Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling

Who Said We Need to Relax *All* Variables?

Michael Katz and **Jörg Hoffmann**
Saarland University
Saarbrücken, Germany
{katz, hoffmann}@cs.uni-saarland.de

Carmel Domshlak
Technion
Haifa, Israel
dcarmel@ie.technion.ac.il

Abstract

Despite its success in both satisficing and optimal planning, the 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. Of course, this relaxation is useful as a basis for devising heuristic functions only if the resulting red-black planning task is polynomial-time solvable. We herein investigate the tractability region of red-black planning, extending Chen and Gimenez' characterization theorems for regular planning to the more general red-black setting. In particular, we identify significant islands of tractable red-black planning, opening the road to the efficient computation of very powerful heuristics.

Introduction

Monotonic, or delete, relaxation played a key role in advances of planning systems over the last decade. State variables in the monotonic relaxation accumulate their values, rather than switching between them. While regular planning is PSPACE-complete even for simple formalisms, monotonic planning is polynomial-time (Bylander 1994). Despite this, plans for the monotonic relaxation often yield very useful search guidance. Starting with the HSP (Bonet and Geffner 2001) and FF (Hoffmann and Nebel 2001) planners, exploitation of the monotonic relaxation became a key ingredient of many competitive planning systems.

While some of the most effective heuristics to date are obtained as the estimated cost of the monotonic relaxation of the original planning task, this relaxation has significant pitfalls. A prime example is planning with non-replenishable resources, whose consumption is completely ignored within the relaxation. More generally, the monotonic relaxation seriously under-estimates goal distances whenever planning decisions have non-local implications, where the effect of some action may be detrimental for the ability to execute other parts of the plan later on.

Numerous works have designed heuristics that take *some* deletes into account, e. g. (Fox and Long 2001; Gerevini, Saetti, and Serina 2003; Helmert 2004; Helmert and Geffner 2008; Keyder and Geffner 2008; Cai, Hoffmann, and Helmert 2009). It has proved daunting, however, to devise frameworks that actually allow to do so systematically, interpolating all the way between real planning and monotonic planning. The first approach of this kind was put forward only very recently, enriching the monotonic relaxation with an explicitly represented set of fact conjunctions, forcing the heuristic to become perfect as that set becomes large (Haslum 2012; Keyder, Hoffmann, and Haslum 2012). We herein devise a much simpler method enabling the same kind of interpolation: *we relax only some of the state variables*.

Our investigation was initiated by a very simple question: Does planning remain polynomial-time if all state variables except a single binary-valued variable v_0 have monotonic (value-accumulating) semantics, while v_0 keeps the regular (value-switching) semantics? When, somewhat surprisingly, the answer to this question turned out to be "yes", we broadened our investigation into what we baptize *red-black planning*. Some state variables, called *red*, take the relaxed semantics accumulating their values, while all other variables, called *black*, keep their regular semantics.[1] This framework is relevant because, as we show, red-black planning gives rise to several interesting fragments of polynomial-time planning, and thus has the potential to allow the efficient computation of very powerful heuristic functions. Exploiting the complexity characterization theorems of Chen and Giménez (2010), we draw a sharp borderline for the identified islands of tractability. Finally, considering several IPC benchmark domains, we provide theoretical evidence that these tractability islands may yield highly informative heuristic functions in practice.

We provide the background, then introduce red-black planning. We examine how Chen and Gimenez' characterization theorems extend to this more general setting. We describe our findings in IPC benchmarks, and conclude.

[1] Earlier attempts applied a special treatment, half-way between red and black planning, to restricted subsets of variables (Fox and Long 2001; Keyder and Geffner 2008). We get back to this later, once we formally introduced our framework.

Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Background

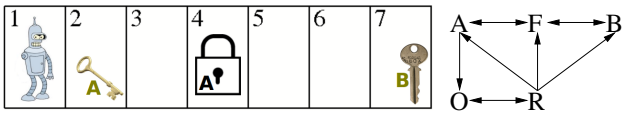
A **finite-domain representation** (FDR) planning task is given by a quadruple $\Pi = \langle V, A, I, G \rangle$. V is a set of *state variables*, where each $v \in V$ is associated with a finite domain $\mathcal{D}(v)$. A complete assignment to V is called a *state*. I is the *initial state*, and the *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. If a has a precondition on v but does not change it, we say that a is *prevailed* by v.

The semantics of FDR tasks is as follows. For a partial assignment p, $\mathcal{V}(p) \subseteq V$ denotes the subset of state variables instantiated by p. In turn, for any $V' \subseteq \mathcal{V}(p)$, by $p[V']$ we denote the value of V' in p. 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 $v \in \mathcal{V}(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$; the resulting state is denoted by $s[\![a]\!]$. By $s[\![\langle a_1, \ldots, a_k \rangle]\!]$ we denote the state obtained from sequential application of the (respectively applicable) actions $a_1, \ldots, a_k$ starting at state s. Such an action sequence is an *s-plan* if $s[\![\langle a_1, \ldots, a_k \rangle]\!][\mathcal{V}(G)] = G$, 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.

Figure 1 (a) illustrates an example that we use throughout the paper. The example is akin to the GRID benchmark. We encode it in FDR via 5 state variables: R, the robot position in $\{1, \ldots, 7\}$; A, the key A position in $\{R, 1, \ldots, 7\}$; B, the key B position in $\{R, 1, \ldots, 7\}$; F, a Boolean encoding whether or not the robot hand is free; O, a Boolean encoding whether or not the lock is open. The actions *move* the robot provided either the target position is $\neq 4$ or the lock is open, *take* a key provided the hand is free, *drop* a key, or *open* the lock provided the robot is at position 3 or at position 5 and holds key A. The goal is for key B to be at position 1. An optimal plan moves to position 2, takes key A, moves to position 3, opens the lock, moves to position 7, drops key A and takes key B, moves back to position 1 and drops 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. Informally, in MFDR tasks the state variables accumulate their values, rather than switching between them. 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 action a is applicable in state s iff $\mathsf{pre}(a)[v] \in s[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$, and applying it in s changes the value of $v \in \mathcal{V}(\mathsf{eff}(a))$ to $s[v] \cup \mathsf{eff}(a)[v]$. An action sequence $\langle a_1, \ldots, a_k \rangle$ applicable in state s is an s-plan if $G[v] \in s[\![\langle a_1, \ldots, a_k \rangle]\!][v]$ for all $v \in \mathcal{V}(G)$. In all other respects, the semantics of MFDR is identical to that of FDR.

While FDR planning is PSPACE-complete even for propositional state variables, planning for MFDR tasks is polynomial time (Bylander 1994). Starting with the HSP (Bonet and Geffner 2001) and FF (Hoffmann and Nebel 2001) planning systems, exploiting this attractive property of MFDR for deriving heuristic estimates became a key ingredient of many modern 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 $\langle V, A, s, G \rangle$. If π^+ is a plan for Π^+, then π^+ is referred to as a **relaxed plan** for Π. 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^+.



(a) (b)

Figure 1: Our running example (a), and its causal graph (b).

A relaxed plan for our running example 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 it drops key B at position 1 (without having to first move back there).

For each of FDR and MFDR, we sometimes distinguish between planning tasks in terms of a pair of standard graphical structures induced by them. The **causal graph** CG_Π of a planning task Π is a digraph over vertices V. An arc (v, v') is in CG_Π iff $v \neq v'$ and there exists an action $a \in A$ such that $(v, v') \in [\mathcal{V}(\mathsf{eff}(a)) \cup \mathcal{V}(\mathsf{pre}(a))] \times \mathcal{V}(\mathsf{eff}(a))$. The **domain transition graph** $DTG_\Pi(v)$ of a variable $v \in V$ is an arc-labeled digraph over the vertices $\mathcal{D}(v)$ such that an arc (d, d') labeled with $\mathsf{pre}(a)[V \setminus \{v\}]$ belongs to the graph iff $\mathsf{eff}(a)[v] = d'$, and either $\mathsf{pre}(a)[v] = d$ or $v \notin \mathcal{V}(\mathsf{pre}(a))$.

Figure 1 (b) illustrates the definition of causal graphs. 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, which influences R.

Red-Black Relaxation

FDR and MFDR can be viewed as two extremes in which *all* state variables adopt value-switching and value-accumulating semantics, respectively. It is then obvious that there is an entire spectrum of possibilities in between the two extremes, 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} is a set of *black state variables*, V^{R} is a set of *red state variables*, and everything else is exactly as for FDR and MFDR tasks. The semantics is defined 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 $v \in \mathcal{V}^{\mathsf{B}}(\mathsf{eff}(a))$ to $\{\mathsf{eff}(a)[v]\}$, and changes the value of $v \in \mathcal{V}^{\mathsf{R}}(\mathsf{eff}(a))$ to $s[v] \cup \mathsf{eff}(a)[v]$.

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

In our running example, if we paint variables R, A, B, O red but paint F black, then the robot needs to drop key A before taking key B. If R is also black, then the robot needs to move back to position 1 before dropping key B.

RB is probably the most direct and intuitive generalization of both FDR and MFDR: Given an FDR task $\langle V, A, I, G\rangle$, we obtain an identical RB task as $\langle V, \emptyset, A, I, G\rangle$, and given an MFDR task $\langle V, A, I, G\rangle$, we obtain an identical RB task as $\langle \emptyset, V, A, I, G\rangle$. In other words:

Proposition 1 RB *generalizes both* FDR *and* MFDR.

We will sometimes refer to the **monotonic relaxation** of an RB task $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$, defined as the MFDR task $\Pi^+ = \langle V^{\mathsf{B}} \cup V^{\mathsf{R}}, A, I, G\rangle$.

The main operation we will use RB tasks for is to (partially) relax 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 $\Pi^{*+}_{V^{\mathsf{R}}} = \langle V \setminus V^{\mathsf{R}}, V^{\mathsf{R}}, A, I, G\rangle$. The **optimal red-black relaxation heuristic** $h^{*+}_{V^{\mathsf{R}}}(I)$ of Π relative to V^{R} is defined as the length of an optimal plan for $\Pi^{*+}_{V^{\mathsf{R}}}$. For arbitrary states s, $h^{*+}_{V^{\mathsf{R}}}(s)$ is defined via the RB task $\langle V \setminus V^{\mathsf{R}}, V^{\mathsf{R}}, A, s, G\rangle$. If π^{*+} is a plan for Π^{*+}, then we will refer to π^{*+} as a **red-black relaxed plan** for Π. Trivially, we have:

Proposition 2 *$h^{*+}_{V^{\mathsf{R}}}$ is consistent and dominates h^+. If $V^{\mathsf{R}} = \emptyset$, then $h^{*+}_{V^{\mathsf{R}}}$ is perfect.*

Some words are in order regarding two earlier works that "un-relax" particular kinds of state variables. Fox and Long (2001) automatically recognize transportation subproblems, and un-relax all vehicle position variables v. Keyder and Geffner (2008) un-relax any single variable v moving which does not have any side effects on other variables. In both cases, the actions moving v in a relaxed plan π^+ are replaced by an approximate TSP solution visiting the set $D(v, \pi^+) \subseteq \mathcal{D}(v)$ of values required in π^+. 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 $j_1, \ldots, j_n$ of jobs, each at a location d_i. Then the TSP approximation follows some path visiting each of $D(v, \pi^+) = \{d_1, \ldots, d_n\}$ once. In a red-black plan, v follows a path traversing the sequence $d_1, \ldots, d_n$.

The question remains how to actually compute h^{*+}. That computation is NP-hard because h^{*+} generalizes h^+. We adopt the popular strategy of upper-approximation by satisficing relaxed planning. Hence we investigate the tractability borderline of satisficing planning for RB tasks. One notion that will be central in this is the causal graph. While the causal graph CG_Π of an RB task Π is defined exactly as for FDR, the red/black coloring of its vertices suggests considering the structure of the two respective sub-graphs of CG_Π. By the **black causal graph** CG^{B}_Π of Π, we denote the subgraph of CG_Π induced by the black variables V^{B}.

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:

Algorithm
: RB2-PLANGEN(Π)
main
// $\Pi = \langle \{v_0\}, V^{\mathsf{R}}, A, I, G\rangle$
$R \leftarrow I$
$R \leftarrow R \cup$ RELAXEDFIXEDPOINT$(A_\emptyset \cup A_{I[v_0]})$
if $G \subseteq R$
  return "$solvable$"
for $I[v_0] \neq d \in \mathcal{D}(v_0), a_1 \in A_{I[v_0]\to d}$ s.t. $\mathsf{pre}(a_1) \subseteq R$
  $R \leftarrow R[\![a_1]\!]$
  $R \leftarrow R \cup$ RELAXEDFIXEDPOINT$(A_\emptyset \cup A_d)$
  if $G \subseteq R$
    return "$solvable$"
  for $a_2 \in A_{d\to I[v_0]}$ s.t. $\mathsf{pre}(a_2) \subseteq R$
    everywhere in Π and R,
      replace $I[v_0]$ and d with a new value $d_{\{I[v_0],d\}}$
    $R \leftarrow R \cup$ RELAXEDFIXEDPOINT(A)
    if $G \subseteq R$
      return "$solvable$"
return "$unsolvable$"

Figure 2: Algorithm used in the proof of Theorem 1.

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

A set $\{v_1, \ldots, v_n\}$ of black variables can be compiled into a single black variable v with domain $\bigotimes_{i=1}^n \mathcal{D}(v_i)$. This compilation is exponential in n, but incurs polynomial overhead for fixed n, so it suffices to prove Theorem 1 for the single-variable case. In what follows, we detail the proof for domain size 2. We will outline below the extension of that proof to arbitrary fixed-size domains; to ease that extension, some of our formulations for the binary-valued case are a little more general than needed.

A pseudo-code algorithm for solving RB tasks $\Pi = \langle \{v_0\}, V^{\mathsf{R}}, A, I, G\rangle$ with $|\mathcal{D}(v_0)| = 2$ is shown in Figure 2. The following notation is used:

- $A_\emptyset = \{a \in A \mid v_0 \notin \mathcal{V}(\mathsf{pre}(a)) \cup \mathcal{V}(\mathsf{eff}(a))\}$ are the actions that do not touch v_0 at all.
- For each value $d \in \mathcal{D}(v_0)$ of v_0, $A_d = \{a \in A \mid \mathsf{pre}(a)[v_0] = d, v_0 \notin \mathcal{V}(\mathsf{eff}(a))\}$ are the actions with a prevail condition on d.
- For each pair $d \neq d' \in \mathcal{D}(v_0)$ of values of v_0, $A_{d\to d'} = \{a \in A \mid \mathsf{pre}(a)[v_0] = d \neq d' = \mathsf{eff}(a)[v_0]\}$ are the actions changing v_0 from d to d'.

We assume wlog that $v_0 \in \mathcal{V}(\mathsf{eff}(a))$ implies $v_0 \in \mathcal{V}(\mathsf{pre}(a))$, and that $\mathsf{pre}(a)[v_0] \neq \mathsf{eff}(a)[v_0]$. Thus every action affecting v_0 is in exactly one of these sets.

The algorithm RB2-PLANGEN starts with (fully) relaxed planning on a restricted set of actions, then enters two nested for-loops, each of which ranges over a subset of actions, and each of whose bodies consists of another instance of relaxed planning. Obviously, the algorithm terminates in polynomial time. We need to prove that it succeeds if and only if a plan π for the (non-relaxed) RB task Π exists, and that we can efficiently construct such a plan in the positive case.

Assume first that π is a plan for Π. Say π does not use any action of the form $A_{I[v_0]\to d}$. Then the first relaxed planning fixed point R, achieved on the set $A_\emptyset \cup A_{I[v_0]}$ of actions that

either do not touch v_0 at all or are only prevailed by its initial value, reaches the goal, and RB2-PLANGEN succeeds as desired. If π does use an action affecting v_0, then the first such action a_1 is in $A_{I[v_0]\to d}$, and is found in the first for-loop. Application of (not relaxed) a_1 to R changes v_0 to d and adds all the red side effects of a. If π does not use any action from $A_{d\to I[v_0]}$, then the second relaxed planning fixed point R, now including also all facts that can be reached with the actions $A_\emptyset \cup A_d$, succeeds. If π does use an action a_2 from $A_{d\to I[v_0]}$, then a_2 will be found in the second for-loop. The algorithm then switches to relaxed planning over the entire action set – trivializing v_0 by replacing its two values with a new aggregate value that is true in the new initial state – and will definitely reach the goal.

Vice versa, if RB2-PLANGEN succeeds, we prove that we can construct a plan for Π. That is obvious if RB2-PLANGEN succeeds before the first for-loop, where the actions supporting R do not affect v_0, and all we need to do is sequence all the actions in the relaxed plan fixed point. If the algorithm succeeds in the first for-loop, then we attach to that sequence the action $a_1 \in A_{I[v_0]\to d}$ we are currently considering, and then again sequence all the actions in the relaxed plan fixed point. Finally, suppose that the algorithm reaches the point in the second for-loop where $I[v_0]$ and d are replaced with the new value $d_{\{I[v_0],d\}}$. In that case, we attach the culprit action $a_2 \in A_{d\to I[v_0]}$ to our prefix. This provides us with an action sequence π' that is applicable in the initial state I of Π, and in which a_1 switches v_0 from its initial value to d, and a_2 switches it back. Of these two actions, all preconditions other than those on v_0 are red, and are thus true in $I[\![\pi']\!]$, as well as in every state reachable from it. Hence, starting from $I[\![\pi']\!]$, we can switch the value of v_0 back and forth as much as we want. Replacing the two values of v_0 with $d_{\{I[v_0],d\}}$ represents exactly that. Given this, if the relaxed plan fixed point of the second for-loop returns a relaxed plan π^+, then a non-relaxed execution of π^+ in $I[\![\pi']\!]$ can be obtained simply by inserting a_1 in between any two actions of π^+ whenever v_0 currently has value $I[v_0]$ but d is needed, and inserting a_2 whenever v_0 currently has value d but $I[v_0]$ is needed. This ends in a plan for Π as desired.

Let us illustrate this with our running example, before considering domain sizes larger than 2. Say we paint all variables except F red. Then the first relaxed planning fixed point is unable to pick up key A, and ends in front of the lock. The first for-loop finds the action that picks up key A, and the second relaxed fixed point opens the lock and moves over to position 7, but cannot pick up key B. The second for-loop finds the action that drops key A, and the last relaxed fixed point reaches the goal. Note that, once the action dropping key A has become applicable, whenever we need to free the hand we can do so by dropping key A (because, A being painted red, that key remains in the hand).

To extend the algorithm to the more general case where the single black variable v_0 has a domain of size $k = O(1)$, observe that RB2-PLANGEN essentially consists of two action-choice branching points, captured by the two for-loops. For non-binary domains, we replace these nested loops with a tree search, each of whose branching points tries all actions changing the current value of v_0 to any other value of v_0. The key observation is that we can bound the depth of the tree search by a function depending only on k, that is, by a constant. Say that, during the search, we encounter a value d of v_0 that has already appeared on the active search path. Say further that D is the set of all v_0 values encountered on that search path between these two occurrences of d, including d itself. Then, similarly to the binary-valued case of $I[v_0]$ and d demonstrated above, from this point on we can switch back and forth between all values of D as needed. So we do not need to distinguish between these values anymore, and can replace D with a new aggregate value d_D. A search branch stops when either a branching point finds no suitable actions, or when the domain of v_0 has reduced to a single value. In the latter case, we execute a final relaxed plan fixed point over all actions, similarly to how it is done in the body of the second for-loop of RB2-PLANGEN. A simple upper bound on the number of steps after which that happens (if it happens) is $(k+1)(k-1)$: Discovering a duplicate takes at most $k+1$ steps, and each time that happens, we reduce the domain size of v_0 by at least 1. This concludes the proof of Theorem 1.

While the algorithm we just described 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.[2] 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 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.*

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 $x_1, \ldots, x_n$ which encode the propositional variables in φ, the domain of each x_i comprising true, false, and unassigned; (2) Boolean variables $c_1, \ldots, c_m$ which encode satisfaction of the clauses in φ; and (3) a variable v_0 with domain $\{1, \ldots, n+1\}$. Initially, $v_0 = 1$, all clause variables are false, and all the proposition variables are unassigned. 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 $v_0 = i$, and applying such an action also changes the value of v_0 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

[2]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.

only v_0 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 is bounded by the sum of the domain sizes of these neighbors. This concludes the proof of Theorem 2.

Theorem 3 *Plan existence for* RB *tasks where all black variables have fixed-size domains is* PSPACE*-complete, and is* NP*-complete even if* CG^{B}_{Π} *contains no arcs.*

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 (Bylander 1994), 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 previous proof, but now without the special variable v_0 (where thus the x_i variables can freely change their value from unassigned to either true or false). 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 above for Theorem 2: for each black variable, all causal graph neighbors are red so we get a polynomial bound on the number of moves required. This concludes the proof of Theorem 3.

We now relate Theorems 1-3 to the main characterization theorem of Chen and Giménez (2010), that establishes a general relation between FDR planning complexity and the structure of the causal graph. Adopting the notation of Chen and Giménez, for a digraph G, let $\#\mathsf{vertices}(G)$, $\mathsf{cc\text{-}size}(G)$, and $\mathsf{scc\text{-}size}(G)$ 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 $\#\mathsf{vertices}(\mathcal{G})$ is *bounded* if there exists a constant k such that $\#\mathsf{vertices}(G) \leq k$ for all $G \in \mathcal{G}$. Bounded $\mathsf{cc\text{-}size}(\mathcal{G})$ and bounded $\mathsf{scc\text{-}size}(\mathcal{G})$ are defined similarly.

$\mathsf{PlanExist}(\mathcal{G})$ and $\mathsf{PlanGen}(\mathcal{G})$ are the plan existence and plan generation problems restricted to FDR planning tasks whose causal graphs are elements of $\mathcal{G}$. The main characterization theorem of Chen and Giménez (2010) is as follows:

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 (Flum and Grohe 2006). 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 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.*

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 so that $s[\![\pi]\!] = I$. 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 (Helmert 2006).)

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. 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 iff the reachable part of each domain transition

graph is strongly connected. In contrast, even if RB is restricted to tasks with *empty* 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 contains no arcs.*

The proof is by reduction from DNF tautology testing. Given a propositional DNF formula φ over l clauses $c_1, \ldots, c_l$, consider an RB planning encoding $\Pi(\varphi)$ with: black variables $x_1, \ldots, x_n$ with $\mathcal{D}(x_i) = \{$*unassigned*, *true*, *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 $x_1, \ldots, x_n$. 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 $\{a_1, \ldots, a_l\}$, 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 $x_1, \ldots, x_n$ 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. This concludes the proof of Theorem 7.

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. This substantially extends Chen and Gimenez' tractability result (Theorem 6) for PlanExist($\mathcal{G}$) to the red-black setting.

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

Note the subtle difference in claims, between 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 the 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 scc-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 causal graphs. We show that red-black plan existence, in this setting, is equivalent to relaxed plan existence:

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

The "only if" direction is trivial. We prove the "if" direction with three successive lemmas. The first one makes the simple observation that, in any reversible RB task, we can achieve all reachable red values up front. We need a few notations. Let $\Gamma \subseteq \bigcup_{v \in V^{\mathsf{R}}} \mathcal{D}(v)$ be the minimal set such that, for every reachable state s, $s[V^r] \subseteq \Gamma$. Given a reachable state s, by $\Gamma(s)$ we denote the completion of s with all achievable red values, i. e., $\Gamma(s) = s \cup \Gamma$.

Lemma 1 *Let Π be a reversible* RB *task, and let s be a state reachable in Π. Then $\Gamma(s)$ is reachable in Π as well.*

We can complete I into $\Gamma(I)$ by going to every reachable state in turn, reverting the black vars to I in between every two states. In $\Gamma(I)$, the action sequence reaching s from I is applicable and leads to $\Gamma(s)$ as desired.

In our running example, say we paint all variables except F and O black. This yields a reversible RB task with acyclic black causal graph (compare Figure 1 (b)). To Γ-complete I into $\Gamma(I)$, we move to position 2, take key A, move to position 3, open the lock, and revert R and A to I.

Henceforth, we consider only Γ-completed states, which by Lemma 1 is not a restricting condition for red-black reachability. Our second lemma introduces the remaining prerequisite – acyclic black causal graph – and observes that we then have a degree of freedom in choosing *which* black variables to revert to their initial value. For the remainder of this section, let $\{v_1, \ldots, v_n\}$ be a topological ordering of V^{B} with respect to CG^{B}_{Π}. We can revert any top subset of the variables, leaving the remaining ones untouched:

Lemma 2 *Let $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G \rangle$ be a reversible* RB *task with acyclic black causal graph CG^{B}_{Π}, and s be a reachable state of Π with $\Gamma(s) = s$. Then, for any $v_i \in V^{\mathsf{B}}$, there exists a sequence of actions π applicable in s s.t. (i) $s[\![\pi]\!][v_1, \ldots, v_i] = I[v_1, \ldots, v_i]$, and (ii) π does not touch (in any precondition or effect) any of $v_{i+1}, \ldots, v_n$.*

By reversibility, there exists a reverting sequence ρ for s, i. e., a plan for $\langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, s, I[V^{\mathsf{B}}] \rangle$. Let ρ' be an action sequence obtained from ρ by removing all actions that have no effect on any of the variables $v_1, \ldots, v_i$. We show that ρ' has the desired properties (i) and (ii). Provided ρ' is applicable in s, its outcome is $I[v_j]$ for all vars v_j with $j \leq i$ because the responsible actions from ρ are contained in ρ', yielding property (i). To see applicability, first note that any red preconditions along ρ' are true simply because $s = \Gamma(s)$.

Algorithm
: UNRELAX($\Pi, \pi^+, g^{\mathsf{B}}$)
main
// $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ and $g^{\mathsf{B}} \subseteq I[\![\pi^+]\!]$
// Assume with Lemma 1 that $I = \Gamma(I)$
global $\pi \leftarrow \langle\rangle$
let $v_1, \ldots, v_n$ be a topological ordering of V^{B} wrt CG^{B}_{Π}
ACHIEVE($g^{\mathsf{B}} \cup I[\![\pi^+]\!][V^{\mathsf{R}}]$)
procedure ACHIEVE(g)
$g^{\mathsf{B}} \leftarrow g[V^{\mathsf{B}}]$ *// $g[V^{\mathsf{R}}] \subseteq \Gamma(I)$, see Lemma 3*
for $i = n$ **to** 1
if $v_i \notin \mathcal{V}(g^{\mathsf{B}})$
continue
let ρ be a reverting sequence for $v_1, \ldots, v_i$ *// Lemma 2*
$\pi \leftarrow \pi \circ \rho$
if $g^{\mathsf{B}}[v_i] = I[v_i]$
continue
let a be the first action in π^+ s.t. $\mathsf{eff}(a)[v_i] = g^{\mathsf{B}}[v_i]$
ACHIEVE($\mathsf{pre}(a)$)
$\pi \leftarrow \pi \circ \langle a\rangle$

Figure 3: Algorithm used in the proof of Theorem 9.

By acyclicity of CG^{B}_{Π} every action affects at most one black variable, so by construction of ρ' every action a in ρ' affects exactly one v_j where $j \leq i$. But then, since $\{v_1, \ldots, v_n\}$ is a topological ordering of V^{B}, any black preconditions of a are on variables v_l for $l \leq j \leq i$, and the actions supporting them in ρ (if needed) are contained in ρ'. So ρ' is applicable in s. These arguments also show that neither the effects nor the preconditions of the actions in ρ' touch the variables $v_{i+1}, \ldots, v_n$. Thus we have (ii), concluding the argument.

In our running example, say we have completed I into $\Gamma(I)$ as above (trivializing the variables F and O), then executed an arbitrary action sequence moving around the robot and keys. If ρ reverts to I, then clearly ρ' obtained by removing all except the robot moves from ρ reverts only R to I. Similar if we remove all except the moves of R and A.

We now have the machinery in place for our last and central lemma, proving that we can find a red-black plan π for any sub-goal achieved in a relaxed plan π^+. Figure 3 depicts pseudo-code for the construction of π.

Lemma 3 *Let $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ be a reversible* RB *task with acyclic black causal graph and $I = \Gamma(I)$, 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) $I[\![\pi]\!][V^{\mathsf{R}}] \supseteq I[\![\pi^+]\!][V^{\mathsf{R}}]$.*

The proof is by induction over the length of π^+. 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) π does not touch (neither in preconditions nor in effects) any black variable v_j with $j > m(g^{\mathsf{B}})$, where $m(g) = \max\{i \mid v_i \in \mathcal{V}(g)\}$ denotes the largest variable index in any partial assignment g.

For the base case, if π^+ is empty then the claim is trivially satisfied by the empty sequence π. For the inductive case, assume that the claim holds for all relaxed plans of length k; we show it for π^+ of length $k+1$.

First, consider the sequence π^+_k containing the first k actions of π^+; denote the last action in π^+ by a_{k+1}. By induction assumption applied to π^+_k, we can reach a state s_k in Π so that $\mathsf{pre}(a_{k+1}) \subseteq s_k$ and $I[\![\pi^+_k]\!][V^{\mathsf{R}}] \subseteq s_k$. Given this, we can apply a_{k+1} to s_k, which obviously will yield a state s_{k+1} where $I[\![\pi^+]\!][V^{\mathsf{R}}] \subseteq s_{k+1}$. 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 Γ-completed, i. e., $I = \Gamma(I)$. Thus, part (ii) of the claim is trivially satisfied, for any sequence π we construct.

We now show how to construct an action sequence π achieving g^{B}. We start with $\pi = \langle\rangle$. We process the variables in g^{B} in descending order, from v_n to v_1. If $v_i \notin \mathcal{V}(g^{\mathsf{B}})$, there is nothing to do. Else, we first invoke Lemma 2 to obtain a sequence of actions reverting the variables v_j for $j \leq i$, without touching any v_j for $j > i$. We append that sequence to π. If $g^{\mathsf{B}}[v_i] = I[v_i]$, then we have achieved this sub-goal and proceed with the next variable v_{i-1} (or stop if $i = 1$). Otherwise, π^+ must contain an action a that achieves $g^{\mathsf{B}}[v_i]$. Applying the induction assumption to the prefix π^+_a of π^+ that precedes a, we obtain a sequence π_a that is both applicable in Π and achieves $\mathsf{pre}(a)$ without touching any variables indexed higher than $m(\mathsf{pre}(a))$. Clearly, $m(\mathsf{pre}(a)) \leq i$: a affects v_i and can thus only be preconditioned on v_i itself and variables preceding it. But then, π_a is applicable at the end state $I[\![\pi]\!]$ of our current sequence π, because $I[\![\pi]\!][v_1, \ldots, v_i] = I[v_1, \ldots, v_i]$. Given that, we append π_a, and then a, to π. This extended sequence still does not touch any variables v_j for $j > i$, and thus achieves $g^{\mathsf{B}}[v_i]$ while satisfying property (iii). Continuing with the next lower variable, we achieve the rest of g^{B} without invalidating any sub-goals already achieved, thus accomplishing property (i). This concludes the argument.

Together, Lemma 3 and Lemma 1 finish the proof of Theorem 9: Given a relaxed plan, first turn I into $\Gamma(I)$ using Lemma 1, then construct a plan for G using Lemma 3.

To illustrate Lemma 3 with our running example, consider the point, in the relaxed plan π^+ for the task, where both keys have been picked up. The construction of π, for $g^{\mathsf{B}} := \{R = 7, A = R, B = R\}$, proceeds as follows. We first Γ-complete I into $\Gamma(I)$ as above. We then achieve $B = R$ by the π^+ action a taking key B at position 7. We recursively establish a's precondition by moving R from 1 to 7. We proceed to the sub-goal $A = R$. We revert R to I (moving back to 1), then use the π^+ action a taking key A at position 2, whose precondition we achieve recursively by moving from 1 to 2. Finally, working on the sub-goal $R = 7$, we revert R to I and then move it from 1 to 7 at which point π has the desired properties (i) and (ii).

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 over-estimation when used inside a heuristic function. Much worse, while the proof of Lemma 1 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 1 (b)). Thus, it is unclear how to tackle red and black variables in combination, rather than separating the red variables out using Lemma 1. In particular, we can *not* in general follow the ordering of values achieved in the relaxed plan, because that might require reverting black variables, which in turn might require red values not achieved by the relaxed plan yet.

Red-Black Relaxed Plan Heuristics

Our ultimate objective in this investigation is to enable the design of practical heuristic functions that will improve on the commonly used approximations of h^+, by approximating h^{*+} for tractable fragments of RB instead. Theorem 1 provides a constructive proof for red-black plan generation, yet the asymptotic runtime required suggests that this approach is not generally practical. As for the tractable fragment identified by Theorem 8, denoted in what follows by rSCC, this pertains to red-black plan existence rather than red-black plan generation. So rSCC cannot be used directly for approximating h^{*+}, and anyhow, its reversibility prerequisite cannot be checked efficiently. We get back to this in the conclusion. For now, we provide a first analysis of the proposed approach from a theoretical perspective, by identifying benchmark domains in which h^{*+} is perfect within the tractable fragment rSCC. We start with a simple sufficient condition for h^{*+} to be perfect.

Lemma 4 *Let Π be an FDR planning task and V' be a subset of its variables. If all variables in V' have no outgoing arcs in CG_Π, then $h^{*+}_{V'}$ is perfect.*

Indeed, in this setting, all non-redundant red-black relaxed plans (that do not contain any superfluous actions), and thus in particular all optimal red-black relaxed plans, are valid plans for the original task Π. To see this, observe that 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 activities in the red-black relaxed plan. This concludes the argument.

Lemma 4 can be applied to natural FDR encodings of several IPC benchmark domains:

Theorem 10 *In any planning task Π of the* LOGISTICS*,* MICONIC*,* SATELLITE*,* VISIT-ALL*, and* ZENOTRAVEL *domains, one can choose a variable subset V^{R} so that $h^{*+}_{V^{\mathsf{R}}}$ is perfect and $\Pi^{*+}_{V^{\mathsf{R}}} \in$ rSCC.*

In LOGISTICS and MICONIC, sets of red variables satisfying the claim are those representing the locations of the packages and passengers, respectively. These variables are causal graph leaves, and so we can apply Lemma 4 to obtain the first half of the claim. For the second half, the resulting red-black planning tasks are reversible and their black causal graphs are acyclic. The same argument can be applied to (the straightforward FDR encoding of) any transportation domain with reversible road maps and with no capacity and fuel constraints. If vehicles are associated with individual replenishable fuel supplies, then reversibility is maintained while the strongly connected components in the black causal graph have size 2 (each vehicle location variable forms a cycle with the respective fuel variable). This implies the result for ZENOTRAVEL. The argument does *not* hold up in the presence of capacity constraints, however, because those introduce cycles between the capacity variables and the object locations. Indeed, in the GRIPPER domain, a set of red variables as claimed by Theorem 10 does not exist. For example, if the red variables are the ball locations, then $\Pi^{*+}_{V^{\mathsf{R}}} \in$ rSCC but $h^{*+}_{V^{\mathsf{R}}}$ under-estimates significantly because we can free the gripper hands by dropping balls in the wrong room without having to pick them up again.

In SATELLITE, we assume an FDR encoding that associates each satellite with 3 variables encoding the direction, which instrument is presently powered on, and which instrument is presently calibrated. Choosing the image variables to be red, the only black cycles are those between each power and calibration variable, and are thus of bounded size.

In VISIT-ALL, we choose the "visited" variables to be red, and obtain $\Pi^{*+}_{V^{\mathsf{R}}} \in$ rSCC. On the one hand, Lemma 4 does not apply here directly because the actions changing the location also mark, as a side effect, the target location as being visited. On the other hand, these side effects are not harmful because the values they delete – location "not being visited" – are required neither by action preconditions nor by the goal, and are thus irrelevant. Lemma 4 can easily be extended to capture this situation.[3]

Conclusion

Red-black planning relaxes only a subset of the state variables, thus generalizing both regular and delete-relaxed planning, allowing to smoothly interpolate between the two. We have extended Chen and Gimenez' characterization theorems to this more general setting, and identified two significant islands of tractability. Theoretical evidence suggests that at least one of these (rSCC) can, in principle, yield highly informed heuristic functions.

The main obstacle in making the approach practical is that the tractability of rSCC pertains not to plan generation, needed to compute a heuristic, but to plan existence. We believe that this can be addressed by replacing the notion of reversibility with that of invertibility (as defined, e. g., by Hoffmann, 2011). A variable is invertible if every arc (d, d') in its DTG has an inverse (d', d) whose condition is contained in that of (d, d'). This can be tested efficiently, and red-black reversibility follows if all black variables are invertible. Our initial results indicate that the construction

[3]Note that, for LOGISTICS and SATELLITE, even $V^{\mathsf{R}} = \emptyset$ satisfies the claim of Theorem 10, since the original tasks are reversible, and their causal graphs have bounded-size SCCs. This is not so for the other domains concerned: MICONIC and SATELLITE are not reversible, VISIT-ALL satisfies neither requirement.

underlying Theorem 9 can be simplified in this setting, allowing to generate red-black plans efficiently.

Acknowledgments. Carmel Domshlak's work was partially supported by ISF grant 1045/12 and the Technion-Microsoft E-Commerce Research Center.

References

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds. 2012. *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*. AAAI Press.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1–2):165–204.

Cai, D.; Hoffmann, J.; and Helmert, M. 2009. Enhancing the context-enhanced additive heuristic with precedence constraints. In Gerevini et al. (2009), 50–57.

Chen, H., and Giménez, O. 2010. Causal graphs and structurally restricted planning. *Journal of Computer and System Sciences* 76(7):579–592.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Springer-Verlag.

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

Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds. 2009. *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*. AAAI Press.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research* 20:239–290.

Haslum, P. 2012. Incremental lower bounds for additive cost planning problems. In Bonet et al. (2012), 74–82.

Helmert, M., and Geffner, H. 2008. Unifying the causal graph and additive heuristics. In Rintanen, J.; Nebel, B.; Beck, J. C.; and Hansen, E., eds., *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*, 140–147. AAAI Press.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In Koenig, S.; Zilberstein, S.; and Koehler, J., eds., *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 161–170. Whistler, Canada: Morgan Kaufmann.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

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

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

Keyder, E., and Geffner, H. 2008. Heuristics for planning with action costs revisited. In Ghallab, M., ed., *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, 588–592. Patras, Greece: Wiley.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2012. Semi-relaxed plan heuristics. In Bonet et al. (2012).