Pushing the Limits of Partial Delete Relaxation: Red-Black DAG Heuristics

Michael Katz
IBM Haifa Research Labs
Haifa, Israel
katzm@il.ibm.com

Jörg Hoffmann
Saarland University
Saarbrücken, Germany
hoffmann@cs.uni-saarland.de

Abstract

Red-black planning is a systematic approach to partial delete relaxation, taking into account *some* of the delete effects: Red variables take the relaxed (value-accumulating) semantics, while black variables take the regular semantics. Prior work on red-black plan heuristics has identified a powerful tractable fragment requiring the *black causal graph* – the projection of the causal graph onto the black variables – to be a DAG; but all implementations so far use a much simpler fragment where the black causal graph is required to not contain any arcs at all. We close that gap here, and we design techniques aimed at making red-black plans executable, short-cutting the search. Our experiments show that these techniques can yield significant improvements on those IPC benchmarks where non-trivial DAG black causal graphs exist.

Introduction

The *delete relaxation*, where state variables accumulate their values rather than switching between them, has played a key role in the success of satisficing planning systems, e. g. (Bonet and Geffner 2001; Hoffmann and Nebel 2001; Richter and Westphal 2010). Still, the delete relaxation has well-known pitfalls, for example the fundamental inability to account for moves back and forth (as done, e. g., by vehicles in transportation). It has thus been an actively researched question from the outset how to take *some* deletes into account, e. g. (Fox and Long 2001; Gerevini, Saetti, and Serina 2003; Helmert 2004; Helmert and Geffner 2008; Baier and Botea 2009; Cai, Hoffmann, and Helmert 2009; Haslum 2012; Keyder, Hoffmann, and Haslum 2012). Herein, we continue the most recent attempt, *red-black planning* (Katz, Hoffmann, and Domshlak 2013b; 2013a; Katz and Hoffmann 2013) where a subset of *red* state variables takes on the relaxed value-accumulating semantics, while the other *black* variables retain the regular semantics.

Katz et al. (2013b) introduced the red-black framework and conducted a theoretical investigation of tractability. Following up on this (2013a), they devised practical *red-black plan heuristics*, non-admissible heuristics generated by repairing fully delete-relaxed plans into red-black plans. Observing that this technique often suffers from dramatic overestimation incurred by following arbitrary decisions taken in delete-relaxed plans, Katz and Hoffmann (2013) refined the approach to rely less on such decisions, yielding a more flexible algorithm delivering better search guidance.

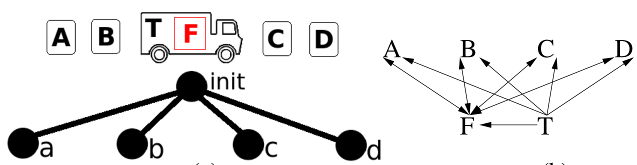
The *black causal graph* is the projection of the causal graph onto the black variables only. Both Katz et al. (2013a) and Katz and Hoffmann (2013) exploit, in theory, a tractable fragment characterized by *DAG black causal graphs*, but confine themselves to *arc-empty black causal graphs* – no arcs at all – in practice. Thus current red-black plan heuristics are based on a simplistic, almost trivial, tractable fragment of red-black planning. We herein close that gap, designing *red-black DAG heuristics* exploiting the full tractable fragment previously identified. To that end, we augment Katz and Hoffmann's implementation with a DAG-planning algorithm (executed several times within every call to the heuristic function). We devise some enhancements targeted at making the resulting red-black plans executable in the real task, stopping the search if they succeed in reaching the goal. In experiments on the relevant IPC benchmarks, we find that the gained informativity often pays off, reducing search and improving overall performance.

Background

Our approach is placed in the *finite-domain representation (FDR)* framework. To save space, we introduce FDR and its delete-relaxation as special cases of red-black planning. A **red-black (RB)** planning task is a tuple $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$. V^{B} is a set of *black state variables* and V^{R} is a set of *red state variables*, where $V^{\mathsf{B}} \cap V^{\mathsf{R}} = \emptyset$ and each $v \in V := V^{\mathsf{B}} \cup V^{\mathsf{R}}$ is associated with a finite domain $\mathcal{D}(v)$. The *initial state* I is a complete assignment to V, the *goal* G is a partial assignment to V. Each action a is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ of partial assignments to V called *precondition* and *effect*. We often refer to (partial) assignments as sets of *facts*, i. e., variable-value pairs $v = d$. For a partial assignment p, $\mathcal{V}(p)$ denotes the subset of V instantiated by p. For $V' \subseteq \mathcal{V}(p)$, $p[V']$ denotes the value of V' in p.

A state s assigns each $v \in V$ a non-empty subset $s[v] \subseteq \mathcal{D}(v)$, where $|s[v]| = 1$ for all $v \in V^{\mathsf{B}}$. An action a is applicable in state s if $\mathsf{pre}(a)[v] \in s[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$. Applying a in s changes the value of $v \in \mathcal{V}(\mathsf{eff}(a)) \cap V^{\mathsf{B}}$ to $\{\mathsf{eff}(a)[v]\}$, and changes the value of $v \in \mathcal{V}(\mathsf{eff}(a)) \cap V^{\mathsf{R}}$ to $s[v] \cup \{\mathsf{eff}(a)[v]\}$. By $s[\![\langle a_1, \dots, a_k\rangle]\!]$ we denote the state obtained from sequential application of $a_1, \dots, a_k$. An action sequence $\langle a_1, \dots, a_k\rangle$ is a *plan* if $G[v] \in I[\![\langle a_1, \dots, a_k\rangle]\!][v]$ for all $v \in \mathcal{V}(G)$.

Π is a **finite-domain representation (FDR)** planning task if $V^{\mathsf{R}} = \emptyset$, and is a **monotonic finite-domain representation (MFDR)** planning task if $V^{\mathsf{B}} = \emptyset$. Plans for MFDR tasks (i. e., for delete-relaxed tasks) can be generated in polynomial time. A key part of many satisficing planning systems is based on exploiting this property for deriving heuristic estimates, via delete-relaxing the task at hand. Generalizing this to red-black planning, the **red-black relaxation** of an FDR task Π relative to V^{R} is the RB task $\Pi^{*+}_{V^{\mathsf{R}}} = \langle V \setminus V^{\mathsf{R}}, V^{\mathsf{R}}, A, I, G \rangle$. A plan for $\Pi^{*+}_{V^{\mathsf{R}}}$ is a **red-black plan** for Π, and the length of a shortest possible red-black plan is denoted $h^{*+}_{V^{\mathsf{R}}}(\Pi)$. 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 $V^{\mathsf{R}} = V$, then red-black plans are **relaxed plans**, and $h^{*+}_{V^{\mathsf{R}}}$ coincides with the optimal delete relaxation heuristic h^+.



(a) (b)

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

In Figure 1, truck T needs to transport each package $X \in \{A, B, C, D\}$ to its respective goal location $x \in \{a, b, c, d\}$. The truck can only carry one package at a time, encoded by a Boolean variable F ("free"). A real plan has length 15 (8 loads/unloads, 7 drives), a relaxed plan has length 12 (4 drives suffice as there is no need to drive back). If we paint (only) T black, then $h^{*+}_{V^{\mathsf{R}}}(I) = 15$ as desired, but red-black plans may not be applicable in the real task, because F is still red so we can load several packages consecutively. Painting T *and* F black, that possibility disappears.[1]

Tractable fragments of red-black planning have been identified using standard structures. The **causal graph** CG_Π of Π is a digraph with vertices V. An arc (v, v') is in CG_Π if $v \neq v'$ and there exists an action $a \in A$ such that $(v, v') \in [\mathcal{V}(\mathsf{eff}(a)) \cup \mathcal{V}(\mathsf{pre}(a))] \times \mathcal{V}(\mathsf{eff}(a))$. The **domain transition graph** $\mathit{DTG}_\Pi(v)$ of a variable $v \in V$ is a labeled digraph with vertices $\mathcal{D}(v)$. The graph has an arc (d, d') induced by action a if $\mathsf{eff}(a)[v] = d'$, and either $\mathsf{pre}(a)[v] = d$ or $v \notin \mathcal{V}(\mathsf{pre}(a))$. The arc is labeled with its **outside condition** $\mathsf{pre}(a)[V \setminus \{v\}]$ and its **outside effect** $\mathsf{eff}(a)[V \setminus \{v\}]$.

The **black causal graph** $\mathit{CG}^{\mathsf{B}}_\Pi$ of Π is the sub-graph of CG_Π induced by V^{B}. An arc (d, d') is **relaxed side effects invertible**, **RSE-invertible** for short, if there exists an arc (d', d) with outside condition $\phi' \subseteq \phi \cup \psi$ where ϕ and ψ are the outside condition respectively outside effect of (d, d'). A variable v is RSE-invertible if all arcs in $\mathit{DTG}_\Pi(v)$ are RSE-invertible, and an RB task is RSE-invertible if all its black variables are. Prior work on red-black plan heuristics proved that plan generation for RSE-invertible RB tasks with **DAG** (acyclic) black causal graphs is tractable, but used the much simpler fragment restricted to **arc-empty** black causal graphs in practice. In Figure 1, both T and F are RSE-invertible; if we paint only T black then the black causal graph is arc-empty, and if we paint both T and F black then the black causal graph is (not arc-empty but) a DAG.

[1]Indeed, all optimal red-black plans (but not some non-optimal ones) then are real plans. We will get back to this below: As we shall see, the ability to increase red-black plan applicability is a main advantage of our red-black DAG heuristics over the simpler red-black plan heuristics devised in earlier work.

Algorithm : REDBLACKPLANNING(Π, R^+)

main
$\pi \leftarrow \langle\rangle$
while $R \not\supseteq R^+$
do
- $A' = \{a \in A \mid \mathsf{pre}(a) \subseteq B \cup R, \mathsf{eff}(a) \cap (R^+ \setminus R) \neq \emptyset\}$
- Select $a \in A'$
- **if** $\mathsf{pre}(a)[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
  then $\pi \leftarrow \pi \circ$ ACHIEVE($\mathsf{pre}(a)[V^{\mathsf{B}}]$)
- $\pi \leftarrow \pi \circ \langle a \rangle$

if $G[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
then $\pi \leftarrow \pi \circ$ ACHIEVE($G[V^{\mathsf{B}}]$)
return π

procedure ACHIEVE(g)
$I^{\mathsf{B}} \leftarrow I[\![\pi]\!][V^{\mathsf{B}}]; G^{\mathsf{B}} \leftarrow g$
$A^{\mathsf{B}} \leftarrow \{a^{\mathsf{B}} \mid a \in A, a^{\mathsf{B}} = \langle \mathsf{pre}(a)[V^{\mathsf{B}}], \mathsf{eff}(a)[V^{\mathsf{B}}] \rangle,$
$\mathsf{pre}(a) \subseteq B \cup R\}$
$\Pi^{\mathsf{B}} \leftarrow \langle V^{\mathsf{B}}, A^{\mathsf{B}}, I^{\mathsf{B}}, G^{\mathsf{B}} \rangle$
return DAGPLANNING(Π^{B})

Figure 2: Katz and Hoffmann's (2013) red-black planning algorithm (abbreviated; for explanations see text).

Red-Black DAG Heuristics

As indicated, we augment Katz and Hoffmann's (2013) implementation with a DAG-planning algorithm. To provide the context, Figure 2 shows (the main parts of) Katz and Hoffmann's pseudo-code. The algorithm assumes as input the set R^+ of preconditions and goals on red variables in a fully delete-relaxed plan, i. e., $R^+ = G[V^{\mathsf{R}}] \cup \bigcup_{a \in \pi^+} \mathsf{pre}(a)[V^{\mathsf{R}}]$ where π^+ is a relaxed plan for Π. It then successively selects achieving actions for R^+, until all these red facts are true. Throughout the algorithm, R *denotes the set of red facts already achieved by the current red-black plan prefix* π*;* B *denotes the set of black variable values that can be achieved using only red outside conditions from* R. We have omitted the (simple) maintenance of R and B here as it is not needed to understand the present paper.

For each action $a \in A'$ selected to achieve new facts from R^+, and for the global goal condition at the end, there may be black variables that do not have the required values. For example, say we paint T and F black in Figure 1. Then R^+ will have the form $\{A = T, A = a, B = T, B = b, C = T, C = c, D = T, D = d\}$. In the initial state, A' will contain only load actions. Say we execute $a =$load(A, init), entering $A = T$ into R and thus including unload(A, a) into A' in the next iteration. Trying to execute that action, we find that its black precondition $T = a$ is not satisfied. The call to ACHIEVE($\{T = a\}$) is responsible for rectifying this.

ACHIEVE(g) creates a task Π^{B} over Π's black variables, asking to achieve g. As Katz and Hoffmann showed, Π^{B} is solvable, has a DAG causal graph, and has strongly connected DTGs (when restricting to actions a where $\mathsf{pre}(a) \subseteq I[\![\pi]\!]$). From this and Theorem 4.4 of Chen and Gimenez (2010), it directly follows that a plan for Π^{B}, *in a succinct*

Algorithm : DAGPLANNING(Π^{B})

```
main
 π^B ← ⟨⟩
 for i = n downto 1
   do { // Denote π^B = ⟨a_1, ..., a_k⟩
        d ← I[v_i]
        for j = 1 to k
          do { π_j ← ⟨⟩
               if pre(a_j)[v_i] is defined
                 then { π_j ← π_{v_i}(d, pre(a_j)[v_i])
                        d ← pre(a_j)[v_i] }
             }
        π_{k+1} ← ⟨⟩
        if G[v_i] is defined
          then π_{k+1} ← π_{v_i}(d, G[v_i])
        π^B ← π_1 · ⟨a_1⟩ · ... · π_k · ⟨a_k⟩ · π_{k+1}
      }
 return π^B
```

Figure 3: Planning algorithm for FDR tasks Π^{B} with DAG causal graph $CG_{\Pi^{\mathsf{B}}}$ and strongly connected DTGs. $v_1, \dots, v_n$ is an ordering of variables V consistent with the topology of $CG_{\Pi^{\mathsf{B}}}$. $\pi_v(d, d')$ denotes an action sequence constituting a shortest path in $DTG_v(\Pi)$ from d to d'.

plan representation, can be generated in polynomial time.

The "succinct plan representation" just mentioned consists of recursive macro actions for pairs of initial-value/other-value within each variable's DTG; it is required as plans for Π^{B} may be exponentially long. Chen and Gimenez' algorithm handling these macros involves the exhaustive enumeration of shortest paths for the mentioned value pairs in all DTGs, and it returns highly redundant plans moving precondition variables back to their initial value in between every two requests. For example, if a truck unloads two packages at the same location, then it is moved back to its start location in between the two unload actions.

Katz and Hoffmann (2013) shunned the complexity of DAG planning, and considered Π^{B} with arc-empty causal graphs, solving which is trivial. In our work, after exploring a few options, we decided to use the simple algorithm in Figure 3: Starting at the leaf variables and working up to the roots, the partial plan π^{B} is augmented with plan fragments bringing the supporting variables into place (a similar algorithm was mentioned, but not used, by Helmert (2006)).

Proposition 1 *The algorithm* DAGPLANNING(Π^{B}) *is sound and complete, and its runtime is polynomial in the size of* Π^{B} *and the length of the plan* π^{B} *returned.*

Note here that the length of π^{B} is worst-case exponential in the size of Π^{B}, and so is the runtime of DAGPLANNING(Π^{B}). We trade the theoretical worst-case efficiency of Chen and Gimenez' algorithm against the practical advantage of not having to rely on exhaustive computation of shortest paths – anew for every call of DAGPLANNING, with "initial values" and DTGs from Π^{B} – for input tasks Π^{B} that typically have small plans (achieving the next action's black preconditions) anyhow.[2]

[2]One could estimate DAG plan length (e.g., using Helmert's (2006) causal graph heuristic), computing a red-black plan *length estimate* only. But that would forgo the possibility to actually execute DAG red-black plans, which is a key advantage in practice.

Unlike the macro-based algorithm of Chen and Gimenez, our DAGPLANNING algorithm does not superfluously keep switching supporting variables back to their initial values. But it is not especially clever, either: If variable v_0 supports two otherwise independent leaf variables v_1 and v_2, then the sub-plans for v_1 and v_2 will be inserted sequentially into π^{B}, losing any potential for synergies in the values of v_0 required. We developed a more flexible algorithm addressing that weakness through using a partially-ordered π^{B}, but that algorithm resulted in significantly worse empirical performance, so we do not include it here.

Enhancing Red-Black Plan Applicability

One crucial advantage of red-black plans, over fully-delete relaxed plans, is that they have a much higher chance of actually working for the original planning task. This is especially so for the more powerful DAG red-black plans we generate here. In Figure 1, as already mentioned, if we paint just T black then the red-black plan *might* work; but if we paint both T and F black – moving to a non-trivial DAG black causal graph – then *every optimal red-black plan definitely works*. A simple possibility for exploiting this, already implemented in Katz and Hoffmann's (2013) earlier work, is to *stop search* if the red-black plan generated for a search state s is a plan for s in the original task.

There is a catch here, though – the red-black plans we generate are not optimal and thus are not guaranteed to execute in Figure 1. In our experiments, we observed that the red-black plans often were not executable due to simple reasons. We fixed this by augmenting the algorithms with the two following applicability enhancements.

(1) Say that, as above, $R^+ = \{A = T, A = a, B = T, B = b, C = T, C = c, D = T, D = d\}$ and REDBLACKPLANNING started by selecting load$(A, init)$. Unload(A, a) *might* be next, but the algorithm might just as well select load$(B, init)$. With T and F black, load$(B, init)$ has the black precondition $F = true$. Calling ACHIEVE($\{F = true\}$) will obtain that precondition using unload$(A, init)$. Note here that variable A is red so the detrimental side effect is ignored. The same phenomenon may occur in any domain with renewable resources (like transportation capacity). We tackle it by giving a preference to actions $a \in A'$ getting whose black preconditions does not involve deleting R^+ facts already achieved beforehand. To avoid excessive overhead, we approximate this by recording, in a pre-process, which red facts may be deleted by moving each black variable, and prefer an action if none of its black preconditions may incur any such side effects.

(2) Our second enhancement pertains to the DTG paths chosen for the black precondition variables in DAGPLANNING (after REDBLACKPLANNING has already selected the next action). The red outside conditions are by design all reached (contained in R), but we can prefer paths whose red outside conditions are "active", i. e., true when executing the current red-black plan prefix in the real task. (E.g., if a capacity variable is red, then this will prefer loads/unloads that use the actual capacity instead of an arbitrary one.) In some special cases, non-active red outside conditions can be easily fixed by inserting additional supporting actions.

Domain		Coverage							Evaluations hFF/Own						DLS	hFF/DLS		Coverage DL			Eval hFF/DL		
		hFF	K12	K13	EAS	ELS	DAS	DLS	K12	K13	EAS	ELS	DAS	DLS	Init	Plan	Time	-S	-(1,2)	-(2)	-S	-(1,2)	-(2)
Barman	20	17	18	13	**20**	**20**	**20**	**20**	3.4	1.6	6	67.8	6	67.8	0	0.9	56.8	**20**	**20**	**20**	6	67.8	67.8
Driverlog	20	20	20	20	20	20	20	20	1.3	2	1	1.1	1.1	1	1	1	1	19	**20**	**20**	0.9	1	1
Elevators	20	**20**	18	17	**20**	18	**20**	**20**	1.2	1.5	1.2	1.6	1.5	5920	20	1.1	15.5	18	**20**	**20**	1.4	2911	5920
Gripper	20	20	20	20	20	20	20	20	1	0.7	4.2	1	344	344	20	1	1	20	20	20	3.7	344	344
Rovers	29	29	29	29	29	29	29	29	1.1	1.2	1.2	1.2	1.4	1.4	1	1	0.8	29	29	29	1.1	1.4	1.4
Tidybot	20	13	10	**16**	12	13	12	13	2.2	1.3	1.2	1.1	1.2	1.1	0	1	0.8	12	13	12	1	1.1	1.1
Transport	20	10	10	10	11	11	**20**	**20**	0.6	1.2	1	0.9	3.4	3071	20	1.5	8.3	15	16	**20**	0.8	33.2	3071
Trucks	30	**19**	15	14	18	18	18	18	0.5	0.9	1.1	1.1	0.5	0.5	0	1	0.5	18	18	18	0.5	0.5	0.5
Sum	179	148	140	139	150	149	159	**160**							62			151	156	159			

Table 1: Experiments results. Ratios: median over instances solved by both planners involved. Explanations see text.

Experiments

The experiments were run on Intel Xeon CPU E5-2660 machines, with time (memory) limits of 30 minutes (2 GB). We ran *all IPC STRIPS benchmark instances whose causal graphs have at least one directed arc* (v, v') *between RSE-invertible variables* v *and* v'*, with no backwards arc* (v', v). These are exactly the tasks for which there exists a choice of black variables so that (a) the resulting red-black planning task is inside the tractable fragment, and (b) the black causal graph is a non-arc-empty DAG. The domains/instances where that happens are as shown in Table 1. For IPC'08 domains also used in IPC'11, we used only the IPC'11 version. For simplicity, we consider uniform costs throughout (i. e., we ignore action costs where specified).

We compare our **D**AG heuristics against Katz and Hoffmann's (2013) arc-**E**mpty ones, and against two variants of Keyder et al.'s (2012) partial delete relaxation heuristics: **K12** is best in their published experiments, **K13** is best in more recent (yet unpublished) experiments. **S** stops the search if a red-black plan works for the original planning task. Our baseline is the **hFF** heuristic implemented in Fast Downward. All configurations run greedy best-first search with lazy evaluation and a second open list for states resulting from preferred operators (Helmert 2006). All red-black heuristics return the same preferred operators as **hFF**: This enhances comparability; we found that changing the preferred operators was typically not beneficial anyway.

Katz and Hoffmann explored a variety of *painting strategies*, i. e., strategies for selecting the black variables. We kept this simple here because, as we noticed, there actually is little choice, at least when accepting the rationale that we should paint black as many variables as possible: In all our domains except Tidybot, there are at most 2 possible paintings per task. To illustrate, consider Figure 1: We can paint T and F black, or paint T and the packages black. All other paintings either do not yield a DAG black causal graph, or are not set-inclusion maximal among such paintings. We thus adopted only Katz and Hoffmann's 3 basic strategies, ordering the variables either by incident arcs (**A**), or by conflicts (**C**), or by causal graph level (**L**), and iteratively painting variables red until the black causal graph is a DAG (Katz and Hoffmann's original strategies continue until that graph is arc-**E**mpty). **A** works best for Katz and Hoffmann's heuristics (both here and in their experiments), and **L** works best for ours, so we show data for these two.

Consider Table 1 from left to right. Our red-black DAG heuristics have the edge in coverage, thanks to excelling in Transport and being reliably good across these domains. For search space size (number of state evaluations, i. e., calls to the heuristic function), there are smallish differences in half of the domains, and huge differences in the other half: Under the **L** painting strategy, in Barman the arc-**E**mpty heuristic already does well, and in Elevators, Gripper, and Transport our new **D**AG heuristic excels. A look at the "DLS Init" column, i. e., the number of instances solved by **S** in the initial state (without any search), shows that the latter 3 domains are exactly those where the superior applicability of DAG red-black plans makes the difference. Column "hFF/DLS Plan" shows that the plans found using **DLS** (even when constructed within the heuristic by **S**) are about as good as those found using **hFF**. Column "hFF/DLS Time" shows that the search space reductions do pay off in total runtime, except in Gripper (where all except **K12** and **K13** terminate in split seconds). The maximum speed-ups are 413 in Barman, 7.5 in Driverlog, 722 in Elevators, 12.9 in Tidybot, and 683 in Transport (none in the other domains); the maximum slow-down factors are 1.2 in Barman, 25 in Driverlog, 4.8 in Rovers, 4 in Tidybot, and 625 in Trucks.

The remainder of Table 1 sheds light on the contribution of stop search in **DLS**: **-S** switches stop search off, **-(1,2)** leaves it on but switches both applicability enhancements off, **-(2)** switches only enhancement (2) off.[3] We see that stop search helps even in domains (Driverlog, Tidybot) not solved directly in the initial state, and we see that the superior coverage in Transport is half due to the more informed heuristic, and half due to stop search with enhancement (1).

To further investigate the effect of stop search, we generated additional instances of the three domains that are fully solved directly in the initial state. Table 2 summarizes the results for additional 100 instances of increasing size in each of the three domains, namely Elevators, Gripper, and Transport. Focusing on our best performer, we compare **DLS** to both switching stop search off and our base **FF** heuristic. Note that **DLS** still solves all, even extremely large[4] instances directly in the initial state. Switching **S** off drastically reduces coverage without improving plan length. Comparing to **FF**, the picture is similar. Due to a large number of evaluations, even the much faster heuristic times out

[3]The data for **-(1)** is left out of Table 1 as it is equal to that for **-(1,2)**. Enhancement (2) has more impact in other (non-DAG) domains, especially NoMystery and Zenotravel, where **S** solves 8 respectively 19 more tasks directly with (2) than without it.

[4]Largest Gripper instance has 3942 balls.

		Coverage			Evals hFF/Own		Plan hFF/Own		Time hFF/Own	
		FF	DLS	DL	DLS	DL	DLS	DL	DLS	DL
Elevators	100	38	**100**	31	**10788.00**	1.28	**1.12**	1.06	**28.33**	0.63
Gripper	100	47	**100**	52	**148778.00**	68.59	1.00	1.00	**34.14**	0.75
Transport	100	30	**100**	26	**4150.00**	0.79	**1.44**	1.10	**13.78**	0.53
Sum	300	115	**300**	109	**10788.00**	1.28	**1.12**	1.06	**28.33**	0.63

Table 2: Experiments results. Coverage, evaluations, plan length, and total time for generated instances. Ratios: median over instances solved by both planners involved.

before finding a solution. Interestingly, in Gripper, **DL** has linear search space[5]. However, even in such case, due to the costly per node evaluation time of **DL**, large enough instances are not solved under the 30 minutes time bound.

Conclusion

Our work provides one more step on the road towards systematic interpolation between delete-relaxed and non-relaxed (real) planning. Our experience, as reflected by the presented experiments, suggests that the key advantage of heavy interpolation may lie in the ability to produce approximate plans that are very close to real plans (or that already *are* real plans). We believe that further progress will be achieved by developing search methods exploiting that ability in a targeted manner, for example by using partially relaxed plans to initialize plan-space searches (e. g., (Nguyen and Kambhampati 2001; Gerevini, Saetti, and Serina 2003)).

[5]So does FF heuristic when using enhanced hill climbing search and helpful actions in FF planning system.

References

Baier, J. A., and Botea, A. 2009. Improving planning performance using low-conflict relaxed plans. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 10–17. AAAI Press.

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

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

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

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

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, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 74–82. AAAI Press.

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'08)*, 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.

Katz, M., and Hoffmann, J. 2013. Red-black relaxed plan heuristics reloaded. In Helmert, M., and Röger, G., eds., *Proceedings of the 6th Annual Symposium on Combinatorial Search (SOCS'13)*, 105–113. AAAI Press.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013a. Red-black relaxed plan heuristics. In desJardins, M., and Littman, M., eds., *Proceedings of the 27th National Conference of the American Association for Artificial Intelligence (AAAI'13)*, 489–495. Bellevue, WA, USA: AAAI Press.

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

Keyder, E.; Hoffmann, J.; and Haslum, P. 2012. Semi-relaxed plan heuristics. In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 128–136. AAAI Press.

Nguyen, X., and Kambhampati, S. 2001. Reviving partial order planning. In Nebel, B., ed., *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 459–464. Seattle, Washington, USA: Morgan Kaufmann.

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