 2.06\times10^6$	–	–
p02-6-5-2	122	–	–	–	$\geq 1.66\times10^6$	–

Domain: grid-visit-all						
Problem	$\|\vec{a}\|$	PRF	MT, tw = 2	RP, tw = 2	MT, tw = 5	RP, tw = 5
p-1-3	12	1	16	3	68	6
p-1-4	22	1	140	2	4	6
p-1-5	38	1	4140	3	–	9
p-1-6	88	1	2	4	2	40
p-1-7	54	1	3	3	3	3
p-1-8	78	1	1	1	1	1
p-05-5	50	1	$\geq 1.41\times10^7$	2640	$\geq 1.34\times10^7$	2640
p-05-6	66	1	210	4	210	15120
p-05-7	57	1	17280	4	17280	60
p-05-8	154	1	3	3	3	3

Domain: scanalyzer						
Problem	$\|\vec{a}\|$	PRF	MT, tw = 2	RP, tw = 2	MT, tw = 5	RP, tw = 5
p10	40	13	13	13	13	13
p11	64	1	1	1	1	1
p12	34	1	1	1	1	1
p13	28	1	1	1	1	1
p14	46	1	1	1	1	1
p15	74	1	1	1	1	1
p16	52	1	1	1	1	1
p17	84	1	1	1	1	1
p18	21	4	576	4	576	4
p19	61	5	7	5	7	5

Table 1: Relaxation results for PRF and MKTR configured with the MinimiseThreats (MT) and Relax Producers (RP) policies, and treewidths of 2 and 5. The input plan length is shown by $|\vec{a}|$, and results indicate the number of instantiations of the final partial plan as counted by gecode in one hour, with "–" indicating that none were found within the time limit.

Results Table 1 shows the results of the empirical evaluations. Set-wise comparisons of the outputs of MKTR and PRF reveal that regardless of the input plan, treewidth or policy, the instantiations found by MKTR always include those found by PRF. Depending on the domain, MKTR can improve on the flexibility provided by PRF by finding additional instantiations with different variable bindings. In 69.6% of *barman* test cases, MKTR found a strict superset of PRF's results. Supersets were also found in *child-snack* (93.1%), *floor-tile* (28.9%), *grid-visit-all* (80.0%) and the relaxed *parking* problems (57.5%). Interestingly, in six such cases (e.g., *barman* p536-1-4 with MT and tw = 2) MKTR *found reorderings not found by* PRF. This demonstrates that, as described in Section 2, changing variable bindings can yield more reorderings. When exact instantiation counts are available (Table 1), they show that when MKTR improves on PRF, it typically finds twice as many instantiations, and in some cases can find significantly more. For example, in *barman* p435-2-4, MKTR finds 48 times as many instantiations as PRF, and in *grid-visit-all* MKTR *finds rebindings of plans which* PRF *cannot deorder at all*.

MKTR did not improve on PRF in any *scanalyzer* or standard *parking* problems. However, both algorithms struggle on these instances, suggesting that they are too tightly constrained to allow many relaxations. Indeed, comparing results from the original and relaxed *barman* and *parking* problems confirms that both MKTR and PRF perform better when resources are less constrained.

Set-wise comparisons of the MKTR results under different treewidths demonstrate that allowing MKTR to search for high-treewidth, structurally complex constraints can yield partial plans with more instantiations. The benefit

gained is dependent on the domain. In 44.7% of the *barman* test cases, MKTR with $\mathrm{tw} = 5$ found a strict superset of the instantiations found when $\mathrm{tw} = 2$. Although in some cases, the higher treewidth sends MKTR into a dead-end, e.g., *barman* p-435-1-2. Similar results occur in the *child-snack* (40.0%) and *grid-visit-all* (42.1%) domains, although increasing the treewidth has little effect on the other, more tightly constrainted domains such as *scanalyzer*.

Interestingly, the flexibility benefit of increasing the treewidth comes at little computational cost. When $tw = 2$, 80.6% of the final partial plans can be instantiated by gecode in under 200ms, and with $\mathrm{tw} = 5$ this reduces just slightly, to 77.5% (although in both cases some outliers timed out after an hour).

The choice of relaxation policy is also significant. RP is more effective in *barman* and *parking*, and MT is more effective in *grid-visit-all*, *child-snack* and, when relaxations are feasable, *scanalyzer*.

8 Discussion

Much of the literature on plan flexibility is on plan de/reordering with the aim to find *minimum orderings*. It thus differs from this work in two ways. Firstly, this work is concerned with modifying both *variable bindings* and *action orderings*. Secondly, while the problems of finding minimum de/reorderings are intractable, the equivalent problem for partial plans – finding a *minimum relaxation* – is trivial (Theorem 5). Thus, this paper aimed to find minimal but *tractable relaxations*. More concretely, it extended the de/reordering works to also cover the optimisation of variable bindings, and used parameterised complexity analysis to study the problem of finding tractable partial plans.

The seminal work on action ordering for plan flexibility is that of Bäckström (1998), which studies the complexity of deordering and reordering partial order plans. Recently, Aghighi and Bäckström (2017) studied the problem from a parameterised complexity perspective.

Muise, Beck, and McIlraith (2016) find optimal de/reorderings of a partial order plan by encoding it as a MAXSAT problem. The propositional formula and clause weights are constructed such that the optimal solution corresponds to, depending on the encoding, either a minimum de- or reordering of the input plan. Plan validity is preserved by encoding minimal causal requirements (the MTC for partial order plans) as "hard" clauses that must be present in any solution. This is the equivalent of the MTC-based soundness requirement for partial plans (Theorem 4), that is the basis for the process for encoding a PC plan into a partial plan (Definition 12). Interestingly, a variation in the MAXSAT encoding allows the number of actions appearing in the final partial order plan to be minimised. While the approach in this paper can in principle be extended to also reduce the number of actions, this is left for later work.

Siddiqui and Haslum (2012) propose the notion of *block deordering*. A block is partially ordered set of actions that cannot be interleaved with steps outside the block. Blocks behave like macro-operators, i.e., they produce and consume propositions, and are themselves partially ordered. Because producer/consumer dependencies between actions in the same block can be ignored when reordering blocks, block deordering is sometimes able to deorder plans that cannot be deordered by standard methods. Future work could generalise this to cover variable debinding, i.e., by considering a block as set of non-ground operators that produce and consume non-ground literals, and then using an algorithm (akin to Siddiqui and Haslum's RESOLVE) to relax the ordering and binding constraints between operators within the same block, and between the blocks themselves.

One work that does address the problem of relaxing both action orderings *and* variable bindings is that of Kambhampati and Kedar (1994). There, a polynomial time algorithm is presented, that uses an MTC-based explanation of the input plan's correctness as a guide for relaxing its ordering and variable bindings. Unlike MKTR, though, no guarantees are given for either the optimality of the resulting relaxation or the computational cost of instantiating it. Interesting further work could compare the two approaches, and examine the feasibility of Kambhampati and Kedar's algorithm even without those guarantees.

In many planning instances, flexibility can be achieved by modelling constants (e.g., hands) as a numeric resource (e.g., n_hands), and delaying its selection until runtime. This works, however, only when the resources are freely substitutable, such as fuel or energy. If, for instance, the resources are trucks in a logistic domain, then they are not freely substitutable as each truck may be at a different place and reasoning about their locations is required.

While it is assumed here that a partial plan's flexibilty is due to the number of classical plans that it represents, a more sophisticated approach could measure the *diversity* within those plans. Unfortunately, existing flexibility measures are unsuitable for sets of plans that differ in both their ordering and variable bindings. The so-called *flex* value (Nguyen and Kambhampati 2001) is commonly used to measure the flexibility of a partial order plan. However, as it is computed from the number of unordered actions, it cannot account for any additional flexibility provided by relaxed variable bindings.

Typically, the *diversity* of a set of plans is derived from the "distance" between each pair of plans in the set. Measures based on a comparison of the plans' ground actions, e.g., edit distance or *stability* (i.e., the proportion of shared actions (Fox et al. 2006; Coman and Muñoz-Avila 2011)), can obscure degrees of difference between variable bindings. For example, the three one-step plans $\langle grasp(\mathsf{LH}, \mathsf{SHOT}_1)\rangle$, $\langle grasp(\mathsf{LH}, \mathsf{SHOT}_2)\rangle$ and $\langle grasp(\mathsf{RH}, \mathsf{SHOT}_3)\rangle$ would be considered equally different from each other despite the common resource in the first two. Distances derived from a set difference of the plans' state space transitions (Srivastava et al. 2007) suffer the same problem. Comparisons of plans' causal structures (Srivastava et al. 2007) can distinguish between variable bindings, but not reorderings of plans with the same causal links. Thus, better diversity measures for partial plans, and MKTR relaxion policies that aim to maximise their diversity are worth investigating.

References

Aghighi, M., and Bäckström, C. 2017. Plan reordering and parallel execution - A parameterized complexity view. In

Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., 3540–3546.

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

Bodlaender, H. L. 1993. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proc. of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, 226–234.

Bodlaender, H. L. 1998. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science* 209(1-2):1–45.

Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32(3):333 – 377.

Coman, A., and Muñoz-Avila, H. 2011. Generating diverse plans using quantitative and qualitative plan distance metrics. In *Proc. of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: replanning versus plan repair. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Freuder, E. C. 1990. Complexity of k-tree structured constraint satisfaction problems. In *Proc. of the Eighth National Conference on Artificial Intelligence - Volume 1*, 4–9.

Gecode Team. 2018. Gecode: Generic constraint development environment. `http://www.gecode.org`.

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.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2016. Optimal partial-order plan relaxation via MaxSAT. *Journal of Artificial Intelligence Research* 57:113 – 149.

Nguyen, X., and Kambhampati, S. 2001. Reviving partial order planning. In *Proc. of the 17th International Joint Conference on Artificial Intelligence - Volume 1 (IJCAI)*, 459–464.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39(1):127–177.

Robertson, N., and Seymour, P. D. 1986. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms* 7(3):309–322.

Siddiqui, S. H., and Haslum, P. 2012. Block-structured plan deordering. In *Proc. of the 25th Australasian Joint Conference on Advances in Artificial Intelligence*, 803–814.

Srivastava, B.; Nguyen, T. A.; Gerevini, A.; Kambhampati, S.; Do, M. B.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2016–2022.