Red-Black Planning: A New Systematic Approach to Partial Delete Relaxation

Carmel Domshlak

Technion, Haifa, Israel

Jörg Hoffmann

Saarland University, Saarbrücken, Germany

Michael Katz

IBM Research, Haifa, Israel

Abstract

To date, delete relaxation underlies some of the most effective heuristics for deterministic planning. Despite its success, however, delete relaxation has significant pitfalls in many important classes of planning domains, and it has been a challenge from the outset to devise heuristics that take *some* deletes into account. We herein devise an elegant and simple method for doing just that. In the context of finite-domain state variables, we define *red* variables to take the relaxed semantics, in which they accumulate their values rather than switching between them, as opposed to *black* variables that take the regular semantics. Red-black planning then interpolates between relaxed planning and regular planning simply by allowing a subset of variables to be painted red. We investigate the tractability region of red-black planning, extending Chen and Giménez' characterization theorems for regular planning to the more general red-black setting. In particular, we identify significant islands of tractable red-black planning, use them to design practical heuristic functions, and experiment with a range of "painting strategies" for automatically choosing the red variables. Our experiments show that these new heuristic functions can improve significantly on the state of the art in satisficing

Email addresses: `dcarmel@ie.technion.ac.il` (Carmel Domshlak), `hoffmann@cs.uni-saarland.de` (Jörg Hoffmann), `katzm@il.ibm.com` (Michael Katz)

planning.[1]

Keywords: Planning, Heuristic Search, Delete Relaxation

1. Introduction

In deterministic, also know as "classical", planning for action selection, the world states are represented by complete assignments to a set of variables, the actions allow for deterministic modifications of these assignments, and the objective is to find a sequence of actions that modifies a given initial assignment to an assignment that satisfies a goal property. In the last two decades, solvers for classical planning have made spectacular advances in their empirical efficiency. A key role in this progress, especially in the context of satisficing planning where no optimality guarantee is required, is played by the *monotonic*, or "delete-free", *relaxation* [4, 5, 6].

At a high level, state variables in the monotonic relaxation accumulate their values, rather than switching between them. The key property of such a relaxation is that applying actions under value accumulating semantics does not reduce the applicability of actions in the future. As a result, while regular satisficing planning is PSPACE-complete even for simple formalisms, monotonic satisficing planning is polynomial-time [7]. Despite this, plans for the monotonic relaxation – commonly referred to as *relaxed plans* – often yield very useful heuristic functions [8, 9], and have been a key ingredient of state-of-the-art satisficing planners (e. g., [5, 6, 10, 11]) since more than a decade.

While some of the most effective heuristics to date are obtained in this manner, the delete relaxation has significant pitfalls. A striking example (see, e. g., [8, 12]) is the inability to account for "having to move back" on a road map: Under the relaxation, once we traversed the map once, we are in all locations simultaneously so there never is a need to move back. In effect, if, say, a truck needs to move across a line of road segments to pick up a cargo and then move back to deliver it, then the heuristic value remains constant (equal to the length of the line) until the truck reaches the cargo. In many domains that involve transportation or other

[1]Many of the results presented herein were previously published in three conference papers [1, 2, 3]. The present article discusses these results much more comprehensively, and makes several extensions, including a refinement to the theoretical analysis, detailed proofs, additional experiments, and implementing heuristic functions relying on acyclic black causal graphs instead of arcless ones.

types of movement, this leads to huge plateaus, i. e., regions of states with identical heuristic value. Another prominent issue (see, e. g., [13, 14, 15]) is "resource persistence", that is, inability to account for the consumption of non-replenishable resources. The monotonic relaxation furthermore ignores any detrimental side effects an action may have on other parts of the plan, trivializing domains with a puzzle nature. For example, monotonic relaxation of Rubic's Cube tasks loses the dependencies across the subcubes.

Given these weaknesses of monotonic relaxation, it has been an actively pursued research challenge from the outset to design heuristics that take *some* deletes into account. This resulted in a wealth of approaches, taking into account conflicts in the relaxed plan [16, 10, 17, 13, 18, 15], keeping track of (some) side effects [12, 19, 20], and incorporating TSP solvers responsible for the movements of particular variables in the relaxed plan [21, 22, 23]. It has proved daunting, however, to devise frameworks that fully *interpolate* between regular planning and monotonic planning, by providing a choice of *which*, and the ability to scale freely with *how many*, deletes are taken into account. The first such interpolation framework was put forward in 2012, enriching the monotonic relaxation with an explicitly represented set of fact conjunctions, forcing the heuristic to eventually become perfect as that set gets larger [24, 25, 26]. We herein propose a much simpler interpolation framework: *we relax only some of the state variables*.

In this framework, which we baptize *red-black planning*, some state variables, called *red*, take the relaxed semantics and accumulate their values, while all other variables, called *black*, keep the regular semantics and thus switch between their values.[2] Consider again our previous example where a truck needs to move across a line of road segments. Say we paint the truck-position variable black, and we paint the cargo-position variable red. A *red-black plan* then needs to include the backward truck moves, and the length of an optimal red-black plan is equal to that of an optimal real plan. The same applies to VisitAll when painting the robot position black. A heuristic function generated this way would be perfect.

The problematic word here of course is the "would". Apart from the quality of the heuristic, we also need to consider the computational effort required to compute it. As red-black planning generalizes monotonic planning – the spe-

[2]The aforementioned works of Fox and Long [21, 22] and Keyder and Geffner [23] can also be viewed as "un-relaxing" some of the state variables (those controlled by TSP solvers). However, this applies only to restricted subsets of variables, and the TSP treatment is weaker than ours in the sense that it considers only the *set* of variable values that must be visited, ignoring duplicates and ordering. We will get back to this in more detail later, once we formally introduced our framework.

cial case where all variables are painted red – and optimal monotonic planning is NP-hard [7], optimal red-black planning also is (at least) NP-hard. Therefore, in analogy to commonly used relaxed plan heuristics, we will generate an (inadmissible) heuristic function by generating some, not necessarily optimal, red-black plan. Still, the question is *whether there are significant tractable fragments of satisficing red-black planning*.

Fortunately, the answer is "yes". Analyzing the complexity of satisficing red-black planning, we in particular show tractability for planning tasks whose *black causal graph* – the projection of the standard causal graph [27, 28, 29] onto the black variables – is acyclic, and whose black variables satisfy a certain invertibility condition. Specifically, we show that any fully relaxed (aka monotonic) plan for a problem in this fragment can be "repaired" into a valid red-black plan with only a polynomial runtime overhead.

Investigating the corresponding heuristic function from a practical perspective, we find that its bias to follow decisions made in the "basis" monotonic plans can be harmful, leading to dramatic over-estimation even in very simple toy benchmarks. Targeting this pitfall, we devise an improved red-black planning algorithm that relies less on monotonic plans, getting rid of much of this over-estimation phenomenon. We fill in important realization details, pertaining in particular to planning with acyclic causal graphs and invertible variables (a sub-procedure of our heuristic function). We devise optimizations enhancing red-black plan applicability, short-cutting the search if the red-black plan is applicable in the original planning task.

To obtain an automatic planning methodology, we finally require *painting strategies* for automatically deciding which variables should adopt each color. We devise a family of such strategies, and analyze their performance. We finally run comparative experiments against the state of the art on the IPC benchmarks, showing that our new heuristic functions bring significant advantages over standard delete-relaxation heuristics, as well as over alternative partial delete-relaxation heuristics, in several domains and overall.

The rest of the paper is structured as follows. In Section 2 we provide the necessary background, and in Section 3 we formally introduce red-black planning as a framework for relaxation. We analyze the complexity of satisficing red-black planning (Section 4), discuss the practical aspects of generating heuristic functions in this context (Section 5), investigate painting strategies (Section 6), and run experiments against the state of the art (Section 7). We conclude with a brief summary and discussion of future work (Section 8). Some proofs are replaced by proof sketches in the main text. The full proofs are in Appendix A.

2. Background

A planning task in **finite-domain representation** (FDR) is given by a quadruple $\Pi = \langle V, A, I, G\rangle$, where:

- V is a set of *state variables*, with each $v \in V$ being associated with a finite domain $\mathcal{D}(v)$.
  - A partial variable assignment p is a function on a variable subset $\mathcal{V}(p) \subseteq V$ that assigns each $v \in \mathcal{V}(p)$ a value $p[v] \in \mathcal{D}(v)$.
  - For a partial assignment p and a variable subset $V' \subseteq \mathcal{V}(p)$, by $p[V']$ we denote the assignment provided by p to V'.
  - We identify partial assignments with sets of *facts* "variable v takes value d", denoted by $\langle v/d\rangle$.
  - Complete assignments to V are called *states*.
- I is an *initial state*. The *goal* G is a partial assignment to V.
- A is a finite set of *actions*. Each action a is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ of partial assignments to V called *precondition* and *effect*, respectively.

The semantics of FDR tasks is as follows. An action a is applicable in a state s iff $s[\mathcal{V}(\mathsf{pre}(a))] = \mathsf{pre}(a)$, i. e., iff $s[v] = \mathsf{pre}(a)[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$. Applying a in state s changes the value of every $v \in \mathcal{V}(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$; the resulting state is denoted by $s[\![a]\!]$. If a has a precondition on v but does not change it, we say that a is *prevailed* by v; the set of all such preconditions is denoted $\mathsf{prevail}(a)$.

By $s[\![\langle a_1, \dots, a_k\rangle]\!]$ we denote the state obtained from sequential application of the (respectively applicable) actions $a_1, \dots, a_k$ starting at state s. Such an action sequence is an *s-plan* if $s[\![\langle a_1, \dots, a_k\rangle]\!][\mathcal{V}(G)] = G$, and it is an *optimal* s-plan if its length is minimal among all s-plans. The computational task of **(optimal) planning** is finding an (optimal) I-plan.

Example 1 *Figure 1 illustrates an example that we use throughout the paper.*

The example is akin to the Grid benchmark, and is encoded in FDR *using the following state variables: R, the robot position in* $\{1, \dots, 7\}$*; A, the key A position in* $\{R, 1, \dots, 7\}$*; B, the key B position in* $\{R, 1, \dots, 7\}$*; F in* $\{0, 1\}$ *encoding whether the robot hand is free; O in* $\{0, 1\}$ *encoding whether the lock is open. The robot can move from i to $i + 1$, or vice versa, provided that either the*

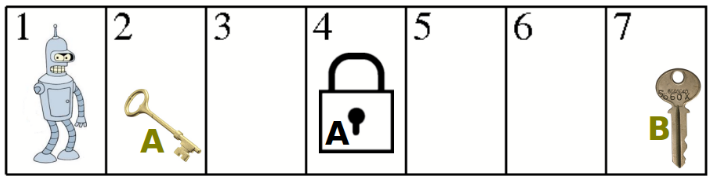


Figure 1: An illustrative example: "SimpleGrid".

lock is open or $\{i, i+1\} \cap \{4\} = \emptyset$.[3] *The robot can take a key if its hand is free, drop a key it is holding, or open the lock if the robot is at* 3 *or* 5 *and holds key* A*. The goal requires key* B *to be at* 1*. An optimal plan for the robot is to move to* 2*, take key* A*, move to* 3*, open the lock, move to* 7*, drop key* A *and take key* B*, move back to* 1*, and drop key* B*.*

A **monotonic finite-domain representation** (MFDR) planning task is given by a quadruple $\Pi = \langle V, A, I, G \rangle$ exactly as for FDR tasks, but the semantics is different.[4] Informally, the state variables in MFDR tasks accumulate their values, rather than switching between them. That is, if an FDR action switches the value of a variable v from x to y, then the monotonic version of that action "extends" the value of v from $\{x\}$ to $\{x, y\}$. More specifically,

- An MFDR state s is a function that assigns each $v \in V$ a non-empty *subset* $s[v] \subseteq \mathcal{D}(v)$ of its domain.
- An MFDR action a is applicable in state s iff $\mathsf{pre}(a)[v] \in s[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$.
- Applying an MFDR action a in state s changes the value of $v \in \mathcal{V}(\mathsf{eff}(a))$ from $s[v]$ to $s[v] \cup \{\mathsf{eff}(a)[v]\}$.

Respectively, an MFDR action sequence $\langle a_1, \dots, a_k \rangle$ applicable in state s is an s-plan if $G[v] \in s[\![\langle a_1, \dots, a_k \rangle]\!][v]$ for all $v \in \mathcal{V}(G)$. In all other respects, MFDR and FDR semantics are identical.

[3] Note that we make the non-standard (but sensible) requirement for the lock to be open also when moving *away* from it. This is used later on to comprehensively illustrate our algorithms.

[4] As was recently noted [30], it is not entirely clear to whom the original formulation of monotonic relaxation for multi-valued variable domains should be attributed, but it can be traced back at least to Malte Helmert's work on the Fast Downward planning system [29], to Jörg Hoffmann's planning lecture (winter 2004/05), and to work applying the monotonic relaxation in the context of timed automata [31].

While FDR planning is PSPACE-complete even for binary state variables, satisficing plan generation for MFDR tasks is polynomial time (this follows directly from Bylander's [7] results). Exploiting the latter for deriving heuristic estimates is a key ingredient of many competitive satisficing planning systems, via the notion of monotonic, or delete, relaxation. Given an FDR planning task $\Pi = \langle V, A, I, G\rangle$, the **monotonic relaxation** of Π is the MFDR task $\Pi^+ = \Pi$. The **optimal delete relaxation heuristic** $h^+(\Pi)$ is defined as the length of an optimal plan for Π^+. For arbitrary states s, $h^+(s)$ is defined via the MFDR task $\langle V, A, s, G\rangle$.

For a state s and applicable action sequence π in Π, we sometimes use $s[\![\pi^+]\!]$ to denote the outcome of executing π in the same state of Π^+. In general, if π^+ is a plan for Π^+, then π^+ is referred to as a **relaxed plan** for Π. For example, a relaxed plan for our SimpleGrid task in Example 1 is for the robot to move to 2, take key A, move to 3, open the lock, move over to position 7, take key B (without having to first drop key A), and then immediately drop key B at position 1 (without having to first move back there).

To characterize fragments of planning, we use two standard graphical structures induced by the description of FDR tasks.

- The **causal graph** CG_Π of Π is a digraph with vertices V. An arc (v, v') is in CG_Π iff $v \neq v'$ and there exists an action $a \in A$ such that $(v, v') \in \mathcal{V}(\mathsf{eff}(a)) \cup \mathcal{V}(\mathsf{pre}(a)) \times \mathcal{V}(\mathsf{eff}(a))$, that is, a affects v' while either affecting v or being preconditioned by the value of v.

  Special cases of interest will be those where the causal graph is **acyclic** (a **DAG**), or where the causal graph is **arcless**, i. e., does not contain any arcs at all.

- The **domain transition graph** $\mathit{DTG}_\Pi(v)$ of a variable $v \in V$ is an arc-labeled multi-digraph with vertices $\mathcal{D}(v)$. $\mathit{DTG}_\Pi(v)$ has an arc from d to d' induced by a iff $\mathsf{eff}(a)[v] = d'$, and either $\mathsf{pre}(a)[v] = d$ or $v \notin \mathcal{V}(\mathsf{pre}(a))$. We denote such arcs by (d, a, d'), and by (d, d') for convenience where the action referred to is clear from context, or where there is no need for referring to a particular action. Each arc is labeled with its **outside condition** $\mathsf{ocon}(d, a, d') = \mathsf{pre}(a)[V \setminus \{v\}]$ and its **outside effect** $\mathsf{oeff}(d, a, d') = \mathsf{eff}(a)[V \setminus \{v\}]$.

Example 2 *Figure 2 (a) illustrates the notion of causal graphs in the SimpleGrid task from Example 1. R is a prerequisite for changing every other variable. Each* *key is interdependent with F because taking/dropping them affects both. Key A influences O, and O influences R.*

$$\begin{tikzcd}
A \arrow[r, leftrightarrow] \arrow[d] & F \arrow[r, leftrightarrow] & B \\
O \arrow[r, leftrightarrow] & R \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
1 \arrow[d, bend left, "{\langle R/x\rangle, \langle y/x\rangle? \; \langle y/R\rangle!}"] \\
0 \arrow[u, bend left, "{\langle R/x\rangle, \langle y/R\rangle? \; \langle y/x\rangle!}"]
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
1 \arrow[r, leftrightarrow] & 2 \arrow[r, leftrightarrow] & 3 \arrow[r, leftrightarrow, "{\langle O/1\rangle?}"'] & 4 \arrow[r, leftrightarrow, "{\langle O/1\rangle?}"'] & 5 \arrow[r, leftrightarrow] & 6 \arrow[r, leftrightarrow] & 7
\end{tikzcd}$$

(c)

Figure 2: Causal graph (a), and domain transition graphs of variables F (b) and R (c) in SimpleGrid (Example 1). Outside conditions are displayed with a question mark "?", outside effects are displayed (in red and) with an exclamation mark "!". In (b), each of the two arcs in the picture corresponds to several arcs in the DTG, namely one for each combination of location $x \in \{1, \dots, 7\}$ and key $y \in \{A, B\}$.

Figures 2 (b) and 2 (c) illustrate the definition of domain transition graphs on the variables F and R in the SimpleGrid task. Note that the transitions of F in (b) are each induced by several actions—all drop(x, y) actions for the transition $0 \to 1$ and all take(x, y) actions for the transition $1 \to 0$), whereas each of the transitions of R in (c) is induced by a single move(i, j) action. The label $\langle O/1\rangle$ in DTG$_\Pi(R)$ is the same for the transitions $3 \to 4$ and $4 \to 3$ (as well as for the transitions $5 \to 4$ and $4 \to 5$) because moving away from the lock also requires it to be open.

Adopting the notation of Chen and Giménez [32], for a digraph G, let #vertices(G), cc-size(G), and scc-size(G) denote the number of vertices, the size of the largest connected component in (the undirected graph induced by) G, and the size of the largest strongly connected component in G, respectively. For a set of digraphs $\mathcal{G}$, we say that #vertices$(\mathcal{G})$ is *bounded* if there exists a constant k such that #vertices$(G) \leq k$ for all $G \in \mathcal{G}$. Bounded cc-size$(\mathcal{G})$ and bounded scc-size$(\mathcal{G})$ are defined similarly. PlanExist$(\mathcal{G})$ and PlanGen$(\mathcal{G})$ are the plan existence and plan generation problems restricted to FDR planning tasks whose causal graphs are elements of $\mathcal{G}$.

3. Red-Black Relaxation

In FDR respectively MFDR, *all* state variables adopt the value-switching respectively the value-accumulating semantics. Formulated that way, it is obvious that FDR and MFDR can be viewed as the two extreme ends of an entire spectrum of relaxations, choosing the semantics on a variable-by-variable basis. We baptize this approach *red-black planning*.

A **red-black finite-domain representation** (RB) planning task is given by a quintuple $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$, where V^{B} and V^{R} are sets of finite-domain state variables, called *black variables* and *red variables*, respectively, and A, I, and G are exactly as in FDR and MFDR tasks, specified with respect to the union of the black and red variables. The semantics of RB is as follows:

(i) A state s assigns each $v \in V^{\mathsf{B}} \cup V^{\mathsf{R}}$ a non-empty subset $s[v] \subseteq \mathcal{D}(v)$, where $|s[v]| = 1$ for all $v \in V^{\mathsf{B}}$.

(ii) An action a is applicable in state s iff $\mathsf{pre}(a)[v] \in s[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$.

(iii) Applying an action a in state s changes the value of each black variable $v \in \mathcal{V}^{\mathsf{B}}(\mathsf{eff}(a))$ to $\{\mathsf{eff}(a)[v]\}$, and changes the value of each red variable $v \in \mathcal{V}^{\mathsf{R}}(\mathsf{eff}(a))$ to $s[v] \cup \mathsf{eff}(a)[v]$.

(iv) An action sequence $\langle a_1, \dots, a_k\rangle$ applicable in s is an s-plan if $G[v] \in s[\![\langle a_1, \dots, a_k\rangle]\!][v]$ for all $v \in \mathcal{V}(G)$.

Example 3 *Consider again the SimpleGrid task from Example 1. Recall that the optimal relaxed plan for this task takes key A, opens the lock, and moves over to position 7. It then takes key B (without having to first drop key A), and drops key B at position 1 (without having to first move back there).*

Now, say we paint variables R, A, B, O red, but paint F black. The robot in the resulting RB *task needs to drop key A before taking key B. If R is also painted black, then the robot also needs to move back to position* 1 *before dropping key B.*

Obviously, if $\langle V, A, I, G\rangle$ is an FDR task, then we obtain an equivalent RB task as $\langle V, \emptyset, A, I, G\rangle$, and if $\langle V, A, I, G\rangle$ is an MFDR task, then we obtain an equivalent RB task as $\langle \emptyset, V, A, I, G\rangle$. In other words, RB generalizes both FDR and MFDR.

In what follows, we use RB tasks primarily as relaxations of FDR tasks. Given an FDR planning task $\Pi = \langle V, A, I, G\rangle$ and a subset $V^{\mathsf{R}} \subseteq V$ of its variables, the **red-black relaxation** of Π relative to V^{R} is the RB task $\Pi^{\mathsf{RB}}_{V^{\mathsf{R}}} = \langle V \backslash V^{\mathsf{R}}, V^{\mathsf{R}}, A, I, G\rangle$.

The **optimal red-black relaxation heuristic** $h^{\mathsf{RB}*}_{V^{\mathsf{R}}}(I)$ of Π relative to V^{R} is defined as the length of an optimal plan for $\Pi^{\mathsf{RB}}_{V^{\mathsf{R}}}$. For arbitrary states s, $h^{\mathsf{RB}*}_{V^{\mathsf{R}}}(s)$ is defined via the RB task $\langle V \setminus V^{\mathsf{R}}, V^{\mathsf{R}}, A, s, G\rangle$. When the set of red variables is clear from context, and when our discussion pertains to arbitrary such sets, we drop the subscript "V^{R}". If π^{RB} is a plan for Π^{RB}, then we refer to π^{RB} as a **red-black relaxed plan** for Π. It is easy to see that:

Proposition 1 *$h^{RB*}_{V^{\mathsf{R}}}$ is consistent and dominates h^+. If $V^{\mathsf{R}\prime} \subseteq V^{\mathsf{R}}$, then $h^{RB*}_{V^{\mathsf{R}\prime}}$ dominates $h^{RB*}_{V^{\mathsf{R}}}$. If $V^{\mathsf{R}} = \emptyset$, then $h^{RB*}_{V^{\mathsf{R}}}$ is perfect.*

Proof: Regarding the last part of the claim, if $V^{\mathsf{R}} = \emptyset$ then $\Pi^{\mathsf{RB}}_{V^{\mathsf{R}}}$ is obviously equivalent to Π and in particular $h^{\mathsf{RB}*}_{V^{\mathsf{R}}}$ is perfect. Domination of h^+ holds because, as follows directly from definition, if π^{RB} is a red-black relaxed plan for any state s, then π^{RB} also is a relaxed plan for s, i. e., is a valid plan in Π^+. The same argument shows that, if $V^{\mathsf{R}\prime} \subseteq V^{\mathsf{R}}$, then $h^{\mathsf{RB}*}_{V^{\mathsf{R}\prime}}$ dominates $h^{\mathsf{RB}*}_{V^{\mathsf{R}}}$.

Regarding consistency, we need to show that, for any state s and applicable action a in Π, denoting $s' := s[\![a]\!]$ we have that $h^{\mathsf{RB}*}_{V^{\mathsf{R}}}(s) \leq 1 + h^{\mathsf{RB}*}_{V^{\mathsf{R}}}(s')$. Let $\pi^{\mathsf{RB}}(s')$ be an optimal s'-plan in $\Pi^{\mathsf{RB}}_{V^{\mathsf{R}}}$. Then $\langle a\rangle \cdot \pi^{\mathsf{RB}}(s')$ is necessarily an s-plan in $\Pi^{\mathsf{RB}}_{V^{\mathsf{R}}}$, because, for every variable $v \in V^{\mathsf{B}} \cup V^{\mathsf{R}}$, $s'[v] \subseteq s[\![a]\!][v]$. $\square$

Some words are in order regarding the previous works of Fox and Long [21, 22] and Keyder and Geffner [23], which can also be viewed as "un-relaxing" some of the state variables. Keyder and Geffner allow to un-relax any single variable v moving which does not have any side effects on other variables. Fox and Long automatically recognize transportation sub-problems, and un-relax all vehicle position variables v. Note that the un-relaxed variables v have no direct influence on each other. This is in contrast to some of the tractable fragments of red-black planning that we will identify. In both approaches, the actions moving v in a relaxed plan π^+ are replaced by an approximate TSP solution visiting the subset of values from $\mathcal{D}(v)$ required in π^+; denote that set by $D(v, \pi^+)$. This is different from painting v black because it disregards repetitions and ordering of these values. For example, say v is a worker that must perform a sequence $j_1, \dots, j_n$ of jobs, each at a location d_i, where d_i and d_j may be the same for $i \neq j$. Then the TSP approximation follows some path visiting each member of the set $D(v, \pi^+) = \{d_1, \dots, d_n\}$ once. In a red-black plan, v follows a path visiting the sequence $d_1, \dots, d_n$.[5] Indeed, in the relaxed plan repair algorithm

[5]Fox and Long do tackle ordering constraints within the transportation sub-problem, e. g. that

introduced in Section 4.3, moves for the black variables are inserted following the sequence of values required in the relaxed plan. The outcome of that procedure, given the same relaxed plan as input, can only be larger than the outcome of the TSP treatment.

Computing $h^{\mathsf{RB}*}_{V^{\mathsf{R}}}$ is NP-hard even in the simplest case, where there are no black variables, simply because that special case corresponds to h^+. We therefore adopt the popular strategy of upper-approximation by satisficing relaxed planning. That, in turn, requires to investigate the tractability boundary of satisficing planning for RB tasks, and we proceed with this task in the next section.

4. Tractability

Central to our investigation of tractable fragments of RB is the notion of causal graph. While the causal graph CG_Π of an RB task Π is defined exactly as for FDR, it is essential to exploit the red/black coloring of its vertices. In particular, by the **black causal graph** $\mathit{CG}^{\mathsf{B}}_\Pi$ of Π, we refer to the sub-graph of CG_Π induced by only the black variables V^{B} of Π. Another notion that we use in some parts of our analysis is the application of monotonic relaxation to RB instead of FDR. This just means to paint all black variables red, i. e., the monotonic relaxation of an RB task $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ is the MFDR task $\Pi^+ = \langle V^{\mathsf{B}} \cup V^{\mathsf{R}}, A, I, G\rangle$.

Our investigation is sub-divided into three parts, which we present in different sub-sections. The first two parts are oriented at Chen and Giménez' complexity characterization theorems for FDR, analyzing planning complexity as a function of the causal graph structure, with and without imposing the additional requirement of *reversibility* [32]. Throughout, we consider the structure of the *black* causal graph, hence imposing no restrictions on the red variables. In Section 4.1 we find that causal graph structure on its own is insufficient for tractability. In fact, we show that, even if the black causal graph does not contain any arcs at all, both the number and domain size of the black variables must be bounded to obtain a tractable problem. In Section 4.2 we examine the prospects of the reversibility requirement, adapted to the specifics of RB. In analogy to the results of Chen and Giménez', we find that it suffices to bound the size of the strongly connected components in the causal graph. While that result is positive – it suggests, e. g., that it suffices to paint at least one variable on each cycle red – (a) reversibility cannot be easily tested, and (b) the result pertains to plan *existence* only, not to plan

"load" actions must precede "unload" actions for the same object. They cannot, of course, capture ordering constraints caused by aspects of the planning task outside the transportation sub-problem.

generation, while it is the latter that is needed to devise a goal distance estimate. We fix these issues in Section 4.3 where we replace reversibility with a stronger and easily testable notion of *invertibility*, under which satisficing red-black plan generation is tractable.

Our practical heuristic functions, which will be the concern of the entire rest of the paper after this section, rely on the results of Section 4.3 exclusively. Therefore, to not distract from the flow of the paper, we do not give full details in Sections 4.1 and 4.2, instead providing only the core of the proofs and delegating technical details to Appendix A.

4.1. Bounding Variable Number and Size

A first question one might ask is, what about restricting the *number* of black variables? For example, is the red-black planning problem still easy when taking into account the deletes on a single binary-valued variable? It is this question that initiated our investigation, and it turns out the answer is "yes". We now prove the following more general result:[6]

Theorem 1 *Plan generation for* RB *tasks with a fixed number of black variables, each with a fixed size domain, is polynomial-time.*

Proof: A set $\{v_1, \dots, v_n\}$ of black variables can be compiled into a single black variable v with domain $\bigotimes_{i=1}^{n} \mathcal{D}(v_i)$ [33]. This compilation is exponential in n, but incurs polynomial overhead for fixed n. Thus, it suffices to prove the claim for RB tasks $\Pi = \langle \{v_{\mathsf{B}}\}, V^{\mathsf{R}}, A, I, G \rangle$ with a single black variable v_{B}, $|\mathcal{D}(v_{\mathsf{B}})| = k = O(1)$. For ease of presentation, in what follows we assume that actions $A_{v_{\mathsf{B}}}$ affecting v_{B} are also all preconditioned by the value of v_{B}; this assumption is without loss of generality because any action $a \in A_{v_{\mathsf{B}}}$ with no precondition on v_{B} can be replaced with k actions that are preconditioned on different values of v_{B}, and otherwise are identical to a.

A straightforward property of RB tasks with a single black variable that plays an important role in the proof is that the outside conditions of all the actions $A_{v_{\mathsf{B}}}$ are all red. Thus, in particular, once an action from $A_{v_{\mathsf{B}}}$ is applied along a plan for

[6]Note that, trivially, FDR planning with a fixed number of variables is polynomial-time (simply because the state space size is polynomial in the size of the input then). Theorem 1 deals with the strictly more general case of red-black planning, and allows to add an arbitrary number of red variables, arbitrarily used in the task's actions, on top of the fixed number of (black) FDR variables. In difference to the simpler FDR special case, this generality necessitates the restriction to fixed domain sizes.

Π, it can then be re-applied any time, as long as its inside condition on the value of v_{B} is satisfied.

Now, let $\pi^+ = \langle a_1, \dots, a_n \rangle$ be a *relaxed* plan for Π, i. e., a plan for the monotonic relaxation Π^+ of our RB task Π, and, for $0 \leq i \leq n$, let $\pi_i^+ = \langle a_1, \dots, a_i \rangle$ denote the corresponding prefix of π^+.

- For $0 \leq i \leq n$, let Γ_i be the subgraph of the domain transition graph $\mathit{DTG}_{\Pi}(v_{\mathsf{B}})$ induced by the actions $A_{v_{\mathsf{B}}} \cap \{a_1, \dots, a_i\}$. That is, the subgraph Γ_0 comprises only the vertex $I[v_{\mathsf{B}}]$, and, in general, each subgraph Γ_i has the value set $I[\![\pi_i^+]\!][v_{\mathsf{B}}]$ as its vertices, and its arcs constitute the π_i^+-induced subset of the arcs between these vertices in $\mathit{DTG}_{\Pi}(v_{\mathsf{B}})$. From that, it is immediate that each Γ_i is a subgraph of Γ_{i+1}, with the latter extending the former with at most one vertex and at most one arc.

- Let m be the number of times the number of strongly connected components (SCCs) changes from Γ_{i-1} to Γ_i along π^+, and σ be the (increasing) function that captures the indexes of the corresponding SCC-changing actions $\{a_{\sigma(1)}, \dots, a_{\sigma(m)}\}$ along π^+. In what follows, we also use two dummy indexes: $\sigma(0)$, with $a_{\sigma(0)}$ for "establishing" the initial state I, and $\sigma(m+1) \equiv n$, with $\sigma(m+1)$ denoting "the last action of π^+".

Suppose that,

- for all $0 \leq i \leq m$ and all $\sigma(i) < j \leq \sigma(i+1)$, the action a_j either has no precondition on v_{B}, or $\mathsf{pre}(a_j)[v_{\mathsf{B}}]$ and $\mathsf{eff}(a_{\sigma(i)})[v_{\mathsf{B}}]$ belong to the same SCC of $\Gamma_{\sigma(i)}$; and

- if $v_{\mathsf{B}} \in \mathcal{V}(G)$, then $G[v_{\mathsf{B}}]$ and $\mathsf{eff}(a_{\sigma(m)})[v_{\mathsf{B}}]$ belong to the same SCC of $\Gamma_{\sigma(m)}$.

Referring to such relaxed plans π^+ as *SCC-aligned*, in Lemma 2, p. 63, we show that (a) any SCC-aligned relaxed plan can be extended, with only a polynomial overhead, into a proper plan for Π, and (b) the monotonic relaxation π^+ of any plan π for Π is SCC-aligned. Therefore, in particular, any sound and complete procedure for SCC-aligned relaxed planning for RB tasks with singleton V^{B} extends with only a polynomial overhead to a sound and complete red-black planning procedure for that fragment of RB.

Now, let π^+ be an SCC-aligned relaxed plan for Π. Given the sequence of the SCC-changing actions $\langle a_{\sigma(1)}, \dots, a_{\sigma(m)} \rangle$ along π^+, consider a different relaxed plan for Π,

$$\rho^+ = \rho_0^+ \cdot \langle a_{\sigma(1)} \rangle \cdot \rho_1^+ \cdot \langle a_{\sigma(2)} \rangle \cdot \dots \cdot \rho_{m-1}^+ \cdot \langle a_{\sigma(m)} \rangle \cdot \rho_m^+, \quad (1)$$

where, inductively, ρ_i^+ is a relaxed plan from $I[\![\rho_0^+ \cdot a_{\sigma(1)} \cdot \rho_1^+ \cdot \ldots \cdot a_{\sigma(i)}]\!]$ to the relaxed planning fixpoint, *using only those actions from A that are neither preconditioned by values of v_{B} outside of the SCC of $\mathsf{eff}(a_{\sigma(i)})[v_{\mathsf{B}}]$ in Γ_i nor have such values among their effects.*

It is not hard to verify (and we anyway show this in Lemma 3, p. 65), that ρ^+ is also an SCC-aligned relaxed plan for Π and that the SCC-changing actions of ρ^+ are precisely the SCC-changing actions of π^+. Thus, ρ^+ can be seen as a canonical representative of the set of all the relaxed plans for Π that have $\langle a_{\sigma(1)}, \ldots, a_{\sigma(m)}\rangle$ as their sequence of SCC-changing actions. Given that, instead of searching for a general SCC-aligned relaxed plan for Π, we can restrict our search to only the canonical SCC-aligned relaxed plans as in Equation 1 by, e.g.,

1. enumerating all possible candidate sequences $\langle a_{\sigma(1)}, \ldots, a_{\sigma(l)}\rangle$ of SCC-changing actions, and

2. checking whether the corresponding canonical action sequence ρ^+ is a plan for Π^+.

Note that each action $a_{\sigma(i)}$ along a valid candidate $\langle a_{\sigma(1)}, \ldots, a_{\sigma(l)}\rangle$ either extends the set of SCCs of $\Gamma_{\sigma(i-1)}$ with a new SCC in $\Gamma_{\sigma(i)}$, or combines several SCCs in $\Gamma_{\sigma(i-1)}$ into a single SCC in $\Gamma_{\sigma(i)}$. The former type of changes can happen at most $k-1$ times because it corresponds to extending $\Gamma_{\sigma(i-1)}$ with a new vertex, and Γ_0 already contains the vertex $I[v_{\mathsf{B}}]$. In turn, the latter type of changes can also happen at most $k-1$ times because it decreases the number of SCCs from $\Gamma_{\sigma(i-1)}$ to $\Gamma_{\sigma(i)}$ by at least one. In sum, we have $l \leq 2(k-1)$, and thus the number of candidate action sequences is $O(|A_{v_{\mathsf{B}}}|^{2k-2})$. Given that $k = O(1)$, and that test (2) corresponds to $l+1$ calls to (polynomial-time) monotonic planning, putting things together results in a polynomial-time procedure for solving RB tasks with a single fixed-domain black variable. □

We remark that the algorithm we just described differs from that in our previous conference paper [1], and provides a stronger bound, $2k-2$ instead of $(k+1)(k-1)$, on "the amount of search" needed. The bound $2k-2$ is tight in the sense that there are cases where $2k-2$ SCC-changing actions are required to solve Π.

While our algorithm runs in polynomial time for fixed k, it is exponential in that parameter, which itself is exponential in the number of black variables in case we have to pre-compile a fixed number of these. This makes it doubtful whether the algorithm is of practical value for computing heuristic functions.[7] Having this in mind, a natural next question is whether the tractability result of Theorem 1 can be extended either to a fixed number of black variables with unrestricted domains, or to an unrestricted number of black variables with fixed-size domains. Theorems 2 and 3 below show that the answer to this question is "no", even under some additional restrictions on the black variables.

Theorem 2 *Plan existence for* RB *tasks with a fixed number of black variables is* NP*-complete.*

The NP-hardness part of the proof of Theorem 2 is by a reduction from CNF satisfiability testing; the proof appears in Appendix A, p. 66.

Theorem 3 *Plan existence for* RB *tasks where all black variables have fixed-size domains is* PSPACE*-complete, and it is* NP*-complete even if* CG^{B}_{Π} *is arcless.*

The first part of the proof for Theorem 3 is straightforward: If we fix the domain size of the black variables, but not their number, then we obtain a problem that is as hard as FDR with fixed-size domains, which is PSPACE-hard [7]. In turn, PSPACE membership is straightforward because the addition of red variables still allows for a proof similar to that for FDR. The proof of NP-hardness for the second part of the claim is, again, by a reduction from CNF satisfiability testing. The complete proof is given in Appendix A, p. 67

We now relate the statement of Theorems 2 and 3 to the main characterization theorem of Chen and Giménez (2010) on the relation between the FDR planning complexity and the structure of the causal graph:

Theorem 4 (Chen and Giménez, 2010) *Let* $\mathcal{G}$ *be a set of directed graphs. If* $\mathsf{cc\text{-}size}(\mathcal{G})$ *is bounded, then* $\mathsf{PlanGen}(\mathcal{G})$ *is polynomial-time solvable. Otherwise,* $\mathsf{PlanExist}(\mathcal{G})$ *is not polynomial-time decidable (unless* $\mathsf{W}[1] \subseteq \mathsf{nu\text{-}FPT}$*).*

$\mathsf{W}[1] \not\subseteq \mathsf{nu\text{-}FPT}$ is a standard assumption on parametric complexity hierarchy [34]. Note that Theorem 4 is not a dichotomy result: unless $\mathsf{P} = \mathsf{NP}$, there are digraph sets $\mathcal{G}$ for which $\mathsf{PlanExist}(\mathcal{G})$ is neither in P nor NP-hard. As the

[7]Matters may be different if *the actual planning problem we want to solve* can be framed as a member of this tractable fragment. A relevant example where that is the case is monotonic planning with a small number of non-replenishable resources.

tractability result in Theorem 4 covers only trivial planning problems, the theorem shows that valuable islands of tractability within FDR must be characterized in terms that go beyond the structure of the causal graph.

Focusing on the structure of CG^{B}_{Π}, let $\mathsf{RB\text{-}PlanExist}(\mathcal{G})$ and $\mathsf{RB\text{-}PlanGen}(\mathcal{G})$ be respectively the plan existence and plan generation problems restricted to RB planning tasks whose *black causal graphs are elements of $\mathcal{G}$.* Note that these problems put no restriction on the structure of the causal graph itself beyond being a super-graph of some element of $\mathcal{G}$. Theorem 5 below refines the complexity characterization for RB with respect to the structure of the black causal graph, providing a valuable fragment of tractability via Theorem 1, and establishing P/NP *dichotomies* for both general RB, as well as for RB restricted to fixed-size domain variables. The first part of Theorem 5 is by the polynomial-time plan generation for MFDR and Theorem 2, and the second part is by Theorems 1 and 3.

Theorem 5 *Let $\mathcal{G}$ be a set of directed graphs.*

- *If* $\#\mathsf{vertices}(\mathcal{G}) = 0$, *then* $\mathsf{RB\text{-}PlanGen}(\mathcal{G})$ *is polynomial-time solvable. Otherwise,* $\mathsf{RB\text{-}PlanExist}(\mathcal{G})$ *is* NP*-hard.*

- *If* $\#\mathsf{vertices}(\mathcal{G})$ *is bounded, then* $\mathsf{RB\text{-}PlanGen}(\mathcal{G})$ *restricted to* RB *tasks with bounded black variable domains is polynomial-time solvable. Otherwise,* $\mathsf{RB\text{-}PlanExist}(\mathcal{G})$ *for* RB *with bounded black variable domains is* NP*-hard.*

4.2. Causal Graph Structure and Reversibility

Departing from the conclusive yet pessimistic statement of Theorem 4, Chen and Giménez (2010) considered so-called *reversible* FDR tasks: An FDR task $\Pi = \langle V, A, I, G \rangle$ is reversible if, for every state s reachable from I, there exists an action sequence π that "reverts" s back to I, i. e., $s[\![\pi]\!] = I$. The characterization theorem of Chen and Giménez for this fragment of FDR, Theorem 6 below, establishes a valuable tractability result. In fact, this fragment has already been successfully exploited for devising heuristic functions [29].

Theorem 6 (Chen and Giménez, 2010) *Let $\mathcal{G}$ be a set of directed graphs. If* $\mathsf{scc\text{-}size}(\mathcal{G})$ *is bounded, then* $\mathsf{PlanGen}(\mathcal{G})$ *restricted to reversible* FDR *is polynomial-time solvable (under a succinct plan representation). Otherwise,* $\mathsf{PlanExist}(\mathcal{G})$ *for reversible* FDR *is not polynomial-time decidable (unless* $\mathsf{W}[1] \subseteq \mathsf{nu\text{-}FPT}$*).*

Adopting the notion of reversibility in red-black planning requires a slight, natural adaptation: Since the value sets of the red variables in states reachable from I will always include their initial values anyway, reversibility should be requested only on the task's black variables. That is, we say that an RB task $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ is **reversible** if, for every state s reachable from I, there exists an action sequence π so that $s[\![\pi]\!][V^{\mathsf{B}}] = I[V^{\mathsf{B}}]$.

While this extension of reversibility to RB may at first sight appear minor and insignificant, at closer inspection it turns out to be quite substantial. In particular, reversibility of FDR tasks with acyclic causal graph can be tested in linear time at the level of the individual domain transition graphs of the variables: such a task is reversible if and only if the reachable part of each domain transition graph is strongly connected. In contrast, even if RB is restricted to tasks with arcless black causal graphs, testing reversibility is not (likely to be) polynomial-time:

Theorem 7 *It is* co-NP*-hard to test whether an* RB *task is reversible, even when restricting the input to* RB *tasks whose black causal graph is arcless.*

The proof of Theorem 7 is by a reduction from DNF tautology testing; see Appendix A, p. 67. We now show that, somewhat surprisingly given Theorem 7, plan existence for reversible RB tasks with acyclic black causal graphs can be decided in polynomial time.

Theorem 8 *Let $\mathcal{G}$ be a set of directed graphs. If* $\mathsf{scc\text{-}size}(\mathcal{G})$ *is bounded, then* $\mathsf{RB\text{-}PlanExist}(\mathcal{G})$ *restricted to reversible* RB *is polynomial-time solvable. Otherwise, the problem* $\mathsf{RB\text{-}PlanExist}(\mathcal{G})$ *for reversible* RB *is not polynomial-time decidable (unless* $\mathsf{W}[1] \subseteq \mathsf{nu\text{-}FPT}$*).*

Note that Theorem 8 substantially extends Chen and Giménez' tractability result for $\mathsf{PlanExist}(\mathcal{G})$ (Theorem 6 here) to the red-black setting, because Theorem 8 puts no constraints on the overall structure of the causal graph. At the same time, note also the subtle difference between the claims of Theorem 6 and Theorem 8 regarding solving *plan generation* vs. deciding *plan existence*. Both the negative and positive parts of Theorem 8 consider plan existence, whereas the positive part of Theorem 6 makes a stronger claim of tractability of plan generation. It is an open question whether plan generation is tractable in the setting of Theorem 8; we conjecture that it is not. We will get back to this at the end of this section, and for now consider plan existence only.

The negative part of Theorem 8 follows immediately from the negative part of Theorem 6. As for the positive part, given bounded $\mathsf{scc\text{-}size}(\mathcal{G})$ we can with polynomial overhead compile each strongly connected component into a single black variable. Since the compilation leaves the semantics of the task intact, if the original task was reversible, so is the compiled task. Thus, it suffices to consider acyclic black causal graphs. In turn, for this setting, we now show that red-black plan existence is equivalent to relaxed plan existence:[8]

Theorem 9 *Any reversible* RB *task with acyclic black causal graph is solvable if and only if its monotonic relaxation is.*

Proof: The "only if" direction is trivial. For the "if" direction, we start with a simple observation that, in any reversible RB task Π, we can achieve all reachable red facts up front. Let $\mathcal{R} \subseteq \bigcup_{v \in V^{\mathsf{R}}} \mathcal{D}(v)$ be the minimal set of red facts such that, for every state s reachable in Π, we have $s[V^{\mathsf{R}}] \subseteq \mathcal{R}$. If s is reachable in Π, then we can "complete" it into $\mathcal{R}(s) := s \cup \mathcal{R}$ by reverting the black variables to $I[V^{\mathsf{B}}]$, then reaching $\mathcal{R}(I)$ by going to every reachable state in turn, reverting the black variables to $I[V^{\mathsf{B}}]$ in between every two states, and then re-achieving the black facts $s[V^{\mathsf{B}}]$, arriving into $\mathcal{R}(s)$ as desired. In sum, if s is a reachable state in a reversible RB task, then $\mathcal{R}(s)$ is reachable in that task as well. Hence, the existence of a plan for $\mathcal{R}(s)$ implies the existence of a plan for s. So, without loss of generality, we can consider only $\mathcal{R}$-completed states, and, in particular, assume that $I = \mathcal{R}(I)$.

Let $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G \rangle$ be a reversible RB task with $I = \mathcal{R}(I)$ and acyclic black causal graph, and $\{v_1, \dots, v_n\}$ be a topological ordering of V^{B} with respect to $\mathit{CG}^{\mathsf{B}}_{\Pi}$. If $\pi^+ = \langle a_1, \dots, a_m \rangle$ is a relaxed plan for Π, consider a red-black action sequence

$$\pi = \pi_n \cdot \pi_{n-1} \cdot \ldots \cdot \pi_1$$

constructed as follows.

For $1 \leq i \leq n$, assume inductively that $\pi_n \cdot \ldots \cdot \pi_{i-1}$ is applicable in I. If $v_i \notin \mathcal{V}(G)$, then we set $\pi_i := \langle \rangle$. Otherwise, by the acyclicity of the black causal graph and the $\mathcal{R}$-completeness of I, we can revert any top subset of the black variables while leaving the remaining ones untouched. In particular, by Lemma 4 (Appendix A, p. 68), there exists an action sequence ρ_i that reverts the black variables $\{v_1, \dots, v_i\}$ to their initial values, and neither is preconditioned by nor

[8]Note that the theorem shows, in particular, that plan existence for reversible FDR tasks with acyclic causal graphs is equivalent to relaxed plan existence. This special case is rather easy to see, but has to our knowledge not been pointed out previously.

$$\begin{tikzcd}
A \arrow[r, leftrightarrow, red] \arrow[d, red] & \color{red}{F} \arrow[r, leftrightarrow, red] & B \\
\color{red}{O} \arrow[r, leftrightarrow, red] & R \arrow[ul] \arrow[u, red] \arrow[ur] &
\end{tikzcd}$$

Figure 3: Causal graph (and the induced black causal graph) of the red-black planning task obtained from our SimpleGrid running example by painting all variables except F and O black.

affects the topologically lower black variables $\{v_{i+1}, \dots, v_n\}$. That is,

$$I[\![\pi_n \cdot \ldots \cdot \pi_{i+1} \cdot \rho_i]\!][v_j] = \begin{cases} I[\![\pi_n \cdot \ldots \cdot \pi_{i+1}]\!][v_j], & j > i \\ I[v_j], & j \leq i \end{cases}. \tag{2}$$

Given that, if $G[v_i] = I[v_i]$, then we set $\pi_i := \rho_i$. Otherwise, by the virtue of π^+ being a relaxed *plan* for Π, we must have an action a_k in π^+ achieving $G[v_i]$ for some $1 \leq k \leq m$, and, by the acyclicity of CG^{B}_{Π}, the black preconditions of a_k may involve only variables $\{v_1, \dots, v_i\}$. In turn, by Lemma 5 (Appendix A, p. 69), there exists an action sequence π_{a_k} that is applicable in I and achieves $\mathsf{pre}(a_k)$ while neither being preconditioned by nor affecting the topologically lower black variables $\{v_{i+1}, \dots, v_n\}$. By Equation 2 we then have π_{a_k} being applicable in $I[\![\pi_n \cdot \ldots \cdot \pi_{i+1} \cdot \rho_i]\!]$, and, for each black variable $v \in \mathcal{V}(\mathsf{pre}(a_k)) \cap V^{\mathsf{B}}$,

$$I[\![\pi_n \cdot \ldots \cdot \pi_{i+1} \cdot \rho_i \cdot \pi_{a_k}]\!][v] = \mathsf{pre}(a_k)[v].$$

Given that, we set $\pi_i := \rho_i \cdot \pi_{a_k} \cdot \langle a_k \rangle$. It is easy to verify that the action sequence π constructed via this iterative bottom-up achievement of the black sub-goals is applicable in I and is a plan for Π. $\square$

Example 4 *To illustrate with our SimpleGrid example the construction in the proof of Theorem 9, say we paint all variables except F and O black. This yields a reversible* RB *task with acyclic black causal graph as in Figure 3. Assume that I is $\mathcal{R}$-complete, $I = \mathcal{R}(I)$, i.e. $I[F] = \{0, 1\}$ and $I[O] = \{0, 1\}$. (If this is not the case, we can complete I as described in the proof, or more effectively by moving to position 2, taking key A, moving to position 3, opening the lock, and then reverting R and A to their initial values.)*

Consider a goal for the robot to reach location 7 with both keys being picked up. Let π^+ be the relaxed plan reaching this goal by moving to position 2, taking key A, moving to position 7, and taking key B.

The construction of π for this goal $G = \{\langle R/7\rangle, \langle A/R\rangle, \langle B/R\rangle\}$ proceeds as follows. Presume that we process the black goals in the order B, A, R. (The only other choice would be A, B, R.) Starting with the goal fact $\langle B/R\rangle$, we commit to using π^+'s action a achieving that fact, namely taking key B at position 7. This action's precondition generates the new sub-goal $\langle R/7\rangle$. To achieve that sub-goal, we first revert R to its initial value (for which the empty action sequence suffices, in this case), and then following π^+ we achieve $\langle R/7\rangle$ by moving R from 1 to 7. This move sequence has no black outside conditions (if there were such conditions, they would be handled recursively), and all red outside conditions (on O, in this case) are true because $I = \mathcal{R}(I)$. Once the move sequence has executed, we can apply a, finishing with our treatment of variable B.

We next process the goal fact $\langle A/R\rangle$, committing to π^+'s action a' taking key A at position 2. The new sub-goal $\langle R/2\rangle$ is achieved by reverting R to its initial value (moving back from 7 to 1), then following π^+ to move R from 1 to 2. We then apply a' and are done with A. Finally, processing the goal fact $\langle R/7\rangle$, we revert R to its initial value by moving it from 2 to 1, then establish the precondition of the π^+'s action a'' that moves R from 6 to 7 by moving R from 1 to 6. We finalize the red-black plan by applying a''.

As the example illustrates, while the constructed π is a red-black plan, it certainly is not a *good* red-black plan, which it should be in order to avoid overestimation when used inside a heuristic function. Much worse, while the proof of Theorem 9 is constructive, executing that construction involves enumerating all reachable red-black states, so the overall construction of π is *not* polynomial time.

As pointed out, it is an open question whether plan generation for reversible RB with acyclic black causal graphs is tractable, and we conjecture that it is not. To understand this pessimism, recall that the overall causal graph – over black *and* red variables – may contain cycles (e. g., Figure 2 (a)). Thus, it is unclear how to tackle red and black variables in combination, rather than separating the red variables out using $\mathcal{R}$-completeness. In particular, we can *not* in general follow the ordering of goal and sub-goal values achieved in the relaxed plan, because achieving these might require reverting black variables, which in turn might require red values not achieved by the relaxed plan yet.

4.3. Causal Graph Structure and Invertibility

To resolve the issues just observed regarding reversibility, we now replace reversibility with a stronger restriction, i. e., a sufficient criterion for reversibility.

This criterion is easily testable, and we show that, given the criterion applies, satisficing red-black plan generation is tractable for acyclic black causal graphs.

Our criterion is based on the idea of *invertibility*, where every action application can be directly "undone". Invertibility criteria have been devised previously, e. g. by Koehler and Hoffmann [35]. Straightforwardly translated to FDR, Koehler and Hoffmann's criterion postulates, for every action a, the existence of a corresponding inverse action a'. That action must be always applicable behind a, ensured in FDR by $\mathsf{pre}(a') \subseteq \mathsf{prevail}(a) \cup \mathsf{eff}(a)$; and it must undo a exactly, ensured in FDR by $\mathcal{V}(\mathsf{eff}(a')) = \mathcal{V}(\mathsf{eff}(a)) \subseteq \mathcal{V}(\mathsf{pre}(a))$ and $\mathsf{eff}(a') = \mathsf{pre}(a)[\mathcal{V}(\mathsf{eff}(a))]$. For any reachable state s, we can then revert to the initial state I simply by inverting the path that lead from I to s.

It turns out that our setting admits a much less restrictive definition. What's more, the definition is per-variable, identifying a subset $V_I \subseteq V$ of FDR variables (the invertible ones) that can be painted black in principle. This enables efficient red-black relaxation design: Identify the set V_I, paint all other variables red, keep painting more variables red until there are no more cycles in the black causal graph.

Note that, once the selection of red variables is completed, every action will affect at most one black variable, or otherwise there would be cycles between the black effect variables. Since red variables accumulate their values anyway, there is no need to "undo" any effects on them. Therefore, inverting an action a now comes down to undoing its single effect (if any) on a black variable. Denoting that variable by v_{B}, inverting a corresponds to inverting an arc (d, a, d') in the domain transition graph of v_{B}. Furthermore, both the outside condition and the outside effect of (d, a, d') will remain true and can be used as conditions for the inverse arc: For $\langle u/e \rangle \in \mathsf{oeff}(d, a, d')$, this is simply because u is affected by a but $u \neq v_{\mathsf{B}}$, and so u must be red. For $\langle u/e \rangle \in \mathsf{ocon}(d, a, d')$, if u is red there is nothing to show, and if u is black then $\langle u/e \rangle$ must be a prevail condition because all the outside effects are red. Putting these observations together leads us to the following definition.

An arc (d, a, d') is **relaxed side effects invertible**, or **RSE-invertible** for short, if there exists an arc (d', a', d) with outside condition

$$\mathsf{ocon}(d', a', d) \subseteq \mathsf{ocon}(d, a, d') \cup \mathsf{oeff}(d, a, d').$$

A variable v is RSE-invertible if all arcs in $\mathit{DTG}_{\Pi}(v)$ are RSE-invertible, and an RB task is RSE-invertible if all its black variables are.

Theorem 10 *Any RSE-invertible* RB *task with acyclic black causal graph is reversible.*

Proof Sketch: The proof in Appendix A, p. 70 shows that, for any state s and action a applicable in s, from $s[\![\langle a\rangle]\!]$ one can reach a state s' so that $s'[V^{\mathsf{B}}] = s[V^{\mathsf{B}}]$ and, for every $v \in V^{\mathsf{R}}$, $s'[v] \supseteq s[v]$. This is trivial if all variables in $\mathcal{V}(\mathsf{eff}(a))$ are red. Otherwise, exactly one variable v_{B} affected by a is black. Considering the arc (d, a, d') in $\mathit{DTG}_{\Pi}(v_{\mathsf{B}})$ taken by a in s, there exists an inverse arc (d', a', d) by prerequisite, and the claim follows easily from the arguments outlined above. $\square$

Obviously, RSE-invertibility can be tested in polynomial time. Note that it generalizes the earlier definition for actions, given above: v_{B} being the black effect variable of a, and assuming as above that $\mathcal{V}(\mathsf{eff}(a)) \subseteq \mathcal{V}(\mathsf{pre}(a))$, it corresponds to the requirement that $\mathsf{pre}(a') \subseteq \mathsf{pre}(a) \cup \mathsf{eff}(a)$ (compared to $\mathsf{pre}(a') \subseteq \mathsf{prevail}(a) \cup \mathsf{eff}(a)$) and $\mathsf{eff}(a')[v_{\mathsf{B}}] = \mathsf{pre}(a)[v_{\mathsf{B}}]$ (compared to $\mathsf{eff}(a') = \mathsf{pre}(a)[\mathcal{V}(\mathsf{eff}(a))]$).

Example 5 *Consider the SimpleGrid example from Figure 1, and the domain transition graphs of F and R as illustrated in Figure 2 (b) and (c). These variables are RSE-invertible. For R, arcs $(i, i+1)$ and $(i+1, i)$ where $\{i, i+1\} \cap \{4\} = \emptyset$ have empty outside conditions, and thus are trivially RSE-invertible. The other arcs all have outside condition $\{O = 1\}$, and so they are RSE-invertible, too.*

For F, arcs $(1, 0)$ induced by take(x, y) are inverted by arcs $(0, 1)$ induced by drop(x, y): $\mathsf{ocon}(0, 1) = \{\langle R/x\rangle, \langle y/R\rangle\}$ *is contained in* $\mathsf{ocon}(1, 0) = \{\langle R/x\rangle, \langle y/x\rangle\} \cup \mathsf{oeff}(1, 0) = \{\langle y/R\rangle\}$*. Similarly vice versa, i. e., arcs $(0, 1)$ are inverted by the corresponding arcs $(1, 0)$. Note here that* $\mathsf{ocon}(0, 1) \not\subseteq \mathsf{ocon}(1, 0)$*, i. e., it is important to allow the inverse arc to make use of conditions established by the original arc's outside effect.*[9]

The outside condition of $\mathit{DTG}_{\Pi}(R)$ arcs $(4, 3)$ and $(4, 5)$ in our example is non-standard in the sense that it is not explicitly specified in the IPC version of the Grid domain. There, instead, that condition is an invariant based on the implicit assumption that the robot is initially in an open position. We have chosen this example to illustrate that, like previous similar notions, RSE-invertibility can be subject to modeling details. It would be an option to use invariance analysis (e. g.,

[9]We remark that the more restrictive version, requiring $\mathsf{ocon}(d', a', d) \subseteq \mathsf{ocon}(d, a, d')$ instead of $\mathsf{ocon}(d', a', d) \subseteq \mathsf{ocon}(d, a, d') \cup \mathsf{oeff}(d, a, d')$, corresponds to Hoffmann's [36] notion of DTG invertibility.

[37, 38, 39, 40]) to detect implicit preconditions, but we have not done so for the moment.

Together with Theorem 9, Theorem 10 immediately implies that:

Corollary 1 *Any RSE-invertible* RB *task with acyclic black causal graph is solvable if and only if its monotonic relaxation is.*

In other words, existence of a relaxed plan implies existence of a red-black plan. These results, however, do not tell us anything about how to actually find such a plan efficiently. Indeed, recall our conjecture that there is no polynomial-time "how to" when relying on reversibility only. The stronger notion of RSE-invertibility solves that issue: If all black variables are RSE-invertible, then we can efficiently *repair* a relaxed plan to form a valid red-black plan. We will spell this algorithm out in detail below. In a nutshell, we simply execute the relaxed plan action-by-action. Whenever the black precondition of the next action (or, at the end, the goal) is not satisfied, we project the planning task onto the black variables, and then project these variables' domains onto the values already visited along our red-black plan prefix. We then solve this projected task to move the black variables into place. The projected tasks have black variables only, so are in FDR. Their causal graph is acyclic because the black causal graph of the original task Π is. Thanks to RSE-invertibility, all their domain transition graphs are strongly connected. These two properties together ensure that we can solve each projected task efficiently:

Lemma 1 *Let $\mathcal{G}$ be a set of directed graphs. If all graphs in $\mathcal{G}$ are acyclic, then* $\mathsf{PlanGen}(\mathcal{G})$ *restricted to* FDR *with strongly connected domain transition graphs is polynomial-time (under a succinct plan representation).*

Proof: By Chen and Giménez [32], as stated in Theorem 6, $\mathsf{PlanGen}(\mathcal{G})$ restricted to reversible FDR is polynomial-time, under a succinct plan representation. So it suffices to show that any FDR task Π with acyclic causal graph and strongly connected domain transition graphs is reversible. Say that s is any reachable state in $\Pi = \langle V, A, I, G \rangle$. We construct a modified task as $\Pi' = \langle V, A, s, I \rangle$, i. e., we take s as the initial state and we take the original initial state as the goal. Obviously, it now suffices to show that Π' is solvable. Since Π', like Π itself, has an acyclic causal graph and strongly connected domain transition graphs, that follows directly from Observation 7 of Helmert [29], which shows that any FDR task with these properties is solvable. $\square$

Results closely related to Lemma 1 have been mentioned at various places in the literature (e. g., [27, 41, 28, 29, 32]), but to our knowledge the lemma has never been stated in this precise form, as is needed in our context.

The succinct plan representation in Chen and Giménez' proof exploits recursive macros for value pairs within domain transition graphs. This representation trick is required as plans in this setup may be exponentially long (e. g., [42, 36]).[10] We remark that, in our actual implementation (detailed below in Section 5.3), we use an explicit plan representation, not relying on macros, as long plans do not tend to occur in our context (inside the heuristic function), and building the macros would incur way too much overhead. In that sense, the importance of Lemma 1 is mainly theoretical. It enables us to prove the main result of this section. We include the full proof here as it essentially consists of our *relaxed plan repair* algorithm, and thus directly underlies our basic heuristic function:[11]

Theorem 11 *Let $\mathcal{G}$ be a set of directed graphs. If all graphs in $\mathcal{G}$ are acyclic, then* $\mathsf{RB\text{-}PlanGen}(\mathcal{G})$ *restricted to RSE-invertible* RB *is polynomial-time (under a succinct plan representation).*

Proof: Figure 4 provides pseudo-code for an algorithm that turns, with polynomial overhead, a relaxed plan into a red-black plan. We use the notations from the figure in what follows.

Consider an iteration i of the main loop. Any red preconditions of a_i are true in the current state $I[\![\pi]\!]$ because the red-black plan prefix π includes the relaxed plan actions $a_1, \dots, a_{i-1}$ processed so far. Unsatisfied black preconditions g are tackled by $\textsc{Achieve}(\pi, g)$, solving an FDR task Π^{B} with goal g. The returned action sequence π^{B} is attached to π.

We next prove that (i) Π^{B} is well-defined, that (ii) all its domain transition graphs are strongly connected, and that (iii) any plan π^{B} for Π^{B} is, in our RB task Π, applicable in the current state $I[\![\pi]\!]$. This suffices to prove the claim: As the causal graph of Π^{B} is (obviously) acyclic, with (ii) and Lemma 1 we know that

[10]The basic idea is to design chain causal graphs $v_1 \to v_2 \to \cdots \to v_n$ where, in a plan, every move of v_i requires several moves of v_{i-1} for achieving its precondition. In this situation, Chen and Giménez' macros for v_i just record the necessary moves of v_{i-1} via recursively pointing to the macros for v_{i-1}, without ever spelling out the actual action sequence represented by the macros (this is akin to the commonly known recursive solution to Towers of Hanoi).

[11]Note that, in difference to Theorems 8 and 9, we require the causal graph to be acyclic, as opposed to having bounded-size strongly connected components. As we show in Theorem 12 below, this restriction is necessary.

```
Algorithm : RelaxedPlanRepair($\Pi, \pi^+$)
 main
  // $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ and $\pi^+ = \langle a_1, \dots, a_n\rangle$
  $\pi \leftarrow \langle a_1 \rangle$
  for $i = 2$ to $n$
    do { if $\mathsf{pre}(a_i)[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
       {   then { $\pi^{\mathsf{B}} \leftarrow$ Achieve($\mathsf{pre}(a_i)[V^{\mathsf{B}}]$)
       {        { $\pi \leftarrow \pi \cdot \pi^{\mathsf{B}}$
       { $\pi \leftarrow \pi \cdot \langle a_i \rangle$
  if $G[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
    then { $\pi^{\mathsf{B}} \leftarrow$ Achieve($G[V^{\mathsf{B}}]$)
         { $\pi \leftarrow \pi \cdot \pi^{\mathsf{B}}$
 procedure Achieve($\pi, g$)
  $F \leftarrow I \cup \bigcup_{a \in \pi} \mathsf{eff}(a)$
  for $v \in V^{\mathsf{B}}$
    do $\mathcal{D}^{\mathsf{B}}(v) \leftarrow \mathcal{D}(v) \cap F$
  $I^{\mathsf{B}} \leftarrow I[\![\pi]\!][V^{\mathsf{B}}]$
  $G^{\mathsf{B}} \leftarrow g$
  $A^{\mathsf{B}} \leftarrow \{a^{\mathsf{B}} \mid a \in A, a^{\mathsf{B}} = \langle \mathsf{pre}(a)[V^{\mathsf{B}}], \mathsf{eff}(a)[V^{\mathsf{B}}]\rangle,$
            $\mathsf{pre}(a) \subseteq F, \mathsf{eff}(a)[V^{\mathsf{B}}] \subseteq F\}$
  $\Pi^{\mathsf{B}} \leftarrow \langle V^{\mathsf{B}}, A^{\mathsf{B}}, I^{\mathsf{B}}, G^{\mathsf{B}}\rangle$
  $\pi^{\mathsf{B}} \leftarrow$ an FDR plan for $\Pi^{\mathsf{B}}$   // $\Pi^{\mathsf{B}}$ is necessarily solvable
  return $\pi^{\mathsf{B}}$
```

Figure 4: The *relaxed plan repair* algorithm for solving a red-black planning task Π. The algorithm underlies the proof of Theorem 11.

we can solve Π^{B} in polynomial time (under a succinct plan representation), and so the overall algorithm runs in polynomial time. As Π^{B} ignores the red variables, but effects on these cannot hurt anyway, with (iii) a_i is applicable in $I[\![\pi \cdot \pi^{\mathsf{B}}]\!]$. Iterating that argument shows that the algorithm returns a red-black plan.

For (i), we need to show that all variable values $\langle v/d \rangle$ occurring in Π^{B} are indeed members of the respective variable domains, i. e., $d \in \mathcal{D}^{\mathsf{B}}(v)$. This is obvious for I^{B} and A^{B}. It holds for G^{B} because by construction these are facts made true by the relaxed plan actions $a_1, \dots, a_{i-1}$ already processed.

For (ii), observe that all values in $DTG_{\Pi^{\mathsf{B}}}(v)$ are, by construction, reached from $I[v]$ by a sequence of arcs (d, d') induced by actions in π. So it suffices to prove that every such arc has a corresponding arc (d', d) in $DTG_{\Pi^{\mathsf{B}}}(v)$. Say $v \in V^{\mathsf{B}}$, and (d, a^{B}, d') is an arc in $DTG_{\Pi^{\mathsf{B}}}(v)$ induced by a^{B} where $a \in \pi$. Because (d, a, d') is RSE-invertible in Π, there exists an action $a' \in A$ inducing an arc (d', a', d) in $DTG_{\Pi}(v)$ whose outside condition is contained in $\mathsf{pre}(a) \cup \mathsf{eff}(a)$. Since, obviously, $\mathsf{pre}(a) \cup \mathsf{eff}(a) \subseteq F$, we get $\mathsf{pre}(a') \subseteq F$. Since a' can have only one black effect, $\mathsf{eff}(a')[V^{\mathsf{B}}] = \{\langle v/d \rangle\}$ which is contained in F. Thus $a'^{\mathsf{B}} \in A^{\mathsf{B}}$, and (d', d) is an arc in $DTG_{\Pi^{\mathsf{B}}}(v)$ as desired.

Finally, (iii) holds because, with $\mathsf{pre}(a) \subseteq F$ for all actions where $a^{\mathsf{B}} \in A^{\mathsf{B}}$, the red preconditions of all these actions a are true in the current state $I[\![\pi]\!]$. So applicability of π^{B} in $I[\![\pi]\!]$, in Π, depends on the black variables only, all of which are contained in Π^{B}. This concludes the proof. □

Example 6 *Consider again the SimpleGrid example from Figure 1. For $V^{\mathsf{B}} = \{R, F\}$, the example is in our tractable fragment: the black causal graph is acyclic (compare Figure 2 (a)), and R and F are RSE-invertible (Figure 2 (b) and (c), cf. Example 5).*

Say the relaxed plan π^+ is: $\langle$move$(1,2)$, take$(2,A)$, move$(2,3)$, open$(3,4,A)$, move$(3,4)$, move$(4,5)$, move$(5,6)$, move$(6,7)$, take$(7,B)$, drop$(1,B)\rangle$.

In RELAXEDPLANREPAIR(Π, π^+)*, there will be no calls to* ACHIEVE(π, g) *until a_i=take$(7,B)$. To achieve the precondition of that action,* ACHIEVE(π, g) *constructs Π^{B} with initial state $\{\langle R/7\rangle, \langle F/0\rangle\}$ and goal $\{\langle R/7\rangle, \langle F/1\rangle\}$, and an action set all of whose "drop" actions have the form "drop(x,A)" (because the red fact $B = R$ has not been reached yet). We get the plan $\pi^{\mathsf{B}} = \langle$drop$(7,A)\rangle$. Thus drop$(7,A)$ and take$(7,B)$ are added to π. The next iteration calls* ACHIEVE(π, g) *for the precondition of drop$(1,B)$, constructing Π^{B} with initial state $\{\langle R/7\rangle, \langle F/0\rangle\}$ and goal $\{\langle R/1\rangle\}$, yielding π^{B} moving R back to 1. Subsequently, π^{B} and drop$(1,B)$ are added to π, and the algorithm stops. The returned action sequence is a real plan and its length corresponds to the perfect heuristic value 17.*

Theorem 11 does *not* hold for the weaker requirement of bounded-size strongly connected components. This is because multiplying fixed-size sets of variables into single variables may lose RSE-invertibility. Indeed:

Theorem 12 *There exists a set $\mathcal{G}$ of directed graphs where* scc-size$(\mathcal{G})$ *is bounded by* 2*, and* RB-PlanExist$(\mathcal{G})$ *restricted to RSE-invertible* RB *is* NP*-hard.*

Proof Sketch: Given a CNF formula ϕ, for each Boolean variable p occurring in ϕ we include two binary state variables x_p and y_p with initial value 0, an action a_{px1y1} with precondition $\{\langle x_p/0\rangle, \langle y_p/0\rangle\}$ and effect $\{\langle x_p/1\rangle, \langle y_p/1\rangle\}$, an action a_{px0} with precondition $\{\langle x_p/1\rangle, \langle y_p/0\rangle\}$ and effect $\{\langle x_p/0\rangle\}$, and an action a_{py0} with precondition $\{\langle x_p/0\rangle, \langle y_p/1\rangle\}$ and effect $\{\langle y_p/0\rangle\}$. Each individual variable x_p and y_p is RSE-invertible, but their product is not: we can transition to $\{\langle x_p/1\rangle, \langle y_p/1\rangle\}$ but we cannot go back. Thus, for each p in ϕ, we can encode the decision whether to set p to true ($\langle x_p/1\rangle, \langle y_p/1\rangle\}$) or false ($\{\langle x_p/0\rangle, \langle y_p/0\rangle\}$). Adding, in a straightforward manner, variables and actions that allow to test satisfaction of ϕ for given decisions on each p concludes the argument. The detailed proof is given in Appendix A, p. 71. □

We remark that the proof does not actually require any red variables, so the same result holds for the special case of FDR planning. Note that the issue here lies in the assumption underlying RSE-invertibility, that all outside-effect variables will be painted red: The transition $(0, 1)$ of x_p (and likewise of y_p) is RSE-invertible because the outside condition $\langle y_p/0\rangle$ of the inverse transition $(1, 0)$ is contained in $\mathsf{ocon}(0, 1)$, and the over-writing effect $\langle y_p/1\rangle \in \mathsf{oeff}(0, 1)$ is, under our assumption, red and thus harmless. When merging x_p with y_p into a single (black) variable, this assumption breaks because the "side effect" is now compiled into that single variable, with black semantics. It remains an open question whether stronger notions of invertibility can yield the ability to naturally handle bounded-size strongly connected components, and whether that yields any advantage in practice.

5. Heuristic Functions

With tractability of (satisficing) red-black plan generation, Theorem 11 establishes an essential prerequisite for generating heuristic functions in analogy to relaxed plan heuristics, and it already provides a red-black planning algorithm for doing so. The resulting basic heuristic function, which we denote by $h^{\text{RB}}_{\text{repair}}$, is obtained for any state s by

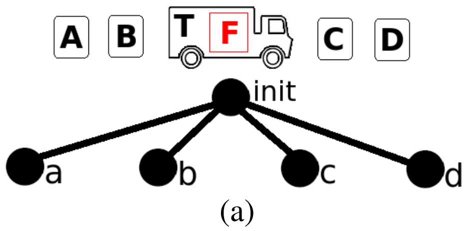
- generating a relaxed plan π^+ for s,
- returning $h^{\text{RB}}_{\text{repair}}(s) := \infty$ if no relaxed plan exists and hence (by Corollary 1) no red-black plan exists either,
- else using relaxed plan repair to transform π^+ into a red-black plan π^{RB} for s, and
- returning the length of π^{RB} as the heuristic value $h^{\text{RB}}_{\text{repair}}(s)$.

We herein tackle the obstacles involved in making this practical. Foremost, we observe in Section 5.1 that $h^{\text{RB}}_{\text{repair}}$ is prone to dramatic over-estimation even in trivial examples. This is due to bad decisions inherited from the relaxed plan π^+, which we fix in Section 5.2 by devising a refined algorithm, *red facts following*, that relies less on π^+; the refined heuristic is denoted $h^{\text{RB}}_{\text{follow}}$. In Section 5.3, we fill in some details, pertaining in particular to our handling of acyclic black causal graphs. Section 5.4 concludes our treatment by empirically showing the merits of $h^{\text{RB}}_{\text{follow}}$ vs. $h^{\text{RB}}_{\text{repair}}$, and of using acyclic black causal graphs as opposed to arcless ones (which were used in our prior works on red-black plan heuristics). We will keep the "winning techniques" ($h^{\text{RB}}_{\text{follow}}$, acyclic black causal graphs) fixed for the rest of the paper, simplifying and focusing the subsequent empirical evaluations.

5.1. Over-Estimation in Relaxed Plan Repair

As a first illustration, consider that in Example 6 we used the relaxed plan $\langle$*move*$(1,2)$, *take*$(2,A)$, *move*$(2,3)$, *open*$(3,4,A)$, *move*$(3,4)$, *move*$(4,5)$, *move*$(5,6)$, *move*$(6,7)$, *take*$(7,B)$, *drop*$(1,B)\rangle$. We showed that relaxed plan repair (with R and F painted black) ends up returning an optimal real plan. What we did *not* say is that this relies on a particular sequencing of the relaxed plan. Under the delete relaxation, the relaxed plan may just as well start with $\langle$*move*$(1,2)$, *move*$(2,3)$, *take*$(2,A)$, *open*$(3,4,A)\rangle$ instead. If that happens, then a call to ACHIEVE(π, g) before *take*$(2,A)$ will insert *move*$(3,2)$, and another call before *open*$(3,4,A)$ will insert *move*$(2,3)$, needlessly undoing and re-doing the work involved in getting from position 2 to position 3. While this may seem benign, similar forms of over-estimation are triggered massively by the typical behavior of standard relaxed plan extraction schemes. Consider the following illustrative example:

Example 7 *In Figure 5, truck T needs to transport each package $X \in \{A,B,C,D\}$ to its respective goal location $x \in \{a,b,c,d\}$. The truck can only carry one package at a time, encoded by a binary variable F ("free"). A real plan has length* 15 *(*8 *load/unload actions,* 7 *drive actions). A relaxed plan has length* 12 *(*4 *drive actions suffice as there is no need to drive back).*



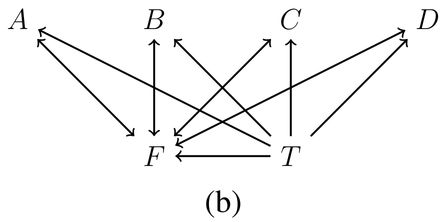


Figure 5: (a) An illustrative example "StarShapeLogistics", and (b) its causal graph

Standard relaxed plan extraction schemes (e. g. [6, 23]) can be characterized as starting at the goal facts, selecting a "best supporter" action a for each (minimizing precondition achievement cost, estimated using h^{max} or h^{add} [5]), marking a's preconditions as new sub-goals, and iterating until the sub-goals are true in the initial state. Doing this in a breadth-first fashion, and scheduling the actions a in reverse order of selection, we obtain a sequential relaxed plan π^+ as needed as input for relaxed plan repair. In our example here, π^+ will start with the 4 *drive$(init, x)$ actions, followed by the* 8 *load$(X, init)$ and unload(X, x) actions.*

Say we paint T and F black and paint the packages red, obtaining an RSE-invertible task with acyclic black causal graph as desired. Example issues are:

(A) Say that $\pi^+ = \langle$*drive*$(init, a), \dots,$ *drive*$(init, d)$, *load*$(A, init)$, *unload*(A, a), $\dots$, *load*$(D, init)$, *unload*$(D, d)\rangle$. *Handing this over to relaxed plan repair, processing the drive*$(init, x)$ *actions will result in a valid path of drives navigating the truck across the entire map – without performing any actual loads or unloads! Subsequently, when load*$(A, init)$ *comes up, we drive back to* $init$*, and drive on to* A *for unload*(A, a)*; and similarly for* B, C, D*. The resulting red-black plan duplicates the effort for driving across the entire map.*

(B) Matters are even worse if π^+ *schedules load*$(A, init), \dots,$ *load*$(D, init)$ *in front of unload*$(A, a), \dots,$ *unload*(D, d)*: When load*$(B, init)$ *comes up, we need to achieve the black precondition* $\langle F/1\rangle$*, the shortest plan for which is to apply unload*$(A, init)$*. Note that, after the latter action,* A *is "still in the truck" because the position of* A *is painted red. The resulting red-black plan duplicates drives* and *contains superfluous unloading actions (which furthermore cause the red-black plan to be inapplicable in the original task).*

(C) Finally, say we extend the example by including N *endpoint locations,* N *packages that need to be transported to these, and* N *trucks, where all packages and trucks are initially in the middle. Then all optimal plans use one truck per package. An optimal relaxed plan, however, can use a single truck. Starting from this, relaxed plan repair will use a single truck as well.*[12]

To mention a concrete IPC benchmark in which such things happen, consider, e. g., IPC'11 Elevators. Assume we paint the lift positions black and paint everything else red. Like in Example 7 (A), *board/leave* actions ("loading/unloading" passengers) will tend to be scheduled behind the elevator moves, resulting in a red-black plan that first moves the elevators all over the place without actually transporting anybody. Similarly as in Example 7 (B), since the relaxed plan is free to choose any *board/leave* actions, it may decide to use the same lift capacity precondition (typically, the one initially true one for that lift) for all boarding actions. This forces the red-black plan to achieve the desired capacity by applying useless instances of board/leave.

An issue not represented in Example 7 are cumbersome solutions enforced by the need to incorporate the relaxed plan actions, whereas not incorporating them

[12]Similar phenomena may occur in any domain with alternative resources that can be flexibly assigned to sub-tasks. Relaxed plans, generated using standard techniques such as extraction from a relaxed planning graph [6], will tend to always use the same resource for all sub-tasks (assuming they always use the same arbitrary tie-breaking). That bad decision will be inherited in relaxed plan repair.

allows much simpler (and shorter!) solutions. A prime example for this is IPC'11 VisitAll, painting the robot position black. If, for example, in the current state the robot is located in the right bottom corner of a grid, then the relaxed plan is likely to visit the grid in a breadth-first fashion, going outwards in all directions from that corner. Given this, during relaxed plan repair, when the robot reaches, say, the top right corner, instead of just moving one step to the left (to the nearest yet un-visited grid cell), we move it all the way back to the bottom before moving out from there again.

5.2. A Refined Algorithm: Red Facts Following

The major source of the observed difficulties is that relaxed plan repair commits to the particular *actions* chosen by the relaxed plan, as well as to their *ordering*. We now define an algorithm, *red facts following*, that makes do with a much weaker commitment, namely to *the set of red facts employed by the relaxed plan*. Namely, we consider the set

$$R^+ := G[V^{\mathsf{R}}] \cup \bigcup_{a \in \pi^+} \mathsf{pre}(a)[V^{\mathsf{R}}]$$

of red facts that are required either by the goal, or by an action precondition in π^+.[13] Pseudo-code for our algorithm is shown in Figure 6.

The algorithm maintains two monotonically increasing sets of variable values: R is the set of all *currently already achieved* red variable values, and B is the set of all black variable values *currently achievable under* R, i. e., that we can achieve based on using only (red) outside conditions from R. Both R and B are maintained by the UPDATE procedure. Consider that procedure first. For $v \in V^{\mathsf{B}}$, $\textit{DTG}_\Pi(v)|_{R\cup B}$ is obtained as follows. Let G be the subgraph of $\textit{DTG}_\Pi(v)$ obtained by removing all arcs whose outside condition is not contained in $R\cup B$. The graph $\textit{DTG}_\Pi(v)|_{R\cup B}$ is obtained from G by removing all vertices (and incident arcs) that are not reachable from $I[v]$. Abusing notation, we use $\textit{DTG}_\Pi(v)|_{R\cup B}$ to denote both the DTG subgraph and the set of vertices (variable values) of that graph. The updating is done in a topological order, i. e., from the roots of the black causal graph to its leaves, because each variable depends on its parent variables so we need to determine the reachable values for the parent variables first.

[13]Any relaxed plan π^+ works for us, and different relaxed plans may yield different sets R^+. In that sense, in particular, our approach is unrelated to landmarks (e. g., [43, 44, 45, 11]), which are concerned with things that happen in *all* possible (relaxed) plans.

```
Algorithm : RedFactsFollowing($\Pi$, $R^+$)
 main
  // $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G \rangle$
  global $R \leftarrow \emptyset$,   $B \leftarrow \emptyset$,   $\pi \leftarrow \langle\rangle$
  Update()
  while $R \not\supseteq R^+$
    do { $A_0 = \{a \in A \mid \mathsf{pre}(a) \subseteq B \cup R, \mathsf{eff}(a) \cap (R^+ \setminus R) \neq \emptyset\}$
       { Select $a \in A_0$
       { if $\mathsf{pre}(a)[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
       {   then $\pi^{\mathsf{B}} \leftarrow$ Achieve($\mathsf{pre}(a)[V^{\mathsf{B}}]$), $\pi \leftarrow \pi \cdot \pi^{\mathsf{B}}$
       { $\pi \leftarrow \pi \cdot \langle a \rangle$
       { Update()
  if $G[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
    then $\pi^{\mathsf{B}} \leftarrow$ Achieve($G[V^{\mathsf{B}}]$), $\pi \leftarrow \pi \cdot \pi^{\mathsf{B}}$
  return $\pi$

 procedure Update()
  $R \leftarrow I[\![\pi]\!][V^{\mathsf{R}}]$
  $B \leftarrow B \cup I[\![\pi]\!][V^{\mathsf{B}}]$
  for $v \in V^{\mathsf{B}}$, ordered topologically by the black causal graph
    do $B \leftarrow B \cup DTG_{\Pi}(v)|_{R \cup B}$

 procedure Achieve($g$)
  $\overleftarrow{V}^{\mathsf{R}} \leftarrow \bigcup_{v \in V^{\mathsf{B}}} V^{\mathsf{R}}(\mathsf{ocon}(\overleftarrow{DTG_{\Pi}(v)|_{R \cup B}}))$
  for $v \in V^{\mathsf{B}} \cup \overleftarrow{V}^{\mathsf{R}}$
    do { $\mathcal{D}^{\mathsf{B}}(v) \leftarrow \mathcal{D}(v)$
       { if $v \in V^{\mathsf{B}}$
       {   then $\mathcal{D}^{\mathsf{B}}(v) \leftarrow \mathcal{D}^{\mathsf{B}}(v) \cap B$
  $I^{\mathsf{B}} \leftarrow I[\![\pi]\!][V^{\mathsf{B}} \cup \overleftarrow{V}^{\mathsf{R}}]$, $G^{\mathsf{B}} \leftarrow g$
  $A^{\mathsf{B}} \leftarrow \{a^{\mathsf{B}} \mid a \in A, a^{\mathsf{B}} = \langle \mathsf{pre}(a)[V^{\mathsf{B}} \cup \overleftarrow{V}^{\mathsf{R}}], \mathsf{eff}(a)[V^{\mathsf{B}} \cup \overleftarrow{V}^{\mathsf{R}}]\rangle$,
           $\mathsf{pre}(a) \subseteq R \cup B$ or ex. $v \in V^{\mathsf{B}} : a \in A(\overleftarrow{DTG_{\Pi}(v)|_{R \cup B}})\}$
  $\Pi^{\mathsf{B}} \leftarrow \langle V^{\mathsf{B}}, \overleftarrow{V}^{\mathsf{R}}, A^{\mathsf{B}}, I^{\mathsf{B}}, G^{\mathsf{B}} \rangle$
  $\langle a_1'^{\mathsf{B}}, \dots, a_k'^{\mathsf{B}} \rangle \leftarrow$ an RB plan for $\Pi^{\mathsf{B}}$ // $\Pi^{\mathsf{B}}$ is necessarily solvable
  return $\langle a_1', \dots, a_k' \rangle$
```

Figure 6: The *red facts following* algorithm for solving a red-black planning task Π, refining relaxed plan repair (cf. Figure 4) to reduce over-estimation. The algorithm assumes that $R^+ = G[V^{\mathsf{R}}] \cup \bigcup_{a \in \pi^+} \mathsf{pre}(a)[V^{\mathsf{R}}]$ where π^+ is a relaxed plan for Π. For explanation of notations, see text.

Getting back to the main procedure, consider the **while** loop. Our candidate actions for inclusion in the red-black plan prefix, in every iteration of the loop, are given by the set A_0 of actions whose preconditions are contained in $B \cup R$ (the red precondition is true and the black precondition is achievable), and that achieve at least one fact from $R^+ \setminus R$ (the action makes progress on R^+). Once an action $a \in A_0$ is selected, the call to Achieve($\mathsf{pre}(a)[V^{\mathsf{B}}]$) serves to find a plan fragment π^{B} establishing its black precondition, if needed. We append π^{B} to the red-black plan prefix π, and we append a itself, then we iterate. In other words, the red facts R^+ employed in the relaxed plan serve as targets which we are free to achieve in any order, and using any actions we like, as long as we know we will be able to achieve their black preconditions. Once all of R^+ has been achieved, we know that we will be able to achieve the black goal.

It remains to explain the $\textsc{Achieve}(g)$ procedure, responsible for generating the plan fragments π^{B} establishing black sub-goals g corresponding to either the precondition of an action in A_0, or to the black part of the original goal. Similarly as in relaxed plan repair, this is accomplished via a projected planning task Π^{B} designed for that purpose. However, the design is more complicated than before. By $\overleftarrow{\mathit{DTG}_\Pi(v)|_{R\cup B}}$, we denote the set of "complementary inverse transitions" (d', a', d) for $\mathit{DTG}_\Pi(v)|_{R\cup B}$: For every arc (d, a, d') in $\mathit{DTG}_\Pi(v)|_{R\cup B}$, we include into $\overleftarrow{\mathit{DTG}_\Pi(v)|_{R\cup B}}$ all inverse arcs (d', a', d) (i. e., arcs with $\mathsf{ocon}(d', a', d) \subseteq \mathsf{ocon}(d, a, d') \cup \mathsf{oeff}(d, a, d')$) that are not already contained in $\mathit{DTG}_\Pi(v)|_{R\cup B}$ itself. Such inverse arcs must make use of at least one yet non-established red outside condition $\langle v/d\rangle \in \mathsf{ocon}(d', a', d) \setminus R$, where by construction we must have $\langle v/d\rangle \in \mathsf{oeff}(d, a, d')$.[14] By $V^{\mathsf{R}}(\mathsf{ocon}(\overleftarrow{\mathit{DTG}_\Pi(v)|_{R\cup B}}))$, we denote the set of red variables appearing in the outside conditions of the arcs $\overleftarrow{\mathit{DTG}_\Pi(v)|_{R\cup B}}$. By $A(\overleftarrow{\mathit{DTG}_\Pi(v)|_{R\cup B}})$, we denote the set of actions inducing the arcs in $\overleftarrow{\mathit{DTG}_\Pi(v)|_{R\cup B}}$.

Let us explain why the more complicated construction of Π^{B} is required. In relaxed plan repair, the sub-goals g to be achieved always consist of facts (variable values) already visited on the plan prefix π. However, in the present algorithm *g may contain facts that we can reach given the current R, but that we haven't actually visited yet*. This leads to complications with our generous definition of invertibility, where the inverse transition (d', d) may make use of red outside conditions that will be established only through the red outside effect *when executing the original transition* (d, d'). To capture this behavior, we need to keep track of which potential red outside conditions of inverse transitions – values of variables $V^{\mathsf{R}}(\mathsf{ocon}(\overleftarrow{\mathit{DTG}_\Pi(v)|_{R\cup B}}))$ – are currently true, and we need to include the actions $A(\overleftarrow{\mathit{DTG}_\Pi(v)|_{R\cup B}})$ which will then be usable to execute the activated inverse transitions. This is best understood through an example:

Example 8 *Consider an* RB *task with two black variables b_1, b_2 and one red variable r. All variables are binary-valued and initialized to* 0*. The goal is* $\langle b_1/0\rangle, \langle b_2/1\rangle$*. The actions are a_{1fwd} with precondition $\langle b_1/0\rangle$ and effect $\langle b_1/1\rangle, \langle r/1\rangle$;*

[14]As $\langle v/d'\rangle$ is in B but (d', a', d) is not in $\mathit{DTG}_\Pi(v)|_{R\cup B}$, (d', a', d) must have at least one outside condition $p \notin R \cup B$. By construction, $p \in \mathsf{ocon}(d, d') \cup \mathsf{oeff}(d, d')$. We cannot have $p \in \mathsf{ocon}(d, d')$ as $\mathsf{ocon}(d, a, d') \subseteq R \cup B$. Hence $p \in \mathsf{oeff}(d, a, d')$; by construction, all these outside effects must be red.

*a_{1bwd} with precondition $\langle b_1/1\rangle, \langle r/1\rangle$ and effect $\langle b_1/0\rangle$; a_{2fwd} with precondition $\langle b_2/0\rangle, \langle b_1/1\rangle$ and effect $\langle b_2/1\rangle$; and a_{2bwd} with precondition $\langle b_2/1\rangle, \langle b_1/1\rangle$ and effect $\langle b_2/0\rangle$. The black causal graph is acyclic and both black variables are RSE-invertible. Say the relaxed plan is $\langle a_{1fwd}, a_{2fwd}\rangle$, hence $R^+ = \emptyset$, and hence the **while** loop in Figure 6 terminates immediately and we get a single call of* ACHIEVE(g)*, on the original goal.*

Assume for the moment that, in ACHIEVE(g)*, we would be using a simpler construction Π^{B} with only the black variables, and not including the actions $A(\overleftarrow{\textbf{DTG}_{\Pi}(v)|_{R \cup B}})$. Then Π^{B} would have variables b_1 and b_2, including their values $\langle b_1/1\rangle$ and $\langle b_2/1\rangle$ because $\langle b_1/1\rangle$ can be reached given $R = \{\langle r/0\rangle\}$ and $\langle b_2/1\rangle$ can be reached given $\langle b_1/1\rangle$, but* not *including a_{1bwd} because that requires the red outside condition $\langle r/1\rangle \notin R$. This Π^{B} is unsolvable because, while we can bring b_1 to value 1 as required for moving b_2, we cannot bring b_1 back into value 0 as required for its own goal.*

To ensure solvability of Π^{B}, we must ascertain that we can always "go back". RSE-invertibility does ensure that, but subject to red outside conditions that will be established "on the way out". Using ACHIEVE(g) *as stated in Figure 6, $\overleftarrow{\textbf{DTG}_{\Pi}(b_1)|_{R \cup B}}$ consists of the single arc $(1, 0)$ with outside condition $r = 1$, and $\overleftarrow{\textbf{DTG}_{\Pi}(b_2)|_{R \cup B}}$ is empty. Thus Π^{B} has all three variables because $r \in \overleftarrow{V}^{\mathsf{R}} = V^{\mathsf{R}}(\mathsf{ocon}(\overleftarrow{\textbf{DTG}_{\Pi}(b_1)|_{R \cup B}})) \cup V^{\mathsf{R}}(\mathsf{ocon}(\overleftarrow{\textbf{DTG}_{\Pi}(b_2)|_{R \cup B}}))$, and includes all actions because $a_{1bwd} \in A(\overleftarrow{\textbf{DTG}_{\Pi}(b_1)|_{R \cup B}})$.*

Note that Π^{B} here is not a standard FDR task, but is a red-black planning task itself. It is a benign kind of red-black planning task though, because its black causal graph is acyclic, there aren't any goals on the red variables, and for each black DTG we know that (a) from our initial position we can reach all values, and (b) once we traversed any arc (d, d') we will have the red outside conditions required for at least one inverse arc (d', d). Like acyclic FDR tasks with strongly connected DTGs, such RB tasks are always solvable (and, as we shall describe further below, can be solved very similarly). Hence our algorithm works as desired:

Theorem 13 *Let $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ be an RSE-invertible* RB *planning task with acyclic black causal graph, π^+ be a relaxed plan for Π, and $R^+ = G[V^{\mathsf{R}}] \cup \bigcup_{a \in \pi^+} \mathsf{pre}(a)[V^{\mathsf{R}}]$. Then, assuming a complete solver for the sub-tasks Π^{B} generated,* REDFACTSFOLLOWING(Π, R^+) *terminates, and the action sequence π it returns is a plan for Π.*

Proof Sketch: The proof is very similar to that of Theorem 11, and it is given in Appendix A, p. 72. The major differences lie in the structure of the main loop (following R^+ instead of the actions in the relaxed plan) and in the details regarding the sub-tasks Π^{B}.

The **while** loop terminates because, as long as $R \not\supseteq R^+$, we always have $A_0 \neq \emptyset$. This is simply because there always exists an action in π^+ which is a member of A_0: As π^+ achieves all of R^+, there must be at least one action $a_i \in \pi^+$ with $\mathsf{eff}(a_i) \cap (R^+ \setminus R) \neq \emptyset$; for the smallest such index i, it is easy to see that the π^+ prefix up to i cannot make use of any preconditions outside $R \cup B$. Similarly, once the **while** loop has terminated, we must have $G[V^{\mathsf{R}}] \subseteq R$ simply because $G[V^{\mathsf{R}}] \subseteq R^+$, and we must have $G[V^{\mathsf{B}}] \subseteq B$ because π^+ cannot make use of any preconditions outside $R \cup B$.

The correctness of precondition and goal achievement, i. e., the correct functioning of the calls to ACHIEVE(g), follows, similarly as in the proof of Theorem 11, from three properties of the sub-tasks Π^{B}: (i) Π^{B} is well-defined; (ii) Π^{B} is solvable; and (iii) any plan π^{B} for Π^{B} is, in the RB task Π, applicable in $I[\![\pi]\!]$. Thanks to (i) and (ii), we will obtain a plan π^{B} for Π^{B}. Thanks to (iii) it is valid to append π^{B} to π. Furthermore, while Π^{B} ignores some of the red variables, effects on these cannot hurt anyway, so (in the **while** loop) the action a is applicable in $I[\![\pi \cdot \pi^{\mathsf{B}}]\!]$.

The proofs for (i) and (iii) are minor extensions to those in the proof of Theorem 11. The proof of (ii) is by an extension of Helmert's Observation 7. Recall that, by this observation, any FDR task with acyclic causal graph and strongly connected domain transition graphs is solvable. This follows from:

(1) Acyclicity of the causal graph implies that we can solve the planning task "top-down", from causal graph leaves to roots, fixing a DTG path for each variable v and propagating the required preconditions as sub-goals to v's parents.

(2) As every DTG is strongly connected, every required path is available, i. e., every variable can always move from its current value to any other value it is required to achieve as a sub-goal (or its own goal).

(1) is preserved in our setting, for the black variables; the red variables are handled exclusively as part of our adaptation of (2) below, which is possible as they have no own goals (G^{B} does not mention the red variables). To ascertain (2), we employ the following two observations, valid for every black variable $v \in V^{\mathsf{B}}$:

(a) From v's start value, $I^{\mathsf{B}}[v] = I[\![\pi]\!][v]$, all values $d \in \mathcal{D}^{\mathsf{B}}(v)$ of v's reduced domain are reachable in $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$.

(b) Assume that π^{B} is any applicable action sequence in Π^{B} which has, at some point, executed action a^{B} traversing $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$ arc (d, d'). Then there exists an action $a'^{\mathsf{B}} \in A^{\mathsf{B}}$ inducing an inverse arc (d', d) whose red outside conditions are contained in the outcome $I^{\mathsf{B}}[\![\pi^{\mathsf{B}}]\!]$ of applying π^{B} in Π^{B}.

To see that (a) and (b) together show (2), as the paths whose existence are postulated in (2) may make use of arbitrary black outside conditions, we need to worry only about red outside conditions. When v makes its first move, by (a) any path that may be required is available and relies only on the red facts R which are true. In any subsequent move of v, for all $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$ arcs we have traversed so far, by (b) there exists a suitable inverse action a' relying only on red outside conditions that are already true. So, to reach any value d_g that may be required, we can go back to the start value $I^{\mathsf{B}}[v]$ exploiting (b), and subsequently move from $I^{\mathsf{B}}[v]$ to d_g exploiting (a).

It remains to prove (a) and (b). Both arguments are very similar to how we showed in Theorem 11 that DTGs are strongly connected: Once we actually executed an action in Π^{B}, the red outside conditions needed for the inverse action are true. This shows (a) because, by construction, all values in $\mathcal{D}^{\mathsf{B}}(v) = \mathit{DTG}_{\Pi}(v)|_{R \cup B}$ are reachable from $I[v]$, and as π induces a path from $I[v]$ to $I[\![\pi]\!][v]$ in $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$ we can invert that path to go back from $I[\![\pi]\!][v]$ to $I[v]$. For (b), once we applied a^{B} in Π^{B} where a' is the corresponding inverse action, if $\mathsf{pre}(a') \subseteq R \cup B$ then the claim is trivial. If $\mathsf{pre}(a') \not\subseteq R \cup B$ then $a' \in A(\overleftarrow{\mathit{DTG}_{\Pi}(v)|_{R \cup B}})$, so we have $a'^{\mathsf{B}} \in A^{\mathsf{B}}$ and $\mathcal{V}(\mathsf{pre}(a')) \cap V^{\mathsf{R}} \subseteq \overleftarrow{V}^{\mathsf{R}}$. This concludes the proof as, by construction, $\mathsf{pre}(a')[V^{\mathsf{R}}] \subseteq \mathsf{pre}(a) \cup \mathsf{eff}(a)$. □

Does our refined algorithm actually yield a benefit? Does $h^{\text{RB}}_{\text{follow}}$ reduce overestimation, compared to $h^{\text{RB}}_{\text{repair}}$? Further below, we provide empirical data strongly supporting that the answer is "yes". For now, let's reconsider the examples from Section 5.1.

Example 9 *In the StarShapeLogistics example from Figure 5, no matter how we order the relaxed plan, the set R^+ will consist of the facts $\{\langle A/\mathit{init}\rangle, \langle A/T\rangle, \langle A/a\rangle, \dots, \langle D/\mathit{init}\rangle, \langle D/T\rangle, \langle D/d\rangle\}$. The truck moves are then completely up to the calls of the* ACHIEVE(g) *procedure, and the* ***while*** *loop is free to choose the order in which to tackle the sub-goals (loading a package, bringing it to its goal position). Hence, assuming that* ACHIEVE(g) *finds short plans, and the choice of $a \in A_0$ is done intelligently enough to prefer delivering a package once it is loaded – which we do accomplish in our implementation, see Section 5.3.2 – the issues pointed out in Example 7 (A) and (B) are solved.*

The issue pointed out in Example 7 (C), however, persists. If the relaxed plan uses a single truck T only, then R^+ will have the same form as above, fixing usage of T through the red facts $\langle A/T \rangle, \dots, \langle D/T \rangle$. It remains an open question how to solve this, and related issues of resource assignment. Perhaps low-conflict relaxed plans [18] could deliver input better suited to that purpose.

As far as the two IPC benchmarks we pointed out are concerned, matters are fine. In Elevators, painting the lift positions black and painting everything else red, R^+ consists of passenger initial, intermediate (in a lift) and goal locations, as well as of lift capacities required along π^+. The actions achieving these facts are *board/leave*; these are the actions selected by the main loop, and moving the lifts is entirely up to $\textsc{Achieve}(g)$, solving issue (A). Regarding issue (B), assume that all *board* actions in π^+ are preconditioned by the initially true capacity of that lift (c_l). Then all *leave* actions in π^+ will be preconditioned by the $c_l - 1$ capacity, and c_l and $c_l - 1$ are the only capacity-related facts in R^+. As c_l is true in the initial state, and $c_l - 1$ is achieved by the first *board* action into the respective lift, after that action there are no more capacity-related facts in $R^+ \setminus R$. Thus action selection in the main loop will be based exclusively on following red facts related to the passenger locations.

In VisitAll, painting the robot variable black and everything else red, R^+ consists of "visited(location)" facts exclusively, and the robot moves are mainly determined by how the next such fact is selected for achievement, i. e., how we select $a \in A_0$. Based on the selection criteria we use in our implementation (Section 5.3.2), the red-black planner will always move to a yet non-visited location nearby, and the issue is solved.

5.3. Realization Details

We first fill in the details how we handle the sub-tasks Π^{B} that need to be solved for black precondition (and goal) achievement within both relaxed plan repair and red facts following. We then describe important optimizations regarding the choice points in red facts following.

5.3.1. Solving the Sub-Tasks Π^{B}

Consider first the sub-tasks Π^{B} in relaxed plan repair, which are FDR tasks with acyclic causal graph and strongly connected DTGs (namely, $\mathit{DTG}_{\Pi^{\mathsf{B}}}(v)$ is the subgraph of $\mathit{DTG}_{\Pi}(v)$ induced by the values $\mathcal{D}(v) \cap F$, cf. Figure 4). As discussed in the context of Lemma 1, plan generation for such tasks is polynomial time, using a succinct plan representation (required as plans may be exponentially

```
Algorithm : AcyclicPlanning($\Pi^{\mathsf{B}}$)
 main
  $\pi^{\mathsf{B}} \leftarrow \langle\rangle$
  for $i = n$ downto 1
    do
      // Denote $\pi^{\mathsf{B}} = \langle a_1, \dots, a_k \rangle$
      $d \leftarrow I[v_i]$
      for $j = 1$ to $k$
        do
          $\pi_j \leftarrow \langle\rangle$
          if $\mathsf{pre}(a_j)[v_i]$ is defined
            then
              $\pi_j \leftarrow \pi_{v_i}(d, \mathsf{pre}(a_j)[v_i])$
              $d \leftarrow \mathsf{pre}(a_j)[v_i]$
      $\pi_{k+1} \leftarrow \langle\rangle$
      if $G[v_i]$ is defined
        then $\pi_{k+1} \leftarrow \pi_{v_i}(d, G[v_i])$
      $\pi^{\mathsf{B}} \leftarrow \pi_1 \cdot \langle a_1 \rangle \cdot \dots \cdot \pi_k \cdot \langle a_k \rangle \cdot \pi_{k+1}$
  return $\pi^{\mathsf{B}}$
```

Figure 7: Planning algorithm for FDR tasks Π^{B} with acyclic causal graph $CG_{\Pi^{\mathsf{B}}}$ and strongly connected DTGs. $v_1, \dots, v_n$ is an ordering of variables V consistent with the topology of $CG_{\Pi^{\mathsf{B}}}$. $\pi_v(d, d')$ denotes an action sequence constituting a shortest path in $DTG_{\Pi^{\mathsf{B}}}(v)$ from d to d'.

long). The succinct plan representation, devised by Chen and Giménez [32], consists of recursive macro actions for pairs of initial-value/other-value within each variable's DTG. That approach, while superior in theory, has several disadvantages that make its practicality in our setting more than doubtful. First, generating the macros involves the exhaustive enumeration of shortest paths for initial-value/other-value pairs in all DTGs, which must be done anew for every call of $\textsc{Achieve}(g)$ (i. e., several times inside each invocation of the heuristic!), as each task Π^{B} has its own individual initial state and DTGs. Second, the macros yield highly redundant plans moving parent variables back to their initial value in between every two sub-goal requests. For example, if a truck unloads two packages at the same location, then it is moved back to its start location in between the two unload actions. Finally, the tasks Π^{B} will typically have small plans anyhow – after all, we merely wish to achieve the next action's black preconditions – so why should we bother to represent these plans compactly?

Given these considerations, we decided to use an explicit plan representation instead, trading the theoretical worst-case efficiency of Chen and Giménez' macros against the practical advantages of less overhead and (potentially) shorter plans. After exploring a few options, we settled on the simple algorithm in Figure 7. Starting at the leaf variables and working up to the roots, the partial plan π^{B} is augmented with plan fragments (DTG paths) bringing the supporting variables into place. This is essentially the same algorithm as described by Helmert [29] as a proof for his Observation 7 (Helmert did not actually implement and use that algorithm, though). It is easy to see that:

Proposition 2 *The algorithm* ACYCLICPLANNING(Π^{B}) *is sound and complete, and its runtime is polynomial in the size of* Π^{B} *and the length of the plan* π^{B} *returned.*

Proof Sketch: In Appendix A, p. 75, we show by induction that, at the end of each iteration i of the **for**-loop, π^{B} is a plan for Π^{B} projected on variables $v_i, \dots, v_n$. This is trivial for $i = n$. Given it holds for $i+1, \dots, n$, it also holds for i because, by acyclicity of the causal graph, the actions inserted to move v_i do not affect any other variables, and do not rely on any preconditions on the variables $v_{i+1}, \dots, v_n$. □

As indicated, the length of π^{B} here is worst-case exponential in the size of Π^{B}, and so is the runtime of ACYCLICPLANNING(Π^{B}).[15] Unlike the macro-based algorithm of Chen and Giménez, our algorithm does not superfluously keep switching supporting variables back to their initial values. But it is not especially clever, either: If variable v_0 supports two otherwise independent leaf variables v_1 and v_2, then the sub-plans for v_1 and v_2 will be inserted sequentially into π^{B}, losing any potential for synergies in the values of v_0 required. We performed a limited investigation into more flexible algorithms addressing that weakness through using a partially-ordered π^{B}, but these algorithms required non-trivial book-keeping, and initial implementations did not yield any apparent benefits. It remains an open question whether something can be gained by a more complex machinery here.

In red facts following, the sub-tasks Π^{B} are red-black planning tasks, with the weaker properties discussed above: the black causal graph is acyclic, there aren't any goals on the red variables, and for each black DTG we know that (a) from our initial position we can reach all values, and (b) once we traversed any arc (d, d') we will have the red outside conditions required for at least one inverse arc (d', d). As argued in the proof of Theorem 13, these tasks are still guaranteed to be solvable. We can use the same decomposition method (solving Π^{B} from leaves to roots of the black causal graph), and whenever we need a DTG path from the current value to a sub-goal value d_g of a black variable v, by (b) we know that

[15]One could choose to merely estimate the plan length of Π^{B} (e. g., using Helmert's causal graph heuristic [29]), computing a red-black plan *length estimate* only. But that would forgo the possibility to actually execute red-black plans (cf. below), which is a key advantage in practice.

v can "go back" to its start value, and by (a) v can reach d_g from there. Therefore, to appropriately extend the algorithm as depicted in Figure 7, there is no need to consider the red-black planning tasks Π^{B} as defined in Figure 6. Instead, we focus on the black variables in these Π^{B} tasks, and initialize their DTGs as $\mathit{DTG}_{\Pi^{\mathsf{B}}}(v) := \mathit{DTG}_{\Pi}(v)|_{R \cup B}$. We collect, for each variable v_i individually (i. e., within each iteration of the **for** loop in Figure 7), the red side effects R_{v_i} of the current π^{B} prefix affecting v_i, extending $\mathit{DTG}_{\Pi^{\mathsf{B}}}(v)$ with the transitions whose red outside conditions are contained in $R \cup R_{v_i}$. In fact, we do so only in case there is no path in $\mathit{DTG}_{\Pi^{\mathsf{B}}}(v)$ to the current sub-goal value d_g. This is to avoid unnecessary computational overhead: In all IPC benchmarks, and in all search states that were encountered in these benchmarks during our experiments, the necessary DTG paths during ACYCLICPLANNING(Π^{B}) were present in $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$ already, without enabling any new transitions based on red side effects. (In other words, situations as in Example 8 appear to be extremely rare in practice.)

In previous works on red-black plan heuristics [2, 3], we made use of a simpler tractable fragment of red-black plan generation, namely that where the black causal graph is arcless. In this special case, the planning tasks solved inside ACHIEVE(g) are trivial: All black variables v are completely independent, and it suffices to find a DTG path from v's current value to $g[v]$ (if defined) for each v individually. This reduces the computational effort required to compute the heuristic function, at the price of a potential loss in accuracy. As we shall see below in Section 5.4, the more complex heuristics based on acyclic black causal graphs tend to pay off in domains where non-trivial acyclic black causal graphs occur. (In case the black causal graph is arcless, our more complex heuristics simplify to the heuristics defined for that special case.)

5.3.2. Instantiating the Choice Points

The main choice points are (i) selecting actions $a \in A_0$ in the **while** loop (pertains to red facts following), and (ii) selecting DTG paths for black variables in the solution to Π^{B}, i. e., inside ACHIEVE(g) (pertains to both, red facts following and relaxed plan repair). We would like to make both choices in a way such that the returned red-black plan π is (a) short, and (b) executable as much as possible in the original FDR planning task we are trying to solve.

The importance of (a) should be self-evident (avoiding over-estimation). (b) is important because red-black plans, compared to fully-delete relaxed plans, have a much higher chance of actually working in reality. We exploit this property by a simple method we refer to as **stop search**: If the red-black plan π generated for a search state s is a plan for s in the original FDR task, then we stop and output π

pre-fixed by the action sequences that lead to s. For illustration, in StarShapeLogistics (Figure 5), a fully relaxed plan *will not work* unless we have already transported all but one package. If we paint just T black, then the red-black plan for the initial state *might* work (in case it happens to make the right choices where to place the *load* and *unload* actions). If we paint both T and F black, then *every optimal red-black plan for the initial state definitely works*, and we can stop the search before we have even started it. (The restriction to *optimal* red-black plans is needed here because, package variables being red, non-optimal red-black plans may contain superfluous *load* and *unload* actions.)

Towards finding (a) short red-black plans, our simple measure in choice point (ii) is to find shortest DTG paths (cf. Figure 7). Choice point (i) is more important, and more difficult to handle. The set of actions A_0 achieving *some* fact from $R^+ \setminus R$ often is quite large (for example, in VisitAll when painting just the robot position black, A_0 contains every action moving into any yet non-visited position in the grid). The straightforward criterion is to select an action achieving whose precondition takes the smallest number of steps. We approximate this number, for any action $a \in A_0$, by pre-computing all-pairs shortest path distances for each black variable v, and summing up the distance from $I[\![\pi]\!][v]$ to $\mathsf{pre}(a)[v]$ over all of a's black precondition variables $v \in V^{\mathsf{B}} \cap \mathcal{V}(\mathsf{pre}(a))$. (Note that these estimates are exact if, and only if, the paths underlying these distances do not rely on any outside conditions that we would need to establish.)

Towards enhancing (b) FDR-executability of the red-black plan, we designed a simple criterion for each of the choice points (i) and (ii). To illustrate our criterion for (i), say that, in StarShapeLogistics, T and F are painted black, $R^+ = \{\langle A/init\rangle, \langle A/T\rangle, \langle A/a\rangle, \dots, \langle D/init\rangle, \langle D/T\rangle, \langle D/d\rangle\}$, and red facts following started by selecting *load*$(A, init)$. Then *unload*(A, a) *might* be selected next, but the algorithm might just as well select *load*$(B, init)$ because the (estimated and real) number of steps for achieving the precondition is 1 for each of these actions: Exactly one black precondition needs to be achieved, namely $\langle T/a\rangle$ for *unload*(A, a) and $\langle F/1\rangle$ for *load*$(B, init)$, and each of these takes a single transition in the respective DTG, induced by *drive*$(init, a)$ respectively *unload*$(A, init)$. Regarding *unload*$(A, init)$, note that variable A is red, so the detrimental side effect is ignored, and later on the red-black plan will apply *unload*(A, a) without re-loading A in between, losing FDR-executability. In other words, unless we manage to distinguish between *drive*$(init, a)$ respectively *unload*$(A, init)$ here, we suffer from a similar issue as pointed out in Example 7 (B). The same phenomenon may occur in any domain with renewable resources. We tackle it by giving a preference to actions $a \in A_0$ getting whose black preconditions does not involve deleting R^+ facts already achieved beforehand. To avoid excessive overhead, we approximate this by recording, in a pre-process, which red facts may be deleted by moving each black variable, and prefer an action if none of its black preconditions may incur any such side effects. In our example, moving F may incur such side effects, but moving T may not.

In choice point (ii), we enhance FDR-executability simply by preferring executable DTG paths. This pertains exclusively to the red outside conditions on the paths. We say that such a condition is "active" if it is true when executing the current red-black plan prefix under the FDR (fully un-relaxed) semantics. As long as there exists at least one DTG path all of whose red outside conditions are active, we use a shortest such path. (E.g., if a storage-capacity variable is red, then this will prefer *load*s/*unload*s that use the actual capacity instead of an arbitrary one.)

5.4. Evaluation

As indicated, we include experiments at this point already in order to evaluate specific aspects of our algorithm design so far, and to simplify the experiments in the remainder of the paper by fixing the "winning techniques". We evaluate, in this order, (i) the advantage of red facts following over relaxed plan repair, (ii) the impact of the stop search method and the associated FDR applicability enhancements, and (iii) the advantage of using acyclic black causal graphs over using arcless black causal graphs.

All our techniques are implemented on top of Fast Downward (FD). All experiments in this paper are run on a cluster of Intel E5-2660 machines running at 2.20 GHz, with runtime (memory) limits of 30 minutes (2 GB). We run all satisficing-track STRIPS benchmarks from the FD benchmark collection, i. e., benchmarks from the IPC satisficing/deterministic/sequential tracks (where distinguished from other forms of planning). We consider uniform costs throughout, ignoring action costs where specified. Since we cannot paint any variable black if there are no RSE-invertible variables, we omit instances in which that is the case, and we omit domains where it is the case for all instances. These domains are (all variants of) Airport, Freecell, Openstacks, Parking, and Pathways.

To keep things simple, in all of (i) – (iii) here we fix a canonical search algorithm, namely FD's greedy best-first search with lazy evaluation and a second open list using preferred operators [29, 46]. We also fix a simple painting strategy, i. e., a method for deciding which variables to paint red or black. The strategy is based on Fast Downward's *level* ordering of the variables [12, 29]. It prefers to paint red the variables with higher level, i. e., the variables "close to the causal graph leaves". We will describe the strategy in detail, along with all other painting strategies we use, in Section 6. The qualitative picture of the following results is similar for our other painting strategies.

		Coverage				Average $h(I)$		Coverage			Evaluations		Solved in I		
		No PO									RDL/		Coverage		Plan length
Domain	#	**FDL**	**RDL**	**FDLS**	**RDLS**	**FDLS**	**RDLS**	**RDL**	**RDLS-**	**RDLS**	**RDLS-**	**RDLS**	**RDLS-**	**RDLS**	**RDL/RDLS**
Barman	20	20	19	20	20	36.3	36.3	20	20	20	1.0	1.0	0	0	
Blocksworld	35	27	33	31	35	18.0	18.0	35	35	35	1.0	1.0	1	1	1.0
Depots	22	11	15	18	18	48.6	37.2	18	18	18	1.0	1.0	1	1	1.0
Driverlog	20	15	18	20	20	56.7	38.8	19	20	20	1.1	1.1	1	2	0.9
Elevators08	30	18	29	28	30	202.1	85.2	30	30	30	398.0	398.0	30	30	1.0
Elevators11	20	2	10	8	20	501.6	209.6	20	20	20	2899.0	4079.5	16	20	1.1
Floortile	20	4	4	6	7	147.6	64.1	6	6	7	1.0	1.1	0	0	
Grid	5	4	4	5	5	31.0	31.0	5	5	5	1.1	1.1	0	0	
Gripper	20	20	20	20	20	137.8	91.0	20	20	20	93.0	93.0	20	20	1.0
Logistics98	35	5	35	35	35	148.1	103.1	35	35	35	167.0	167.0	35	35	1.0
Logistics00	28	28	28	28	28	54.6	43.9	28	28	28	99.5	99.5	28	28	1.0
Miconic	150	150	150	150	150	98.5	55.1	150	150	150	113.5	113.5	150	150	1.0
Mprime	35	31	30	35	35	7.3	7.0	35	35	35	1.2	1.3	0	0	
Mystery	28	18	18	17	16	8.2	7.3	16	16	16	1.5	1.5	0	0	
NoMystery	20	9	14	6	13	72.6	36.6	13	13	13	1.1	71.0	0	10	1.0
ParcPrinter08	20	15	20	20	20	54.2	50.4	20	20	20	1.1	1.1	0	0	
ParcPrinter11	13	4	13	13	13	93.2	87.2	13	13	13	1.0	1.0	0	0	
PegSol08	30	30	30	30	30	11.9	11.4	30	30	30	1.0	1.0	0	0	
PegSol11	20	20	20	20	20	14.4	13.8	20	20	20	1.0	1.0	0	0	
Pipesworld-NoTankage	40	18	22	33	33	26.6	24.1	33	33	33	1.2	1.2	0	0	
Pipesworld-Tankage	40	15	16	29	30	30.4	25.9	30	30	30	1.0	1.0	0	0	
PSR	50	50	50	50	50	3.1	3.1	50	50	50	1.0	1.0	0	0	
Rovers	40	16	20	40	40	143.8	86.7	40	40	40	1.1	1.1	3	3	0.8
Satellite	36	26	34	36	36	236.1	128.0	36	36	36	13.1	13.1	10	10	1.0
Scanalyzer08	21	18	19	21	21	27.6	21.7	21	21	21	1.0	1.0	0	0	
Scanalyzer11	14	11	12	14	14	33.0	25.5	14	14	14	1.0	1.0	0	0	
Sokoban08	30	29	28	29	27	71.2	42.9	29	29	27	1.0	1.0	0	0	
Sokoban11	20	19	18	19	17	85.2	50.2	18	19	17	1.0	1.0	0	0	
Tidybot	20	14	15	14	13	32.7	32.7	13	13	13	1.0	1.0	0	0	
TPP	30	21	23	30	30	79.0	66.4	30	30	30	1.0	1.0	5	5	1.0
Transport08	30	21	25	26	30	262.8	86.2	28	28	30	40.0	319.0	12	30	1.0
Transport11	20	4	7	9	20	768.1	212.7	15	15	20	7.9	3038.0	0	20	1.2
Trucks	30	14	15	16	18	76.9	56.8	18	18	18	1.0	1.0	0	0	
VisitAll	20	1	19	20	20	19476.8	1102.9	19	20	20	1684.0	1684.0	20	20	1.0
Woodworking08	30	30	30	30	30	56.3	47.0	30	30	30	1.0	1.0	1	1	1.0
Woodworking11	20	20	20	20	20	90.0	76.7	20	20	20	1.0	1.0	0	0	
Zenotravel	20	20	20	20	20	45.2	33.3	20	20	20	1.6	41.0	6	20	1.0
$\sum$	1082	778	903	966	1004			997	1000	1004			339	406	

Table 1: Data analyzing (i) the advantage of red facts following over relaxed plan repair (left half) and (ii) the impact of stop search and its FDR applicability enhancements (right half). The averages in "Average $h(I)$", and all ratios, are over those instances commonly solved by the pair of planners involved; for ratios we show the median per domain. "#" is the number of instances in our benchmark set. The two "No PO" configurations do not use preferred operators. By **S-** we refer to stop search without the enhancements to (b) as described in Section 5.3.2. The "Solved in I" data regards those instances where stop search fires on the initial state already.

Even though each of our experiments will focus on particular algorithm parameters and/or performance aspects only, for easier reference across experiments, we employ a system of acronyms to identify configurations. Again for easier cross-reference, we choose these acronyms to be consistent with those in our earlier works [2, 3]. Red facts following is denoted by **R** and relaxed plan repair is denoted by **F** (this seemingly unintuitive notation was used by Katz et al. [2] in reference to a minor optimization of re-ordering the relaxed plan by **F**orwarding actions with no black effects). The use of acyclic black causal graphs is denoted

D (for "DAG") and that of arcless black causal graphs is denoted **E**. The painting strategy described above is denoted **L**. If stop search is in use, we denote that by **S**. For example, "**FEL**" is the configuration that uses relaxed plan repair with arcless black causal graphs and painting strategy **L** and stop search switched off, whereas "**RDLS**" is the configuration that uses red facts following with DAG black causal graphs and painting strategy **L** and stop search switched on. (Other configuration parameters are less important so we don't define acronyms for them.) Table 1 shows our data regarding (i) and (ii).

Consider first the left half of the table. In the "No PO" configurations, we measure the quality of the respective heuristic functions $h^{\text{RB}}_{\text{repair}}$ (**F**) vs. $h^{\text{RB}}_{\text{follow}}$ (**R**) in the most basic setting possible, plugging them into a plain best-first search without the two search enhancements (preferred operators and stop search). The coverage data resulting from this – overall, 778 for $h^{\text{RB}}_{\text{repair}}$ vs. 903 for $h^{\text{RB}}_{\text{follow}}$ – impressively confirms the advantage of red facts following over relaxed plan repair. This drastic advantage gets watered down when turning on the search enhancements, as these can yield substantial improvements even where the underlying heuristic has low quality (e. g., in VisitAll, see also next).

The "Average $h(I)$" columns illuminate the difference between $h^{\text{RB}}_{\text{repair}}$ and $h^{\text{RB}}_{\text{follow}}$ in terms of their plan length estimation in the initial state. As is evident, $h^{\text{RB}}_{\text{follow}}$ *does* yield much shorter red-black plans than $h^{\text{RB}}_{\text{repair}}$ in many domains. Consistently across all domains, $h^{\text{RB}}_{\text{follow}}$ never yields larger average red-black plan length. Consider VisitAll, where the behavior is most extreme. $h^{\text{RB}}_{\text{repair}}$ overestimates dramatically so is not useful in search, in contrast to $h^{\text{RB}}_{\text{follow}}$, cf. our discussion of over-estimation in this domain (Sections 5.1 and 5.2) and the coverage data for **FDL** and **RDL**. Stop search fixes this issue in **FDLS**, but the plans correspond to the "Average $h(I)$" column so are of extremely poor quality. Note that shorter red-black plans (even if they are not FDR-executable) are always better in the sense that, ideally, we would like our heuristic to return $h^{\text{RB}*}$: any value larger than this, even if it happens to be closer to h^* than $h^{\text{RB}*}$, is not justified by the red-black relaxation and must be attributed to arbitrary phenomena in the practical heuristic, as opposed to systematic estimates of goal distance.

Consider now the right half of Table 1. For coverage, the **RDLS** column is identical to that in the left half of the table, we repeat it here just for ease of reading. As can be seen, the impact of stop search in terms of coverage is small. Without our FDR-executability enhancements, i. e., for **RDLS-**, coverage is identical to **RDL** except in Driverlog, Sokoban11, and VisitAll, in each of which **RDLS-** solves a single instance more. With the executability enhancements, coverage gets worse in Sokoban but a bit better in Floortile and a lot better in Transport, resulting in an overall "net win" for **RDLS** by a small margin. In other words, when using $h^{\text{RB}}_{\text{follow}}$ with DAG causal graphs, coverage is already too high to allow to discriminate between stop search vs. no stop search (we show data for some larger, non-IPC, test instances below in Table 2). The more fine-grained data regarding the number of evaluated states goes to show that, actually, search space size decreases substantially across a range of IPC domains and IPC test instances: 9 of our 29 domains here, not counting IPC'08/IPC'11 duplicates. In 3 of these 9 domains (NoMystery, Transport, and Zenotravel), **RDLS** has a significant advantage over **RDLS-**. As the "Solved in I" data shows, in most (though not all) of these 9 domains, stop search always fires on the initial state. Consistently across all the domains where it sometimes fires on the initial state, stop search returns plans of very similar quality as would be returned by the search itself.

	IPC instances				Extended instances					
		Coverage		Evaluations		Coverage			Evaluations	
Domain	#	**RELS**	**RDLS**	**RELS/RDLS**	#	**RELS**	**RDLS**	**RDL**	**RELS/RDLS**	**RDL/RDLS**
Barman	20	20	20	1.0						
Driverlog	20	20	20	1.0						
Elevators08	30	30	30	520.5	100	39	100	31	11880	6044
Elevators11	20	19	20	9435.0						
Gripper	20	20	20	369.5	100	11	100	54	70963	2449
Rovers	29	29	29	1.0						
Tidybot	20	13	13	1.0						
Transport08	30	28	30	633.5	100	31	100	26	13149	5748.5
Transport11	20	12	20	5458.0						
Trucks	30	21	18	0.6						
$\sum$	239	212	220		300	81	300	111		

Table 2: Data analyzing (iii) the advantage of using DAG black causal graphs (**D**) over using arcless black causal graphs (**E**). The "extended instances" in the right half of the table were generated by choosing very large parameters (e. g., up to 3942 balls in Gripper), to demonstrate the advantage of DAG causal graphs in extreme situations where stop search (**S**) succeeds on the initial state. Ratios are per-domain median over those instances commonly solved by the pair of planners involved.

Consider finally the issue (iii) of DAG black causal graphs (**D**) vs. arcless black causal graphs (**E**), Table 2. The table is much smaller because, for this comparison, we need to consider only those IPC instances *whose causal graph has at least one directed arc* (v, v') *between RSE-invertible variables* v *and* v'*, with no backwards arc* (v', v). These are exactly the tasks for which there exists a choice of black variables so that (a) the resulting red-black planning task is inside our tractable fragment, and (b) the black causal graph is a non-arcless DAG. Table 2 contains only these tasks. In other tasks, each of our **D** configurations simplifies exactly to the corresponding **E** configuration.

The "main" data for this part of our evaluation, considering IPC benchmark instances, is in the left half of Table 2. In terms of coverage, the richer structure underlying the DAG heuristic pays off mainly in Transport, and a little bit in Elevators; it leads to somewhat worse performance in Trucks. The evaluations data illuminates the advantage in terms of search space size, which extends also to the Gripper domain. This advantage is due to stop search, which in the three domains in question – Elevators, Gripper, Transport – always fires on the initial state (cf. our discussion of Table 1 above). To shed some more light on this, the right half of Table 2 considers extreme cases in these domains, with instances much larger than those used in the IPC. Stop search still fires immediately, scaling to almost arbitrary instance sizes which are completely unfeasible for search, i. e., when either using the weaker **E** relaxation (**RELS**) where stop search does not fire, or when switching stop search off (**RDL**). As the number of evaluations for **RDLS** is constant 1, the evaluations data here also shows us that, even without stop search, the DAG heuristic has an advantage over the arcless one.

We remark that Table 2 is the only point here where the particular painting strategy **L** we chose for this presentation *does* make a substantial difference. Stop search for the DAG heuristic fires on the initial states of Elevators and Transport for about half of our painting strategies (including **L**). Using one of the painting strategies from the "bad half" does not affect coverage in the IPC instances, but it does affect coverage on our extended instances, and it of course affects the number of state evaluations in those two domains (see also Table 4 in the next section).

6. Painting Strategies

As yet, we have not specified how to automatically choose which variables to paint red, and which variables to paint black. We refer to strategies for making that choice as *painting strategies*. We now introduce a family of such strategies and examine their behavior. We start in Section 6.1 by making a few basic observations and describing our strategies. We evaluate the strategies in Section 6.2, drawing the high-level conclusion that *"the performance differences between different painting strategies are typically minor, and sometimes brittle with respect to small changes"*. To shed additional light on this, we also examine the behavior of random painting strategies. Similarly as we did for the red-black planning variants in Section 5, we make a selection of "winning strategies" and keep these fixed for the rest of the paper.

6.1. Painting Strategy Design

The first observation to be made when designing painting strategies is that there actually are two kinds of variables that can be immediately excluded from those that might end up being painted black. The first condition is obvious and was already discussed beforehand: As our tractable fragment requires all black variables to be RSE-invertible, variables that are *not* RSE-invertible can immediately be painted red. The second condition is slightly less obvious:

Theorem 14 *Let* Π *be an* FDR *planning task and let* V^{R} *be a subset of its variables. If all variables in* V^{R} *have no outgoing arcs in* CG_Π*, then* $h^{\mathit{RB}*}_{V^{\mathsf{R}}}$ *is perfect.*

Proof: It clearly suffices to show that, in this setting, all non-redundant red-black relaxed plans, i. e., red-black relaxed plans that do not contain any superfluous actions, are valid plans for the original FDR task Π. Which is true because the leaf variables v in CG_Π are neither (1) used to support value changes of any other variables, nor (2) affected as a side effect of changing any other variables. Due to (1), any non-redundant red-black relaxed plan either changes v along a simple (acyclic) path from v's initial value to its goal value, or leaves v untouched in case it has no goal. That same path will be executable within Π. Due to (2), such execution is not interfered with by any other actions in the red-black relaxed plan. This concludes the argument. □

In other words, causal graph leaves can be painted red without affecting the (ideal) red-black plan heuristic.

Our second observation regards paintings that are *maximal* in the sense that, if we paint one more variable black, then the black causal graph is no longer acyclic. Such paintings are always preferable over non-maximal ones, at least in theory, because $h^{\mathsf{RB}*}$ grows monotonically with the set of black variables, cf. Proposition 1. We therefore design our painting strategies in a way guaranteeing maximality. We get back to the practical implications of this below, after explaining our painting algorithm.

All our painting strategies proceed by iteratively painting variables red (starting from the set of variables identified above), until all cycles in the causal graph are broken. We then employ a simple post-processing step to ensure maximality. See Figure 8.

Note that this pseudo-code is used for configurations **D**, i. e., DAG black causal graphs, which we keep fixed for the rest of the paper. For configurations **E**, we simply test whether the black causal graph is arcless, as opposed to acyclic. The

```
Algorithm : PAINTING($\Pi$)
 main
  // $\Pi = \langle V, A, I, G \rangle$
  $V^{\mathsf{R}} \leftarrow \{v \mid v$ is not RSE-invertible or $v$ is a leaf vertex in $CG_\Pi\}$
  $list^R \leftarrow \langle\rangle$
  while the sub-graph of $CG_\Pi$ induced by $V \setminus V^{\mathsf{R}}$ is not acyclic
    do { Select $v \in V \setminus V^{\mathsf{R}}$  // choice point
         $V^{\mathsf{R}} \leftarrow V^{\mathsf{R}} \cup \{v\}$
         $list^R \leftarrow list^R \circ \langle v \rangle$
  // post-process ensuring maximality of the painting
  for $v$ in $list^R$ from back to front
    do { $V' \leftarrow V^{\mathsf{R}} \setminus \{v\}$
         if the sub-graph of $CG_\Pi$ induced by $V \setminus V'$ is acyclic
           then $V^{\mathsf{R}} \leftarrow V'$
  return $\Pi^{\mathsf{RB}}_{V^{\mathsf{R}}} = \langle V \setminus V^{\mathsf{R}}, V^{\mathsf{R}}, A, I, G \rangle$
```

Figure 8: The algorithm underlying all our painting strategies.

post-processing step checks, for each variable that was selected to be painted red, whether that variable can now be removed from the set of red variables (i. e., be painted black) without breaking acyclicity. The ordering of these checks is not important as far as the maximality guarantee is concerned. We order them from back to front because, in all our painting strategies, the "least important" variables are painted red first. Given this, the ordering from back to front gives a preference to removing the more important variables from V^{R}.

The post-process is required for maximality because a new variable v added into V^{R} might break all cycles concerning a previously-added variable v'. For example, in StarShapeLogistics (Figure 5), there is a cycle between every package $X \in \{A, B, C, D\}$ and the capacity variable F. If the algorithm selects, say, A, B, and C first, and only thereafter selects F, then painting A, B, and C red has become redundant. This will be recognized by the post-process.

We remark that, in our prior works on red-black plan heuristics [2, 3], we did not use the post-processing step because we overlooked the lack of maximality in this setup. Thus, we effectively employed painting strategies geared at finding maximal paintings, but not giving a guarantee. Now, while in theory the heuristic function can only get better with more black variables, in practice of course this is not as clear. As far as the IPC benchmarks are concerned, our modified (maximality-guaranteeing) painting strategies here give performance very similar to the previous ones, better in some domains, and never worse except for a minor performance loss in Trucks. We also performed limited experimentation with painting strategies geared at finding *non-maximal* paintings: these keep painting variables red even when the black causal graph is already acyclic, stopping at a randomly selected time point. It appears that maximal paintings tend to work better. But exploring this comprehensively is an open topic.

It remains to instantiate the choice point in Figure 8. We have devised a variety of methods for doing so. The common rationale behind all of these is the attempt to paint black as many variables as possible, and/or for the black variables to be as "important" as possible. The strategies differ in how they attempt to achieve these objectives:

- **L**: Selects v with highest level in the causal graph heuristic [12], i. e., "closest to the causal graph leaves". This aims at painting red the "client" variables, which do not tend to move back and forth to suit the needs of other variables.

- **A**: Select v with the maximal number $A(v)$ of incident arcs to black variables, i. e., to variables in $V \setminus V^{\mathsf{R}}$. The intuition behind this is to remove many arcs from the black causal graph quickly so that we minimize the number of red variables. We break ties by smaller variable domain size, and if ties still remain then, break them by the **L** strategy.

- **C**: Select v with the minimal number $C(v)$ of conflicts, i. e., relaxed plan actions with a precondition on v that will be violated when executing the relaxed plan with black v. The intuition is for the "least critical" variables to be red. We break ties by the **L** strategy.

- **C[**N**]**: Extends **C** by sampling N random states, then selecting v with the minimal average number of conflicts in the relaxed plans for the sample states. The motivation is that **C** depends on the relaxed plan for the initial state, in which conflicts (e. g., on resources) that will necessarily occur later on may not be visible. We run $N \in \{5, 25, 100\}$ in our experiments.

- **CA[**p**]**: Interpolation between **C** (with $p = 0$) and **A** (with $p = 1$). Selects v maximizing $P(v) := p * \hat{A}(v) + (1-p) * (1-\hat{C}(v))$, where $\hat{A}(v)$ and $\hat{C}(v)$ are normalized counters of the number of incident edges and the number of conflicts, respectively: $\hat{A}(v)$ is the number of incident edges of v divided by the maximal number of incident edges among all variables in $V \setminus V^{\mathsf{R}}$, and $\hat{C}(v)$ is the number of conflicts of v divided by the maximal number of conflicts among all variables in $V \setminus V^{\mathsf{R}}$.

A subtlety in **CA[p]** regards the tie breaking. Ties are broken differently in **A** and **C**, cf. above. In **CA[p]**, we adopt the tie breaking of **A**. Thus **CA[1]** is equivalent to **A**, but **CA[0]** is a variant of **C** using the tie breaking from **A**. We run $p \in \{0, 0.25, 0.5, 0.75\}$ in our experiments.

We also experiment with corresponding inverse strategies, as well as with random strategies, intended as sanity checks:

- $\overline{\mathbf{L}}$: Like **L** but selecting v with *lowest* level, i. e., "closest to the causal graph roots": What happens if we paint the "servant" variables red?
- $\overline{\mathbf{A}}$: Like **A** but selecting v with the *minimal* number $A(v)$ of incident arcs to black variables: What happens if we try to maximize the number of red variables?
- $\overline{\mathbf{C}}$: Like **C** but selecting v with the *maximal* number of conflicts: What happens if we paint the "most critical" variables red?
- **RND**: Selects v at random, uniformly from $V \setminus V^{\mathsf{R}}$.

6.2. Evaluation

Table 3 shows coverage data for all our painting strategies, with the canonical search algorithm as well as the fixed best-performing configuration parameters **RDS** as per Section 5. The message of this table is easiest to appreciate by observing the sparsity of boldface numbers, which indicate best per-domain coverage performance where there are differences: In all except Barman, Scanalyzer, Sokoban, and Tidybot, coverage is constant across all painting strategies, including the average of 10 runs of our random painting strategy **RND**. Indeed, as the "**RND** Std dev" column shows, the per-domain standard deviation in the distribution of random-painting coverage is 0 except in Barman, Scanalyzer, and Sokoban (IPC'11 only). In the next column to the right, "$|V_{\text{inv}}|$", we see that this is not for lack of RSE-invertible variables (i. e., candidates for being painted black): in most domains, there is quite a few of these. We elucidate this further in the rightmost column, where we indicate the number of maximal paintings per domain. We approximate that number through running **RND** 10000 times (without running the actual planner), and counting how many different paintings were generated. In most domains, this count is very small. Indeed, in the majority of the domains,

		Coverage															$\lvert V_{\text{inv}}\rvert$	$\lvert P\rvert$
			CA[p]					**C[N]**							**RND**			**RND** $10K$
Domain	#	**A**	0.75	0.5	0.25	0	**C**	5	25	100	**L**	$\overline{\textbf{A}}$	$\overline{\textbf{C}}$	$\overline{\textbf{L}}$	Average	Std dev	Average	Average
Barman	20	**20**	**20**	8	8	8	9	9	9	9	**20**	8	**20**	**20**	19.0	1.2	66.0	2.0
Blocksworld	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35.0	0.0	20.3	11.6
Depots	22	18	18	18	18	18	18	18	18	18	18	18	18	18	18.0	0.0	2.8	1.0
Driverlog	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20.0	0.0	18.4	2.0
Elevators08	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30.0	0.0	25.7	2.0
Elevators11	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20.0	0.0	50.3	2.0
Floortile	20	7	7	7	7	7	7	7	7	7	7	7	7	7	7.0	0.0	4.8	1.0
Grid	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5.0	0.0	12.0	12.0
Gripper	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20.0	0.0	3.0	1.0
Logistics98	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35.0	0.0	51.5	1.0
Logistics00	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28.0	0.0	13.4	1.0
Miconic	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150.0	0.0	1.0	1.0
Mprime	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35.0	0.0	21.1	2.0
Mystery	28	16	16	16	16	16	16	16	16	16	16	16	16	16	16.0	0.0	20.9	2.0
NoMystery	20	13	13	13	13	13	13	13	13	13	13	13	13	13	13.0	0.0	11.5	1.0
ParcPrinter08	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20.0	0.0	5.5	1.0
ParcPrinter11	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13.0	0.0	9.2	1.0
PegSol08	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30.0	0.0	7.9	15.0
PegSol11	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20.0	0.0	8.0	16.0
Pipesworld-NoTankage	40	33	33	33	33	33	33	33	33	33	33	33	33	33	33.0	0.0	22.5	1.8
Pipesworld-Tankage	40	30	30	30	30	30	30	30	30	30	30	30	30	30	30.0	0.0	15.0	2.9
PSR	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50.0	0.0	6.0	1.0
Rovers	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40.0	0.0	8.7	1.0
Satellite	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36.0	0.0	29.8	1079.4
Scanalyzer08	21	19	20	20	20	20	20	**21**	**21**	**21**	**21**	19	19	19	19.4	0.5	12.0	12.0
Scanalyzer11	14	12	13	13	13	13	13	**14**	**14**	**14**	**14**	12	12	12	12.2	0.4	13.6	13.6
Sokoban08	30	**27**	**27**	26	26	26	26	26	26	26	**27**	26	**27**	26	**27.0**	0.0	9.8	334.6
Sokoban11	20	**17**	**17**	16	16	16	16	16	16	16	**17**	16	**17**	16	16.9	0.3	13.1	500.5
Tidybot	20	13	13	13	13	13	**14**	**14**	**14**	13	13	13	13	13	13.0	0.0	102.6	9046.9
TPP	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30.0	0.0	4.3	1.0
Transport08	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30.0	0.0	17.6	2.0
Transport11	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20.0	0.0	27.3	2.0
Trucks	30	18	18	18	18	18	18	18	18	18	18	18	18	18	18.0	0.0	26.0	8.6
VisitAll	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20.0	0.0	1.0	1.0
Woodworking08	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30.0	0.0	9.0	2.0
Woodworking11	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20.0	0.0	14.2	2.0
Zenotravel	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20.0	0.0	13.2	1.0
$\sum$	1082	1000	1002	988	988	988	990	992	992	991	**1004**	986	1000	998	999.5			

Table 3: Coverage results for different painting strategies. Best values highlighted in boldface where there are differences. All configurations shown run FD's greedy best-first search with lazy evaluation and a second open list using preferred operators, and use **R** red facts following, **D** DAG black causal graphs, as well as **S** stop search. The configuration using the random painting strategy **RND** has been run 10 times. The $|V_{\text{inv}}|$ column shows the average number of RSE-invertible variables. The $|P|$ column shows the average number of different paintings generated in 10000 runs of **RND**.

the average is 1.0 or 2.0, and in all but one of these (Pipesworld-NoTankage) the count is constant 1 respectively 2 across all instances. This gives a very strong indication that the number of maximal paintings tends to be small, at least in the IPC benchmarks. As an example of how this happens, consider StarShapeLogistics (Figure 5). We can paint T and F black, or paint T and the packages black. All other paintings either do not yield a DAG black causal graph, or are not set-inclusion maximal among such paintings.

That said, as we see in Barman, already a very small number of paintings to choose from (2, in this case) can make a huge performance difference (20 vs. 8 instances solved). Furthermore, Table 3 gives too simple a picture, due to the

		Average number of evaluations															$\lvert P \rvert$
			CA[p]					**C**[N]							**RND**		
Dom	#	**A**	0.75	0.5	0.25	0	**C**	5	25	100	**L**	$\overline{\mathbf{A}}$	$\overline{\mathbf{C}}$	$\overline{\mathbf{L}}$	Avg	Std dev	Avg
Bar	8	**163839**	**163839**	1756579	1756579	1756579	1756579	1756579	1756579	1756579	**163839**	1756579	**163839**	**163839**	401635	205973	2
Blo	35	284425	284425	**318**	**318**	**318**	**318**	**318**	**318**	**318**	284425	284425	63744	284425	150687	110090	12
Dep	18	126356	126356	126356	126356	126356	126356	126356	126356	126356	126356	126356	126356	126356	126356	0	1
Dri	20	249242	249242	248775	248800	248800	248800	248981	249519	249519	245766	**432**	3000	**432**	198794	103143	2
Elv08	30	992	992	13	**1**	**1**	**1**	**1**	**1**	**1**	**1**	998	975	998	843	84	2
Elv11	20	4913	4913	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	4913	4913	4913	4131	484	2
Flo	7	4896204	4896204	4896204	4896204	4896204	4896204	4896204	4896204	4896204	4896204	4896204	4896204	4896204	4896204	0	1
Grd	5	**1886**	**1886**	**1886**	**1886**	**1886**	2021	2021	2021	2021	2021	**1886**	2021	**1886**	4432	4871	12
Grp	20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
Log98	35	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
Log00	28	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
Mic	150	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
Mpr	35	**33**	**33**	36	36	36	36	36	36	36	**33**	**33**	**33**	**33**	**33**	1	2
Mys	18	996622	996622	996751	996751	996751	996751	996751	996751	996751	**996610**	996622	996622	996622	996676	75	2
NoM	13	657	657	657	657	657	657	657	657	657	657	657	657	657	657	0	1
Prc08	20	192	192	192	192	192	192	192	192	192	192	192	192	192	192	0	1
Prc11	13	22732	22732	22732	22732	22732	22732	22732	22732	22732	22732	22732	22732	22732	22732	0	1
Peg08	30	32384	32419	32419	32419	32419	34568	**2797**	7785	796332	135069	32384	6893	32384	138939	253907	15
Peg11	20	48632	48690	48690	48690	48690	51832	**4181**	11628	1194534	202522	48632	10250	48632	310721	316455	16
PNT	33	**135939**	**135939**	135981	135981	135947	135989	135948	135946	135946	**135939**	135946	135947	135946	135943	19	2
PT	30	3706	3706	3632	3632	3632	3636	3632	**2233**	**2233**	3706	3632	3707	3632	3682	35	3
PSR	50	274	274	274	274	274	274	274	274	274	274	274	274	274	274	0	1
Rov	40	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384	0	1
Sat	36	70	70	12	**1**	**1**	**1**	**1**	**1**	**1**	70	**1**	61	**1**	51	11	1079
Sca08	19	51579	24207	24207	24207	24207	24207	3918	**154**	**154**	**154**	51579	55101	51579	40601	19750	12
Sca11	12	81639	38235	38235	38235	38235	38235	6184	**223**	**223**	**223**	81639	87209	81639	66465	38537	14
Sok08	26	600814	600814	**370087**	**370087**	**370087**	**370087**	**370087**	**370087**	**370178**	601278	**370087**	601098	**370087**	584072	79089	335
Sok11	16	959266	959266	**584652**	**584652**	**584652**	**584652**	**584652**	**584652**	584800	960145	**584652**	959853	**584652**	877611	176406	501
Tidy	11	1516	1516	2118	2118	2405	2404	1661	2342	2403	1887	2376	**1354**	2376	1512	447	9047
TPP	30	990	990	990	990	990	990	990	990	990	990	990	990	990	990	0	1
Tra08	30	589	589	583	296	**1**	**1**	**1**	**1**	**1**	**1**	590	539	590	451	54	2
Tra11	20	2035	2035	2035	1668	**1**	**1**	**1**	**1**	**1**	**1**	2035	2035	2035	1804	181	2
Tru	18	755013	755013	755013	755013	755013	755013	755013	755013	755013	755013	755013	755013	755013	755013	0	9
Vis	20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
Woo08	30	10760	10760	10760	10760	10760	10760	10760	10760	10760	10760	10760	10760	10760	10760	0	2
Woo11	20	18517	18517	18517	18517	18517	18517	18517	18517	18517	18517	18517	18517	18517	18517	0	2
Zen	20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
Avg	986	255492	253582	272435	272417	272371	272519	268942	269010	322300	258562	275474	241413	**232427**	263572		

Table 4: Search space size results for different painting strategies. Best values highlighted in boldface where there are differences. Lines separate domains with vs. without an actual choice of painting ($|P| > 1$ vs. $|P| = 1$). Configurations as in Table 3. "#" gives the number of instances commonly solved by all painting strategies shown (including the 10 runs of **RND**). All averages are taken over that instance set. The total number of these instances is 986 (out of 1082).

abstraction level implied by looking at coverage only. Do the 1000s of choices in Satellite really have no effect on performance? Or are we only observing the lack of instances at the borderline of feasibility for the group of planners and computational resources considered? Table 4 shows average search space size, to address these questions.

There can of course not be any performance differences between painting strategies in those domains where there is no actual choice ($|P| = 1$). We keep these in the table merely to point out that our painting strategies do not discover anything not discovered by the 1000 random runs underlying $|P|$, and to give explicit search space size data (as opposed to relative data such as for **RDL**/**RDLS** in Table 1) for these domains as well.

For the domains where different painting choices do exist, an interesting observation is that a small choice $|P|$ of different paintings available does not imply a small scale of performance differences in the domain, as a function of that choice. For a quick look, just compare column "$|P|$" with column "**RND** Std dev", and consider the extreme cases Barman and Driverlog ($|P| = 2$, standard deviation > 100000). There is of course also a long list of domains with both small $|P|$ and little performance variation (e. g., Mystery, Mprime, Woodworking), but then again there are cases like Satellite and Tidybot ($|P| > 1000$, standard deviation < 500) where a huge amount of choice results in comparatively little performance variation. Altogether, the best conclusion we can draw from this is that the performance of different painting strategies is rather unpredictable, and in some domains is extremely brittle even when there is little choice to be made.[16]

Considering our sanity test strategies $\overline{\mathbf{A}}$, $\overline{\mathbf{C}}$, and $\overline{\mathbf{L}}$, the only "expected" observation is that, in terms of coverage, $\overline{\mathbf{A}}$ and $\overline{\mathbf{L}}$ are indeed consistently dominated by their more intuitive counterparts **A** and **L**. However, in coverage, $\overline{\mathbf{C}}$ almost consistently dominates (namely in all domains except Tidybot) its counterpart **C**, and in terms of search space size almost all of the strategies are pairwise complementary, with better or worse performance depending on the domain. Ironically, $\overline{\mathbf{L}}$ ends up doing best when averaging the per-domain average search space size across all domains (see next).

In terms of competitive performance evaluation, our primary intention for including the "global average" row at the bottom of Table 4 is to point out that, overall, the differences are minor. The more interesting parameter to look at is best performance per domain, where dramatic differences do exist. With this in mind, our selection of strategies to focus on in the remainder of the paper is **L**, $\overline{\mathbf{L}}$, and **C**[5]: In 18 out of the 20 domains where performance differences do exist, these three configurations represent the best performance observed.[17] Furthermore, **L** and **C**[5] together represent all best performances in terms of coverage, **L** is best in terms of overall coverage, and $\overline{\mathbf{L}}$ is best in terms of the global average for evaluations.

Note that we restrict to a selection of just three strategies here to make the re-

[16]One particular note here regards domains with search space size average "1". This means that, as previously discussed in the context of Table 1, stop search always fires on the initial state. As we can see in Table 4, in domains with more than one choice of painting – Elevators, Satellite, Transport – this ability of stop search hinges on the painting strategy.

[17]The only exceptions are Pipesworld-Tankage, where **C**[25] and **C**[100] do marginally better (2233 vs. 3632 evaluations on average), and Tidybot, where $\overline{\mathbf{C}}$ does marginally better (1354 vs. 1661).

maining experiments more feasible and accessible. Given the observations of unpredictable cross-strategy variance in Table 4, it could make sense to design methods trying different painting strategies on a domain, e. g., employing re-starting or auto-tuning. We leave this as an open topic for future work.

7. Comparison to the State of the Art

To conclude the empirical evaluation of our work, we now compare its most competitive configurations against the state of the art. Section 7.1 examines the main performance parameters against related heuristic functions and a representation of the state of the art in satisficing planning. Section 7.2 analyzes simple methods for combining our new methods with existing ones, highlighting the mutual benefits. Both of these experiments maintain the benchmarks from the previous experiments, i. e., the FD collection of satisficing-planning STRIPS benchmarks up to IPC 2011. In Section 7.3 we provide an additional brief evaluation on the IPC 2014 benchmarks, in which one of our techniques, implemented in the "Mercury" planner, participated successfully.

7.1. Competitive Performance

As before, we exclude benchmarks without RSE-invertible variables, and we use FD's greedy best-first search with lazy evaluation and a second open list using preferred operators [29, 46] as our base search algorithm. Into that search algorithm, we plug:

- h^{FF}, as a baseline;
- h^{cea}, as a competitive heuristic function from earlier work on (not-interpolating) partial delete relaxation heuristics;
- two versions of $h^{\text{FF}}(\Pi^C_{\mathsf{ce}})$, namely the configurations performing best (overall) in the original experiments [25] (referred to as "$\Pi^C_{\mathsf{ce}}1$") respectively the more recent experiments [26] (referred to as "$\Pi^C_{\mathsf{ce}}2$"), representing the only competing interpolation framework for partial delete relaxation heuristics.

We run the first search iteration of LAMA [11] which we refer to as "LAMA-1st" (short: "LA-1"), as a representation of the state of the art in "agile" satisficing planning, i. e., focusing on runtime performance without investing any extra effort into optimizing plan quality. Exclusively for a comparison regarding plan quality, i. e., the length of the plans returned (recall that we ignore action costs), we also

		Coverage									Total Time									
												LAMA-1st/								
												h^{FF}	h^{cea}	$\Pi^C_{\text{ce}}1$	$\Pi^C_{\text{ce}}2$	**RD$\overline{\text{L}}$S**	**RDC[5]S**	**RDLS**		
Domain	#	Mp	h^{FF}	h^{cea}	$\Pi^C_{\text{ce}}1$	$\Pi^C_{\text{ce}}2$	LA-1	**RD$\overline{\text{L}}$S**	**RDC[5]S**	**RDLS**	#			Med				Min	Med	Max
Bar	20	0	**20**	0	15	18	**20**	**20**	9	**20**	0									
Blo	35	34	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	4	2.9	**3.1**	0.7	0.8	0.0	2.1	0.0	0.0	1.1
Dep	22	**21**	18	18	**21**	**21**	**21**	18	18	18	14	**0.9**	0.3	1.0	1.8	0.8	0.8	0.0	0.8	2.3
Dri	20	16	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	10	1.6	1.8	0.8	1.5	**2.2**	1.6	0.0	1.7	12.4
Elv08	30	14	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	17	1.2	1.6	0.1	0.1	0.6	3.8	1.6	**4.3**	11.6
Elv11	20	1	19	**20**	**20**	**20**	**20**	**20**	**20**	**20**	19	1.3	2.4	0.1	0.1	0.6	11.1	1.6	**11.9**	55.2
Flo	20	16	6	6	**20**	**20**	5	7	7	7	5	1.5	30.0	**351.5**	**351.5**	2.8	2.9	0.5	2.9	61.5
Grd	5	4	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	5	**0.6**	0.5	0.3	0.3	0.4	0.5	0.3	0.5	0.9
Grp	20	14	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	0									
Log98	35	20	33	**35**	**35**	**35**	**35**	**35**	**35**	**35**	20	1.5	2.6	0.1	0.1	3.3	3.3	0.9	**3.4**	24.7
Log00	28	28	28	28	28	28	28	28	28	28	0									
Mic	150	132	**150**	**150**	**150**	**150**	**150**	**150**	**150**	**150**	20	**1.3**	**1.3**	0.3	0.4	**1.3**	**1.3**	1.1	**1.3**	1.6
Mpr	35	31	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	22	0.8	**1.0**	0.4	0.3	0.5	0.6	0.2	0.5	1.2
Mys	28	17	16	**19**	**19**	**19**	**19**	16	16	16	6	0.8	**0.9**	0.3	0.3	0.4	0.7	0.1	0.6	0.9
NoM	20	2	9	7	6	6	**13**	**13**	**13**	**13**	1	2.7	**4.0**	2.3	0.6	1.5	1.5	1.5	1.5	1.5
Prc08	20	**20**	**20**	16	12	11	**20**	**20**	**20**	**20**	0									
Prc11	13	**13**	**13**	5	2	1	**13**	**13**	**13**	**13**	0									
Peg08	30	30	30	30	30	30	30	30	30	30	8	2.5	**7.7**	5.9	7.2	3.4	5.8	0.0	5.4	3826.4
Peg11	20	20	20	20	20	20	20	20	20	20	8	2.5	**7.7**	6.3	7.3	3.4	5.8	0.0	5.4	3829.6
PNT	40	32	33	30	31	32	**34**	33	33	33	20	**3.7**	2.8	2.2	2.6	2.7	2.7	0.0	2.7	48.1
PT	40	20	29	23	30	31	**33**	30	30	30	17	**1.4**	0.6	0.6	1.3	1.2	1.3	0.5	0.8	25.6
PSR	50	48	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	2	1.6	**1.7**	0.6	0.8	1.0	1.0	0.7	0.9	1.2
Rov	40	28	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**	19	**2.2**	1.6	0.1	0.1	0.8	0.8	0.5	0.8	1.5
Sat	36	23	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	22	2.6	4.0	0.1	0.2	**9.4**	**9.4**	1.0	9.2	99.7
Sca08	21	17	19	**21**	20	**21**	**21**	19	**21**	**21**	9	0.3	0.8	0.1	0.2	0.9	1.0	0.7	**1.1**	2.7
Sca11	14	11	12	**14**	13	**14**	**14**	12	**14**	**14**	8	0.3	0.8	0.1	0.2	0.7	**1.1**	0.7	**1.1**	1.7
Sok08	30	7	**29**	5	27	27	**29**	26	26	27	5	**1.5**	0.3	0.5	1.2	0.7	0.7	0.4	0.8	0.9
Sok11	20	2	**19**	3	17	17	**19**	16	16	17	3	**1.9**	0.1	1.1	1.5	0.9	0.9	0.4	0.8	0.9
Tidy	20	7	14	**16**	15	14	**16**	13	14	13	9	0.8	**1.1**	0.4	0.9	0.5	0.5	0.5	0.6	0.9
TPP	30	13	**30**	28	**30**	**30**	**30**	**30**	**30**	**30**	14	**1.7**	0.1	0.1	0.1	1.0	1.0	0.5	0.9	1.4
Tra08	30	11	28	29	26	27	**30**	**30**	**30**	**30**	16	1.5	1.2	0.1	0.2	1.0	**3.1**	0.8	**3.1**	13.2
Tra11	20	0	11	17	9	10	17	**20**	**20**	**20**	8	2.0	2.4	0.2	0.1	1.0	**6.5**	3.0	6.3	47.7
Tru	30	8	**18**	15	14	14	15	**18**	**18**	**18**	8	**13.2**	4.0	1.2	1.0	4.5	4.6	0.5	4.5	16.9
Vis	20	0	3	3	**20**	**20**	**20**	**20**	**20**	**20**	2	0.2	0.1	0.3	0.5	**3.9**	**3.9**	2.2	**3.9**	5.6
Woo08	30	10	**30**	22	**30**	**30**	**30**	**30**	**30**	**30**	16	0.3	**0.9**	0.3	0.4	0.2	0.2	0.1	0.2	2.8
Woo11	20	1	**20**	8	**20**	**20**	**20**	**20**	**20**	**20**	7	0.5	**0.7**	0.2	0.3	0.4	0.4	0.1	0.4	1.7
Zen	20	17	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	7	2.7	2.8	0.1	0.2	**3.0**	2.9	0.7	2.9	5.9
$\sum$	1082	688	968	879	971	977	**1013**	998	992	1004	351									

Table 5: Coverage and runtime performance in comparison to the state of the art. **Best values** highlighted in **boldface** where there are differences. The ratios (per-domain median "Med", minimum "Min", maximum "Max") are over the set of instances commonly solved by all planners in the right half of the table, excluding instances solved by all these planners in ≤ 0.1 seconds.

run LAMA in full (referred to simply as "LAMA"). Finally, to represent the recent competitive approach to satisficing planning via SAT, we run Rintanen's Mp solver [47]. Tables 5, 6, and 7 show the data.

Let us start with a quick look at coverage in Table 5. Mp lags far behind the heuristic search planners in terms of coverage already, so we will not consider it in the remainder of these experiments. h^{cea} overall performs substantially worse than h^{FF} (it has significant advantages in some domains but too many losses in others). $h^{\text{FF}}(\Pi^C_{\text{ce}})$ leads to an overall win over h^{FF}, but only a slight one, excelling mainly in Floortile. Our red-black plan heuristics substantially improve over h^{FF}. They almost never have worse coverage (the only exceptions are Sokoban, Barman for **RDC[5]S**, and Tidybot for **RDLS** and **RD$\overline{\text{L}}$S**). They have much better coverage in VisitAll, Transport, and NoMystery, as well as smaller improvements in a few

		Evaluations $h^{\text{FF}}/$						Plan length LAMA/							
Domain	# common	h^{cea}	$\Pi^C_{\text{ce}}1$	$\Pi^C_{\text{ce}}2$	**RD$\overline{\text{L}}$S**	**RDC[5]S**	**RDLS**	h^{FF}	h^{cea}	$\Pi^C_{\text{ce}}1$	$\Pi^C_{\text{ce}}2$	LAMA-1st	**RD$\overline{\text{L}}$S**	**RDC[5]S**	**RDLS**
Bar	0														
Blo	35	0.7	**2.9**	2.8	0.7	0.9	0.7	0.6	0.5	**1.0**	**1.0**	0.7	0.9	**1.0**	0.9
Dep	18	1.7	**2.4**	**2.4**	1.3	1.3	1.3	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Dri	20	1.0	2.6	2.2	**10.7**	1.2	1.3	0.7	0.7	**0.9**	0.8	0.7	0.8	0.8	0.8
Elv08	30	2.7	1.3	1.2	1.0	**570.5**	**570.5**	1.0	1.0	1.0	1.0	1.0	1.0	**1.1**	**1.1**
Elv11	19	3.4	1.4	1.0	1.4	**6594.0**	**6594.0**	1.0	1.0	1.0	1.0	1.0	1.0	**1.1**	**1.1**
Flo	5	38.3	10185.3	**17382.2**	3.9	3.9	3.9	0.9	**1.0**	0.9	0.9	**1.0**	0.9	0.9	0.9
Grd	5	0.4	1.3	**1.4**	1.1	1.1	1.1	0.8	0.8	0.8	0.7	0.7	0.8	0.8	0.8
Grp	20	3.7	0.7	0.9	**344.5**	**344.5**	**344.5**	0.8	0.8	0.9	0.8	**1.0**	0.8	0.8	0.8
Log98	33	3.4	2.9	2.7	**435.0**	**435.0**	**435.0**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Log00	28	1.6	1.5	1.5	**137.0**	**137.0**	**137.0**	**1.0**	0.9	0.9	0.9	**1.0**	**1.0**	**1.0**	**1.0**
Mic	150	1.4	1.2	1.2	**253.5**	**253.5**	**253.5**	0.9	0.9	0.9	0.9	0.9	**1.0**	**1.0**	**1.0**
Mpr	35	**2.4**	**2.4**	2.3	1.3	1.3	1.3	0.9	**1.0**	**1.0**	**1.0**	0.9	0.9	0.9	0.9
Mys	16	**1.6**	**1.6**	**1.6**	1.5	1.5	1.5	0.9	**1.0**	**1.0**	**1.0**	0.9	0.9	**1.0**	0.9
NoM	5	7.1	3.1	0.7	**710.0**	**710.0**	**710.0**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Prc08	10	0.6	0.7	0.4	**1.3**	**1.3**	**1.3**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Prc11	0														
Peg08	30	1.0	1.3	**1.4**	1.0	1.1	1.0	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Peg11	20	1.2	**2.4**	2.3	1.0	1.3	1.4	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
PNT	29	1.2	0.8	0.8	**1.5**	**1.5**	**1.5**	0.7	0.6	0.6	0.6	0.6	**0.8**	**0.8**	**0.8**
PT	22	0.5	1.0	1.1	**1.3**	**1.3**	1.1	0.8	0.7	0.7	0.7	**0.8**	**0.8**	**0.8**	**0.8**
PSR	50	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Rov	40	0.8	1.1	1.1	**1.4**	**1.4**	**1.4**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Sat	36	1.4	1.8	1.2	**303.5**	**303.5**	28.3	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.9	0.9	**1.0**
Sca08	18	2.3	2.8	2.7	2.4	2.6	**3.5**	1.0	1.0	**1.1**	**1.1**	1.0	1.0	**1.1**	**1.1**
Sca11	11	6.0	2.9	**13.8**	3.0	2.8	11.5	1.0	1.0	**1.1**	**1.1**	1.0	1.0	**1.1**	**1.1**
Sok08	5	0.6	0.7	**1.5**	**1.5**	**1.5**	1.2	**0.9**	0.8	0.6	0.8	0.7	0.8	0.8	0.8
Sok11	3	0.2	1.0	**1.5**	**1.5**	**1.5**	1.2	**0.7**	0.5	0.5	0.6	0.6	**0.7**	**0.7**	0.6
Tidy	9	0.9	1.3	**1.9**	1.0	1.2	1.1	0.8	**0.9**	0.7	0.8	0.8	0.8	0.8	0.8
TPP	28	0.2	1.0	0.9	1.0	1.0	1.0	0.9	0.7	0.8	0.8	**0.9**	**0.9**	**0.9**	**0.9**
Tra08	25	1.6	1.2	1.2	1.4	**421.0**	**421.0**	0.9	1.0	1.0	0.9	1.0	1.0	**1.1**	**1.1**
Tra11	8	2.1	1.5	0.6	1.8	**2293.0**	2293.0	0.9	1.0	0.8	0.8	1.0	0.9	**1.3**	**1.3**
Tru	13	0.3	**1.3**	0.4	1.0	1.0	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Vis	2	1.6	25.0	29.1	**100723.5**	**100723.5**	**100723.5**	0.3	0.1	0.5	0.5	0.9	**1.2**	**1.2**	**1.2**
Woo08	22	1.2	212.1	**214.8**	1.0	1.0	1.0	1.0	1.0	**1.2**	**1.2**	1.0	1.0	1.0	1.0
Woo11	8	1.1	**196.4**	**196.4**	1.0	1.0	1.0	1.0	0.9	**1.1**	1.1	1.0	1.0	1.0	1.0
Zen	20	1.2	1.2	1.2	**61.5**	**61.5**	**61.5**	0.9	0.9	0.9	0.9	0.9	1.0	1.0	1.0

Table 6: Search space size and plan length performance in comparison to the state of the art. **Best values** highlighted in **boldface** where there are differences. The ratios (per-domain median) are over the set of instances commonly solved by all planners in the table.

other domains. LAMA is still best overall, profiting from its use of two different heuristics as opposed to a single one like for all other heuristic search planners here. Our new heuristics beat LAMA's coverage (to small extents) in Floortile, Transport, and Trucks.

We consider runtime and plan length in terms of ratios vs. a representation of the state of the art, namely LAMA-1st respectively LAMA. We consider search space size, i. e. the number of state evaluations (calls to the heuristic function), in terms of ratios vs. the baseline h^{FF}. All these ratios, see Tables 5 and 6, are taken over the set of instances commonly solved by all heuristic search planners in our experiments. So the ratios are directly comparable across columns. In some domains, due to the smaller coverage of h^{FF}, h^{cea}, and $h^{\text{FF}}(\Pi^C_{\text{ce}})$, the set of common instances is small. For these domains, Table 7 shows data over the larger set of instances commonly solved by the most competitive planners, i. e., LAMA and the three variants of our new heuristics (we do not show evaluations data here as

	Total Time						Plan Length				
		LAMA-1st/									
		RD$\overline{\text{L}}$S	**RDC[5]S**	**RDLS**				LAMA/			
Domain	#	Med		Min	Med	Max	#	LAMA-1st	**RD$\overline{\text{L}}$S**	**RDC[5]S**	**RDLS**
Barman	9	*0.7*	*0.0*	*0.0*	*0.7*	**1.3**	9	1.0	1.0	1.0	1.0
Elevators11	20	*0.6*	**14.8**	**1.6**	**15.7**	**55.2**	20	1.0	1.0	**1.1**	**1.1**
Logistics98	23	**3.3**	**3.3**	*0.6*	**3.4**	**24.7**	35	1.0	1.0	1.0	1.0
NoMystery	8	**1.6**	**1.7**	*0.2*	**1.6**	**25.7**	11	1.0	1.0	1.0	1.0
ParcPrinter08	2	**1.3**	**1.3**	**1.3**	**1.3**	**1.4**	20	1.0	1.0	1.0	1.0
ParcPrinter11	4	*0.7*	*0.7*	*0.0*	*0.7*	**1.6**	13	1.0	1.0	1.0	1.0
Pipesworld-NoTankage	25	**2.7**	**2.6**	*0.0*	**2.7**	**48.1**	33	*0.6*	*0.8*	*0.8*	*0.8*
Pipesworld-Tankage	25	**1.4**	**1.4**	*0.5*	**1.2**	**528.7**	30	*0.7*	*0.8*	*0.8*	*0.8*
Scanalyzer08	11	*0.7*	1.0	*0.3*	**1.1**	**2.7**	19	1.0	1.0	**1.2**	**1.2**
Scanalyzer11	9	*0.7*	**1.1**	*0.7*	**1.2**	**2.7**	12	1.0	1.0	**1.1**	**1.1**
Sokoban08	23	*0.7*	*0.7*	*0.2*	*0.8*	**11.9**	26	*0.6*	*0.7*	*0.7*	*0.6*
Sokoban11	16	*0.7*	*0.7*	*0.2*	*0.9*	**11.4**	16	*0.6*	*0.6*	*0.6*	*0.6*
Tidybot	11	*0.5*	*0.5*	*0.1*	*0.5*	*0.9*	11	*0.8*	*0.8*	*0.8*	*0.8*
TPP	17	1.0	1.0	*0.5*	*0.9*	**1.5**	30	*0.9*	*0.9*	*0.9*	*0.9*
Transport08	21	**1.2**	**4.0**	*0.8*	**4.0**	**249.4**	30	1.0	1.0	**1.1**	**1.1**
Transport11	17	**2.4**	**14.3**	**3.0**	**14.3**	**316.7**	17	1.0	*0.9*	**1.3**	**1.3**
Trucks	9	**5.2**	**5.4**	*0.5*	**5.2**	**16.9**	14	*0.9*	*0.9*	*0.9*	*0.9*
VisitAll	20	**3.0**	**3.1**	**1.5**	**3.1**	**5.6**	20	1.0	**1.3**	**1.3**	**1.3**
Woodworking08	25	*0.6*	*0.5*	*0.0*	*0.6*	**6.6**	30	1.0	1.0	1.0	1.0
Woodworking11	19	*0.4*	*0.4*	*0.1*	*0.4*	**7.1**	20	1.0	1.0	1.0	1.0

Table 7: Supplementary data for Tables 5 and 6. Performance **better than LAMA** highlighted in **boldface**, performance *worse than LAMA* highlighted in *italics*. We give the same measures of total time respectively plan length, over the set of instances (total number: 973) commonly solved by the most competitive planners, i. e., LAMA and the three variants using our heuristic. We show only the domains where that set of instances is larger than the set of commonly solved instances in Table 5 (right half) and Table 6. As in Table 5, for runtime we exclude instances solved by all involved planners in ≤ 0.1 seconds.

the ratio to h^{FF} is not defined on this instance set).

Regarding search time, at a high level the picture is somewhat similar to what we observed for coverage. Considering Table 5, we see advantages of h^{cea} over LAMA in some domains, and we see the enormous advantage of $h^{\text{FF}}(\Pi^C_{\mathsf{ce}})$ in Floortile. Our new heuristics yield significantly better runtime than all other planners in Elevators, Satellite, Transport, and VisitAll.

To discuss this in some more detail, consider **RDLS**, the most competitive variant of our heuristics in terms of coverage. In the median runtime ratio, **RDLS** does worse than LAMA-1st in 14 domains (i. e., instance suites), and better in 18. Table 7 complies with these observations, **RDLS** being worse than LAMA-1st in 8 of the shown instance suites, and better in 12. To shed some light on the distribution of ratios, we also include the per-domain minimum and maximum in Tables 5 and 7. This data mainly goes to show that there is a lot of per-domain variance, with very good and very bad behavior mixed in many domains. In Table 5, there are only 5 instance suites (Table 7: 1 suit) where **RDLS** is consistently worse, maximum < 1; and 6 instance suites (Table 7: 4 suites) where **RDLS** is consistently better, minimum > 1. The "consistently better" cases arise in Elevators, Miconic, NoMystery, ParcPrinter, Transport, and VisitAll. In the remaining domains – with ratios on both sides of 1 – we see that the median typically is indicative of whether or not large improvements occur: For median ratios close to 1, the maximum typically is below 3. There are some exceptions to this, most notably Driverlog in Table 5 as well as NoMystery, Pipesworld-Tankage, and Sokoban in Table 7. And there are domains like PegSol where the intra-domain variance is huge (the ratios > 3800 are extreme outliers, the next highest ratio is < 20 in each of these suites). Overall, it is not clear to us what causes this variance, nor how one could get rid of it. A simple promising approach would be to run several different heuristics in parallel. We leave this open for future research.

The data regarding the number of evaluations in Table 6 shows that the domains basically fall into two classes: a large class of domains where the informativity gain over h^{FF} is smallish and roughly similar for all the partial delete relaxation heuristics considered here; and a smaller class of domains where at least one of these heuristics yields dramatic gains. The latter is the case for $h^{\mathrm{FF}}(\Pi^C_{\mathsf{ce}})$ in Floortile and Woodworking, and for our new heuristics in Elevators, Gripper, Logistics, Miconic, NoMystery, Satellite, Transport, VisitAll, and Zenotravel (which almost coincides with the domains with a consistent runtime advantage, cf. above). Note that these latter advantages are mostly due to stop search (compare Table 1).

Regarding plan length, the right half of Table 6, as well as that of Table 7, show that LAMA has a very consistent advantage over all other planners, which is expected, as none of these planners make any effort to minimize plan length after the first solution is found. But LAMA's advantage tends to be small. There are few cases of median ratios worse than 0.8; the largest advantage by far arises in VisitAll for h^{FF} and h^{cea}. For our new heuristics, median ratios worse than 0.8 occur only in Sokoban. In a few cases (Elevators, Scanalyzer, Transport, and VisitAll) the plans found using our heuristics are somewhat shorter than those found by LAMA. Note that, in most of these latter cases, the plans are actually found by stop search, showing that this technique can be useful not only for runtime but also for plan length.

7.2. Simple Combinations

We so far ran our new heuristics *against* competing state-of-the-art techniques, but of course one can instead *combine* all these techniques to exploit their complementary strengths. We explore two straightforward forms of such combination, showing the potential benefits. For simplicity, we consider only the overall strongest variant of our heuristic function, **RDLS**, and we consider only its combinations with LAMA.

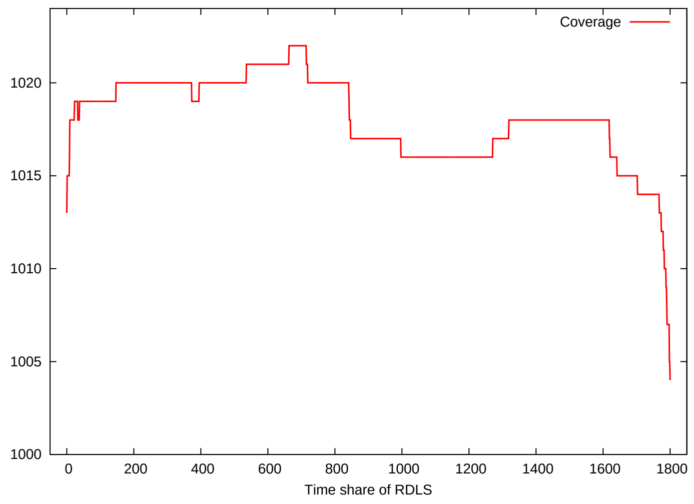


Figure 9: Coverage as a function of the **RDLS** time share T in a sequential portfolio of **RDLS** and LAMA. T is scaled from 0 to 1800 seconds, in steps of 1.

The simplest form of combination is to replace h^{FF} in LAMA by **RDLS**. We refer to this approach by the name "Mercury", as used for a corresponding planner that participated in IPC 2014 (see also the next section). We remark that, while LAMA implements an interaction ("synergy") between h^{FF} and its landmark heuristic, and while that interaction could presumably be adapted to our red-black plan heuristics, we did not perform this adaptation yet. That is, in Mercury, the two heuristics are strictly separate.

Our second combination method is the classical, and highly successful, *sequential portfolio* idea, where the component planners are run in a fixed sequence, each with a fixed *time share* of the total time (1800 seconds, in our case). We take the two portfolio components to be **RDLS** and LAMA. We denote the time share of **RDLS** (in seconds) by T; the time share of LAMA is $1800 - T$. We do implement a basic form of interaction between these components: If **RDLS** succeeded in solving the task at hand (within its time share), but we are trying to minimize plan length, then the length of the plan found by **RDLS** is provided as an initial upper bound on plan length to LAMA.

To clarify the influence of the parameter T, Figure 9 shows portfolio coverage as a function of that parameter. We obtained this data by running each of **RDLS** and **LAMA** once, reconstructing, for each of the 1801 values of T, the coverage the portfolio *would* have had for that setting. Notice first of all that the y-axis shows only the part of the coverage scale above 1000 instances – the differences

		Coverage				Plan length			
							LAMA/		
Domain	#	LAMA-1st	Mercury-1st	P-Unif	P-Best	#	Mercury	P-Unif	P-Best
Barman	20	20	20	20	20	20	1.0	*0.9*	*0.9*
Blocksworld	35	35	35	35	35	35	1.0	1.0	1.0
Depots	22	21	21	21	21	20	1.0	1.0	1.0
Driverlog	20	20	20	20	20	20	1.0	1.0	1.0
Elevators08	30	30	30	30	30	30	**1.1**	**1.1**	**1.1**
Elevators11	20	20	20	20	20	20	**1.1**	**1.1**	**1.1**
Floortile	20	5	5	**7**	**7**	4	*0.9*	*0.9*	*0.9*
Grid	5	5	5	5	5	5	1.0	1.0	1.0
Gripper	20	20	20	20	20	20	1.0	1.0	1.0
Logistics98	35	35	35	35	35	35	1.0	1.0	1.0
Logistics00	28	28	28	28	28	28	1.0	1.0	1.0
Miconic	150	150	150	150	150	150	1.0	1.0	1.0
Mprime	35	35	35	35	35	35	1.0	1.0	1.0
Mystery	28	19	19	19	19	19	1.0	1.0	1.0
NoMystery	20	13	**14**	**15**	**15**	12	1.0	1.0	1.0
ParcPrinter08	20	20	20	20	20	20	1.0	1.0	1.0
ParcPrinter11	13	13	13	13	13	13	1.0	1.0	1.0
PegSol08	30	30	*29*	30	30	29	1.0	1.0	1.0
PegSol11	20	20	*19*	20	20	19	1.0	1.0	1.0
Pipesworld-NoTankage	40	34	34	34	34	34	1.0	1.0	1.0
Pipesworld-Tankage	40	33	*32*	*31*	*32*	31	1.0	1.0	1.0
PSR	50	50	50	50	50	50	1.0	1.0	1.0
Rovers	40	40	40	40	40	40	1.0	1.0	1.0
Satellite	36	36	36	36	36	36	1.0	1.0	1.0
Scanalyzer08	21	21	21	21	21	21	1.0	**1.1**	**1.1**
Scanalyzer11	14	14	14	14	14	14	1.0	**1.1**	**1.1**
Sokoban08	30	29	*26*	*27*	29	26	1.0	1.0	1.0
Sokoban11	20	19	*16*	*17*	19	16	1.0	1.0	1.0
Tidybot	20	16	*14*	16	16	14	1.0	1.0	1.0
TPP	30	30	30	30	30	30	1.0	1.0	1.0
Transport08	30	30	30	30	30	30	**1.1**	**1.1**	**1.1**
Transport11	20	17	**20**	**20**	**20**	17	**1.3**	**1.3**	**1.3**
Trucks	30	15	**18**	**18**	**18**	14	1.0	1.0	1.0
VisitAll	20	20	20	20	20	20	**1.3**	**1.3**	**1.3**
Woodworking08	30	30	30	30	30	30	1.0	1.0	1.0
Woodworking11	20	20	20	20	20	20	*0.9*	1.0	1.0
Zenotravel	20	20	20	20	20	20	1.0	1.0	1.0
$\sum$	1082	1013	1009	**1017**	**1022**	997			

Table 8: Performance for simple combinations of our new heuristics with LAMA. Performance **better than LAMA** highlighted in **boldface**, performance *worse than LAMA* highlighted in *italics*. "Mercury" is like LAMA but using the **RDLS** heuristic in place of h^{FF} (see text). "P-Unif" is the sequential portfolio of LAMA and **RDLS** where each gets 900 seconds; "P-Best" uses a time share of $T = 714$ seconds for **RDLS**, which is best in terms of overall coverage (compare Figure 9). Ratios (per-domain median) are over those instances commonly solved by all planners in the table.

in overall coverage are small (as expected, given Table 5). The sharp drops at each extreme end of the T scale are due to losing coverage in domains where either **RDLS** (T close to 0) or LAMA (T close to 1800) excels, solving tasks within a few seconds that are out of reach for the respective other component planner. In between the extremes, the coverage curve is rather flat, with a performance peak of coverage 1022 for **RDLS** time shares $663 \leq T \leq 714$. The difference between the two extreme values of T in this peak is small; in what follows, we selected $T = 714$ as the "best" time share setting.

Consider Table 8. We do not show data for runtime because, on those instances commonly solved by LAMA and either of the two portfolios in the table, the vast majority of instances is solved by **RDLS** already. Hence the runtime ratio over these instances would essentially come down to the ratios already shown in Tables 5 and 7. Regarding plan length, as before we employ LAMA as a representation of the state of the art. The picture is clear: most of the time the median performance is identical to that of LAMA. There are only five cases (Barman for the uniform portfolio, Floortile for all our planners, Woodworking11 for Mercury) where plans become worse. In Elevators, Scanalyzer, Transport, and VisitAll, plans become shorter (similarly as in Table 6). Note also that the disadvantage on Sokoban when running **RDLS** alone (cf. Tables 6, and 7) disappears when using the portfolio instead, showing that the "plan-improvement post-process" by LAMA in that portfolio is effective.

Regarding coverage, the basic message is that replacing h^{FF} with **RDLS** in LAMA is marginally effective at best, while sequential portfolios of **RDLS** and LAMA do yield substantial improvements. In a little more detail, Mercury improves coverage over LAMA in NoMystery (1 instance), Transport (3), and Trucks (3), but loses in PegSol (2), Pipesworld-Tankage (1), Sokoban (6), and Tidybot (2), for an overall loss of 4. As for the portfolios, with $T = 714$ the portfolio is at least as good as LAMA in all except Pipesworld-Tankage where it loses a single instance, while it performs substantially better in 4 domains. In detail, coverage goes down in Pipesworld-Tankage (by 2 instances for $T = 900$ and by 1 instance for $T = 714$) and Sokoban (by 4 instances for $T = 900$), but goes up in Floortile (2 for each value of T), NoMystery (2 for each), Transport (3 for each), and Trucks (3 for each). Note that, in NoMystery, the portfolio solves more instances than each of its components, showing that complementary strengths can sometimes be exploited even within a domain.

7.3. IPC 2014

Mercury obtained the 2nd prize in the IPC 2014 sequential satisficing track, being outperformed (in terms of IPC quality score which was used to rank the planners in this track) only by the IBaCoP2 portfolio planner [48]. As stated by the organizers in their results presentation, LAMA would have ranked much worse, on place 12th out of 21 in that track. Given our observations above – Mercury having only a marginal performance advantage over LAMA, the portfolio approach being substantially more effective – this raises the questions (1) why Mercury was superior to LAMA in IPC 2014 (due to the different benchmarks, or due to the different ranking criterion?), and (2) how the portfolio approach would

Domain	#	Coverage LAMA	Coverage Mercury	Coverage P-Best	IPC Quality Score LAMA	IPC Quality Score Mercury	IPC Quality Score P-Best
Barman	20	19	20	18	18.22	17.88	17.66
CaveDiving	20	7	7	7	7.00	7.00	7.00
Childsnack	20	0	0	6	0.00	0.00	6.00
Citycar	20	3	5	3	3.00	4.56	3.00
Floortile	20	2	2	2	2.00	2.00	2.00
GED	20	20	20	20	18.18	19.29	15.38
Hiking	20	17	18	11	15.32	17.63	8.99
Maintenance	20	7	7	7	6.46	6.84	6.46
Openstacks	20	20	20	20	19.94	19.96	19.94
Parking	20	20	20	20	19.18	17.68	19.18
Tetris	20	9	17	11	8.43	15.81	9.79
Thoughtful	20	0	0	0	0.00	0.00	0.00
Transport	20	13	20	20	8.21	20.00	16.44
VisitAll	20	20	20	20	15.94	20.00	19.95
$\sum$	280	157	176	165	141.89	168.65	151.80

Table 9: Performance on IPC 2014 benchmarks, sequential satisficing track. IPC quality score is relative to the three planners shown here (not relative to the participants in that IPC track). All three planners use the same basic support for conditional effects, a straightforward "multiply out" compilation realized in FD, in the context of IPC 2014, by Gabriele Röger. "P-Best" is the same portfolio as in Table 8. If there are no RSE-invertible variables (which case previously was excluded up front), the portfolio defaults to LAMA, i. e., assigns the full 30 minutes runtime to LAMA; this happens in all instances of Citycar, Maintenance, Openstacks, and Parking, as well as in a single instance of Tetris. To stick as closely as possible to the IPC 2014 computational setup, the data for LAMA and P-Best were obtained on the original IPC 2014 machines with the original IPC 2014 settings; the data for Mercury is taken from the official IPC 2014 results (adding 4 additional tasks in CaveDiving, solved by Mercury, after a bug fix to the conditional effects compilation, quickly and with the same quality as for LAMA and the portfolio).

have fared. Consider Table 9.[18]

Considering question (1) first, the answer clearly is that Mercury's superiority over LAMA is due to coverage. Mercury never decreases coverage relative to LAMA. It dramatically improves coverage in Tetris and Transport, and yields smaller improvements in Barman, Citycar, and Hiking. In domains with equal coverage, the IPC quality score picture is mixed; Mercury has the edge in VisitAll and GED, but loses in Parking, and even in Barman despite its slight coverage advantage. In the light of our results above, the conclusion is that the IPC 2014 benchmarks, most specifically Tetris, are qualitatively different from the earlier IPC benchmarks, and that this qualitative difference leads to Mercury's advantage in coverage and hence its advantage in the official competition outcome.

Considering now question (2), a similar conclusion applies: In difference to the earlier IPC benchmarks used above, Mercury ends up having the edge in cover-

[18]In Thoughtful, all planners here fail due to a parsing issue.

age over the portfolio. Substantial differences occur in three new domains, Childsnack, Hiking, and Tetris. The portfolio is much more effective in Childsnack, but is much more ineffective in Hiking and Tetris. The portfolio also tends to do a bit worse than Mercury in the quality score, which makes sense as Mercury will plausibly have a tendency to spend more time trying to improve the initial solution (and IPC quality score is more sensitive to that than the median plan length ratios reported above). Overall, the portfolio still beats LAMA on these different benchmarks and ranking criterion, but not as convincingly as Mercury does.

8. Conclusion

Taking "some deletes into account" has been a challenge ever since delete relaxation heuristics were first invented. Red-black planning tackles this challenge by relaxing only a subset of the state variables, thus generalizing both regular and delete-relaxed planning, and allowing to smoothly interpolate between the two. We have provided a first complexity analysis and, focusing on a particular tractable fragment, have shown that practically useful heuristic functions can be derived this way.

In our view, the main virtue of the red-black planning framework lies in its elegance and simplicity. We believe that we have only scratched the surface of both, its analysis, and its potentially useful applications.

On the theory side, a major construction site is the structural analysis of dependencies *across black and red variables*. Can one restrict these in interesting ways to obtain tractability? In particular, as we have shown, just by considering the black causal graph, useful tractable fragments can be obtained. But what about the combined *red-black causal graph*? Can one identify interesting fragments based on criteria on that graph, taking the different vertex colors into account?

On the practical side, a major issue – not just for red-black planning but for partial delete relaxation in general – is the runtime overhead incurred by the more complex reasoning inside the heuristic function. This severely limits the amount of "un-relaxation" one can apply, to the point of rendering the interpolation ability useless. But no one forces us to use red-black planning that way! We believe that there is a space of unlimited possibilities in alternate uses. Instead of generating a new red-black plan for every search state, one could generate one red-black plan, or a small set of red-black plans, for the initial state only, i. e., allow substantial runtime to obtain highly "un-relaxed" *plan templates*. One could then design distance-to-template heuristics for guiding state space search. Another possibility is to use the templates as seed plans in plan-space searches such as partial-order planning (e. g. [49, 50]) or LPG [10], or to seed Nakhost and Müller's plan neighborhood graph search [51]. Finally, an exciting approach is *incremental red-black planning*, where one would obtain a relaxed plan in iteration 0, paint one more variable black in every iteration, and rely on the information obtained when solving iteration i to provide search guidance for iteration $i + 1$.

In summary, red-black planning is an elegant approach to partial delete relaxation, whose exploration has only just begun. We hope that other researchers will be inspired to join us in this effort.

Acknowledgments. We thank Mauro Vallati and Lukas Chrpa for running additional experiments for us on the IPC 2014 machines. We thank the anonymous reviewers for their comments, which helped to improve the paper. Part of this work was done while Michael Katz was working at Saarland University. This work was partially supported by the German Research Foundation (DFG), under grant HO 2169/5-1, and by the EU FP7 Programme under grant agreement no. 295261 (MEALS). The work of Carmel Domshlak was partly supported by the ISF grant 1045/12.

Appendix A. Proofs

Lemma 2 *For any* RB *task* $\Pi = \langle \{v_{\mathsf{B}}\}, V^{\mathsf{R}}, A, I, G \rangle$ *it holds that*

(1) the monotonic relaxation π^+ *of any plan* π *for* Π *is SCC-aligned, and*

(2) any SCC-aligned relaxed plan π^+ *for* Π *can be extended, with only a polynomial overhead, into a plan for* Π.

Proof: SCC-aligned relaxed plans, as well as all the auxiliary notions and notation, are defined in the proof of Theorem 1, p. 12.

(1) The proof of this part is straightforward. If $\pi = \langle a_1, \dots, a_n \rangle$ is a plan for Π, then $\pi^+ = \langle a_1, \dots, a_n \rangle$ is a relaxed plan for Π. Let $\langle a_{\sigma(1)}, \dots, a_{\sigma(m)} \rangle$ be the sequence of all the SCC-changing actions along π^+.

For all $0 \leq i \leq m$, and each $\sigma(i) < j \leq \sigma(i+1)$, the action a_j either has no precondition on v_{B} or $\mathsf{pre}(a_j) = \mathsf{eff}(a_l)$ for some $\sigma(i) \leq l < j$. If $l = \sigma(i)$, then $\mathsf{pre}(a_j)[v_{\mathsf{B}}] = \mathsf{eff}(a_{\sigma(i)})[v_{\mathsf{B}}]$, and thus the two trivially belong to the same SCC of $\Gamma_{\sigma(i)}$. Otherwise, if $l > \sigma(i)$, since $l < \sigma(i+1)$, $\mathsf{pre}(a_j)[v_{\mathsf{B}}] = \mathsf{eff}(a_l)[v_{\mathsf{B}}]$ and $\mathsf{eff}(a_{\sigma(i)})[v_{\mathsf{B}}]$ belong to the same SCC of $\Gamma_{\sigma(i)}$, or otherwise a_l would be an SCC-changing action.

Finally, let a_j be the last v_{B}-changing action along π (and thus along π^+). In particular, it means that $j \geq \sigma(m)$. Since π is a plan for Π, if $v_{\mathsf{B}} \in \mathcal{V}(G)$, then $\mathsf{eff}(a_j)[v_{\mathsf{B}}] = G[v_{\mathsf{B}}]$. Following precisely the same argument as above: If $j = \sigma(m)$, then trivially $\mathsf{eff}(a_{\sigma(m)})[v_{\mathsf{B}}] = G[v_{\mathsf{B}}]$. Otherwise, if $j > \sigma(m)$, then $\mathsf{eff}(a_{\sigma(m)})[v_{\mathsf{B}}]$ and $\mathsf{eff}(a_j)[v_{\mathsf{B}}] = G[v_{\mathsf{B}}]$ belong to the same SCC of $\Gamma_{\sigma(m)}$, or otherwise $a_{\sigma(m)}$ would not be the last SCC-changing action along π^+.

(2) Let $\pi^+ = \langle a_1, \ldots, a_n \rangle$ be a SCC-aligned relaxed plan for Π, and $\langle a_{\sigma(1)}, \ldots, a_{\sigma(m)} \rangle$ be the sequence of all the SCC-changing actions along π^+. These m SCC-changing actions divide π^+ into m consecutive action subsequences $\pi^+_i = \langle a_{\sigma(i-1)+1}, \ldots, a_{\sigma(i)} \rangle$, each ending with the corresponding SCC-changing action, plus, possibly an empty, action subsequence $\pi^+_{m+1} = \langle a_{\sigma(m)+1}, a_n \rangle$.

Given that, we construct a plan

$$\pi = \pi_1 \cdot \pi_2 \cdot \ldots \cdot \pi_n \cdot \pi_{n+1}$$

for Π such that, for all $v \in V^{\mathsf{R}}$,

$$I[\![\pi_1 \cdot \ldots \cdot \pi_i]\!][v] = \begin{cases} I[\![\langle a^+_1, \ldots, a^+_i \rangle]\!][v], & 1 \leq i \leq n \\ I[\![\langle a^+_1, \ldots, a^+_n \rangle]\!][v], & i = n+1 \end{cases}, \tag{A.1}$$

and

$$I[\![\pi_1 \cdot \ldots \cdot \pi_i]\!][v_{\mathsf{B}}] = \begin{cases} \mathsf{eff}(a_i)[v_{\mathsf{B}}], & i \in \{\sigma(1), \ldots, \sigma(m)\}, \\ I[v_{\mathsf{B}}], & i < \sigma(1), \\ \mathsf{eff}(a_{\sigma(j)})[v_{\mathsf{B}}], & \sigma(j) < i < \sigma(j+1) \text{ and } j < m \\ \mathsf{eff}(a_{\sigma(m)})[v_{\mathsf{B}}], & \sigma(m) < i \leq n \\ \mathsf{eff}(a_{\sigma(m)})[v_{\mathsf{B}}], & i = n+1 \text{ and } v_{\mathsf{B}} \notin \mathcal{V}(G) \\ G[v_{\mathsf{B}}], & i = n+1 \text{ and } v_{\mathsf{B}} \in \mathcal{V}(G) \end{cases}. \tag{A.2}$$

The plan π is constructed, and the satisfaction of Eqs. A.1-A.2 is proven, inductively, as follows. For $i = 1$, we use $\pi_1 := \langle a_1 \rangle$. This choice trivially satisfies Eq. A.1, and, since $1 \leq \sigma(1)$, Eq. A.2 is satisfied as well (via either its first or second cases). This provides us with a basis for our induction. Assuming now that Eqs. A.1-A.2 hold for $\pi_1 \cdot \ldots \cdot \pi_{i-1}$, $i > 1$, we construct π_i so that $\pi_1 \cdot \ldots \cdot \pi_i$ also satisfy Eqs. A.1-A.2. We do that as follows.

Considering $i \leq n$, let $\sigma(j) < i \leq \sigma(j+1)$ for some $0 \leq j \leq m$. Given that, we set

$$\pi_i := \begin{cases} \pi'_i \cdot \langle a_i \rangle, & i = \sigma(j+1) \\ \pi'_i \cdot \langle a_i \rangle \cdot \pi''_i, & i < \sigma(j+1) \end{cases}, \tag{A.3}$$

where, if a_i is not preconditioned by v_{B} (and thus, in particular, does not affect v_{B}), then $\pi'_i = \pi''_i = \langle\rangle$, and otherwise, π'_i and π''_i are a pair of sequences of actions from $\{a_{\sigma(1)}, \dots, a_{\sigma(j)}\}$ that change v_{B} from $\mathsf{eff}(a_{\sigma(j)})[v_{\mathsf{B}}]$ to $\mathsf{pre}(a_i)[v_{\mathsf{B}}]$ and back, from $\mathsf{eff}(a_i)[v_{\mathsf{B}}]$ to $\mathsf{eff}(a_{\sigma(j)})[v_{\mathsf{B}}]$, respectively.

First, by the structure of π^+ and our inductive assumption on $\pi_1 \cdot \ldots \cdot \pi_{i-1}$ with respect to Eq. A.1, all the (red) preconditions of a_i hold in $I[\![\pi_1 \cdot \ldots \cdot \pi_{i-1}]\!]$. Second, if a_i *is* preconditioned by v_{B}, then, by the SCC-alignment of π^+, $\mathsf{eff}(a_{\sigma(j)})$, $\mathsf{pre}(a_i)[v_{\mathsf{B}}]$, and, if $i < \sigma(j+1)$, $\mathsf{eff}(a_i)[v_{\mathsf{B}}]$, belong to the same SCC of $\Gamma_{\sigma(j)}$. Therefore, an action sequence π'_i, and, if needed, an action sequence π''_i as in Eq. A.3 are guaranteed to exit. Third, by our inductive assumption on Eq. A.2, $I[\![\pi_1 \cdot \ldots \cdot \pi_{i-1}]\!][v_{\mathsf{B}}] = \mathsf{eff}(a_{\sigma(j)})[v_{\mathsf{B}}]$. Forth, since our inductive construction ensures that all the actions $\{a_{\sigma(1)}, \dots, a_{\sigma(j)}\}$ are already executed along π prior to executing π_i, all the red preconditions of actions in π'_i and π''_i hold from $I[\![\pi_1 \cdot \ldots \cdot \pi_{i-1}]\!]$ onwards.

Together, these four arguments imply that π_i is applicable in $I[\![\pi_1 \cdot \ldots \cdot \pi_{i-1}]\!]$, and, since the only action in π_i that is novel to $\pi_1 \cdot \ldots \cdot \pi_{i-1}$ is a_i, $\pi_1 \cdot \ldots \cdot \pi_i$ satisfies Eq. A.1. In turn, if $\pi'_i = \pi''_i = \langle\rangle$, then $I[\![\pi_1 \cdot \ldots \cdot \pi_i]\!][v_{\mathsf{B}}] = I[\![\pi_1 \cdot \ldots \cdot \pi_{i-1}]\!][v_{\mathsf{B}}]$, and thus our inductive assumption on Eq. A.2 directly implies that Eq. A.2 is satisfied by $\pi_1 \cdot \ldots \cdot \pi_i$. Otherwise, if $i = \sigma(j+1)$, then, by Eq. A.3, π_i ends with a_i, and thus $I[\![\pi_1 \cdot \ldots \cdot \pi_i]\!][v_{\mathsf{B}}] = \mathsf{eff}(a_{\sigma(j+1)})$ satisfies Eq. A.2. Finally, if $i < \sigma(j+1)$, then $I[\![\pi_1 \cdot \ldots \cdot \pi_i]\!][v_{\mathsf{B}}] = \mathsf{eff}(a_{\sigma(j)})$, satisfying Eq. A.2 as well.

This finalizes both the construction step and the proof of its correctness for $i \leq n$. Considering now the end-case of $i = n+1$, if $v_{\mathsf{B}} \notin \mathcal{V}(G)$, then we set $\pi_{n+1} = \langle\rangle$. Otherwise, if $v_{\mathsf{B}} \in \mathcal{V}(G)$, then π_{n+1} is set to an action sequence from $\{a_{\sigma(1)}, \dots, a_{\sigma(m)}\}$ that change v_{B} from $\mathsf{eff}(a_{\sigma(m)})$ to $G[v_{\mathsf{B}}]$. Such a sequence of actions from $\{a_{\sigma(1)}, \dots, a_{\sigma(m)}\}$ exists by the SCC-alignment of π^+, and it is applicable in $I[\![\pi_1 \cdot \ldots \cdot \pi_n]\!]$ because (1) by Eq. A.2, $I[\![\pi_1 \cdot \ldots \cdot \pi_n]\!][v_{\mathsf{B}}] = \mathsf{eff}(a_{\sigma(m)})[v_{\mathsf{B}}]$, and (2) all the red preconditions of the actions in π_{n+1} hold in $I[\![\pi_1 \cdot \ldots \cdot \pi_n]\!]$ since $\pi_1 \cdot \ldots \cdot \pi_n$ already contains an instance of each action in $\{a_{\sigma(1)}, \dots, a_{\sigma(m)}\}$. The latter argument also implies that $I[\![\pi_1 \cdot \ldots \cdot \pi_{n+1}]\!]$ satisfies Eq. A.1, and the satisfaction of Eq. A.2 is immediate by the construction of π_{n+1}. □

Lemma 3 *Let* $\Pi = \langle \{v_{\mathsf{B}}\}, V^{\mathsf{R}}, A, I, G \rangle$ *be an* RB *task,* $\pi^+ = \langle a_1, \dots, a_n \rangle$ *be an SCC-aligned relaxed plan for* Π*, and* $\langle a_{\sigma(1)}, \dots, a_{\sigma(m)} \rangle$ *be the sequence of all the SCC-changing actions along* π^+*. Let*

$$\rho^+ = \rho^+_0 \cdot \langle a_{\sigma(1)} \rangle \cdot \rho^+_1 \cdot \langle a_{\sigma(2)} \rangle \cdot \ldots \cdot \rho^+_{m-1} \cdot \langle a_{\sigma(m)} \rangle \cdot \rho^+_m, \qquad \text{(A.4)}$$

be a sequence of actions from A *in which, inductively,* ρ^+_i *is a relaxed plan from*

$I[\![\rho_0^+ \cdot a_{\sigma(1)} \cdot \rho_1^+ \cdot \ldots \cdot a_{\sigma(i)}]\!]$ to the relaxed planning fixpoint, using only those actions from A that are neither preconditioned by the values of v_{B} outside of the SCC of $\mathsf{eff}(a_{\sigma(i)})[v_{\mathsf{B}}]$ *in $\Gamma_{\sigma(i)}$, nor have such values among their effects.*

Then, similarly to π^+,

1. *ρ^+ is an SCC-aligned relaxed plan for Π, and*
2. *$\langle a_{\sigma(1)}, \ldots, a_{\sigma(m)} \rangle$ is the sequence of all the SCC-changing actions along ρ^+.*

Proof: First, *if* the action sequence ρ^+ constructed as above is a relaxed plan for Π, then (2) is immediate by the very restrictions put on the segments ρ_i^+. Now, let $\pi^+ = \pi_0^+ \cdot \langle a_{\sigma(1)} \rangle \cdot \pi_1^+ \cdot \ldots \cdot \pi_{n-1}^+ \cdot \langle a_{\sigma(m)} \rangle \cdot \pi_m^+$. Showing by induction that, for $0 \leq i \leq m$,

$$I[\![\pi_0^+ \cdot \langle a_{\sigma(1)} \rangle \cdot \pi_1^+ \cdot \ldots \cdot \langle a_{\sigma(i)} \rangle \cdot \pi_i^+]\!][v] \subseteq I[\![\rho_0^+ \cdot \langle a_{\sigma(1)} \rangle \cdot \rho_1^+ \cdot \ldots \cdot \langle a_{\sigma(i)} \rangle \cdot \rho_i^+]\!][v] \quad \text{(A.5)}$$

for all $v \in \{v_{\mathsf{B}}\} \cup V^{\mathsf{R}}$, we prove that ρ^+ is a relaxed plan for Π.

For $i = 0$, by the definition of SCC-alignment, π_0^+ comprises a subset of actions that (a) appear on action sequences applicable in $I[\pi_0^+]$, and (b) neither are preconditioned by, nor have among their effects, values of v_{B} except for $I[v_{\mathsf{B}}]$. In turn, by the construction, ρ_0^+ contains *all* the actions satisfying (a)-(b). Thus, we have $I[\![\pi_0^+]\!][v] \subseteq I[\![\rho_0^+]\!][v]$ for all $v \in V^{\mathsf{R}}$, and $I[\![\pi_0^+]\!][v_{\mathsf{B}}] = I[\![\rho_0^+]\!][v_{\mathsf{B}}] = \{I[v_{\mathsf{B}}]\}$.

Assuming now that Eq. A.5 holds for $i - 1 \geq 0$, we show that it holds for i, using the arguments very close to these used for the induction basis. Since, by the induction hypothesis,

$$I[\![\pi_0^+ \cdot \ldots \cdot \langle a_{\sigma(i-1)} \rangle \cdot \pi_{i-1}^+]\!][v] \subseteq I[\![\rho_0^+ \cdot \ldots \cdot \langle a_{\sigma(i-1)} \rangle \cdot \rho_{i-1}^+]\!][v]$$

for all $v \in \{v_{\mathsf{B}}\} \cup V^{\mathsf{R}}$, we have $a_{\sigma(i)}$ applicable in $I[\![\rho_0^+ \cdot \ldots \cdot \langle a_{\sigma(i-1)} \rangle \cdot \rho_{i-1}^+]\!]$ and

$$I[\![\pi_0^+ \cdot \ldots \cdot \langle a_{\sigma(i-1)} \rangle \cdot \pi_{i-1}^+ \cdot a_{\sigma(i)}]\!][v] \subseteq I[\![\rho_0^+ \cdot \ldots \cdot \langle a_{\sigma(i-1)} \rangle \cdot \rho_{i-1}^+ a_{\sigma(i)}]\!][v] \quad \text{(A.6)}$$

for all $v \in \{v_{\mathsf{B}}\} \cup V^{\mathsf{R}}$. In turn, by the definition of SCC-alignment, π_i^+ comprises a subset of actions that (a) appear on action sequences applicable in $I[\![\pi_0^+ \cdot \ldots \cdot \pi_{i-1}^+ \cdot \langle a_{\sigma(i)} \rangle]\!]$, and (b) neither are preconditioned by, nor have among their effects, values of v_{B} outside of the SCC of $\mathsf{eff}(a_{\sigma(i)})[v_{\mathsf{B}}]$ in $\Gamma_{\sigma(i)}$. By Eq. A.6, (a) implies that π_i^+ can be seen as comprising a subset of actions that (a') appear on action sequences applicable in $I[\![\rho_0^+ \cdot \ldots \cdot \rho_{i-1}^+ \cdot \langle a_{\sigma(i)} \rangle]\!]$, while, by the construction of ρ_i^+, ρ_i^+ contains *all* the actions satisfying (a') and (b). This finalizes the proof of the induction step, and thus, of the lemma. □

Theorem 2 *Plan existence for* RB *tasks with a fixed number of black variables is* NP*-complete.*

Proof: For NP-hardness, we construct an RB task, with a single black variable, that is solvable iff an input CNF formula φ is satisfiable. Consider a planning encoding $\Pi(\varphi)$ with: (1) ternary-valued variables $x_1, \dots, x_n$ which encode the propositional variables in φ, the domain of each x_i comprising true, false, and "unassigned"; (2) Boolean variables $c_1, \dots, c_m$ which encode satisfaction of the clauses in φ; and (3) a variable v_{B} with domain $\{1, \dots, n + 1\}$. Initially, $v_{\mathsf{B}} = 1$, all clause variables are false, and all the proposition variables are "unassigned", and the goal is to make all clause variables true. There are actions changing a clause variable c_j to true provided an x_i variable corresponding to one of c_j's literals is assigned the correct truth value. Other actions change the value of x_i from "unassigned" to either false or true provided $v_{\mathsf{B}} = i$, and applying such an action also changes the value of v_{B} to $i + 1$. This encoding does not work in the delete relaxation – that is, if we paint all variables of $\Pi(\varphi)$ red – because x_i may be set to both truth values. However, if only v_{B} is painted black, then plans are forced to assign each x_i exactly one truth value, yielding the desired equivalence.

For membership in NP, just observe that there always exists a polynomial-length plan. We can pre-compile the fixed number of black variables into a single black variable, whose domain size is polynomial. Since all causal graph neighbors of that variable are red, the number of times it needs to change its value (i. e., traverse its DTG from some node to another one) is bounded by the sum of the domain sizes of these neighbors. $\square$

Theorem 3 *Plan existence for* RB *tasks where all black variables have fixed-size domains is* PSPACE*-complete, and it is* NP*-complete even if CG^{B}_{Π} is arcless.*

Proof: If we fix the domain size of the black variables, but not their number, then we obtain a problem that is as hard as FDR with fixed-size domains, which is PSPACE-hard [7], and PSPACE membership is straightforward because the addition of red variables still allows for a proof similar to that for FDR. For the second part of the claim, consider the CNF encoding from the proof of Theorem 2, but now without the special variable v_{B}. If all the clause variables c_j, but none of the variables x_i, are painted red, then the black causal graph of the resulting RB task contains no arcs – each arc in the causal graph involves a red clause variable. At the same time, since the x_i variables are all black, the plans are forced to assign each x_i exactly one truth value, and thus our RB task is solvable iff φ is satisfiable. Membership in NP follows from basically the same argument as in the proof of

Theorem 2: for each black variable, all causal graph neighbors are red so we get a polynomial bound on the number of moves required. □

Theorem 7 *It is* co-NP*-hard to test whether an* RB *task is reversible, even when restricting the input to* RB *tasks whose black causal graph is arcless.*

Proof: The proof is by reduction from DNF tautology testing. Given a propositional DNF formula φ over l clauses $c_1, \dots, c_l$, consider an RB planning encoding $\Pi(\varphi)$ with: black variables $x_1, \dots, x_n$ with $\mathcal{D}(x_i) = \{\mathit{unassigned}, \mathit{true}, \mathit{false}\}$, encoding the propositional variables in φ; and a Boolean *red* variable r, encoding whether or not φ is satisfied under a given assignment to $x_1, \dots, x_n$. All x_i variables are initially *unassigned*, and r is initially false; the goal does not matter here. The value of x_i can be changed from *unassigned* to either false or true with no preconditions, and back from *false* or *true* to *unassigned* with precondition r. We can set r to true using actions $\{a_1, \dots, a_l\}$, where the precondition of a_j requires that all x_i participating in c_j are assigned to their values required by c_j.

Let $\mathcal{M}$ be the set of all 2^n valuations of φ's propositions. For every $m \in \mathcal{M}$, let $S_m \subseteq S$ be the set of reachable states in which *all* variables x_i are assigned as in m. We observe:

(1) For every $m \in \mathcal{M}$ that does satisfy φ, the states in S_m are reversible in the red-black sense.

(2) For every $m \in \mathcal{M}$ that does not satisfy φ, none of the states in S_m is reversible in the red-black sense.

(3) For every state s reachable in $\Pi(\varphi)$, and every $m \in \mathcal{M}$ that complies with the (partial) valuation to $x_1, \dots, x_n$ defined by s, there exists a state $s_m \in S_m$ such that s_m is reachable from s.

For (1), S_m consists of two states in this case, one of which has r true and the other of which has an applicable a_j action achieving r. This allows to assign the x_i variables back to their initial *unassigned* values. For (2), S_m is a singleton in this case, setting the x_i to m and r to false. (3) is obvious.

If φ is a tautology, then by (3) and (1) every reachable state in $\Pi(\varphi)$ is reversible. If φ is not a tautology, then there exists a valuation m that does not satisfy φ. Applying (3) to the initial state, we can reach a state $s_m \in S_m$, which by (2) is not reversible. □

Lemma 4 *Let* $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ *be a reversible* RB *task with acyclic black causal graph,* $\{v_1, \dots, v_n\}$ *be a topological ordering of* V^{B}*, and* s *be a reachable* *state of* Π *with* $\mathcal{R}(s) = s$*. Then, for any* $v_i \in V^{\mathsf{B}}$*, there exists a sequence of actions* π *applicable in* s *such that*

(i) $s[\![\pi]\!][v_1, \dots, v_i] = I[v_1, \dots, v_i]$*, and*

(ii) actions along π *neither are preconditioned by nor affect variables* $v_{i+1}, \dots, v_n$.

Proof: By reversibility, there exists a reverting sequence π' for s, that is, a plan for $\langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, s, I[V^{\mathsf{B}}]\rangle$. Let π be an action sequence obtained from π' by removing all actions that have no effect on any of the variables $v_1, \dots, v_i$. We show that π has the desired properties (i) and (ii). Provided π is applicable in s, its outcome is $I[v_j]$ for all vars v_j with $j \leq i$ because the responsible actions from π' are contained in π, yielding property (i). To see applicability, first note that any red preconditions along π are true simply because $s = \mathcal{R}(s)$. By acyclicity of $\mathit{CG}^{\mathsf{B}}_{\Pi}$ every action in A affects at most one black variable, and so, by the construction of π, every action a in π affects at most one v_j where $j \leq i$. But then, since $\{v_1, \dots, v_n\}$ is a topological ordering of V^{B}, any black preconditions of a are on variables v_l for $l \leq j \leq i$, and thus the actions supporting these preconditions in π', if any, are contained in π. Therefore, π is applicable in s. These arguments also show that neither the effects nor the preconditions of the actions in ρ' touch the variables $v_{i+1}, \dots, v_n$. Thus we have (ii), concluding the argument. □

Lemma 5 *Let* $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ *be a reversible* RB *task with acyclic black causal graph and* $I = \mathcal{R}(I)$*,* π^+ *be an action sequence applicable in the monotonic relaxation* Π^+*, and* g^{B} *be an assignment to* V^{B} *such that* $g^{\mathsf{B}} \subseteq I[\![\pi^+]\!]$*. Then, there exists an action sequence* π *applicable in* I *such that*

(i) $I[\![\pi]\!][V^{\mathsf{B}}] = g^{\mathsf{B}}$*, and*

(ii) $I[\![\pi]\!][V^{\mathsf{R}}] \supseteq I[\![\pi^+]\!][V^{\mathsf{R}}]$.

Proof:

Let $\{v_1, \dots, v_n\}$ be a topological ordering of V^{B}. We prove the stronger claim that there exists π with the claimed properties (i) and (ii), as well as with the third property that (iii) actions in π neither are preconditioned nor affect black variables v_j with $j > m(g^{\mathsf{B}})$, where, for a partial assignment g,

$$m(g) = \begin{cases} \max\{i \mid v_i \in \mathcal{V}(g) \cap V^{\mathsf{B}}\}, & \mathcal{V}(g) \cap V^{\mathsf{B}} \neq \emptyset, \\ 0, & \text{otherwise} \end{cases}$$

denotes the index of the topologically lowest black variable assigned by g.

The proof is by induction over the length of $\pi^+ = \langle a_1, \dots, a_k \rangle$. For the base case, if π^+ is empty, then the claim is trivially satisfied by the empty sequence π. Assuming now that the claim holds for all relaxed plans of length $k - 1 \geq 0$, we prove it for π^+ of length k.

First, consider the sequence π^+_{k-1} containing the first $k-1$ actions of π^+. Since π^+ is applicable in I, we have $\mathsf{pre}(a_k) \subseteq I[\pi^+_{k-1}]$. Thus, by the induction assumption applied to π^+_{k-1}, there is an action sequence ρ_{k-1} for Π applicable in I such that $\mathsf{pre}(a_{k+1}) \subseteq I[\![\rho_{k-1}]\!]$ and $I[\![\pi^+_{k-1}]\!][V^{\mathsf{R}}] \subseteq I[\![\rho_{k-1}]\!][V^{\mathsf{R}}]$. In turn, action sequence $\rho_{k-1} \cdot \langle a_k \rangle$ is thus applicable in I, and obviously, $I[\![\pi^+]\!][V^{\mathsf{R}}] \subseteq I[\![\rho_{k-1} \cdot \langle a_k \rangle]\!][V^{\mathsf{R}}]$. In other words, the red variable values achieved by π^+ are all reachable in Π. But then, $I[\![\pi^+]\!][V^{\mathsf{R}}] \subseteq I$ because, by prerequisite, the initial state is $\mathcal{R}$-completed, i. e., $I = \mathcal{R}(I)$. Thus, part (ii) of the claim is trivially satisfied, for *any* red-black action sequence π applicable in Π.

Given that, the action sequence $\pi = \pi_n \cdot \pi_{n-1} \cdot \dots \cdot \pi_1$ achieving g^{B} is constructed in a way similar to the construction in the proof of Theorem 9. There, however, the construction relied on Lemma 5 we prove here, while here, the corresponding reliance will be on the induction assumption.

For $1 \leq i \leq n$, assume inductively that $\pi_n \cdot \dots \cdot \pi_{i-1}$ is applicable in I; the basis case of the empty action sequence for $i = n$ trivially holds. If $v_i \notin \mathcal{V}(g^{\mathsf{B}})$, then we set $\pi_i := \langle \rangle$. Otherwise, by Lemma 4 above, there exists an action sequence ρ_i that (1) reverts the black variables $\{v_1, \dots, v_i\}$ to their initial values, and (2) neither is preconditioned by nor affects the topologically lower black variables $\{v_{i+1}, \dots, v_n\}$. That is,

$$I[\![\pi_n \cdot \dots \cdot \pi_{i+1} \cdot \rho_i]\!][v_j] = \begin{cases} I[\![\pi_n \cdot \dots \cdot \pi_{i+1}]\!][v_j], & j > i \\ I[v_j], & j \leq i \end{cases}. \tag{A.7}$$

Given that, if $g^{\mathsf{B}}[v_i] = I[v_i]$, then we set $\pi_i := \rho_i$. Otherwise, by the virtue of π^+ being a relaxed *plan* for Π, we must have $\mathsf{eff}(a_j)[v_i] = g^{\mathsf{B}}[v_i]$ for some π^+'s action a_j, $1 \leq j \leq k$. Furthermore, by the acyclicity of $\mathit{CG}^{\mathsf{B}}_{\Pi}$, the black preconditions of a_j may involve only variables $\{v_1, \dots, v_i\}$, and thus $m(\mathsf{pre}(a_j)) \leq i$. In turn, by our induction assumption on the prefix $\pi^+_{a_j}$ of π^+ that precedes a_j, there exists an action sequence π_{a_j} that is applicable in I and achieves $\mathsf{pre}(a_j)$ while neither being preconditioned by nor affecting the black variables $\{v_{i+1}, \dots, v_n\}$. By Eq. A.7 we then have π_{a_j} being applicable in $I[\![\pi_n \cdot \dots \cdot \pi_{i+1} \cdot \rho_i]\!]$, and, for each black variable $v \in \mathcal{V}(\mathsf{pre}(a_j)) \cap V^{\mathsf{B}}$,

$$I[\![\pi_n \cdot \dots \cdot \pi_{i+1} \cdot \rho_i \cdot \pi_{a_j}]\!][v] = \mathsf{pre}(a_j)[v].$$

Given that, we set $\pi_i := \rho_i \cdot \pi_{a_j} \cdot \langle a_j \rangle$. Based on the applicability of each π_i in $I[\![\pi_n \cdot \ldots \cdot \pi_{i+1}]\!]$, the overall action sequence π constructed this way is applicable in I. Finally, since π_i does not touch (neither in preconditions nor in effects) the black variables $\{v_{i+1}, \ldots, v_n\}$, it achieves $g^{\mathsf{B}}[v_i]$ while satisfying (iii), and thus, in particular, invalidates invalidates no sub-goal already achieved by $\pi_n \cdot \ldots \cdot \pi_{i+1}$ on the topologically lower variables $v_{i+1}, \ldots, v_n$. Therefore, the action sequence π satisfies (i), concluding the argument. $\square$

Theorem 10 *Any RSE-invertible* RB *task with acyclic black causal graph is reversible.*

Proof: Say $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G \rangle$. We prove that we can "undo" all action applications. For any state s and action a applicable in s, we show that from $s[\![\langle a \rangle]\!]$ we can reach a state s' so that $s'[V^{\mathsf{B}}] = s[V^{\mathsf{B}}]$ and, for every $v \in V^{\mathsf{R}}$, $s'[v] \supseteq s[v]$. If all variables $\mathcal{V}(\mathsf{eff}(a))$ affected by a are red, there is nothing to do. Otherwise, because the black causal graph is acyclic, exactly one variable v_{B} affected by a is black. Let (d, a, d') be the corresponding arc in $\mathit{DTG}_{\Pi}(v_{\mathsf{B}})$, and (d', a', d) be the inverse arc that exists by prerequisite. We show that we can set $s' := s[\![\langle a, a' \rangle]\!]$. First, a' is applicable in $s[\![\langle a \rangle]\!]$: The precondition of a' is $\mathsf{pre}(a') = \mathsf{ocon}(d', a', d) \cup \{\langle v_{\mathsf{B}}/d' \rangle\}$. Obviously, $\langle v_{\mathsf{B}}/d' \rangle \in s[\![\langle a \rangle]\!]$. Further, we have $\mathsf{ocon}(d', a', d) \subseteq \mathsf{ocon}(d, a, d') \cup \mathsf{oeff}(d, a, d')$. If $\langle v/d'' \rangle \in \mathsf{oeff}(d, a, d')$, then it is made true by a. If $\langle v/d'' \rangle \in \mathsf{ocon}(d, a, d')$, then it is true in s, and remains true if v is black because all variables affected by $\mathsf{oeff}(d, a, d')$ must be red. Applying a' to $s[\![\langle a \rangle]\!]$, the black variable v_{B} is reverted to its value d in s; all other effects of a' are red. This concludes the proof of Theorem 10. $\square$

Theorem 12 *There exists a set $\mathcal{G}$ of directed graphs where* $\mathsf{scc\text{-}size}(\mathcal{G})$ *is bounded by* 2*, and* $\mathsf{RB\text{-}PlanExist}(\mathcal{G})$ *restricted to RSE-invertible* RB *is* NP*-hard.*

Proof: The proof is by a polynomial reduction from CNF satisfiability testing: Given a CNF formula ϕ, we construct an RSE-invertible RB task with strongly connected black causal graph components of size 2 that has a plan iff ϕ is satisfiable.

For each Boolean variable p occuring in ϕ, we include two binary state variables x_p and y_p with initial value 0. We furthermore include an action a_{px1y1} with precondition $\{\langle x_p/0 \rangle, \langle y_p/0 \rangle\}$ and effect $\{\langle x_p/1 \rangle, \langle y_p/1 \rangle\}$, an action a_{px0} with precondition $\{\langle x_p/1 \rangle, \langle y_p/0 \rangle\}$ and effect $\{\langle x_p/0 \rangle\}$, and an action a_{py0} with prec $\{\langle x_p/0 \rangle, \langle y_p/1 \rangle\}$ and effect $\{\langle y_p/0 \rangle\}$. Each individual variable x_p and y_p is RSE-invertible, but their product is not; we can transition to $\{\langle x_p/1 \rangle, \langle y_p/1 \rangle\}$ but we cannot go back. Thus, for each p in ϕ, we can encode the decision whether to set p to true ($\{\langle x_p/1\rangle, \langle y_p/1\rangle\}$) or false ($\{\langle x_p/0\rangle, \langle y_p/0\rangle\}$).

We conclude our construction by adding, in a straightforward manner, variables and actions that allow to test satisfaction of ϕ for given decisions on each p. Namely, for each clause $c = \{l_1, \dots, l_k\}$ in ϕ, we include a binary state variable x_c with initial value 0, and we include an action a_{ci} for every literal l_i in c. If $l_i = p$ then a_{ci} has precondition $\{\langle x_p/1\rangle, \langle y_p/1\rangle\}$ and effect $\{\langle x_c/1\rangle\}$. If $l_i = \neg p$ then a_{ci} has precondition $\{\langle x_p/0\rangle, \langle y_p/0\rangle\}$ and effect $\{\langle x_c/1\rangle\}$.

We paint all variables black. The resulting RB task obviously has the desired properties, concluding the proof. $\square$

Theorem 13 *Let* $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ *be an RSE-invertible* RB *planning task with acyclic black causal graph,* π^+ *be a relaxed plan for* Π*, and* $R^+ = G[V^{\mathsf{R}}] \cup \bigcup_{a \in \pi^+} \mathsf{pre}(a)[V^{\mathsf{R}}]$. *Then, assuming a complete solver for the sub-tasks* Π^{B} *generated,* RedFactsFollowing(Π, R^+) *terminates, and the action sequence* π *it returns is a plan for* Π.

Proof: We start by showing that, as long as $R \not\supseteq R^+$, we always have $A_0 \neq \emptyset$. This is done with the help of the relaxed plan π^+. Let $a_i \in \pi^+$ be the action with the smallest index i such that $\mathsf{eff}(a_i) \cap (R^+ \setminus R) \neq \emptyset$ (at least one such action must exist as not all of R^+ has been established yet). We show that $a_i \in A_0$, i. e., that $\mathsf{pre}(a_i) \subseteq R \cup B$. First, regarding the red variables, assume to the contrary that there exists $v \in \mathcal{V}^{\mathsf{R}}(\mathsf{pre}(a_i))$ such that $\mathsf{pre}(a_i)[v] \notin R$. Then $\mathsf{pre}(a_i)[v] \neq I[v]$ and thus there exists $1 \leq j \leq i-1$ such that $\mathsf{eff}(a_j)[v] = \mathsf{pre}(a_i)[v] \in R^+$. But then, $\mathsf{eff}(a_j) \cap (R^+ \setminus R) \neq \emptyset$, in contradiction to the assumption that i is the *smallest* index i with $\mathsf{eff}(a_i) \cap (R^+ \setminus R) \neq \emptyset$. Second, regarding the black variables, because π^+ is a relaxed plan $a_1 \cdot \ldots \cdot a_{i-1}$ correspond, for each black variable $v \in \mathcal{V}^{\mathsf{B}}(\mathsf{pre}(a_i))$, to a path in $\boldsymbol{DTG}_\Pi(v)$ that visits the value $\mathsf{pre}(a_i)[v]$. Because $R \cup B$ is a superset of the facts achieved in the relaxed plan using $a_1 \cdot \ldots \cdot a_{i-1}$, the path $\boldsymbol{DTG}_\Pi(v)$ uses only actions with outside conditions in $R \cup B$. Therefore, $\mathsf{pre}(a_i) \subseteq R \cup B$ and we have $a_i \in A_0$ as desired.

Consider an iteration of the **while** loop. By definition of A_0, any red preconditions of the selected action $a \in A_0$ are true in the current state $I[\![\pi]\!]$. For the unsatisfied black preconditions $g = \mathsf{pre}(a)[V^{\mathsf{B}}]$, we have $g \subseteq B$, and they are tackled by Achieve(g), solving an FDR task Π^{B} with goal g. We show below that:

(i) Π^{B} is well-defined;

(ii) Π^{B} is solvable; and

(iii) any plan π^{B} for Π^{B} is, in our RB task Π, applicable in $I[\![\pi]\!]$.

Thanks to (i) and (ii), we will obtain a plan π^{B} for Π^{B}. Thanks to (iii), while Π^{B} ignores some of the red variables, effects on these cannot hurt anyway, so a is applicable in $I[\![\pi \cdot \pi^{\mathsf{B}}]\!]$.

Since $\mathsf{eff}(a) \cap (R^+ \setminus R) \neq \emptyset$, $|R^+ \setminus R|$ decreases by at least 1 in each iteration, the **while** loop terminates after at most $|R^+|$ iterations. Upon termination, we have $R^+ \subseteq I[\![\pi]\!][V^{\mathsf{R}}] = R$. The latter implies that $G[V^{\mathsf{B}}] \subseteq B$ because the relaxed plan achieves the goal without using any other red variable values: for each black variable $v \in V^{\mathsf{B}} \cap \mathcal{V}(G)$, π^+ corresponds to a path in $\boldsymbol{DTG}_{\Pi}(v)$, visiting $G[v]$ and using red outside conditions from R^+ only. Calling $\textsc{Achieve}(G[V^{\mathsf{B}}])$ (if needed), as we will show the FDR task Π^{B} constructed will again have properties (i)–(iii), so it is solvable and appending its solution will turn π into a plan for our RB task Π.

Regarding (i), we need to show that all variable values occuring in Π^{B} are indeed members of the respective variable domains, i. e., the reduced domains $\mathcal{D}^{\mathsf{B}}(v)$ for black variables v. This is obvious for I^{B}. It holds for $G^{\mathsf{B}} = \mathsf{pre}(a)[V^{\mathsf{B}}]$ as $a \in A_0$, and it holds for $G^{\mathsf{B}} = G[V^{\mathsf{B}}]$ because, then, $G[V^{\mathsf{B}}] \subseteq B$. Finally, for actions, for the red variables there is nothing to show as they keep their original domains. For the black variables, if $\mathsf{pre}(a) \subseteq R \cup B$ then the (single) black effect must be contained in B as well. If $a \in A(\overleftarrow{\boldsymbol{DTG}_{\Pi}(v)|_{R \cup B}})$ then by construction of $\overleftarrow{\boldsymbol{DTG}_{\Pi}(v)|_{R \cup B}}$ its black precondition and effect on v are contained in $\boldsymbol{DTG}_{\Pi}(v)|_{R \cup B}$, and any black (outside) preconditions on variables other than v are contained in $\mathsf{ocon}(d, d') \cup \mathsf{oeff}(d, d')$ for some arc (d, d') in $\boldsymbol{DTG}_{\Pi}(v)|_{R \cup B}$ and are thus contained in B; by construction, a cannot have any black outside effects.

Regarding (iii), for all actions $a^{\mathsf{B}} \in A^{\mathsf{B}}$ where we have $\mathsf{pre}(a) \subseteq R \cup B$, the red preconditions are true in the current state $I[\![\pi]\!]$. For all other actions $a^{\mathsf{B}} \in A^{\mathsf{B}}$ we have $a \in A(\overleftarrow{\boldsymbol{DTG}_{\Pi}(v)|_{R \cup B}})$ for some black variable v, which implies that all red (outside) precondition variables are contained in $\overleftarrow{V}^{\mathsf{R}}$. So applicability, in Π, of π^{B} in $I[\![\pi]\!]$ depends only on variables that are contained in Π^{B}, which immediately implies (iii).

We finally show (ii), i. e., that Π^{B} is solvable. By Observation 7 of Helmert [29], any FDR task with acyclic causal graph and strongly connected domain transition graphs is solvable. By contrast to relaxed plan repair (cf. the proof of Theorem 11), Π^{B} as constructed here does not fit that profile; however, the argument can be easily adapted. Helmert's observation relies on the facts that:

(1) Acyclicity of the causal graph implies that we can solve the planning task

"top-down", from causal graph leaves to roots, fixing a DTG path for each variable v and propagating the required preconditions as sub-goals to v's parents.

(2) As every DTG is strongly connected, every required path is available, i. e., every variable can always move from its current value to any other value it is required to achieve as a sub-goal (or its own goal).

(1) is preserved in our setting, for the black variables; the red variables are handled exclusively as part of our adaptation of (2) below, which is possible as they have no own goals (G^{B} does not mention the red variables). So how do we ascertain (2)? We employ the following two observations, valid for every black variable $v \in V^{\mathsf{B}}$:

(a) From v's start value, $I^{\mathsf{B}}[v] = I[\![\pi]\!][v]$, all values $d \in \mathcal{D}^{\mathsf{B}}(v)$ of v's reduced domain are reachable in $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$.

(b) Assume that π^{B} is any applicable action sequence in Π^{B} which has, at some point, executed action a^{B} traversing $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$ arc (d, d'). Then there exists an action $a'^{\mathsf{B}} \in A^{\mathsf{B}}$ inducing an inverse arc (d', d) whose red outside conditions are contained in the outcome $I^{\mathsf{B}}[\![\pi^{\mathsf{B}}]\!]$ of applying π^{B} in Π^{B}.

To see that (a) and (b) together show (2), observe first that the paths whose existence are postulated in (2) may make use of arbitrary black outside preconditions ("arbitrary" subject to causal graph acyclicity, of course). To achieve this in our setting, all we need to show is that we can reach any other value of v, from its current value, without having to rely on any *red* outside conditions that are not already true. Now, when v makes its first move, by (a) any path that may be required is available and relies only on the red facts R which are already true at the start. In any subsequent move of v, for all $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$ arcs we have traversed so far, by (b) there exists a suitable inverse action a' (relying only on red outside conditions that are already true) inducing the respective inverse arc. So, say we need to reach any value d_g that may be required as a sub-goal. We can go back to the start value $I^{\mathsf{B}}[v]$, exploiting the fact that by (b) we can invert any $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$ arc we traversed, and exploiting the fact that any non-$\mathit{DTG}_{\Pi}(v)|_{R \cup B}$ arc (d', d) we traversed must be the inverse of a $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$ arc (d, d'), so we can take (d, d') for our path back to $I^{\mathsf{B}}[v]$. From $I^{\mathsf{B}}[v]$, by (a) we can move to d_g, concluding this argument.

It remains to prove (a) and (b). Regarding (a), by construction $\mathcal{D}^{\mathsf{B}}(v)$ consists exactly of the values in $\mathit{DTG}_{\Pi}(v)|_{R \cup B}$, which by construction are all reachable from $I[v]$. So it suffices to prove that $\mathit{DTG}_\Pi(v)|_{R \cup B}$ contains a path from $I^{\mathsf{B}}[v] = I[\![\pi]\!][v]$ to $I[v]$. Clearly, π induces a path from $I[v]$ to $I[\![\pi]\!][v]$ in $\mathit{DTG}_\Pi(v)|_{R \cup B}$. Let (d, d') induced by a^{B} (where $a \in \pi$) be any arc on that path. We show that there exists an inverse arc (d', d) in $\mathit{DTG}_\Pi(v)|_{R \cup B}$. Because (d, d') is RSE-invertible in Π, there exists an action $a' \in A$ inducing an arc (d', d) in $\mathit{DTG}_\Pi(v)$ whose outside condition is contained in $\mathsf{pre}(a) \cup \mathsf{eff}(a)$. Since, obviously, $\{\langle v/d' \rangle\} \cup \mathsf{pre}(a) \cup \mathsf{eff}(a) \subseteq R \cup B$, we get $\mathsf{pre}(a') \subseteq R \cup B$. Thus $a'^{\mathsf{B}} \in A^{\mathsf{B}}$, and (d', d) is an arc in $\mathit{DTG}_{\Pi^{\mathsf{B}}}(v)$ as desired.

To show (b), a similar argument can be applied. If π^{B} is an applicable action sequence in Π^{B}, containing a^{B} traversing the $\mathit{DTG}_\Pi(v)|_{R \cup B}$ arc (d, d'), then by RSE-invertibility there exists an inverse arc whose new red outside conditions (if any) have been established by a^{B}. In more detail, because (d, d') is RSE-invertible in Π, there exists an action $a' \in A$ inducing an arc (d', d) in $\mathit{DTG}_\Pi(v)$ whose outside condition is contained in $\mathsf{pre}(a) \cup \mathsf{eff}(a)$. If $\mathsf{ocon}(d', d)[V^{\mathsf{R}}] \subseteq R$, then the claim is trivial as $R \subseteq I^{\mathsf{B}}$. If $\mathsf{ocon}(d', d)[V^{\mathsf{R}}] \not\subseteq R$, then (d', d) is contained in $\overleftarrow{\mathit{DTG}_\Pi(v)|_{R \cup B}}$: It is an inverse arc to an arc in $\mathit{DTG}_\Pi(v)|_{R \cup B}$, and is not itself contained in $\mathit{DTG}_\Pi(v)|_{R \cup B}$. Thus its red outside condition variables are contained in $\mathcal{V}(\mathsf{ocon}(\overleftarrow{\mathit{DTG}_\Pi(v)|_{R \cup B}})) \subseteq \overleftarrow{V}^{\mathsf{R}}$, and $a'^{\mathsf{B}} \in A^{\mathsf{B}}$ as $a' \in A(\overleftarrow{\mathit{DTG}_\Pi(v)|_{R \cup B}})$. By construction, $\mathsf{ocon}(d', d)[V^{\mathsf{R}}] \subseteq \mathsf{pre}(a) \cup \mathsf{eff}(a)$. Obviously, all red facts in $\mathsf{pre}(a) \cup \mathsf{eff}(a)$ are true in $I^{\mathsf{B}}[\![\pi^{\mathsf{B}}]\!]$. This concludes the proof. $\square$

Proposition 2 *The algorithm* ACYCLICPLANNING(Π^{B}) *is sound and complete, and its runtime is polynomial in the size of* Π^{B} *and the length of the plan* π^{B} *returned.*

Proof: Note that all DTGs are strongly connected, and thus the required sequences of actions $\pi_v(d, d')$ exist for every pair of values $d, d'' \in \mathit{DTG}_\Pi(v)$. The algorithm starts with an empty sequence, and stops after n iterations. At each iteration the current sequence $\pi = \langle a_1 \cdot \ldots \cdot a_k \rangle$ is extended with sequences π_j, $1 \leq j \leq k+1$, resulting in a new sequence $\pi_1 \cdot \langle a_1 \rangle \cdot \ldots \cdot \pi_k \cdot \langle a_k \rangle \cdot \pi_{k+1}$. We need to show that the final sequence is a plan.

We show by induction that the sequence obtained at the end of each iteration is a plan for Π^{B} projected on variables $v_i, \ldots, v_n$. For the first iteration, with $i = n$ we have the final sequence being empty if $G[v_n]$ is not defined, and $\pi_{v_n}(I[v_n], G[v_n])$ otherwise. Assume now that $\pi = \langle a_1 \cdot \ldots \cdot a_k \rangle$ is a plan for Π^{B} projected on variables $v_{i+1}, \ldots, v_n$. Let π_j, $1 \leq j \leq k+1$ be the sequences as defined in iteration i of the algorithm. Since the causal graph is acyclic, we know that (a) all actions in π_j effect only the variable v_i, and (b) none of the actions in

π_j are preconditioned on any of the variables $v_{i+1}, \ldots, v_n$. Thus, focusing on the projection Π_i^{B} of Π^{B} on variables $v_i, \ldots, v_n$, the actions in π_j, $1 \leq j \leq k+1$ are preconditioned only on the variable v_i, and effect only that variable. In addition, the sequence $\pi_1 \cdot \ldots \cdot \pi_{k+1}$ induces a path in *DTG*$_{\Pi_i}(v_i)$ from $I[v_i]$ to $G[v_i]$.

Furthermore, if $\mathsf{pre}(a_j)[v_i]$ is defined, then the sequence $\pi_1 \cdot \ldots \cdot \pi_j$ induces a path in *DTG*$_{\Pi_i}(v_i)$ from $I[v_i]$ to $\mathsf{pre}(a_j)[v_i]$.

Thus, $\pi_1 \cdot \langle a_1 \rangle \cdot \ldots \cdot \pi_k \cdot \langle a_k \rangle \cdot \pi_{k+1}$ is an applicable sequence of actions that leads to the goal of Π_i^{B}, hence is a plan for Π_i^{B}. $\square$

References

[1] M. Katz, J. Hoffmann, C. Domshlak, Who said we need to relax *all* variables?, in: D. Borrajo, S. Fratini, S. Kambhampati, A. Oddi (Eds.), Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13), AAAI Press, Rome, Italy, 2013, pp. 126–134.

[2] M. Katz, J. Hoffmann, C. Domshlak, Red-black relaxed plan heuristics, in: M. desJardins, M. Littman (Eds.), Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI'13), AAAI Press, Bellevue, WA, USA, 2013, pp. 489–495.

[3] M. Katz, J. Hoffmann, Red-black relaxed plan heuristics reloaded, in: M. Helmert, G. Röger (Eds.), Proceedings of the 6th Annual Symposium on Combinatorial Search (SOCS'13), AAAI Press, 2013, pp. 105–113.

[4] D. V. McDermott, Using regression-match graphs to control search in planning, Artificial Intelligence 109 (1999) 111–159.

[5] B. Bonet, H. Geffner, Planning as heuristic search, Artificial Intelligence 129 (2001) 5–33.

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

[7] T. Bylander, The computational complexity of propositional STRIPS planning, Artificial Intelligence 69 (1994) 165–204.

[8] J. Hoffmann, Where 'ignoring delete lists' works: Local search topology in planning benchmarks, Journal of Artificial Intelligence Research 24 (2005) 685–758.

[9] M. Helmert, R. Mattmüller, Accuracy of admissible heuristic functions in selected planning domains, in: D. Fox, C. Gomes (Eds.), Proceedings of the 23rd National Conference of the American Association for Artificial Intelligence (AAAI-08), AAAI Press, Chicago, Illinois, USA, 2008, pp. 938–943.

[10] A. Gerevini, A. Saetti, I. Serina, Planning through stochastic local search and temporal action graphs, Journal of Artificial Intelligence Research 20 (2003) 239–290.

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

[12] M. Helmert, A planning heuristic based on causal graph analysis, in: S. Koenig, S. Zilberstein, J. Koehler (Eds.), Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS'04), Morgan Kaufmann, Whistler, Canada, 2004, pp. 161–170.

[13] A. Coles, M. Fox, D. Long, A. Smith, A hybrid relaxed planning graph'lp heuristic for numeric planning domains, in: J. Rintanen, B. Nebel, J. C. Beck, E. Hansen (Eds.), Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08), AAAI Press, 2008, pp. 52–59.

[14] H. Nakhost, J. Hoffmann, M. Müller, Resource-constrained planning: A monte carlo random walk approach, in: B. Bonet, L. McCluskey, J. R. Silva, B. Williams (Eds.), Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12), AAAI Press, 2012, pp. 181–189.

[15] A. J. Coles, A. Coles, M. Fox, D. Long, A hybrid LP-RPG heuristic for modelling numeric resource flows in planning, Journal of Artificial Intelligence Research 46 (2013) 343–412.

[16] M. B. Do, S. Kambhampati, Sapa: A domain-independent heuristic metric temporal planner, in: A. Cesta, D. Borrajo (Eds.), Recent Advances in AI Planning. 6th European Conference on Planning (ECP-01), Lecture Notes in Artificial Intelligence, Springer-Verlag, Toledo, Spain, 2001, pp. 109–120.

[17] M. van den Briel, J. Benton, S. Kambhampati, T. Vossen, An LP-based heuristic for optimal planning, in: C. Bessiere (Ed.), Proceedings of the

Thirteenth International Conference on Principles and Practice of Constraint Programming (CP'07), volume 4741 of *Lecture Notes in Computer Science*, Springer-Verlag, 2007, pp. 651–665.

[18] J. A. Baier, A. Botea, Improving planning performance using low-conflict relaxed plans, in: A. Gerevini, A. Howe, A. Cesta, I. Refanidis (Eds.), Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09), AAAI Press, 2009, pp. 10–17.

[19] M. Helmert, H. Geffner, Unifying the causal graph and additive heuristics, in: J. Rintanen, B. Nebel, J. C. Beck, E. Hansen (Eds.), Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08), AAAI Press, 2008, pp. 140–147.

[20] D. Cai, J. Hoffmann, M. Helmert, Enhancing the context-enhanced additive heuristic with precedence constraints, in: A. Gerevini, A. Howe, A. Cesta, I. Refanidis (Eds.), Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09), AAAI Press, 2009, pp. 50–57.

[21] M. Fox, D. Long, Hybrid STAN: Identifying and managing combinatorial optimisation sub-problems in planning, in: B. Nebel (Ed.), Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Morgan Kaufmann, Seattle, Washington, USA, 2001, pp. 445–450.

[22] M. Fox, D. Long, Stan4: A hybrid planning strategy based on subproblem abstraction, The AI Magazine 22 (2001) 81–84.

[23] E. Keyder, H. Geffner, Heuristics for planning with action costs revisited, in: M. Ghallab (Ed.), Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08), Wiley, Patras, Greece, 2008, pp. 588–592.

[24] P. Haslum, Incremental lower bounds for additive cost planning problems, in: B. Bonet, L. McCluskey, J. R. Silva, B. Williams (Eds.), Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12), AAAI Press, 2012, pp. 74–82.

[25] E. Keyder, J. Hoffmann, P. Haslum, Semi-relaxed plan heuristics, in: B. Bonet, L. McCluskey, J. R. Silva, B. Williams (Eds.), Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12), AAAI Press, 2012, pp. 128–136.

[26] E. Keyder, J. Hoffmann, P. Haslum, Improving delete relaxation heuristics through explicitly represented conjunctions, Journal of Artificial Intelligence Research 50 (2014) 487–533.

[27] C. Knoblock, Automatically generating abstractions for planning, Artificial Intelligence 68 (1994) 243–302.

[28] R. Brafman, C. Domshlak, Structure and complexity in planning with unary operators, Journal of Artificial Intelligence Research 18 (2003) 315–349.

[29] M. Helmert, The Fast Downward planning system, Journal of Artificial Intelligence Research 26 (2006) 191–246.

[30] C. Domshlak, A. Nazarenko, The complexity of optimal monotonic planning: The bad, the good, and the causal graph, Journal of Artificial Intelligence Research 48 (2013) 783–812.

[31] S. Kupferschmid, J. Hoffmann, H. Dierks, G. Behrmann, Adapting an AI planning heuristic for directed model checking, in: A. Valmari (Ed.), Proceedings of the 13th International SPIN Workshop (SPIN 2006), volume 3925 of *Lecture Notes in Computer Science*, Springer-Verlag, 2006, pp. 35–52.

[32] H. Chen, O. Giménez, Causal graphs and structurally restricted planning, Journal of Computer and System Sciences 76 (2010) 579–592.

[33] J. Seipp, M. Helmert, Fluent merging for classical planning problems, in: ICAPS 2011 Workshop on Knowledge Engineering for Planning and Scheduling, 2011, pp. 47–53.

[34] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer-Verlag, 2006.

[35] J. Koehler, J. Hoffmann, On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm, Journal of Artificial Intelligence Research 12 (2000) 338–386.

[36] J. Hoffmann, Analyzing search topology without running any search: On the connection between causal graphs and h^+, Journal of Artificial Intelligence Research 41 (2011) 155–229.

[37] A. Gerevini, L. Schubert, Inferring state-constraints for domain independent planning, in: J. Mostow, C. Rich (Eds.), Proceedings of the 15th National Conference of the American Association for Artificial Intelligence (AAAI'98), MIT Press, Madison, WI, USA, 1998, pp. 905–912.

[38] M. Fox, D. Long, The automatic inference of state invariants in TIM, Journal of Artificial Intelligence Research 9 (1998) 367–421.

[39] J. Rintanen, An iterative algorithm for synthesizing invariants, in: H. A. Kautz, B. Porter (Eds.), Proceedings of the 17th National Conference of the American Association for Artificial Intelligence (AAAI-00), AAAI Press, Austin, TX, USA, 2000, pp. 806–811.

[40] M. Helmert, Concise finite-domain representations for PDDL planning tasks, Artificial Intelligence 173 (2009) 503–535.

[41] B. C. Williams, P. P. Nayak, A reactive planner for a model-based executive, in: M. Pollack (Ed.), Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), Morgan Kaufmann, Nagoya, Japan, 1997, pp. 1178–1185.

[42] C. Domshlak, Y. Dinitz, Multi-agent offline coordination: Structure and complexity, in: A. Cesta, D. Borrajo (Eds.), Recent Advances in AI Planning. 6th European Conference on Planning (ECP-01), Lecture Notes in Artificial Intelligence, Springer-Verlag, Toledo, Spain, 2001, pp. 34–43.

[43] J. Hoffmann, J. Porteous, L. Sebastia, Ordered landmarks in planning, Journal of Artificial Intelligence Research 22 (2004) 215–278.

[44] E. Karpas, C. Domshlak, Cost-optimal planning with landmarks, in: C. Boutilier (Ed.), Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), Morgan Kaufmann, Pasadena, California, USA, 2009, pp. 1728–1733.

[45] M. Helmert, C. Domshlak, Landmarks, critical paths and abstractions: What's the difference anyway?, in: A. Gerevini, A. Howe, A. Cesta, I. Refanidis (Eds.), Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09), AAAI Press, 2009, pp. 162–169.

[46] S. Richter, M. Helmert, Preferred operators and deferred evaluation in satisficing planning, in: A. Gerevini, A. Howe, A. Cesta, I. Refanidis (Eds.), Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09), AAAI Press, 2009, pp. 273–280.

[47] J. Rintanen, Planning as satisfiability: Heuristics, Artificial Intelligence 193 (2012) 45–86.

[48] I. Cenamor, T. de la Rosa, F. Fernández, IBACOP and IBACOP2 planner, in: IPC 2014 planner abstracts, 2014, pp. 35–38.

[49] J. S. Penberthy, D. S. Weld, UCPOP: A sound, complete, partial order planner for ADL, in: B. Nebel, W. Swartout, C. Rich (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference (KR-92), Morgan Kaufmann, Cambridge, MA, 1992, pp. 103–114.

[50] X. Nguyen, S. Kambhampati, Reviving partial order planning, in: B. Nebel (Ed.), Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Morgan Kaufmann, Seattle, Washington, USA, 2001, pp. 459–464.

[51] H. Nakhost, M. Müller, Action elimination and plan neighborhood graph search: Two algorithms for plan improvement, in: R. I. Brafman, H. Geffner, J. Hoffmann, H. A. Kautz (Eds.), Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10), AAAI Press, 2010, pp. 121–128.