Unchaining the Power of Partial Delete Relaxation, Part II: Finding Plans with Red-Black State Space Search (Technical Report)

Maximilian Fickert, Daniel Gnad, and **Jörg Hoffmann**
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
{fickert,gnad,hoffmann}@cs.uni-saarland.de

Abstract

Red-black relaxation in classical planning allows to interpolate between delete-relaxed and real planning. Yet the traditional use of relaxations to generate heuristics restricts relaxation usage to tractable fragments. How to actually tap into the red-black relaxation's interpolation power? Prior work has devised *red-black state space search (RBS)* for intractable red-black planning, and has explored two uses: proving unsolvability, generating seed plans for plan repair. Here, we explore the generation of plans directly through RBS. We design two enhancements to this end: (A) use a known tractable fragment where possible, use RBS for the intractable parts; (B) check RBS state transitions for *realizability*, spawn relaxation refinements where the check fails. We show the potential merits of both techniques on IPC benchmarks.

1 Introduction

Relaxations are prominently used in AI Planning for the generation of heuristic functions (e. g. [Bonet and Geffner, 2001; Hoffmann and Nebel, 2001; Helmert and Domshlak, 2009; Helmert *et al.*, 2014]). The *delete relaxation* in particular has been highly influential. Under this relaxation, state variables accumulate their values rather than switching between them.

The delete relaxation cannot account for *having to move to-and-fro*, and it ignores *resource consumption*. Hence there is a lot of work on taking some deletes into account (e. g. [Fox and Long, 2001; Helmert and Geffner, 2008; Haslum, 2012; Coles *et al.*, 2013; Keyder *et al.*, 2014]). Here we consider *red-black planning* [Domshlak *et al.*, 2015], a *partial delete relaxation* method that allows to force delete-relaxed plans to behave like real plans in the limit. A subset of ("red") variables take the delete-relaxed semantics, accumulating values, while the remaining ("black") ones retain the true semantics.

The partition into red and black variables is called a *painting*, and its choice obviously allows to interpolate between delete-relaxed and real planning. Yet for use as a heuristic function, the painting must be chosen so that red-black plan generation is tractable. Prior work therefore restricts the black variables to what we will refer to as ACI, with *acyclic* causal-graph dependencies and *invertible* value-transitions.

Acyclic dependencies and invertible value-transitions occur only in small parts of practical planning tasks, so ACI is typically very far from real planning. How can we actually tap into the interpolation power of red-black planning?

We follow up on prior work on this question [Gnad *et al.*, 2016] (*Gnad16* in what follows). Gnad16 have shown how to generate red-black plans for arbitrary paintings, via *red-black state space search (RBS)*, a hybrid of forward search and delete-relaxed planning, where every transition contains a local delete-relaxed planning step over the red variables. Gnad16 explored 1) the generation of red-black seed plans for plan repair with LPG [Gerevini *et al.*, 2003; Fox *et al.*, 2006]; and 2) proving planning tasks unsolvable within the red-black relaxation, via an iteration of more and more refined RBS searches (more and more black variables).

Here, we explore the use of RBS for generating plans. This is the natural complement of 2), in what we envision as a red-black relaxation refinement process. The challenge is to make RBS produce real plans early on, with few black variables. We design two enhancements to this end:

A) We create synergy between RBS and ACI, by replacing delete-relaxed planning with ACI planning in RBS. This uses ACI where possible (e. g., moving to-and-fro on an invertible road map), and uses RBS where not (e. g., non-invertible resource consumption). We identify a maximally permissive condition on the black-variable dependencies under which this combination is possible.

B) We design an adaptive variant of refinement, locally within a single RBS search space where needed. We check every transition $s \xrightarrow{a} s'$ for *realizability* of the red parts, i. e., whether the delete-relaxed plan here works in reality. Non-realizable transitions are pruned, and spawn *refinement options*: red-black planning tasks starting at s, with additional black variables addressing the non-realizability of $s \xrightarrow{a} s'$. The refinement options become search nodes in an overall heuristic search.

We evaluate our techniques on the IPC benchmarks. In overall performance, A) is competitive, while B) often suffers from too many refinement options. Compared to Gnad16's approach 1), A) is better overall, and both A) and B) are highly complementary to 1) per domain. In five domains, our best configurations outperform the state-of-the-art systems LAMA and Mercury by large margins.

2 Preliminaries

We use the *finite-domain representation (FDR)* framework [Bäckström and Nebel, 1995; Helmert, 2009]. An FDR **planning task** is a tuple $\Pi = (V, A, I, G)$. V is a set of **variables** v, each with a finite domain D_v. A complete assignment to V is a **state**. I is the **initial state**, and the **goal** G is a partial assignment to V. A is a finite set of **actions**, where each $a \in A$ is a triple $(\mathsf{pre}_a, \mathsf{eff}_a, c_a)$. The **precondition** pre_a and the **effect** eff_a are partial assignments to V; $c_a \in \mathbb{R}_0^+$ is the action's **cost**. We will sometimes refer to variable-value pairs $v = d$ as **facts**. For a partial assignment p, $\mathcal{V}(p)$ denotes the set of variables instantiated by p. For $V' \subseteq \mathcal{V}(p)$, by $p[V'] := p|_{V'}$ we denote the restriction of p to V'. An action a is **applicable** in a state s if $s[\mathcal{V}(\mathsf{pre}_a)] = \mathsf{pre}_a$. The outcome state $s[\![a]\!]$ is like s except that $s[\![a]\!](v) = \mathsf{eff}_a(v)$ for each $v \in \mathcal{V}(\mathsf{eff}_a)$.

A **transition system** is a tuple $\Theta = (S, L, T, s_0, S_G)$. S is a set of states. L is a set of *labels*. $T \subseteq S \times L \times S$ is a set of **transitions**. $s_0 \in S$ is the **start state** and $S_G \subseteq S$ is the set of **goal states**. A **plan** for a state s is a transition path from s to a state in S_G. The **state space** of Π is the transition system Θ_Π where S is the set of states in Π, $L = A$, $(s, a, s') \in T$ iff a is applicable in s and $s' = s[\![a]\!]$, $s_0 = I$, and $s \in S_G$ if $s[\mathcal{V}(G)] = G$. A plan π for I in Θ_Π is called a **plan** for Π.

The **causal graph** (e. g. [Jonsson and Bäckström, 1995; Helmert, 2006]) is a digraph with vertices V and an arc (v, v') if $v \neq v'$ and there exists an action $a \in A$ such that $(v, v') \in [\mathcal{V}(\mathsf{eff}(a)) \cup \mathcal{V}(\mathsf{pre}(a))] \times \mathcal{V}(\mathsf{eff}(a))$.

3 Red-Black Planning

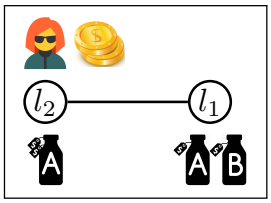
We next give an overview of red-black planning and associated techniques, as needed to understand our contribution.

3.1 Definitions

A **red-black planning task**, or **RB task**, is a tuple $\Pi^{\mathsf{RB}} = (V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G)$ with $V^{\mathsf{B}} \cap V^{\mathsf{R}} = \emptyset$, where $\Pi := (V, A, I, G)$ is an FDR task with $V := V^{\mathsf{B}} \cup V^{\mathsf{R}}$. V^{B} is the set of **black variables**, V^{R} is the set of **red variables**. States are now **RB states** s^{RB}, which map each variable v to a subset of its domain, $s^{\mathsf{RB}}(v) \subseteq D_v$, where $|s^{\mathsf{RB}}(v)| = 1$ for $v \in V^{\mathsf{B}}$. In the **RB initial state** s_0^{RB} each variable v is mapped to $\{I(v)\}$. **RB goal states** are those s^{RB} where $G(v) \in s^{\mathsf{RB}}(v)$ for all $v \in \mathcal{V}(G)$. An action a is applicable in an RB state s^{RB} if $\mathsf{pre}_a(v) \in s^{\mathsf{RB}}(v)$ for all $v \in \mathcal{V}(\mathsf{pre}_a)$. Upon executing a in s^{RB}, $v \in \mathcal{V}(\mathsf{eff}_a) \cap V^{\mathsf{B}}$ is set to $\{\mathsf{eff}_a(v)\}$, and $v \in \mathcal{V}(\mathsf{eff}_a) \cap V^{\mathsf{R}}$ is set to $s^{\mathsf{RB}}(v) \cup \{\mathsf{eff}_a(v)\}$. The outcome state is denoted $s^{\mathsf{RB}}[\![a]\!]$. A plan π^{RB} under this semantics is an **RB plan** for Π^{RB}. We also refer to π^{RB} as an RB plan for Π, viewing Π^{RB} as a **red-black relaxation** of Π, where the choice of V^{B} vs. V^{R} is a **painting** defining the relaxation.

The red-black relaxations of any FDR task Π form a **refinement hierarchy**, with more refined relaxations having larger sets V^{B}. At the extremes, for $V^{\mathsf{B}} = V$ we obtain real planning, and for $V^{\mathsf{B}} = \emptyset$ we obtain fully delete-relaxed planning.

Example 1. *Our example task Π has variables $V = \{T, M, A, B\}$ with domains $D_T = \{l_1, l_2\}$, $D_M = \{0, 1, 2\}$, $D_A = \{0, 1\}$, $D_B = \{0, 1\}$. T encodes a traveling agent with two locations l_1 and l_2, initially l_2. The goal is to be at l_2, and to possess each product A and B. Each product is available at l_1 at price 1; A is also available at l_2, but at price 2. M is the available money. The actions have the form $go(l, l')$ and $buy(l, p, m)$. For example, $go(l_1, l_2)$ has precondition $\{T = l_1\}$ and effect $\{T = l_2\}$, and $buy(l_1, A, 2)$ has precondition $\{T = l_1, M = 2\}$ and effect $\{A = 1, M = 1\}$.*



*A fully delete-relaxed plan for this task has two **flaws**: 1) it does not go back from l_1 to l_2; 2) it may choose to buy A at l_2 instead of l_1, over-spending the budget. We can fix 1) by painting T black, and we can fix 2) by painting M black. In the red-black relaxation where $V^{\mathsf{B}} = \{T, M\}$ and $V^{\mathsf{R}} = \{A, B\}$, every RB plan for Π is a real plan for Π.*

3.2 Tractable Fragment: ACI

The initial line of work on red-black planning [Domshlak *et al.*, 2015], culminating in the Mercury system's success at IPC'14 [Katz and Hoffmann, 2014], generates a heuristic function based on the tractable fragment **ACI**. We simplify some details in what follows, for easier exposition.

ACI requires 1) that the causal graph over the black variables is acyclic, and 2) that every black variable is **invertible**. A variable v is invertible if every value transition can be inverted under the same (or easier) conditions on other variables. An RB plan can then be generated by finding a fully delete-relaxed plan π^+, and running **ACI plan repair** on π^+ to obtain an RB plan π^{RB}. The repair process executes π^+ step-by-step under the red-black semantics; whenever a condition (precondition or goal) g on V^{B} is not satisfied, the process inserts a subsequence π achieving g. The latter is always possible, in time polynomial in the length of π: thanks to 1), V^{B} can be solved in a sequence from clients (variables which can only be modified through actions depending other variables) to servants (the dependent variables); thanks to 2), whenever a servant v must provide a value $d \in D_v$ for a client, v can reach d from its current value.[1]

Example 2. *In Example 1, T is invertible. A relaxed plan is $\pi^+ = \langle buy(l_2, A, 2), go(l_2, l_1), buy(l_1, B, 2)\rangle$. ACI plan repair with $V^{\mathsf{B}} = \{T\}$ finds flaw 1), π^+ does not satisfy the goal $T = l_2$. It inserts $go(l_1, l_2)$ at the end to fix that.*

Given an FDR task Π, the **painting strategies** associated with ACI choose V^{R} so as to guarantee that the resulting relaxed task Π^{RB} is in ACI. A major weakness in practice here is the restriction of V^{B} to invertible variables. In our example, T is the *only* such variable; we cannot paint M black, so we cannot fix flaw 2) pertaining to money consumption.

Intuitively, using ACI instead of full delete relaxation fixes the "moving to-and-fro" issue, for invertible moves now painted black (here: T). But it does not address resource consumption, which involves non-invertible variables (here: M).

3.3 Red-Black State Space Search

To enable convergence to real planning in the limit, red-black planning methods are required that can handle arbitrary pain-

[1] In our implementation, we adapted *red facts following*, the more advanced repair algorithm by Katz and Hoffmann [2013].

tings. Addressing this, Gnad et al. [2016] (*Gnad16*) have introduced **red-black state space search (RBS)**. RBS performs forward search with a relaxed fixed point over the red variables at each transition. At plan extraction time, RBS augments the solution path with a relaxed plan at each transition.

We require some notations. The **red actions** in an RB state s^{RB}, denoted $A^{\mathsf{R}}(s^{\mathsf{RB}})$, are the actions available to the relaxed fixed point at s^{RB}: the actions that comply with the black-variable values. $A^{\mathsf{R}}(s^{\mathsf{RB}}) := \{a^{\mathsf{R}} \mid a \in A, \mathsf{pre}_a[V^{\mathsf{B}}] \subseteq s^{\mathsf{RB}}, \mathsf{eff}_a[V^{\mathsf{B}}] \subseteq s^{\mathsf{RB}}\}$, where a^{R} is the projection of a onto V^{R}.

The relaxed fixed point at s^{RB} is now formalized in terms of a local planning task, namely the RB task $\Pi^+(s^{\mathsf{RB}}) := (\emptyset, V^{\mathsf{R}}, A^{\mathsf{R}}(s^{\mathsf{RB}}), s^{\mathsf{RB}}[V^{\mathsf{R}}], \emptyset)$. The **red completion** of s^{RB} is the RB state $\mathcal{F}^+(s^{\mathsf{RB}})$ where $\mathcal{F}^+(s^{\mathsf{RB}})[V^{\mathsf{B}}] = s^{\mathsf{RB}}[V^{\mathsf{B}}]$, and $\mathcal{F}^+(s^{\mathsf{RB}})[V^{\mathsf{R}}]$ is the set of all facts reachable in $\Pi^+(s^{\mathsf{RB}})$.

Definition 1 (Gnad16). *Let Π^{RB} be an RB planning task. The **RB state space** is the transition system $\Theta^{\mathsf{RB}} = (S^{\mathsf{RB}}, T^{\mathsf{RB}}, A, s_0^{\mathsf{RB}}, S_G^{\mathsf{RB}})$. S^{RB} is the set of RB states. s_0^{RB} is the RB initial state. $S_G^{\mathsf{RB}} = \{s^{\mathsf{RB}} \mid \mathcal{F}^+(s^{\mathsf{RB}})$ is RB goal state$\}$. T^{RB} is the set of transitions $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ where a is applicable to $\mathcal{F}^+(s^{\mathsf{RB}})$, $\mathsf{eff}(a)[V^{\mathsf{B}}] \not\subseteq s^{\mathsf{RB}}[V^{\mathsf{B}}]$, and $t^{\mathsf{RB}} = \mathcal{F}^+(s^{\mathsf{RB}})[\![a]\!]$.*

Example 3. *Setting $V^{\mathsf{B}} = \{M\}$, $\mathcal{F}^+(s_0^{\mathsf{RB}})$ contains $T = l_1$ and $T = l_2$, but neither $A = 1$ nor $B = 1$ as buying a product affects the black variable M. The outgoing transitions of s_0^{RB} are the buy actions. $\langle buy(l_1, A, 2), buy(l_1, B, 1)\rangle$ leads to an RB goal state. For $buy(l_2, A, 2)$, in contrast, the outcome RB state t^{RB} has $t^{\mathsf{RB}}(M) = \{0\}$, so no further actions are applicable here and we detect that this is a dead-end.*

RB plan extraction augments backward solution path extraction with a relaxed plan extraction step at each transition. Assume that $\pi = \langle a_0, \ldots, a_{n-1}\rangle$ is a plan for Θ^{RB}, assume that backward extraction has already extracted an RB plan for the postfix $\pi_k := \langle a_k, \ldots, a_{n-1}\rangle$, and assume that the transition taken by a_{k-1} in π is $s_{k-1}^{\mathsf{RB}} \xrightarrow{a_{k-1}} s_k^{\mathsf{RB}}$. Then the **red goal** for relaxed plan extraction at this transition is $G(s_{k-1}^{\mathsf{RB}}) := \mathsf{Regress}^{\mathsf{R}}(G, a_{k-1} \circ \pi_k) \backslash s_{k-1}^{\mathsf{RB}}[V^{\mathsf{R}}]$, where $\mathsf{Regress}^{\mathsf{R}}$ is regression in the projection onto V^{R}. Intuitively, $G(s_{k-1}^{\mathsf{RB}})$ is the set of red facts that must be achieved before a_{k-1}, and that cannot be achieved further below. Any relaxed plan extraction mechanism can now be used on $\Pi^+(s_{k-1}^{\mathsf{RB}})$ to find a relaxed plan $\pi^+(s_{k-1}^{\mathsf{RB}})$ achieving $G(s_{k-1}^{\mathsf{RB}})$. Then π_k is replaced by $\pi^+(s_{k-1}^{\mathsf{RB}}) \circ a_{k-1} \circ \pi_k$, and we iterate.

Example 4. *In Example 3, denote $\pi = \langle buy(l_1, A, 2), buy(l_1, B, 1)\rangle = \langle a_0, a_1\rangle$. Denote the RB states along π as $s_0^{\mathsf{RB}}, s_1^{\mathsf{RB}}, s_2^{\mathsf{RB}}$. Plan extraction first processes $s_1^{\mathsf{RB}} \xrightarrow{a_1} s_2^{\mathsf{RB}}$. The red goal here is $G(s_1^{\mathsf{RB}}) = \emptyset$, as $\mathsf{Regress}^{\mathsf{R}}(\{A = 1, B = 1\}, buy(l_1, B, 1)) = \{T = l_1, A = 1\}$ and $s_1^{\mathsf{RB}}[V^{\mathsf{R}}] = \{T = l_2, T = l_1, A = 0, A = 1, B = 0\}$. The postfix thus simply is $\pi_1 = \langle buy(l_1, B, 1)\rangle$. In the next step though, at $s_0^{\mathsf{RB}} \xrightarrow{a_0} s_1^{\mathsf{RB}}$, the red goal is $G(s_0^{\mathsf{RB}}) = \{T = l_1\}$, leading to the relaxed plan $\langle go(l_2, l_1)\rangle$ and thus to the overall red-black plan $\pi^{\mathsf{RB}} = \langle go(l_2, l_1), buy(l_1, A, 2), buy(l_1, B, 1)\rangle$.*

Observe that π^{RB} in Example 4 is correct about M, but is flawed regarding T (as π^{RB} does not go back from l_1 to l_2 at the end, leaving the goal $T = l_2$ unsatisfied). This is complementary to the tractable fragment ACI, which can fix T but cannot fix M (cf. Example 2). The first new method we propose here is motivated by this kind of complementarity. We combine RBS with ACI to handle each kind of flaw with the most appropriate method.

4 Combining RBS with ACI

Any flaw in an RB plan π^{RB} can in principle be fixed by painting the respective variable v black, $V^{\mathsf{B}} := V^{\mathsf{B}} \cup \{v\}$, and re-running RBS. Yet Θ^{RB} grows exponentially in $|V^{\mathsf{B}}|$. Can we avoid the computational cost incurred by painting v black?

As we now show, the answer is yes – if, like for $v = T$ in Example 4, we can handle v by ACI instead. We can use ACI to effectively handle a tractable part of the task at hand (e. g. invertible moves to-and-fro), combined with RBS to handle the remainder (e. g. resource consumption).

4.1 The RBS+ACI Framework

Our combined framework, that we baptize **RBS+ACI**, distinguishes black variables of two different kinds, handled by RBS vs. ACI. So a painting is now a partition of V into three subsets $V^{\mathsf{RBS}}, V^{\mathsf{ACI}}, V^{\mathsf{R}}$ where $V^{\mathsf{B}} = V^{\mathsf{RBS}} \cup V^{\mathsf{ACI}}$.

Assume that such a partition is given. We need an RB plan relative to the entire set V^{B} of black variables, i. e. for the RB task $(V^{\mathsf{RBS}} \cup V^{\mathsf{ACI}}, V^{\mathsf{R}}, A, I, G)$. The basic idea is to apply ACI plan repair on the outcome of RBS on the coarser (more relaxed) task $\Pi_+^{\mathsf{RB}} := (V^{\mathsf{RBS}}, V^{\mathsf{R}} \cup V^{\mathsf{ACI}}, A, I, G)$.

ACI plan repair is defined for fully delete-relaxed plans, not RB plans, so we must adapt the repair process. We must make sure that the repair *1) is always possible given the black part V^{RBS} already fixed*, and *2) never affects that fixed part*.

Let π be the plan found by RBS for Π_+^{RB}. Our adapted repair process, **RBS+ACI plan repair**, computes a plan without conflicts on the entire set of black variables $V^{\mathsf{RBS}} \cup V^{\mathsf{ACI}}$, fixing unsatisfied conditions only on V^{ACI} without modifying the conflict-free V^{RBS}.

To ensure 2), an obvious and natural requirement is that there is no $a \in A$ with $\mathcal{V}(\mathsf{eff}_a) \cap V^{\mathsf{ACI}} \neq \emptyset$ and $\mathcal{V}(\mathsf{eff}_a) \cap V^{\mathsf{RBS}} \neq \emptyset$. That is, the repair actions will never affect V^{RBS}.

Ensuring 1) is more tricky. In RBS on Π_+^{RB}, the red completion $\mathcal{F}^+(s^{\mathsf{RB}})$ of any state s^{RB} uses only actions whose precondition is satisfied given the black variable assignment $s^{\mathsf{RB}}[V^{\mathsf{RBS}}]$. So one may think (and we did think at first) that no further restrictions are needed. However, across transitions $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$, the fixed repair context changes from $s^{\mathsf{RB}}[V^{\mathsf{RBS}}]$ to $t^{\mathsf{RB}}[V^{\mathsf{RBS}}]$. This causes problems because, during RBS, the values reached for V^{ACI} in $\mathcal{F}^+(s^{\mathsf{RB}})$ are propagated to t^{RB}. But due to the different context $t^{\mathsf{RB}}[V^{\mathsf{RBS}}]$, the repair process at t^{RB} cannot necessarily reach these values.

Similar to Gnad and Hoffmann [2015], we impose that there is no $a \in A$ with $\mathcal{V}(\mathsf{eff}_a) \cap V^{\mathsf{ACI}} \neq \emptyset$ and $\mathcal{V}(\mathsf{pre}_a) \cap V^{\mathsf{RBS}} \neq \emptyset$, i. e., the repair actions do not have preconditions on V^{RBS}. We next show that this restriction is sufficient (the repair will always work). We then show that the restriction is necessary for computational reasons.

The conjunction of our two restrictions is equivalent to the absence of a causal graph arc from V^{RBS} to V^{ACI}. We say in this case that V^{ACI} **does not depend on** V^{RBS}.

Proposition 1. *Given an RB planning task* $\Pi^{\mathsf{RB}} = (V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G)$*, and a partition of* V^{B} *into* V^{RBS} *and* V^{ACI} *so that* $(V^{\mathsf{ACI}}, V^{\mathsf{R}} \cup V^{\mathsf{RBS}}, A, I, G)$ *is in ACI, and* V^{ACI} *does not depend on* V^{RBS}*. Let* π *be an RB plan for* $\Pi^{\mathsf{RB}}_+ = (V^{\mathsf{RBS}}, V^{\mathsf{R}} \cup V^{\mathsf{ACI}}, A, I, G)$*. Then RBS+ACI plan repair on* π *succeeds, and its output* π^{RB} *is an RB plan for* Π^{RB}*.*

Proof. Any action a that may be inserted by ACI plan repair, and hence by RBS+ACI plan repair, affects a variable in V^{ACI}. Therefore, by prerequisite, 1) a has no effect on V^{RBS}, and 2) a has no precondition on V^{RBS}. So the arguments given by Katz et al. [2013] remain applicable. □

Example 5. *Say we set* $V^{\mathsf{RBS}} = \{M\}$ *and* $V^{\mathsf{ACI}} = \{T\}$*. Note that* M *depends on* T*: this dependency direction is allowed.*

RBS is run on $\Pi^{\mathsf{RB}}_+ = (\{M\}, \{T, A, B\}, A, I, G)$*. The outcome is* $\pi = \langle go(l_2, l_1), buy(l_1, A, 2), buy(l_1, B, 1)\rangle$*. Running ACI plan repair on* π *finds the unsatisfied goal condition* $g = \{T = l_2\}$ *at the end. This is repaired by appending* $\langle go(l_1, l_2)\rangle$ *to* π*, yielding a plan for the original task.*

Proposition 1 shows that our RBS+ACI framework is sound for RB planning in Π^{RB}. Completeness holds, too:

Proposition 2. *Under the prerequisites of Proposition 1, an RB plan for* $\Pi^{\mathsf{RB}} = (V^{\mathsf{RBS}} \cup V^{\mathsf{ACI}}, V^{\mathsf{R}}, A, I, G)$ *exists iff an RB plan for* $\Pi^{\mathsf{RB}}_+ = (V^{\mathsf{RBS}}, V^{\mathsf{R}} \cup V^{\mathsf{ACI}}, A, I, G)$ *exists.*

Proof. The "if" direction holds by Proposition 1. The "only if" direction holds because Π^{RB} is a refinement of Π^{RB}_+. □

So our approach works provided there is no CG arc from V^{RBS} to V^{ACI}. Let us show that this restriction is necessary. Consider the decision problem **RBS-dependent ACI PlanGen**, defined as follows. Given $\Pi^{\mathsf{RB}} = (V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G)$ and a partition of V^{B} into V^{RBS} and V^{ACI} s.t. $(V^{\mathsf{ACI}}, V^{\mathsf{R}} \cup V^{\mathsf{RBS}}, A, I, G)$ is in ACI, and all CG arcs between V^{RBS} and V^{ACI}, if any, go from V^{RBS} to V^{ACI}. Given an RB plan π for $\Pi^{\mathsf{RB}}_+ = (V^{\mathsf{RBS}}, V^{\mathsf{R}} \cup V^{\mathsf{ACI}}, A, I, G)$. Denote by $\pi|_{V^{\mathsf{RBS}}}$ the subsequence of V^{RBS}-affecting actions in π. Decide whether $\pi|_{V^{\mathsf{RBS}}}$ is a subsequence of an RB plan for Π^{RB}.

Theorem 1. *RBS-dependent ACI PlanGen is* ***NP****-hard.*

Proof. By a reduction from SAT. Let ϕ be a CNF formula with propositions $p_1, \ldots, p_n$ and clauses $c_1, \ldots, c_m$. Our planning encoding first chooses values for p_i, then satisfies the clauses c_j. The construction sets V^{RBS} to contain a single "indicator" variable, determining whether we can right now set p_i to 0 or to 1; V^{ACI} represents this choice of values; and V^{R} represents whether or not a clause has been satisfied yet.

In detail, we set $V^{\mathsf{RBS}} = \{v\}$ with domain $\{0, 1\}$, initial value 0, and a single action $a[v01]$ going from 0 to 1. We set $V^{\mathsf{ACI}} = \{v_{p_1}, \ldots, v_{p_n}\}$ with domain $\{u, 0, 1\}$, initial value u, actions going from u to 0 with precondition $v = 0$, and actions going from u to 1 with precondition $v = 1$. We set $V^{\mathsf{R}} = \{v_{c_1}, \ldots, v_{c_m}\}$ with domain $\{0, 1\}$, initial value 0, goal value 1, and an action $a[v_{c_j}01]$ setting v_{c_j} from 0 to 1 with precondition $\{v = 1, v_{p_i} = x\}$ for each $(p_i = x) \in c_j$.

Observe first that this RB planning task Π^{RB} does satisfy the prerequisites: all $v_{p_i} \in V^{\mathsf{ACI}}$ are invertible, and there are no dependencies across these variables; the dependencies between V^{RBS} and V^{ACI} consist in the CG arcs (v, v_{p_i}).

Consider now $\pi|_{V^{\mathsf{RBS}}} := \langle a[v01]\rangle$. This is a subsequence of an RB plan π for Π^{RB}_+: We can move each v_{p_i} to $v_{p_i} = 0$ before the application of $a[v01]$, and to $v_{p_i} = 1$ after that application. Any formula ϕ can be satisfied that way.

But is $\pi|_{V^{\mathsf{RBS}}}$ a subsequence of an RB plan for Π^{RB}? The answer is "yes" iff ϕ is satisfiable. This is because $\pi|_{V^{\mathsf{RBS}}}$ is (trivially) a subsequence of any RB plan for Π^{RB}, and an RB plan for Π^{RB} exists iff ϕ is satisfiable. The latter is true because, in Π^{RB}, each v_{p_i} can support the clause-satisfying actions $a[v_{c_j}01]$ with only a single truth value. First, $v_{p_i} = 1$ can only be reached after $a[v01]$, at which point $v_{p_i} = 0$ is no longer reachable. Second, we can set $v_{p_i} = 0$ before the application of $a[v01]$. But at that point, $a[v_{c_j}01]$ is not yet applicable due to its precondition $v = 1$. So we must apply $a[v01]$, and afterwards we can no longer reach $v_{p_i} = 1$. □

By Theorem 1, given the fixed solution path $\pi|_{V^{\mathsf{RBS}}}$ found by RBS for Π^{RB}_+, augmenting $\pi|_{V^{\mathsf{RBS}}}$ to an RB plan for Π^{RB} is hard. In our framework, such augmentation is done by red (delete-relaxed) planning in Π^{RB}_+ alongside $\pi|_{V^{\mathsf{RBS}}}$, followed by RBS+ACI plan repair. So one of these steps would need to have worst-case exponential runtime (unless $\mathbf{P} = \mathbf{NP}$). In other words, efficient RBS+ACI plan repair is not possible when allowing CG arcs from V^{RBS} to V^{ACI}.

In practice, i. e., in our overall planning algorithm introduced next, one can ameliorate the situation by *attempting* RBS+ACI plan repair even if V^{ACI} does depend on V^{RBS}. If the repair succeeds, all is fine. We only need to act – remove the problematic variable(s) from V^{ACI} – if the repair fails.

4.2 Overall Planning Process: Iterated RBS+ACI

We now know how to solve any RB task Π^{RB} with a painting $V^{\mathsf{RBS}}, V^{\mathsf{ACI}}, V^{\mathsf{R}}$ that qualifies for Proposition 1. But our aim here is to find real plans, for the original FDR input task Π. So RBS+ACI becomes a tool within an overall planning process.

That process is a loop around RBS+ACI searches with increasingly refined paintings. In a pre-process, we compute an ACI painting $V^{\mathsf{B}}_0, V^{\mathsf{R}}_0$ using the default painting strategy in Mercury, which orders the variables by causal graph level and iteratively paints variables red until the black CG is a DAG [Katz and Hoffmann, 2014]. We then initialize our painting as $V^{\mathsf{RBS}} := \emptyset, V^{\mathsf{ACI}} := V^{\mathsf{B}}_0, V^{\mathsf{R}} := V^{\mathsf{R}}_0$. We run RBS+ACI on that painting. If an RB plan does not exist, we know that Π is unsolvable and we stop. Otherwise, we now have an RB plan π^{RB}. We check whether π^{RB} is a real plan for Π. If yes, we stop. Otherwise, we refine our painting. Namely, we simulate the execution of π^{RB} under the real planning semantics in Π, and we count the number of flaws associated with each variable $v \in V^{\mathsf{R}}$. We select $v \in V^{\mathsf{R}}$ with a maximal number of flaws (a criterion adapted from Mercury). We set $V^{\mathsf{RBS}} := V^{\mathsf{RBS}} \cup \{v\}$ and $V^{\mathsf{R}} := V^{\mathsf{R}} \setminus \{v\}$, and iterate.

Adding v to V^{RBS} may introduce dependencies of V^{ACI} on V^{RBS}. Therefore, as discussed above, at some point RBS+ACI plan repair may fail. In that case, we move the culprit variable(s) from V^{ACI} to V^{R}, re-establishing the Proposition 1 guarantee that repair will succeed. The red-black relaxation considered is, then, no longer a refinement of the previous one. But convergence to $V^{\mathsf{B}} = V$ remains intact, so that the completeness of the overall planning process is preserved.

Whenever checking whether an intermediate RB plan π^{RB} works under the real planning semantics in Π, a variant is to *commit* to the prefix that works. We will refer to this as **prefix-execution**. The advantage is that the next iteration of RBS+ACI will not have to start from scratch on the initial state. On the downside, of course this loses completeness.

5 Adaptive Refinement via Realizability

An iterative refinement loop around RBS, as in iterated RBS+ACI, is wasteful in that every iteration of RBS starts from scratch, re-building the entire RB state space. Prefix-execution fixes this, but in a very limited way. Ideally, like other abstraction refinement processes, we ought to refine in an adaptive manner, *only where needed*, and do so incrementally within a single, iteratively refined, relaxed search space.

But how to do this in RBS, and effectively for the purpose of finding real plans? The straightforward approach would be to search until an RB plan π^{RB} is found, execute π^{RB} against the real semantics until the first flaw occurs at RB state s^{RB}, then accordingly refine the painting and re-do the RBS search space below s^{RB}. But there are a number of issues with this. First, it saves us only the work otherwise done above s^{RB} (similarly as the much simpler prefix-execution). Second, with many black variables – as needed to find real plans – finding π^{RB} becomes very expensive so there will be long time intervals between the local refinement steps. Which is especially wasteful as, third, things often go wrong at the root of an RBS sub-tree already. To illustrate the latter, say that the only action applicable at the root s^{RB} has red preconditions p and q, each of which is reached in $\mathcal{F}^+(s^{\mathsf{RB}})$ but which are in conflict so their conjunction is not reachable under the real semantics. Then all search below s^{RB} is wasted.

Given these observations, here we design an eager approach, imposing refinements whenever a transition in Θ^{RB} will not work out in reality. We first show how to do this in RBS, then we discuss the combination with ACI.

5.1 Realizability Refinement: X-RBS

Let s^{RB} be any RB state in Θ^{RB}, and let $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ be any outgoing transition of s^{RB}. By construction, we know that $\mathsf{pre}_a[V^{\mathsf{R}}] \subseteq \mathcal{F}^+(s^{\mathsf{RB}})$. That is, the red preconditions of a can be achieved in the delete-relaxed task $\Pi^+(s^{\mathsf{RB}})$ at s^{RB}. Let now π^+_X be a relaxed plan for the goal $\mathsf{pre}_a[V^{\mathsf{R}}]$ in $\Pi^+(s^{\mathsf{RB}})$, extracted by some relaxed-plan extraction method X. If π^+_X achieves $\mathsf{pre}_a[V^{\mathsf{R}}]$ under the *real* semantics $V^{\mathsf{B}} = V$, we say that $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ is **realized by** π^+_X and is **realizable given X**.

Definition 2. *Let Π^{RB} be an RB planning task, and let X be a relaxed-plan extraction method. The **X-RB state space** is the transition system Θ^{RB}_X defined like Θ^{RB} except that:*

(i) transitions $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ not realizable given X are pruned;

(ii) if $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ is realized by π^+_X, then t^{RB} is the outcome state of executing $\pi^+_X \circ a$ in s^{RB} with $V^{\mathsf{B}} = V$.

Some remarks are in order. First, the rationale behind (i) is that red-black plans will be extracted using X, so if X does not actually achieve pre_a in reality then $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ won't be in a real plan. It is of course a restriction here to commit to X. But there is no systematic alternative: short of a full-scale planning process for pre_a – giving up on the relaxation altogether – if X does not find a real plan, then the best one could do is try another relaxed plan extraction method X'.

Second, that said, Definition 2 is only one half of the story. Whenever a transition $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ is pruned by (i), we spawn a **refinement option**, discussed in detail below. A refinement option is a refined RB planning task at s^{RB}, addressing the reason for non-realizability of $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$.

Finally, (ii) has the immediate effect that every reachable state s^{RB} in Θ^{RB}_X is in fact a real state. It turns the red part of the search (the method X) into a fast macro-generator to the next applicable black-variable affecting action. Observe that this is a natural match with our realizability check. What realizability affirms is that, in reality, we can reach pre_a at s^{RB}. In contrast, the over-approximated state transition, without (ii), would pretend that we can reach *the entire set* $\mathcal{F}^+(s^{\mathsf{RB}})$. Intuitively, we can check the validity of $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ only in a limited way, because we don't a-priori know what the red goal might be here at plan extraction time. So we commit to the minimal way of both, checking and using, the transition. (On the side, realizability checks without (ii) would apply the real semantics starting from an RB state, another mismatch.)

Now, that said, (ii) is a choice we made in our work so far. Exploring alternate definitions is a topic for future work.

Let us now turn to refinement options:

Definition 3. *Let $\Pi^{\mathsf{RB}} = (V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G)$ be an RB planning task. Let $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ be a transition pruned in Θ^{RB}_X, not realized by π^+_X. Let $v \in V^{\mathsf{R}}$ be s.t. π^+_X contains a maximal number of flaws on v. Then $\Pi^{\mathsf{RB}}_{+v}(s^{\mathsf{RB}}) := (V^{\mathsf{B}} \cup \{v\}, V^{\mathsf{R}} \setminus \{v\}, A, s^{\mathsf{RB}}, G)$ is a **refinement option** for $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$.*

Whenever a transition $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$ is pruned in our exploration of Θ^{RB}_X, we generate a refinement option $\Pi^{\mathsf{RB}}_{+v}(s^{\mathsf{RB}})$. That option is inserted as a search node into the overall (heuristic) search. Thus, the search decides not only which states to explore, but also which refinement is used to explore that state. We will refer to this overall search framework as **X-RBS**.

Observe that the under-approximation (ii) loses completeness, i. e., our overall search space may not contain a plan: below realizable transitions, the commitment to π^+_X may exclude the solutions. As an optional fix, **refinement-explored**, we also spawn refinement options at nodes s^{RB} all of whose descendants have been unsuccessfully explored. In such a case, we do not have a concrete flaw to fix, so we pick a variable $v \in V^{\mathsf{R}}$ to paint black arbitrarily.

5.2 Combination with ACI

The number of refinement options can be a major source of computational overhead in X-RBS. One way to ameliorate this is to combine X-RBS with ACI (**X-RBS+ACI**): replacing delete-relaxed planning with tractable red-black planning will result in fewer flaws, and in more realizable transitions.

The combination is simple in X-RBS as relaxed planning occurs only at individual transitions $s^{\mathsf{RB}} \xrightarrow{a} t^{\mathsf{RB}}$. It 1) generates $\mathcal{F}^+(s^{\mathsf{RB}})$ to test whether pre_a is relaxed-reachable; it 2) extracts a relaxed plan using method X, to check realizability.

Using ACI instead, 1) remains unchanged. For 2), we use ACI plan repair on top of X. This uses separate sets V^{RBS} vs. V^{ACI} of black variables as before, but with no constraint on their dependencies: in a realizability check – against the real semantics – a success guarantee cannot be given anyhow.

6 Experiments

Our techniques are implemented on top of Gnad16's RBS, which modifies Fast Downward (FD) [Helmert, 2006] in a minimally intrusive way, exchanging the state and state transition data structures while preserving all search algorithms. All our configurations run FD's greedy best-first dual-queue search with Gnad16's h^{FF} extension and preferred operators.

We run each of RBS and X-RBS with vs. without ACI. We run RBS with vs. without prefix-execution (PE), and X-RBS with vs. without refinement-explored (RE), yielding eight different configurations. Among these, RBS with neither ACI nor prefix-execution is a baseline easily derived from (though not evaluated by) Gnad16. To represent the state of the art in satisficing planning, we run LAMA [Richter and Westphal, 2010] and Mercury [Katz and Hoffmann, 2014]. We also run the best-performing LPG-plan-repair configuration by Gnad16. This paints 90% of the variables black, uses RBS to find an RB plan π^{RB}, then calls LPG to repair π^{RB} into a real plan.

We run all IPC satisficing STRIPS benchmarks. All experiments were run on a cluster of Intel Xeon E5-2650v3 machines, with runtime (memory) limits of 30 minutes (4 GB).

6.1 Coverage

Consider Table 1, and the variants of RBS (leftmost part of the table). Relative to the baseline, our techniques (+ACI and +PE) improve performance substantially. This is clearly visible in overall coverage. Per domain, +PE yields better coverage in 14 domains, +ACI in 12, and the two together in 15. Both techniques also have their drawbacks, as +PE does not work well if the prefix often leads into dead ends (e.g. in Sokoban). Furthermore, +ACI can sometimes introduce more conflicts into the partially relaxed plan. This happens e.g. in Childsnack, where otherwise the RBS+PE configuration only needs to paint the sandwich objects and tray locations black (22-25% of the total variables) to make the red-black plan a real plan, solving all instances in less than 5 seconds.

For the X-RBS method, in the middle part of Table 1, the results are much worse, in many domains and hence in the overall. A key reason is the overhead from too many refinement options. On average, 74% of the generated transitions are realizable, in some domains much less (15% in Parking, 18% in Tetris). As expected, the combination with ACI ameliorates this significantly. But it remains a question for future work how X-RBS can be made competitive overall. While the +RE option helps in domains where X-RBS fails often, it also increases the overhead of too many refinement options.

Consider now RBS+LPG. The empty entries in Table 1 are domains where that architecture did not run properly, for implementation reasons (Gnad16's results do not include these domains either). Filling in the gaps optimistically – assuming that RBS+LPG can solve *all* instances in the missing domains – overall coverage becomes 934. This still lags behind our RBS methods, even the baseline. On a per-domain level though, the methods are highly complementary: of the 32 domains, RBS beats RBS+LPG in 12 and is inferior in 12; RBS+ACI+RE beats RBS+LPG in 16 and is inferior in 11.

	RBS				X-RBS						
		+PE	+ACI	+PE		+RE	+ACI	+RE	RBS +LPG	LAMA	Mercury
Airport (50)	27	28	27	28	41	43	41	**44**	42	32	32
Barman (40)	0	3	0	3	0	7	0	0	24	39	**40**
Blocks (35)	**35**	**35**	**35**	**35**	**35**	**35**	24	33		**35**	**35**
Childsnack (20)	5	**20**	9	10	0	0	0	0	4	5	0
Depots (22)	15	17	16	18	1	9	14	15	**21**	20	**21**
Driverlog (20)	19	18	**20**	19	2	7	3	9	18	**20**	**20**
Elevat (50)	45	47	**50**	**50**	0	12	**50**	**50**	**50**	**50**	**50**
Floortile (40)	3	3	6	7	0	4	0	0	**9**	8	8
Freecell (80)	71	69	71	69	69	61	69	60	35	79	**80**
GED (20)	10	9	10	10	**20**	**20**	14	0	4	**20**	**20**
Grid (5)	4	4	**5**	4	0	2	4	**5**	4	**5**	**5**
Hiking (20)	**20**	**20**	15	17	18	15	18	**20**	19	18	**20**
Logistics (63)	62	62	**63**	**63**	0	12	**63**	**63**	35	**63**	**63**
Maintenan (20)	**11**	7	**11**	7	0	0	0	0		0	7
Mprime (35)	**35**	34	**35**	**35**	3	18	**35**	34	**35**	**35**	**35**
Mystery (19)	16	13	17	13	1	8	**19**	18	16	**19**	**19**
NoMystery (20)	**19**	**19**	**19**	17	0	4	1	4	**19**	11	14
ParcPrin(50)	49	49	49	49	39	48	36	37	35	49	**50**
Parking (40)	12	13	11	13	0	0	0	0	0	**40**	**40**
Pathways (30)	21	28	21	28	27	26	27	26	21	23	**30**
PegSol (50)	**50**	**50**	**50**	**50**	**50**	**50**	**50**	37	16	**50**	**50**
PipesNoT (50)	35	38	36	38	34	25	25	17	39	43	**44**
PipesTank (50)	31	26	28	30	26	20	34	18	24	**42**	**42**
PSR (50)	**50**	**50**	**50**	**50**	0	49	0	49	**50**	**50**	**50**
Rovers (40)	**40**	**40**	**40**	**40**	2	16	18	20		**40**	**40**
Satellite (36)	**36**	**36**	**36**	**36**	0	5	**36**	**36**		**36**	**36**
Scanaly (50)	42	46	42	**50**	43	42	44	44	46	**50**	**50**
Sokoban (50)	20	15	22	13	44	44	29	9	5	**48**	42
Storage (30)	18	20	18	18	16	17	**28**	**28**	25	19	19
Tetris (20)	0	3	0	2	1	0	3	2	0	13	**19**
Thoughtful (20)	6	11	6	10	15	13	9	5		**16**	13
Tidybot (20)	8	6	7	8	0	2	0	0	13	**17**	15
TPP (30)	**30**	**30**	**30**	**30**	0	10	**30**	27	**30**	**30**	**30**
Transpo (70)	31	33	**70**	**70**	0	20	61	57	45	61	**70**
Trucks (30)	12	12	12	12	4	10	0	8	**20**	15	19
VisitAll (40)	3	4	**40**	**40**	3	3	**40**	**40**	4	**40**	**40**
Woodw (50)	**50**	49	**50**	49	17	16	10	13	47	**50**	**50**
Zenotrav (20)	**20**	**20**	**20**	**20**	1	7	**20**	**20**		**20**	**20**
$\sum$ (1385)	961	987	1047	1061	512	680	855	848	755	1211	**1238**

Table 1: Coverage. Best results **highlighted**. We omit domains where all tested planners have full coverage. RBS+LPG is RBS followed by LPG plan repair (empty entries could not be run, see text).

For our X-RBS configurations, the comparison to RBS+LPG is, naturally, less favorable. Complementarity at per-domain level persists though. X-RBS+ACI beats RBS+LPG in 13 domains and is inferior in 14.

Consider finally LAMA and Mercury. All our configurations are far from their performance overall. Our best configuration, RBS+ACI+PE, beats LAMA in 5 domains and is inferior in 20; for Mercury, these numbers are 2 vs. 22.

That said, there are five domains in which at least one of our configurations works exceptionally well. In Airport, our best method gains +12 coverage over the best of LAMA and Mercury; in Childsnack, +15; in Maintenance, +4; in NoMystery, +5; in Storage, +9. So the new methods can potentially contribute in portfolios or per-domain auto-configuration.

6.2 #Black Variables until Solution in RBS

The major motivation behind our +ACI and +PE extensions to RBS is to reduce the size of V^{RBS} required to find a real plan. Figure 1 measures this impact directly.

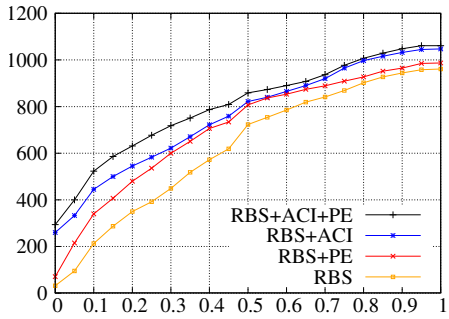


Figure 1: Coverage as a function of the fraction of RBS variables, $|V^{\mathsf{RBS}}|/|V|$, in the first iteration of RBS that finds a real plan.

Both extensions clearly help as intended. Without +ACI, few instances can be solved without search ($|V^{\mathsf{RBS}}| = 0$) as, there, the delete-relaxed plan for the initial state has to be a real plan. The advantage of our extensions remains strong when allowing larger V^{RBS}, until about $|V^{\mathsf{RBS}}|/|V| = 50\%$ where the gap narrows. After that, the difference is mainly due to benchmarks (like Transport) that ACI solves on the initial state but that are beyond reach of RBS search alone.

7 Conclusion

We have shown that RBS can be synergetically combined with ACI tractable red-black planning, and we have started the exploration of adaptive relaxation refinement within RBS. The results for the former show performance improvements due to the smaller number of black variables that need to be searched over. The results for the latter exhibit promise, but the jury is still out how such adaptive refinement is best done.

Overall, our work contributes another piece in the puzzle how to tap into the power of partial delete relaxation without incurring a prohibitive overhead. This fits into the larger puzzle of how to use informative but costly approximations. We believe that such research is valuable to complement the more prominent focus on fast-but-inaccurate approximations, and we hope that our ideas and insights may be useful for approaches other than red-black planning as well.

Acknowledgments

This work was partially supported by the German Research Foundation (DFG), under grants HO 2169/5-1 ("Critically Constrained Planning via Partial Delete Relaxation") and HO 2169/6-1 ("Star-Topology Decoupled State Space Search").

References

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS^+ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[Bonet and Geffner, 2001] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33, 2001.

[Coles *et al.*, 2013] Amanda Jane Coles, Andrew Coles, Maria Fox, and Derek Long. A hybrid LP-RPG heuristic for modelling numeric resource flows in planning. *Journal of Artificial Intelligence Research*, 46:343–412, 2013.

[Domshlak *et al.*, 2015] Carmel Domshlak, Jörg Hoffmann, and Michael Katz. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, 221:73–114, 2015.

[Fox and Long, 2001] Maria Fox and Derek Long. Stan4: A hybrid planning strategy based on subproblem abstraction. *The AI Magazine*, 22(3):81–84, 2001.

[Fox *et al.*, 2006] M. Fox, A. E. Gerevini, D. Long, and I. Serina. Plan stability: Replanning versus plan repair. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS'06)*, pages 212–221, Ambleside, UK, 2006. Morgan Kaufmann.

[Gerevini *et al.*, 2003] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research*, 20:239–290, 2003.

[Gnad and Hoffmann, 2015] Daniel Gnad and Jörg Hoffmann. Red-black planning: A new tractability analysis and heuristic function. In *Proceedings of the 8th Annual Symposium on Combinatorial Search (SOCS'15)*. AAAI Press, 2015.

[Gnad *et al.*, 2016] Daniel Gnad, Marcel Steinmetz, Mathäus Jany, Jörg Hoffmann, Ivan Serina, and Alfonso Gerevini. Partial delete relaxation, unchained: On intractable red-black planning and its applications. In *Proceedings of the 9th Annual Symposium on Combinatorial Search (SOCS'16)*. AAAI Press, 2016.

[Haslum, 2012] Patrik Haslum. Incremental lower bounds for additive cost planning problems. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 74–82. AAAI Press, 2012.

[Helmert and Domshlak, 2009] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 162–169. AAAI Press, 2009.

[Helmert and Geffner, 2008] Malte Helmert and Hector Geffner. Unifying the causal graph and additive heuristics. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, pages 140–147. AAAI Press, 2008.

[Helmert *et al.*, 2014] Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the Association for Computing Machinery*, 61(3), 2014.

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

[Helmert, 2009] Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173:503–535, 2009.

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

[Jonsson and Bäckström, 1995] Peter Jonsson and Christer Bäckström. Incremental planning. In *European Workshop on Planning*, 1995.

[Katz and Hoffmann, 2013] Michael Katz and Jörg Hoffmann. Red-black relaxed plan heuristics reloaded. In *Proceedings of the 6th Annual Symposium on Combinatorial Search (SOCS'13)*, pages 105–113. AAAI Press, 2013.

[Katz and Hoffmann, 2014] Michael Katz and Jörg Hoffmann. Mercury planner: Pushing the limits of partial delete relaxation. In *IPC 2014 planner abstracts*, pages 43–47, 2014.

[Katz *et al.*, 2013] Michael Katz, Jörg Hoffmann, and Carmel Domshlak. Red-black relaxed plan heuristics. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI'13)*, pages 489–495, Bellevue, WA, USA, July 2013. AAAI Press.

[Keyder *et al.*, 2014] Emil Keyder, Jörg Hoffmann, and Patrik Haslum. Improving delete relaxation heuristics through explicitly represented conjunctions. *Journal of Artificial Intelligence Research*, 50:487–533, 2014.

[Richter and Westphal, 2010] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39:127–177, 2010.