Classical Planning@Robotics: Methods and Challenges

Jörg Hoffmann

SAARLAND UNIVERSITY
COMPUTER SCIENCE

June 14, 2017

Agenda

1. Classical Planning: What? Why?
2. The Delete Relaxation
3. Beyond the Delete Relaxation
4. Proving Things about the Space of Plans
5. Future Topics in Robotics

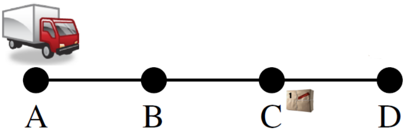
(A Canonical) Classical Planning Language

Definition. *A planning task is a 4-tuple* $\Pi = (V, A, I, G)$ *where:*

- V *is a set of state variables, each* $v \in V$ *with a finite domain* D_v.
- A *is a set of actions; each* $a \in A$ *is a triple* (pre_a, eff_a, c_a), *of precondition and effect (partial assignments), and the action's cost* $c_a \in \mathbb{R}^+_0$.
- *Initial state* I *(complete assignment), goal* G *(partial assignment).*

→ Solution ("Plan"): Action sequence mapping I into s s.t. $s \models G$.

"Logistics" Example:

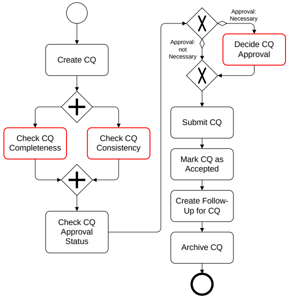


- State variables V: $truck : \{A, B, C, D\}$; $pack1 : \{A, B, C, D, T\}$.
- Initial state I: $truck = A$, $pack1 = C$.
- Goal G: $truck = A$, $pack1 = D$.
- Actions A (unit costs): $drive(x, y)$, $load(p, x)$, $unload(p, x)$. E.g.: $load(pack1, x)$ precondition $truck = x, pack1 = x$, effect $pack1 = T$.

Semantic BPM@SAP

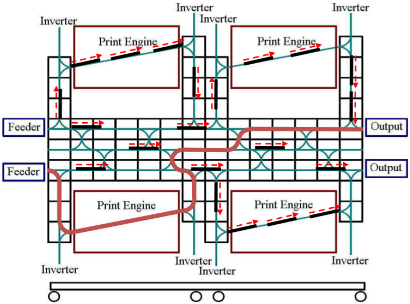
[Hoffmann *et al.* (2012)]

Action name	precondition	effect
Check CQ Completeness	CQ.archiving:notArchived	CQ.completeness:complete OR CQ.completeness:notComplete
Check CQ Consistency	CQ.archiving:notArchived	CQ.consistency:consistent OR CQ.consistency:notConsistent
Check CQ Approval Status	CQ.archiving:notArchived AND CQ.approval:notChecked AND CQ.completeness:complete AND CQ.consistency:consistent	CQ.approval:necessary OR CQ.approval:notNecessary
Decide CQ Approval	CQ.archiving:notArchived AND CQ.approval:necessary	CQ.approval:granted OR CQ.approval:notGranted
Submit CQ	CQ.archiving:notArchived AND (CQ.approval:notNecessary OR CQ.approval:granted)	CQ.submission:submitted
Mark CQ as Accepted	CQ.archiving:notArchived AND CQ.submission:submitted	CQ.acceptance:accepted
Create Follow-Up for CQ	CQ.archiving:notArchived AND CQ.acceptance:accepted	CQ.followUp:documentCreated
Archive CQ	CQ.archiving:notArchived	CQ.archiving:archived



- **Planning model:** SAP-scale model of activities on Business Objects; desired process endpoint.
- **Solution:** Process template leading to this point.
- **Key advantage:** Eases process creation. Automation wrpto activities/objects (i. e., arbitrary models thereof).

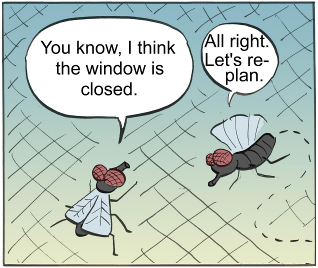
Controlling Modular Printers@Xerox [Ruml *et al.* (2011)]



- **Planning model:** Modular printer components, printer configuration, current jobs.
- **Solution:** Optimal schedule of current jobs.
- **Key advantage:** Automation wrpto possible configurations: A controller for a system whose exact design is not known at software-design time.

Model-Based (aka Automated) Planning@Robotics: Why?

VS.



Plan-Based Control:

- Planning model allows to capture arbitrary states, actions, goals.
- Reacting to arbitrary state/goal change (within fixed model): Re-planning.
- Explicit model/plan allows robot to reason about plan/what went wrong, instead of just following a prescribed recipe/workflow.
- Adapting to arbitrary world change (human supervisor): Model, not code.

→ Long-term autonomy; dynamic environments (unexpected events).

Planning Framework Dimensions (& Classical Planning)

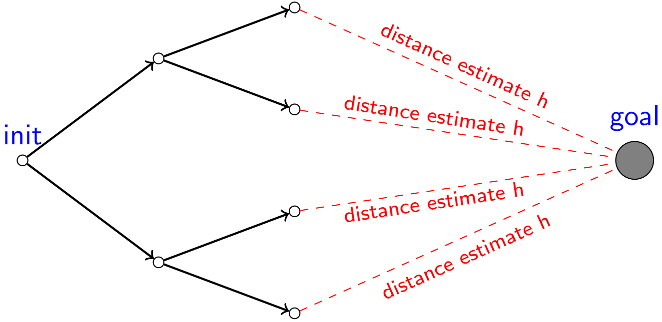
You GET (classical planning)	vs.	You NEED (robotics)
discrete	vs.	continuous
instantaneous actions	vs.	temporal actions
sequential plans	vs.	concurrent plans
fully observable	vs.	partially observable
deterministic	vs.	non-deterministic/stochastic
goal-state condition	vs.	temporal goals/preferences/rewards

So why bother?

- Fast re-planning. (Classical planning already is **PSPACE**-complete, never mind the more accurate variants.)
- "Your need" actually depends on your system architecture/what the "Activity Planning" component is being used for.
- Transfer of algorithmic ideas to richer planning problems.

→ Classical planning is very restricted, but provides fast mechanisms for flexible reactive high-level planning. (And is a good frame for basic algorithms research.)

A Successful Family of Solvers: Heuristic Search

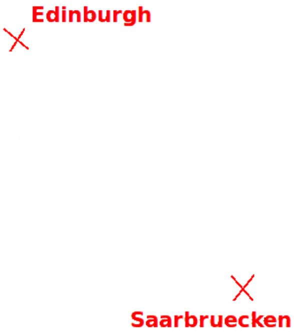


→ Forward state space search. Heuristic function h maps states s to an estimate $h(s)$ of goal distance.

Heuristic Functions

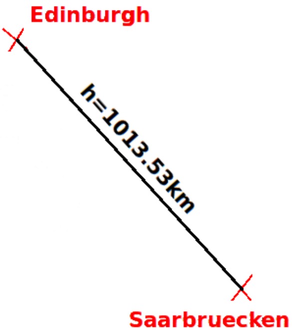
Problem: Find a route from Saarbruecken To Edinburgh.

Heuristic Functions



Relaxed Problem: Throw away the map.

Heuristic Functions



Heuristic function: Straight line distance.

Pretending Things Can Only Get Better

Delete Relaxation = “What was once true remains true forever.”

Relaxed world: (after)

A Little More Formally

Real planning:

Definition (State Transitions). *Let $\Pi = (V, A, I, G)$ be a planning task, s a state in Π, and $a \in A$. We say that a is applicable in s if $pre_a \subseteq s$. In that case, the outcome state s' of applying a to s is:*

$$s'(v) := \begin{cases} \mathit{eff}_a(v) & \mathit{eff}_a \text{ is defined on } v \\ s(v) & \text{otherwise} \end{cases}$$

Delete-relaxed planning:

Definition (Relaxed State Transitions). *Let $\Pi = (V, A, I, G)$ be a planning task. A relaxed state in Π is any set s^+ of variable/value pairs. Let s^+ be a relaxed state, and $a \in A$. We say that a is applicable in s^+ if $pre_a \subseteq s^+$. In that case, the outcome state $s^{+\prime}$ of applying a to s^+ is $s^{+\prime} := s^+ \cup \mathit{eff}_a$.*

→ Under the delete relaxation, state variables accummulate their values, rather than switching between them.

→ In the "Logistics" Example, e. g., we don't need to drive the truck back to A.

The Relaxed Dompteur

(Using a propositional-logic like notation for Boolean variables:)

- $V = \{alive, haveTiger, tamedTiger, haveJump\}$; all Boolean.
- Initial state I: $alive$.
- Goal G: $alive, haveJump$.
- Actions A:

  $getTiger$: pre $alive$; eff $haveTiger$

  $tameTiger$: pre $alive$, $haveTiger$; eff $tamedTiger$

  $jumpTamedTiger$: pre $alive$, $tamedTiger$; eff $haveJump$

  $jumpTiger$: pre $alive$, $haveTiger$; eff $haveJump$, $\neg alive$

$\rightarrow$ Relaxed plan for this task? $\langle getTiger, jumpTiger \rangle$

$\langle getTiger, tameTiger, jumpTamedTiger \rangle$ works as well, but the previous relaxed plan is "better" ...

h^+: The Ideal Delete Relaxation Heuristic

Definition (h^+). *Let $\Pi = (V, A, I, G)$ be a planning task, with state set S. The optimal delete relaxation heuristic h^+ for Π is the function $h^+ : S \mapsto \mathbb{R}_0^+ \cup \{\infty\}$ where $h^+(s)$ is the cost of an optimal relaxed plan for s.*

Good news: (Notation: h^* is the perfect heuristic)

Proposition (h^+ is Admissible). *Let Π be a planning task, with state set S. Then, for every $s \in S$, $h^+(s) \leq h^*(s)$.*

Proof. Every real plan for a state s in Π also is a relaxed plan for $s^+ := s$ in Π.

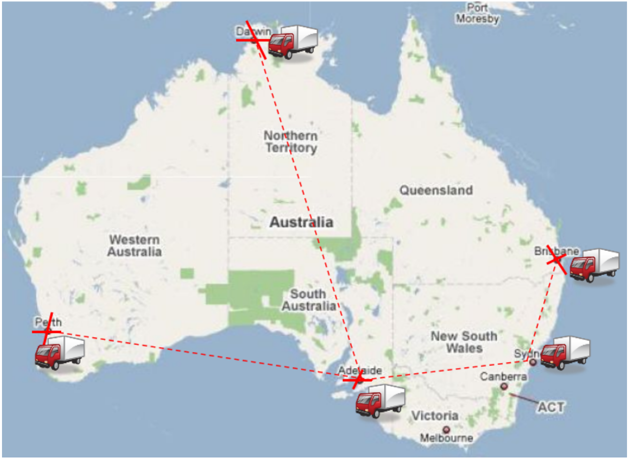
Bad news:

Proposition (Computing h^+ is Hard). *Let PlanOpt$^+$ be the problem of deciding, given a planning task Π and $B \in \mathbb{R}_0^+$, whether there exists a relaxed plan for Π whose cost is at most B. Then PlanOpt$^+$ is* **NP***-complete.*

Proof. Simple reduction from SAT. (First proved by [Bylander (1994)]; proof idea given on my lecture slides, available at `http://fai.cs.uni-saarland.de/teaching/`.)

→ (Quote Patrik Haslum) "So we approximate . . ."

Example: h^+ in TSP



$h^+(\text{TSP}) = $ Minimum Spanning Tree

Approximating h^+, Take I: h^1

Notation (Regression): $regr(g, a) = (g \setminus \mathit{eff}_a) \cup pre_a$ if (i) $\mathit{eff}_a \cap g \neq \emptyset$ and (ii) eff_a does not contradict g on any variable; else, $regr(g, a)$ is undefined.

Definition (h^1). *Let $\Pi = (V, A, I, G)$ be a planning task. The critical path heuristic h^1 is the function $h^1(s) := h^1(s, G)$ where $h^1(s, g)$ satisfies* $h^1(s, g) =$

$$\begin{cases} 0 & g \subseteq s \\ min_{a \in A, regr(g,a) \text{ is defined}}\, c_a + h^1(s, regr(g, a)) & |g| = 1 \\ \max_{p \in g} h^1(s, \{p\}) & |g| > 1 \end{cases}$$

→ Estimate the cost of sets g of subgoal facts (variable/value pairs) by the cost of the most costly fact $p \in g$.

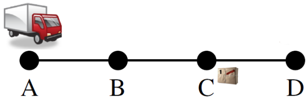
→ This is indeed a delete relaxation heuristic:

Proposition (h^1 estimates h^+). *Let Π be a planning task, with state set S. Then, for every $s \in S$, $h^1(s) \leq h^+(s)$.*

Proof. Regression is used only on singleton $g = \{p\}$, where by (i) $p \in \mathit{eff}_a$ and therefore (ii) is moot. (In terms of negative effects it is easier to see: for any $\neg q \in \mathit{eff}_a$, we have $q \neq p$ so $q \notin g$ and all negative effects are ignored.)

Computing h^1: Forward Fixpoint on Singleton Subgoals

Example "Logistics":



- State variables V: $truck : \{A, B, C, D\}$; $pack1 : \{A, B, C, D, T\}$. Short: t, p.
- Initial state I: $truck = A$, $pack1 = C$.
- Goal G: $truck = A$, $pack1 = D$.
- Actions A (unit costs): $drive(x, y)$, $load(p, x)$, $unload(p, x)$. Short: $dr(x, y), lo(x), ul(x)$.

i	$t = A$	$t = B$	$t = C$	$t = D$	$p = T$	$p = A$	$p = B$	$p = C$	$p = D$
0	0	∞	∞	∞	∞	∞	∞	0	∞
1	0	1	∞	∞	∞	∞	∞	0	∞
2	0	1	2	∞	∞	∞	∞	0	∞
3	0	1	2	3	3	∞	∞	0	∞
4	0	1	2	3	3	4	4	0	4
5	0	1	2	3	3	4	4	0	4

$\rightarrow$ So $h^1(I) = 4$. And for 100 packages with init C and goal D? Still $h^1(I) = 4$.

$\rightarrow$ h^1 is admissible, but is typically VERY uninformed. To the rescue: h^{FF}.

Approximating h^+, Take II: h^{FF}

Relaxed Plan Extraction for state s and **best-supporter function** bs

if $h^1(s) = \infty$ **then return** $h^{\mathsf{FF}}(s) := \infty$ **endif**
$Open := G \setminus s$
$Closed := \emptyset$
$RPlan := \emptyset$
while $Open \neq \emptyset$ **do**:
  select $p \in Open$
  $Open := Open \setminus \{p\}$; $Closed := Closed \cup \{p\}$;
  $RPlan := RPlan \cup \{bs(p)\}$; $Open := Open \cup (pre_{bs(p)} \setminus (s \cup Closed))$
endwhile
return $h^{\mathsf{FF}}(s) := |RPlan|$

$\rightarrow$ Starting with the top-level goal facts $p \in G$, iteratively close open singleton subgoal-facts p by selecting the best supporter $bs(p)$ for p.

$\rightarrow$ Err, so where do we get bs from? From h^1.

Best-Supporter Function from h^1

Definition (Best-Supporters from h^1). *Let $\Pi = (V, A, I, G)$ be a planning task, and let s be a state. Denote $P := \{(v = d) \mid v \in V, d \in D_v\}$. The h^1 supporter function $bs^1_s : \{p \in P \mid 0 < h^1(s, \{p\}) < \infty\} \mapsto A$ is defined by $bs^1_s(p) := \arg\min_{a \in A, p \in add_a} c(a) + h^1(s, pre_a)$.*

Example "Logistics":

Heuristic values:

	$t = A$	$t = B$	$t = C$	$t = D$	$p = T$	$p = A$	$p = B$	$p = C$	$p = D$
h^1	0	1	2	3	3	4	4	0	4

Yield best-supporter function:

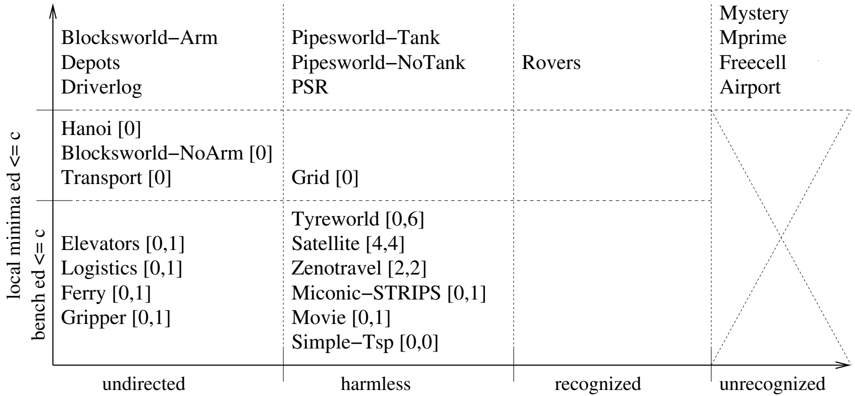
	$t = A$	$t = B$	$t = C$	$t = D$	$p = T$	$p = A$	$p = B$	$p = C$	$p = D$
bs^1	–	$dr(A, B)$	$dr(B, C)$	$dr(C, D)$	$lo(C)$	$ul(A)$	$ul(B)$	–	$ul(D)$

Yield relaxed plan extraction:

$p = D \to ul(D) \to t = D, p = T; t = D \to dr(C, D) \to t = C; t = C \to dr(B, C) \to t = B;$
$t = B \to dr(A, B) \to t = A; p = T \to lo(C) \to t = C, p = C.$

$\to$ So $h^{\mathsf{FF}}(I) = 5$. And for 100 packages with init C and goal D?
$h^{\mathsf{FF}}(I) = 103 = h^+(I) \gg h^1(I) = 4$ (compare slide 18).

Proved Quality of h^+ (Largely Inherited by h^{FF}) [Hoffmann (2005)]



Legend: x-axis: 4 classes regarding dead ends, each domain in "highest" class of any of its instances. y-axis: Does there exist a constant bound on exit distance from bench states and/or local minimum states in the domain? [lm,bench] where both exist, [lm] where only the latter exists; lm=0 means no local minima at all. Bottom right crossed out since unrecognized dead ends imply infinite exit distance.

h^1 vs. h^+ vs. h^{FF}: Properties

Proposition ($h^1 \leq h^+ \leq h^{\mathrm{FF}}$). *Let Π be a planning task, with state set S. Then, for every $s \in S$, $h^1(s) \leq h^+(s) \leq h^{\mathrm{FF}}(s)$. There exist Π and s where $h^{\mathrm{FF}}(s) > h^*(s)$.*

Proof. $h^1(s) \leq h^+(s)$: see slide 17; $h^+(s) \leq h^{\mathrm{FF}}(s)$ as relaxed plan extraction yields a relaxed plan. $h^{\mathrm{FF}}(s) > h^*(s)$ happens when h^1 best supporters make bad decisions (left for you as a little thinking exercise).

$\rightarrow$ h^{FF} is much more informative than $h^{\max}$. But it is not admissible.

$\rightarrow$ h^{FF} is extremely useful for satisficing planning, where plan optimality is not guaranteed (we just try to find *some* reasonably good plan).

Proposition (Agreement on ∞). *Let Π be a planning task, with state set S. Then, for every $s \in S$, $h^1(s) = \infty \Leftrightarrow h^+(s) = \infty \Leftrightarrow h^{\mathrm{FF}}(s) = \infty$.*

Proof. $h^{\mathrm{FF}}(s) = \infty$ iff $h^1(s) = \infty$ by definition. h^1 returns $\infty \Rightarrow$ some subgoal fact cannot be achieved $\Rightarrow h^+(s) = \infty$; done with $h^1 \leq h^+$.

$\rightarrow$ For dead-end detection, all heuristics so far are the same. (More later!)

Historical/Literature Notes

- Critical path heuristics root in Graphplan [Blum and Furst (1997)], and can be defined and computed over arbitrary-size subgoals [Haslum and Geffner (2000)], and even over arbitrary choices of atomic subgoals [Hoffmann and Fickert (2015)]. (We'll get back to the latter later.)
- The delete relaxation goes back to [McDermott (1999); Bonet and Geffner (2001)].
- Relaxed plan extraction and h^{FF} were first proposed in the FF system [Hoffmann and Nebel (2001)]. They were taken up extremely widely and still form a reasonable baseline in satisficing planning today.

  → In particular, an easy-to-modify-and-extend baseline that you might consider in your own work.
- Nevertheless, the delete relaxation has many pitfalls. To the rescue: See next.

Partial Delete Relaxation

Delete Relaxation:

Relaxed world: (after)

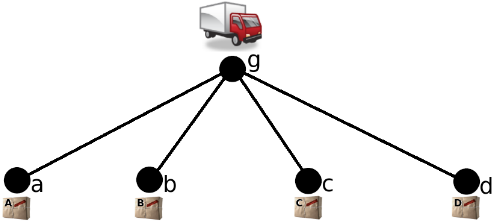
Partial Delete Relaxation:

interpolation parameter

h^+ $\longleftrightarrow$ h^*

Why? Resource consumption! Moving to-and-fro!

Moving To-and-Fro: Example "Star-Shape Logistics"



- State variables: $v_T : \{g, a, b, c, d\}$; $v_A, v_B, v_C, v_D : \{t, g, a, b, c, d\}$.
- Initial state: $v_T = g, v_A = a, v_B = b, v_C = c, v_D = d$.
- Goal: $v_A = g, v_B = g, v_C = g, v_D = g$.
- Actions (unit costs): $drive(x, y), load(x, y), unload(x, y)$.
  E.g., $load(x, y)$ has precondition $v_T = y, v_x = y$ and effect $v_x = t$.

$\rightarrow$ Relaxed plan for this task: $drive(g, a)$, $drive(g, b)$, $drive(g, c)$, $drive(g, d)$, $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$. Thus: $h^+ = 12 < 16 = h^*$.

$\rightarrow$ And with 100 star-leaf locations & packages? $h^+ = 300 \ll 400 = h^*$.

Red-Black Planning

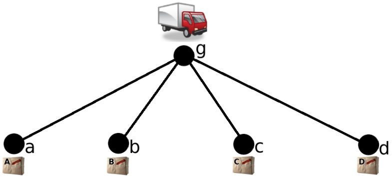
Definition (Red-Black State Transitions). *Let $\Pi = (V, A, I, G)$ be a planning task, and let $V^{\mathsf{B}} \cup V^{\mathsf{R}} = V$ be disjoint. A red-black state s^{RB} in Π is a set of variable/value pairs that assigns each $v \in V$ a subset of its domain, where $|s^{RB}(v)| = 1$ for $v \in V^{\mathsf{B}}$. The outcome state $s^{RB'}$ of applying a to s^{RB} is:*

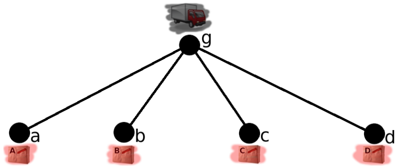
$$s^{RB'}(v) := \begin{cases} \mathit{eff}_a(v) & v \in V^{\mathsf{B}}, \mathit{eff}_a \text{ is defined on } v \\ s^{RB}(v) \cup \mathit{eff}_a(v) & v \in V^{\mathsf{R}}, \mathit{eff}_a \text{ is defined on } v \\ s^{RB}(v) & \text{otherwise} \end{cases}$$

$\rightarrow$ Black variables switch between values ("real semantics)", red variables accumulate them ("relaxed semantics"). $h^{*\mathsf{RB}}$: cost of optimal red-black plan.

b-it bots@Work, 2nd prize 2016 RoboCup@Work, uses Mercury [Domshlak *et al.* (2015)], which uses a red-black plan heuristic.

Red-Black Planning in Star-Shape Logistics





Relaxed plan:

1. Initial state: $\{v_T = g, \dots\}$.
2. Apply $drive(g, a)$:
   $\{v_T = g, v_T = a, \dots\}$.
3. Apply $drive(g, b)$:
   $\{v_T = g, v_T = a, v_T = b, \dots\}$.
4. $\dots$

Red-black plan:

1. Initial state: $\{v_T = g, \dots\}$.
2. Apply $drive(g, a)$:
   $\{v_T = a, \dots\}$.
3. Apply $drive(g, b)$:
   Not applicable!

$\rightarrow$ It's easy to see that any optimal red-black plan is a real plan here. In particular, $h^{*\mathsf{RB}}(I) = h^*(I)$.

Basic Observations about Red-Black Planning

Method: Given a planning task (V, A, I, G), choose a subset $V^{\mathsf{R}} \subseteq V$ of variables. The red-black relaxation of Π then is the red-black planning task $\Pi^{\mathsf{RB}} = (V^{\mathsf{B}} := V \setminus V^{\mathsf{R}}, V^{\mathsf{R}}, A, I, G)$.

- If we set $V^{\mathsf{R}} := V$, then $h^{*\mathsf{RB}} = h^+$.
- If we set $V^{\mathsf{R}} := \emptyset$, then $h^{*\mathsf{RB}} = h^*$.

$\rightarrow$ Red-black planning allows to naturally interpolate between h^+ and h^*.

$\rightarrow$ So, that's it? In our planner, we'll set $V^{\mathsf{R}} := \emptyset$ and be done? Nope: Computing $h^{*\mathsf{RB}}$ would just mean to solve the original planning task.

$\rightarrow$ Choosing V^{R} = Trading off between accuracy and overhead.

$\rightarrow$ *How many variables do we have to paint red in order to obtain a tractable (polynomial-time solvable) planning problem?*

Questionnaire

Question!

What if, in Star-Shape Logistics, instead of the truck we paint the packages black?

(A): $h^{*\mathrm{RB}} = h^*$

(B): $h^{*\mathrm{RB}} = h^+$

(C): We can't paint the packages black

(D): Honestly, I don't care what color the packages have

→ (A): No, because painting the packages black has no effect at all on the relaxed plan. The packages do not "move to-and-fro" anyway, each just makes two transitions to its goal value.

→ (B): Yes, see (A).

→ (C): We can paint whatever variable subset we want.

→ (D): In fact, it doesn't matter (to the heuristic value) what color the packages have: see (A). And that's actually the case for *any* causal graph leaf variables, which are "pure clients" and don't need to move to-and-fro (see [Katz *et al.* (2013)] for details).

"How Many Variables do We Have to Paint Red" $=$ All??

Theorem (Hardness for a Single Black Variable). *The problem of deciding, given a red-black planning task $\Pi^{\mathsf{RB}} = (V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G)$ where $|V^{\mathsf{B}}| = 1$, whether Π^{RB} is solvable, is* **NP***-complete.*

Proof Sketch. (Membership: Omitted) Hardness: By reduction from SAT.

- Red variables: For each variable $v_i \in \{v_1, \dots, v_m\}$ in the CNF, a variable v_i with domain $D_{v_1} = \{none, true, false\}$: Has v_i been assigned yet? And to which value? Initially $v_i = none$.
  For each clause $c_j \in \{c_1, \dots, c_n\}$ in the CNF, a Boolean variable sat_j: Has clause j been satisfied yet? Initially, sat_j is false; the goal requires it to be true.
- Black variable: v_0 with domain $D_{v_0} = \{1, \dots, n+1\}$: Whose variable's turn is it to be assigned? Initially, $v_0 = 1$.
- Actions that allow setting v_i from $none$ to either $true$ or $false$, provided that $v_0 = i$; apart from setting v_i, the actions also set $v_0 := i + 1$.
- Actions that allow to make sat_j true provided one of its literals has already been assigned to the correct truth value.

$\rightarrow$ We cannot "cheat" because the black "index variable" v_0 forces us to assign each v_i exactly once!

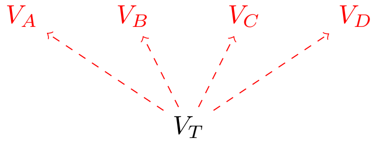
To the Rescue, Part I: The Black Causal Graph

Don't despair: The theorem just stated holds *unless we impose any additional restrictions on the planning task*.

$\rightarrow$ To the rescue: Restrict the structure of the black variables!

Definition (Black Causal Graph). *Let* $\Pi^{\mathsf{RB}} = (V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G)$ *be a red-black planning task. The black causal graph of* Π^{RB} *is the directed graph with vertices* V^{B} *and an arc* (u, v) *whenever there exists an action* $a \in A$ *so that either (i) there exists* $a \in A$ *so that* $pre_a(u)$ *and* $eff_a(v)$ *are both defined, or (ii) there exists* $a \in A$ *so that* $eff_a(u)$ *and* $eff_a(v)$ *are both defined.*

$\rightarrow$ The black causal graph in Star-Shape Logistics:



$\rightarrow$ Relevant for us here: There are no arcs between black variables.

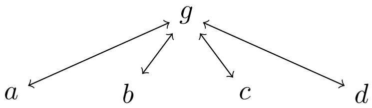
To the Rescue, Part II: Invertible Variables

Definition (Domain Transition Graph). *Let $\Pi = (V, A, I, G)$ be a planning task, and let $v \in V$. The domain transition graph (DTG) of v is the arc-labeled directed graph with vertices D_v, and, for every $d, d' \in D_v$ and $a \in A$ where either (i) $pre_a(v) = d$ and $eff_a(v) = d'$ or (ii) $pre_a(v)$ is not defined and $eff_a(v) = d'$, an arc $d \xrightarrow{a} d'$. We refer to $d \xrightarrow{a} d'$ as a value transition of v. We write $d \xrightarrow{a}_{\varphi} d'$ where $\varphi = pre_a \setminus \{v = d\}$ is the outside condition.*

Let $d \rightarrow_{\varphi} d'$ be a value transition of v. We say that $d \rightarrow_{\varphi} d'$ is invertible if there exists a value transition $d' \rightarrow_{\varphi'} d$ where $\varphi' \subseteq \varphi$.

Notation: A variable is invertible if all transitions in its DTG are invertible.

$\rightarrow$ The DTG of the truck variable v_T in Star-Shape Logistics:



$\rightarrow$ Relevant for us here: v_T is invertible.

The SMS Theorem

Theorem ("The SMS Theorem"). *Let $\Pi = (V, A, I, G)$ be an FDR planning task, and let $V^{\mathsf{R}} \subseteq V$ be a subset of its state variables. Say that, in the red-black relaxation of Π, the black causal graph does not contain any arcs, and all black variables are invertible. Then any relaxed plan for Π can in polynomial time be transformed into a red-black plan for Π.*

- **Idea:** Relaxed Plan Repair. Execute the relaxed plan step-by-step. If a black precondition (or goal) is not satisfied, we can move each black variable concerned into its required precondition/goal value separately.
- **Corollary (a):** If a relaxed plan exists, we can easily generate a red-black plan. **Trivial (b):** If no relaxed plan exists, then no red-black plan can exist either. From (a) + (b), we have a complete and efficient red-black planning procedure.
- **Usage:** On any state s encountered during search, generate a red-black plan for s and take its cost as the heuristic value. (= "In h^{FF}, replace relaxed plan by red-black plan.")

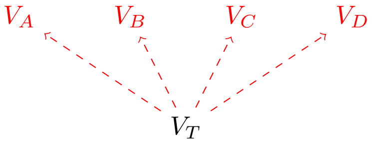
Questionnaire

Question!

Why is this called "The SMS Theorem"?

→ After spending 3 days examining the red-black tractability borderline during a visit to Carmel Domshlak in Haifa, I had this particular idea while already on the train to the airport. The proof took 3 SMS to write . . .

More importantly for us here: (reconsider black causal graph in the example)



→ We can leave "service variables" – like the robot positions! – black, and paint red only the "client variables". In fact, the SMS Theorem can be extended to allow arbitrary acyclic dependencies over the black variables.

(Which is what's inside Mercury, cf. slide 27.)

Relaxed Plan Repair

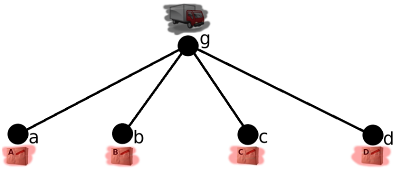
(no details here; see [Domshlak *et al.* (2015)])

// $\Pi = (V, A, c, I, G)$, *relaxed plan* $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$, *black and red variables* V^{B}, V^{R}
$\vec{a} := \langle a_1 \rangle$; $s := appl(I, a_1)$ // *"appl": red-black semantics (slide* **??***)*
for $i = 2$ to n **do** // *Repair black action preconditions*
  if $pre_{a_i}(V^{\mathsf{B}}) \not\subseteq s$ **then**
    $\vec{a}^{\mathsf{B}} :=$ Achieve$(s, pre_{a_i}(V^{\mathsf{B}}))$; $\vec{a} := \vec{a} \circ \vec{a}^{\mathsf{B}}$; $s := appl(s, \vec{a}^{\mathsf{B}})$
  endif
  $\vec{a} := \vec{a} \circ \langle a_i \rangle$; $s := appl(s, a_i)$
endfor
if $G(V^{\mathsf{B}}) \not\subseteq s$ **then** // *Repair black goals*
  $\vec{a}^{\mathsf{B}} :=$ Achieve$(s, G(V^{\mathsf{B}}))$; $\vec{a} := \vec{a} \circ \vec{a}^{\mathsf{B}}$
endif
return $\vec{a}$

Procedure: Achieve(s, g)
$\vec{a}^{\mathsf{B}} := \langle\rangle$
for $v \in V^{\mathsf{B}}$ s.t. $g(v)$ is defined **do** // *Move black variables into place separately*
  $\vec{a}^{\mathsf{B}} := \vec{a}^{\mathsf{B}} \circ$ invert path used by $\vec{a}$ from $I(v)$ to $s(v)$ // *Black variables are invertible*
  $\vec{a}^{\mathsf{B}} := \vec{a}^{\mathsf{B}} \circ$ path used by $\vec{a}^+$ from $I(v)$ to $g(v)$
endfor
return $\vec{a}^{\mathsf{B}}$

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



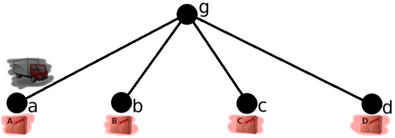
Relaxed Plan Repair:

- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $drive(g, a)$, $drive(g, b)$, $drive(g, c)$, $drive(g, d)$, $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$.

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

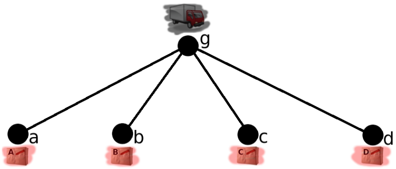
- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $drive(g, a)$, $drive(g, b)$, $drive(g, c)$, $drive(g, d)$, $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

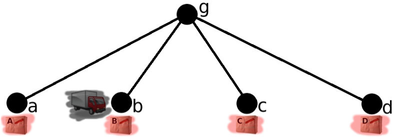
- Relaxed plan $\vec{a}^{+} = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $drive(g, b)$, $drive(g, c)$, $drive(g, d)$, $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g, b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

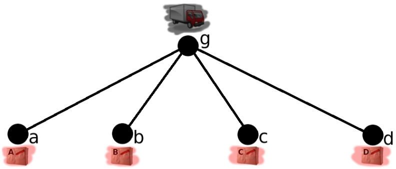
- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $drive(g,b)$, $drive(g,c)$, $drive(g,d)$, $load(A,a)$, $load(B,b)$, $load(C,c)$, $load(D,d)$, $unload(A,g)$, $unload(B,g)$, $unload(C,g)$, $unload(D,g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g,b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

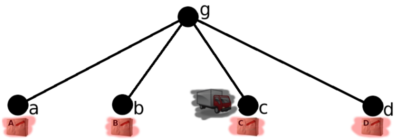
- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $drive(g, c)$, $drive(g, d)$, $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g, b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$
- $a_3 = drive(g, c)$: Move v_T back to g. Apply a_3, giving $v_T = c$, $v_x = \dots$

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

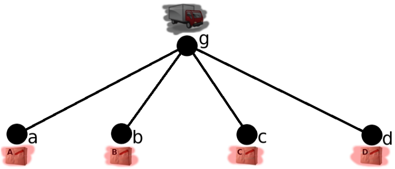
- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $drive(g,c)$, $drive(g,d)$, $load(A,a)$, $load(B,b)$, $load(C,c)$, $load(D,d)$, $unload(A,g)$, $unload(B,g)$, $unload(C,g)$, $unload(D,g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g,b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$
- $a_3 = drive(g,c)$: Move v_T back to g. Apply a_3, giving $v_T = c$, $v_x = \dots$

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

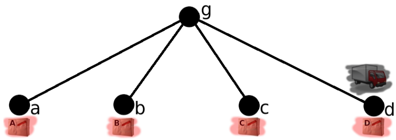
- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $drive(g, d)$, $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g, b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$
- $a_3 = drive(g, c)$: Move v_T back to g. Apply a_3, giving $v_T = c$, $v_x = \dots$
- ...

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

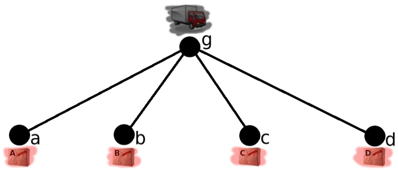
- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $drive(g, d)$, $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g, b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$
- $a_3 = drive(g, c)$: Move v_T back to g. Apply a_3, giving $v_T = c$, $v_x = \dots$
- ...

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

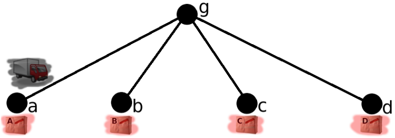
- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g, b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$
- $a_3 = drive(g, c)$: Move v_T back to g. Apply a_3, giving $v_T = c$, $v_x = \dots$
- ...

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

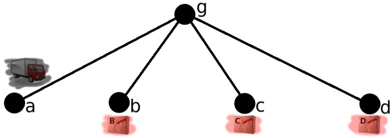
- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g, b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$
- $a_3 = drive(g, c)$: Move v_T back to g. Apply a_3, giving $v_T = c$, $v_x = \dots$
- $\dots$

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

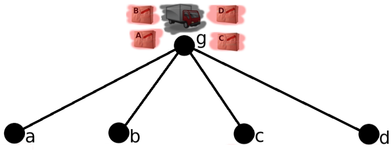
- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder: $load(A, a)$, $load(B, b)$, $load(C, c)$, $load(D, d)$, $unload(A, g)$, $unload(B, g)$, $unload(C, g)$, $unload(D, g)$.

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g, b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$
- $a_3 = drive(g, c)$: Move v_T back to g. Apply a_3, giving $v_T = c$, $v_x = \dots$
- ...

Relaxed Plan Repair in Star-Shape Logistics

[**Note:** In the illustration, the packages move. This is just for simplicity of illustration: as these variables are red, actually the packages accumulate their positions.]



Relaxed Plan Repair:

- Relaxed plan $\vec{a}^+ = \langle a_1, \dots, a_n \rangle$.
- $s :=$ red-black outcome of a_1 in init.
- For any black v, if $s(v) \neq z$ precondition of a_2: Move v to value z.
- $s :=$ red-black outcome of a_2.
- Proceed with $a_3, \dots, a_n$ and the goal.

Relaxed Plan Remainder:

- After a_1 moving v_T to a: $v_T = a$, $v_x = \dots$
- $a_2 = drive(g, b)$: Move v_T back to g. Apply a_2, giving $v_T = b$, $v_x = \dots$
- $a_3 = drive(g, c)$: Move v_T back to g. Apply a_3, giving $v_T = c$, $v_x = \dots$
- ...

Questionnaire

Question!

Does Relaxed Plan Repair yield an accurate heuristic function?

(A): Yes (B): No

→ Pro: It does "take some deletes into account" and can in this way improve over standard relaxed plan heuristics.

→ Contra: It may drastically over-estimate! See previous slide: The relaxed plan schedules all truck moves up front, to the effect that the repaired red-black plan starts off by moving the truck all over the place uselessly, only to have to do it all again when the load/unload actions come up ...

→ Commitments made in the relaxed plan are typically bad commitments in the red-black plan, leading to pver-estimation. We must thus "commit less" to the relaxed plan. Good news: this is possible, see [Domshlak *et al.* (2015)]. (Also good news? Omitted here.)

Historical/Literature Notes

- The weaknesses of the delete relaxation were recognized early, and many proposals have been made to address them (e. g. [Do and Kambhampati (2001); Fox and Long (2001); Gerevini *et al.* (2003); Helmert (2004); Keyder and Geffner (2008); Helmert and Geffner (2008); Baier and Botea (2009)]).
- The first technique able to actually interpolate between h^+ and h^*, however, was put forward only in 2012: atomic conjunctions [Haslum (2012); Keyder *et al.* (2012); Hoffmann and Fickert (2015); Fickert *et al.* (2016)].

  This family of heuristic functions combines critical path heuristics with the delete relaxation, performing relaxed planning over an arbitrary set C of atomic subgoals. One can always choose C so that $h^{C+} = h^*$.
- Red-black planning was put forward a year later. (Although its basic idea is actually much simpler?)
- A third method, that was actually explored before but is somewhat trivial, is variable pre-merging [van den Briel *et al.* (2007); Seipp and Helmert (2011)].

  This replaces subsets of variables with their product in the planning task input. If we pre-merge all variables, we get $h^+ = h^*$.
- Pre-merging can be simulated by explicit conjunctions; other than that, the methods are complementary [Hoffmann *et al.* (2014b)].

Highlight Nr. 1: Summing Arbitrary Admissible Heuristics

1	2	3	4
5	6	7	

		3	
			8
9	10	11	12
13	14		15

→ Is the sum of the two heuristics admissible? No, because the same moves of tile 3 may be counted by both abstractions.

→ But what if, on each side, we count only 0.5 for each move of tile 3? Then yes, because "duplicate moves" will be accounted for as cost 1.

Cost Partitioning

Definition (Cost Partitioning). *Let Π be a planning task with actions A and cost function c. An ensemble of functions $c_1, \dots, c_n : A \mapsto \mathbb{R}_0^+$ is a cost partitioning for Π if, for all $a \in A$, $\sum_{i=1}^{n} c_i(a) \leq c(a)$. If $h_1, \dots, h_n$ are heuristic functions for Π, then their partitioned sum is $\sum_{i=1}^{n} h_i[c_i]$.*

$\rightarrow$ Cost partitionings distribute the cost of each action across a set of otherwise identical planning tasks. This technique can be used to admissibly combine arbitrary admissible heuristic functions.

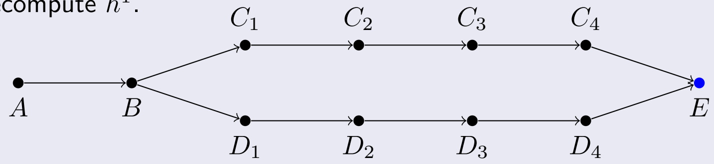
- First proposed by [Katz and Domshlak (2008)], then investigated widely in planning [e. g. Karpas *et al.* (2011); Pommerening *et al.* (2015); Seipp *et al.* (2017)].
- For landmark heuristics and abstraction heuristics (see next), an optimal cost partitioning – one that maximizes the value of the partitioned sum in a given state s – is computable in polynomial time, via an LP encoding [Katz and Domshlak (2008); Karpas and Domshlak (2009); Katz and Domshlak (2010)]!
- The well-known and wide-spread concept of independent heuristics [Korf and Felner (2002); Felner *et al.* (2004)] is the special case of 0/1 cost partitionings!

Highlight Nr. 2: Almost h^+!

LM-cut [Helmert and Domshlak (2009)], here only the rough intuition

Graph over facts p, with an edge $p \xrightarrow{a} q$ for $q \in \mathit{eff}_a$ and $p \in \mathit{pre}_a$ s.t. $h^1(s, \mathit{pre}_a) = h^1(s, \{p\})$.

$h := 0$; **Loop do:** Find a *cut* C between the initial state and the "0-cost goal zone"; $c := \min_{a \in C} c(a)$; $h := h + c$; reduce the cost of each action in C by c, and recompute h^1.

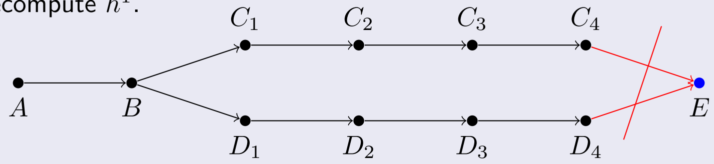


Highlight Nr. 2: Almost h^+!

LM-cut [Helmert and Domshlak (2009)], here only the rough intuition

Graph over facts p, with an edge $p \xrightarrow{a} q$ for $q \in \mathit{eff}_a$ and $p \in \mathit{pre}_a$ s.t. $h^1(s, \mathit{pre}_a) = h^1(s, \{p\})$.

$h := 0$; **Loop do:** Find a *cut* C between the initial state and the "0-cost goal zone"; $c := \min_{a \in C} c(a)$; $h := h + c$; reduce the cost of each action in C by c, and recompute h^1.

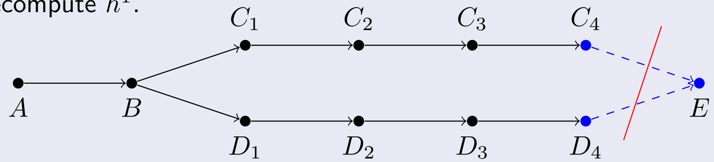


Highlight Nr. 2: Almost h^+!

LM-cut [Helmert and Domshlak (2009)], here only the rough intuition

Graph over facts p, with an edge $p \xrightarrow{a} q$ for $q \in \mathit{eff}_a$ and $p \in \mathit{pre}_a$ s.t. $h^1(s, \mathit{pre}_a) = h^1(s, \{p\})$.

$h := 0$; **Loop do:** Find a *cut* C between the initial state and the "0-cost goal zone"; $c := \min_{a \in C} c(a)$; $h := h + c$; reduce the cost of each action in C by c, and recompute h^1.

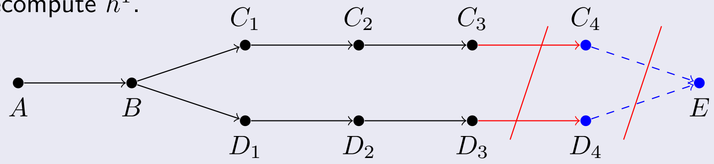


Highlight Nr. 2: Almost h^+!

LM-cut [Helmert and Domshlak (2009)], here only the rough intuition

Graph over facts p, with an edge $p \xrightarrow{a} q$ for $q \in \mathit{eff}_a$ and $p \in \mathit{pre}_a$ s.t. $h^1(s, \mathit{pre}_a) = h^1(s, \{p\})$.

$h := 0$; **Loop do:** Find a *cut* C between the initial state and the "0-cost goal zone"; $c := \min_{a \in C} c(a)$; $h := h + c$; reduce the cost of each action in C by c, and recompute h^1.

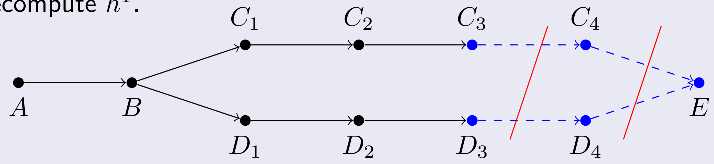


Highlight Nr. 2: Almost h^+!

LM-cut [Helmert and Domshlak (2009)], here only the rough intuition

Graph over facts p, with an edge $p \xrightarrow{a} q$ for $q \in \mathit{eff}_a$ and $p \in \mathit{pre}_a$ s.t. $h^1(s, \mathit{pre}_a) = h^1(s, \{p\})$.

$h := 0$; **Loop do:** Find a *cut* C between the initial state and the "0-cost goal zone"; $c := \min_{a \in C} c(a)$; $h := h + c$; reduce the cost of each action in C by c, and recompute h^1.

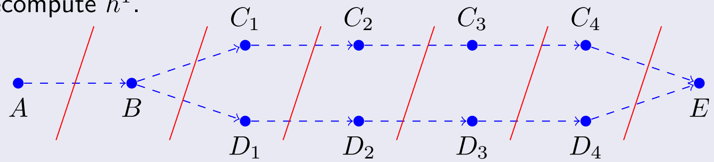


Highlight Nr. 2: Almost h^+!

LM-cut [Helmert and Domshlak (2009)], here only the rough intuition

Graph over facts p, with an edge $p \xrightarrow{a} q$ for $q \in \mathit{eff}_a$ and $p \in \mathit{pre}_a$ s.t. $h^1(s, \mathit{pre}_a) = h^1(s, \{p\})$.

$h := 0$; **Loop do:** Find a *cut* C between the initial state and the "0-cost goal zone"; $c := \min_{a \in C} c(a)$; $h := h + c$; reduce the cost of each action in C by c, and recompute h^1.

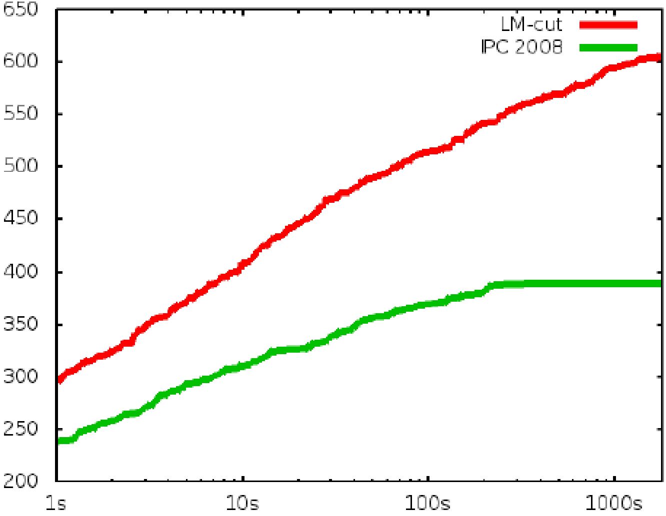


- Each cut is a landmark: a set of actions at least one of which must be used on every plan [Hoffmann *et al.* (2004); Karpas and Domshlak (2009)].
- Given a landmark C, c as above is a lower bound on cost-to-goal.
- Reducing the cost in each step iteratively yields a cost partitioning over these lower bounds.

Almost h^+! (table verbatim from [Helmert and Domshlak (2009)])

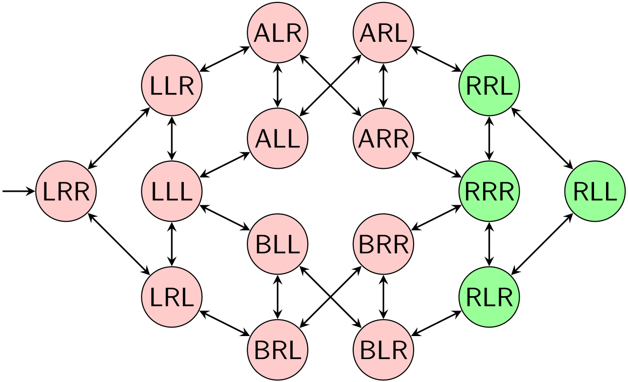
Domain	h^+	$h^{\max}$	HBG	CFLS	h^{LA}	$h^{LM\text{-cut}}$
Airport (37)	114.38	36.68	n/a	110.49	108.97	**114.38**
Blocks (35)	17.37	7.54	16.86	12.00	**17.37**	**17.37**
Gripper (20)	47.00	2.00	n/a	**47.00**	**47.00**	**47.00**
Logistics-2000 (26)	35.12	5.85	31.42	33.81	35.00	**35.12**
Miconic-STRIPS (150)	50.47	2.99	5.11	32.00	**50.47**	**50.47**
Pathways (5)	15.60	5.80	5.80	9.00	7.60	**15.60**
PSR-Small (50)	3.14	1.46	2.78	2.46	**3.14**	**3.14**
TPP (18)	32.17	6.39	n/a	n/a	17.61	**32.17**
Depot (10)	20.90	4.70	14.80	17.40	17.50	**20.50**
Driverlog (14)	15.50	4.71	10.71	12.00	13.43	**15.00**
Grid (2)	15.00	10.50	10.50	11.50	11.50	**14.00**
Logistics-1998 (10)	27.90	5.30	n/a	22.10	23.50	**27.70**
MPrime (24)	5.42	3.54	n/a	4.38	3.42	**4.92**
Rovers (14)	18.21	3.71	12.21	11.43	11.64	**18.00**
Satellite (9)	17.11	3.00	4.22	9.33	15.89	**16.89**
Zenotravel (13)	11.62	2.85	9.08	9.46	11.00	**11.54**
FreeCell (6)	8.33	3.00	7.00	3.33	**7.67**	7.17
Mystery (18)	6.44	3.56	n/a	4.72	3.67	**5.39**
Openstacks (5)	21.00	4.00	12.00	17.00	**21.00**	17.20
Pipesworld-NoTankage (18)	10.28	4.33	n/a	4.50	7.17	**8.28**
Pipesworld-Tankage (11)	8.36	3.91	n/a	3.91	6.27	**6.82**
Trucks (10)	21.70	4.00	**20.50**	9.90	14.60	**20.50**
avg. additive error compared to h^+		27.99	17.37	8.05	1.94	**0.28**
avg. relative error compared to h^+		68.5%	40.9%	25.2%	9.5%	**2.5%**

Huge Performance Improvement! (data courtesy of Malte Helmert)



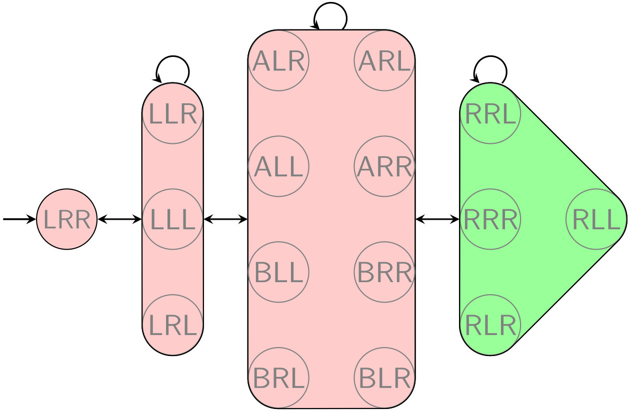
Highlight Nr. 3: Abstraction Heuristics

Concrete state space:

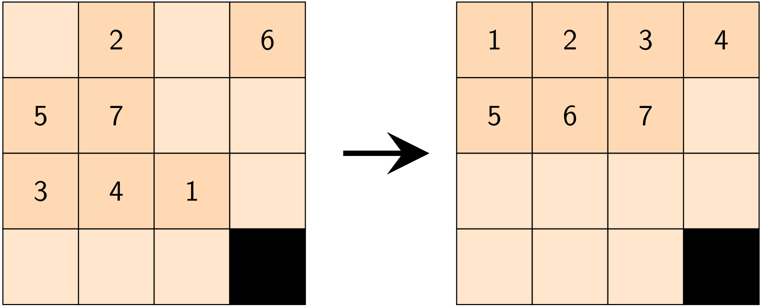


Highlight Nr. 3: Abstraction Heuristics

Abstract state space:



Highlight Nr. 3: Abstraction Heuristics

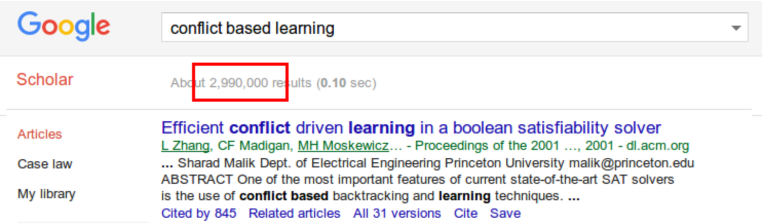


$h :=$ **Solution in abstract state space of projected puzzle**

$\rightarrow$ Lots of methods for doing this effectively! PDBs project onto part of the problem [e. g. Culberson and Schaeffer (1998); Edelkamp (2001); Pommerening *et al.* (2013)]; merge-and-shrink can compactly compute much more general abstractions [e. g. Dräger *et al.* (2009); Helmert *et al.* (2014)]; Cartesian abstraction allows effective abstraction-refinement operations [e. g. Seipp and Helmert (2013)]; saturated cost partitioning allots each abstraction only the minimum cost function needed for the lower bound [e. g. Seipp *et al.* (2017)].

Highlight Nr. 4: State-Space Conflict Based Learning

Conflict Based Learning:

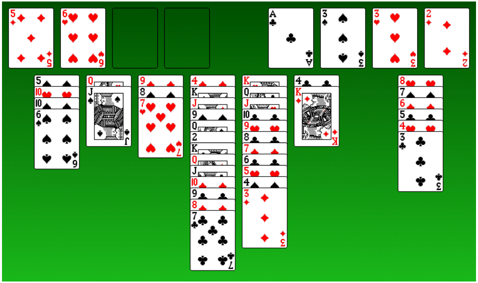


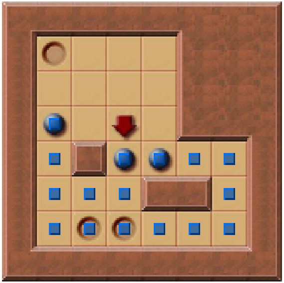
. . . **yes, but:** "State-space"

- Conflict-based learning is ubiquitous in constraint reasoning.
- Planning/reachability checking: Limited to bounded-length reachability (which is a form of constraint reasoning, easily encoded into e. g. SAT).

→ Can we learn from conflicts in unbounded-length state space search?

State Space Conflicts & Learning





→ Conflict = *dead-end* state from which the goal is unreachable.

Idea:

- Run Tarjan's algorithm; learn upon backtracking out of SCC.
- Use dead-end detector Δ.
- When learning on s, refine Δ so that it detects s.
- So far: Δ based on atomic conjunctions [Steinmetz and Hoffmann (2017b)], Δ based on traps [Lipovetzky *et al.* (2016); Steinmetz and Hoffmann (2017a)].

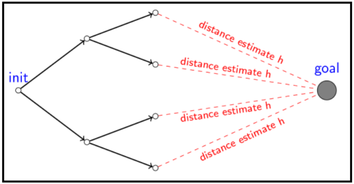
Search Space Reduction (on Resource-Constrained Planning)

vs. merge-and-shrink abstraction Δ^{MS} [Hoffmann et al. (2014a)]:

	Number of Tasks Solved or Proved Unsolvable											
	Transport (30)				Rovers (30)				TPP (5)			
	h^{FF}	DFS		h^{FF}	h^{FF}	DFS		h^{FF}	h^{FF}	DFS		h^{FF}
W	(Δ^1)	Δ^1	Δ^C	Δ^{MS}	(Δ^1)	Δ^1	Δ^C	Δ^{MS}	(Δ^1)	Δ^1	Δ^C	Δ^{MS}
0.5	25	25	**30**	**30**	3	5	**30**	29	4	4	**5**	**5**
0.6	15	15	**30**	**30**	2	2	**30**	25	1	1	**5**	**5**
0.7	5	7	**29**	**29**	0	0	**30**	23	0	0	2	**3**
0.8	0	0	**26**	**26**	0	0	**29**	21	0	0	**1**	**1**
0.9	0	0	14	**24**	0	0	**24**	13	0	0	0	0
1.0	2	1	12	**20**	0	0	**23**	6	0	**1**	**1**	0
1.1	3	0	12	**20**	0	0	**23**	5	0	0	**2**	1
1.2	5	4	14	**21**	0	0	**18**	1	0	2	**3**	2
1.3	6	5	13	**23**	0	0	**20**	2	2	2	**5**	4
1.4	12	3	17	**24**	0	1	**19**	4	3	3	**5**	4
$\sum$	73	60	197	**247**	5	8	**246**	129	10	13	**29**	25

→ Search space reduction factors (min/geomean/max) for learning vs. no-learning, on unsolvable tasks: Transport $6.7/436.5/37561.5$; Rovers $65.0/1286.6/69668.1$; TPP $190.0/711.9/1900.5$.

Planning under a Strict Time Limit



VS.

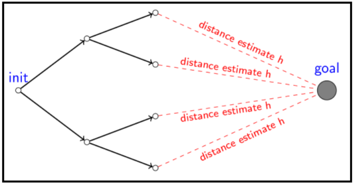
→ Robots usually have little opportunity to "stand around and think".

- Anytime search [e. g. Richter and Westphal (2010)]? Unaware of the deadline.
- Real-time search [e. g. Korf (1990)]? Immediate response, yet very risky.
- Limited horizon search? Controllable response time, but risky.
- Deadline-aware search [e. g. Burns *et al.* (2013)]? Better. (Don't search below s if predicting that not enough time to find a solution below s.) But what if time is not enough to find any complete (full lookahad) solution?

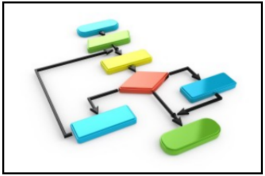
→ Rational search tree of fixed size, maximizing "confidence in decision"? (And/or inspiration from Qiescence search?)

→ Preparation at design-time? E.g. synergies with (partial/approximate) Verification; transfer of ideas from route planning [e. g. Geisberger *et al.* (2012)].

Model Assessment/Certification



VS.



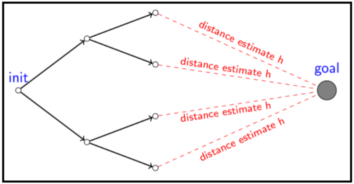
→ Explicit workflows are way easier to understand, assess, and trust than a complex planning machinery working on a general & flexible model.

- Support for modeling/model understanding in planning: knowledge engineering sub-community [e. g. Simpson *et al.* (2007); Vaquero *et al.* (2013)].
- Knowledge engineering competitions, see `http://www.icaps-conference.org/index.php/Main/Competitions`.
- But this is chronically under-addressed and way more needs to be done.

→ Talk to Tiago Vaquero here at the school!

→ Verification to assure that there won't be any "fatal plans"? How will the planner react to rare situations? Systematic testing, validation, certification processes?

Explanation



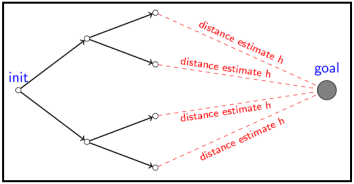
VS.

→ If you want to interact with people, you need to be able to explain yourself.

- E.g. "what I am about to do", to human co-worker in production environment.
- E.g. "why I am doing A and not B", to human supervisor in production environment.
- So far: "what" in terms of inner workings of the chosen plan [e. g. McGuinness *et al.* (2007); Khan *et al.* (2009); Seegebarth *et al.* (2012)]; "excuses" explaining why a plan does not exist (minimal modifications that would make the task solvable) [e. g. Göbelbecker *et al.* (2010)].

→ What about explaining the plan decision itself, "why this plan"? E.g. for simplicity, given state s and applicable actions A vs. B, why A and not B?

Deep Learning



VS.

→ Everybody's going crazy about it, so we must do it as well!

- AlphaGo definitely shows that "search + learning" can be key, and that NN can represent very informative search guidance.
- Same thing we've been doing in classical planning all along, yet using NN instead of model-based approximations for search guidance.

→ How to overcome the limitations of fixed input (game board) size and manual effort for NN/learning-process architecture design?

→ Using models & planning to generate robot-NN training data in simulations? Using model-based simulation to gain more confidence in NN decisions?

Last Slide

Thanks for your attention. Questions?

References I

Jorge A. Baier and Adi Botea. Improving planning performance using low-conflict relaxed plans. In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 10–17. AAAI Press, 2009.

Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1–2):279–298, 1997.

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

Ethan Burns, Wheeler Ruml, and Minh Binh Do. Heuristic search when time matters. *Journal of Artificial Intelligence Research*, 47:697–740, 2013.

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

Joseph C. Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

Minh. B. Do and Subbarao Kambhampati. Sapa: A domain-independent heuristic metric temporal planner. In A. Cesta and D. Borrajo, editors, *Proceedings of the 6th European Conference on Planning (ECP'01)*, pages 109–120. Springer-Verlag, 2001.

References II

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

Klaus Dräger, Bernd Finkbeiner, and Andreas Podelski. Directed model checking with distance-preserving abstractions. *International Journal on Software Tools for Technology Transfer*, 11(1):27–37, 2009.

Stefan Edelkamp. Planning with pattern databases. In A. Cesta and D. Borrajo, editors, *Proceedings of the 6th European Conference on Planning (ECP'01)*, pages 13–24. Springer-Verlag, 2001.

Ariel Felner, Richard Korf, and Sarit Hanan. Additive pattern database heuristics. *Journal of Artificial Intelligence Research*, 22:279–318, 2004.

Maximilian Fickert, Jörg Hoffmann, and Marcel Steinmetz. Combining the delete relaxation with critical-path heuristics: A direct characterization. *Journal of Artificial Intelligence Research*, 56(1):269–327, 2016.

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

References III

Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, 2012.

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.

Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 81–88. AAAI Press, 2010.

Patrik Haslum and Hector Geffner. Admissible heuristics for optimal planning. In S. Chien, R. Kambhampati, and C. Knoblock, editors, *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS'00)*, pages 140–149, Breckenridge, CO, 2000. AAAI Press, Menlo Park.

References IV

Patrik Haslum. Incremental lower bounds for additive cost planning problems. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 74–82. AAAI Press, 2012.

Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 162–169. AAAI Press, 2009.

Malte Helmert and Hector Geffner. Unifying the causal graph and additive heuristics. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric Hansen, editors, *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, pages 140–147. AAAI Press, 2008.

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.

References V

Malte Helmert. A planning heuristic based on causal graph analysis. In Sven Koenig, Shlomo Zilberstein, and Jana Koehler, editors, *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*, pages 161–170, Whistler, Canada, 2004. Morgan Kaufmann.

Jörg Hoffmann and Maximilian Fickert. Explicit conjunctions w/o compilation: Computing $h^{\mathsf{FF}}(\Pi^C)$ in polynomial time. In Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*. AAAI Press, 2015.

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

Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278, 2004.

Jörg Hoffmann, Ingo Weber, and Frank Michael Kraft. SAP speaks PDDL: Exploiting a software-engineering model for planning in business process management. *Journal of Artificial Intelligence Research*, 44:587–632, 2012.

References VI

Jörg Hoffmann, Peter Kissmann, and Álvaro Torralba. "Distance"? Who Cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In Thorsten Schaub, editor, *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, Prague, Czech Republic, August 2014. IOS Press.

Jörg Hoffmann, Marcel Steinmetz, and Patrik Haslum. What does it take to render $h^+(\pi^c)$ perfect? In *ICAPS 2014 Workshop on Heuristics and Search for Domain-Independent Planning (HSDIP'14)*, 2014.

Jörg Hoffmann. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research*, 24:685–758, 2005.

Erez Karpas and Carmel Domshlak. Cost-optimal planning with landmarks. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1728–1733, Pasadena, California, USA, July 2009. Morgan Kaufmann.

Erez Karpas, Michael Katz, and Shaul Markovitch. When optimal is just not good enough: Learning fast informative action cost-partitionings. In Fahiem Bacchus, Carmel Domshlak, Stefan Edelkamp, and Malte Helmert, editors, *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS'11)*, pages 122–129. AAAI Press, 2011.

References VII

Michael Katz and Carmel Domshlak. Optimal additive composition of abstraction-based admissible heuristics. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric Hansen, editors, *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, pages 174–181. AAAI Press, 2008.

Michael Katz and Carmel Domshlak. Optimal admissible composition of abstraction heuristics. *Artificial Intelligence*, 174(12–13):767–798, 2010.

Michael Katz, Jörg Hoffmann, and Carmel Domshlak. Who said we need to relax *all* variables? In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, pages 126–134, Rome, Italy, 2013. AAAI Press.

Emil Keyder and Hector Geffner. Heuristics for planning with action costs revisited. In Malik Ghallab, editor, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08)*, pages 588–592, Patras, Greece, July 2008. Wiley.

References VIII

Emil Keyder, Jörg Hoffmann, and Patrik Haslum. Semi-relaxed plan heuristics. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 128–136. AAAI Press, 2012.

Omar Zia Khan, Pascal Poupart, and James P. Black. Minimal sufficient explanations for factored markov decision processes. In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*. AAAI Press, 2009.

Richard E. Korf and Ariel Felner. Disjoint pattern database heuristics. *Artificial Intelligence*, 134(1–2):9–22, 2002.

Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

Nir Lipovetzky, Christian J. Muise, and Hector Geffner. Traps, invariants, and dead-ends. In Amanda Coles, Andrew Coles, Stefan Edelkamp, Daniele Magazzeni, and Scott Sanner, editors, *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS'16)*, pages 211–215. AAAI Press, 2016.

References IX

Drew V. McDermott. Using regression-match graphs to control search in planning. *Artificial Intelligence*, 109(1–2):111–159, 1999.

Deborah L. McGuinness, Alyssa Glass, Michael Wolverton, and Paulo Pinheiro da Silva. Explaining task processing in cognitive assistants that learn. In *Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference (FLAIRS'07)*, pages 284–289, 2007.

Florian Pommerening, Gabriele Röger, and Malte Helmert. Getting the most out of pattern databases for classical planning. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. AAAI Press/IJCAI, 2013.

Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. From non-negative to general operator cost partitioning. In Blai Bonet and Sven Koenig, editors, *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 3335–3341. AAAI Press, January 2015.

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

References X

Wheeler Ruml, Minh Binh Do, Rong Zhou, and Markus P. J. Fromherz. On-line planning and scheduling: An application to controlling modular printers. *Journal of Artificial Intelligence Research*, 40:415–468, 2011.

Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. Making hybrid plans more clear to human users - A formal approach for generating sound explanations. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*. AAAI Press, 2012.

Jendrik Seipp and Malte Helmert. Fluent merging for classical planning problems. In *ICAPS 2011 Workshop on Knowledge Engineering for Planning and Scheduling*, pages 47–53, 2011.

Jendrik Seipp and Malte Helmert. Counterexample-guided Cartesian abstraction refinement. In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, pages 347–351, Rome, Italy, 2013. AAAI Press.

References XI

Jendrik Seipp, Thomas Keller, and Malte Helmert. Narrowing the gap between saturated and optimal cost partitioning for classical planning. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, pages 3651–3657. AAAI Press, February 2017.

Ron M. Simpson, Diane E. Kitchin, and T. L. McCluskey. Planning domain definition using GIPO. *Knowledge Engineering Review*, 22(2):117–134, 2007.

Marcel Steinmetz and Jörg Hoffmann. Search and learn: On dead-end detectors, the traps they set, and trap learning. In Carles Sierra, editor, *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press/IJCAI, 2017.

Marcel Steinmetz and Jörg Hoffmann. State space search nogood learning: Online refinement of critical-path dead-end detectors in planning. *Artificial Intelligence*, 245:1 – 37, 2017.

Menkes van den Briel, Subbarao Kambhampati, and Thomas Vossen. Fluent merging: A general technique to improve reachability heuristics and factored planning. In *ICAPS 2007 Workshop on Heuristics for Domain-Independent Planning: Progress, Ideas, Limitations, Challenges (HDIP'07)*, 2007.

References XII

Tiago Stegun Vaquero, José Reinaldo Silva, Flavio Tonidandel, and J. Christopher Beck. itsimple: towards an integrated design system for real planning applications. *Knowledge Engineering Review*, 28(2):215–230, 2013.