Incremental Lower Bounds for Additive Cost Planning Problems

P@trik Haslum

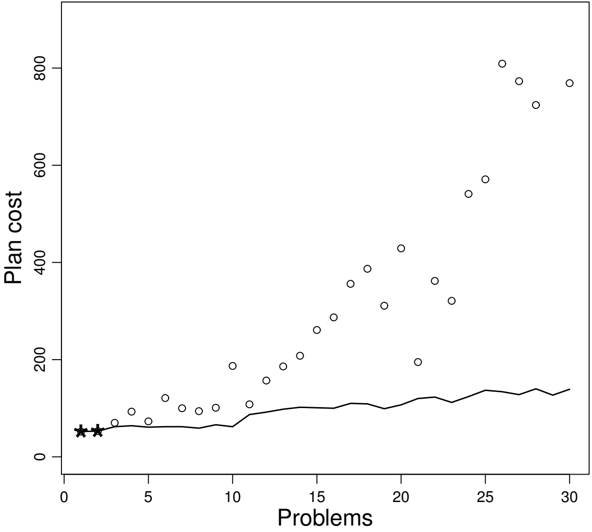
ICAPS 2012

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

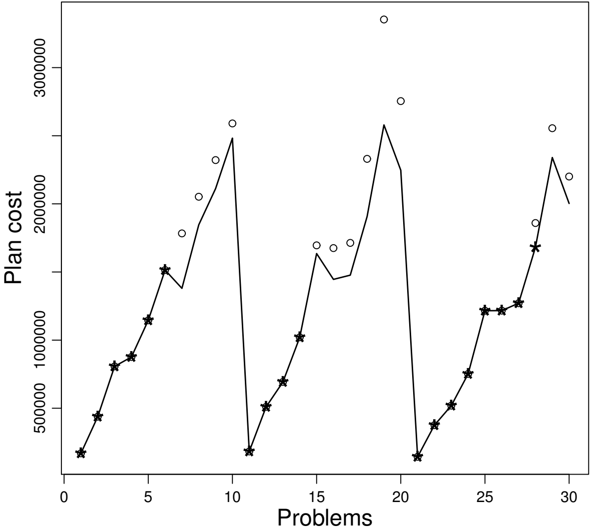
Australian National University
UNSW THE UNIVERSITY OF NEW SOUTH WALES
NSW GOVERNMENT Trade & Investment
Victoria The Place To Be
THE UNIVERSITY OF MELBOURNE
THE UNIVERSITY OF SYDNEY
Queensland Government
Griffith UNIVERSITY
QUT
THE UNIVERSITY OF QUEENSLAND AUSTRALIA

The Plan Quality Gap



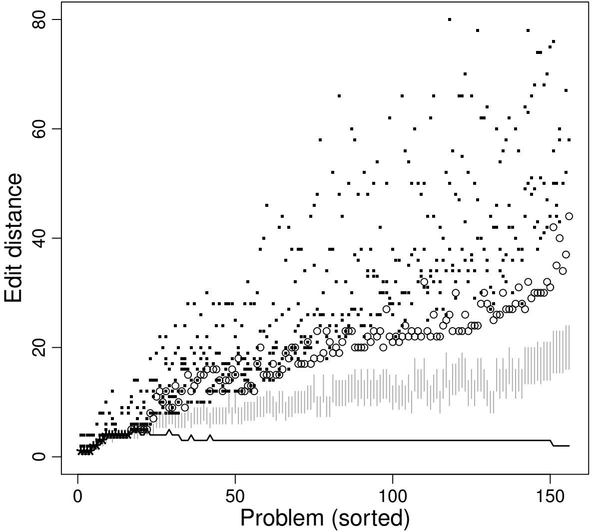
Elevators domain, IPC 2008

The Plan Quality Gap



ParcPrinter domain, IPC 2008

The Plan Quality Gap



Genome Edit Distance

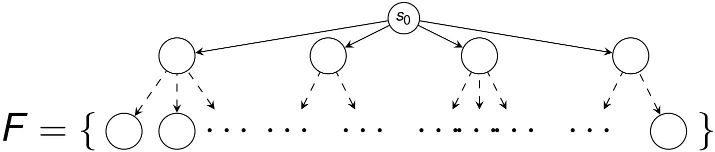
The Plan Quality Gap

- Optimal planners do not scale.
- Non-optimal planners fall far short of achievable plan quality.
- Lack of sufficiently strong lower bounds.

Admissible Heuristics & Search

NICTA

- If h is **admissible**, $h(s_0)$ is a lower bound.
- Strengthening via search (look-ahead):



- $f_{\min} = \min_{s \in F} (\text{cost}(s_0, s) + h(s))$ is also a lower bound.
- Many admissible heuristics for plan cost.

Strengthening via De-Relaxation

- A general idea...

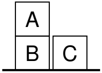
1. Solve a relaxation of the problem, optimally.
2. If the relaxed solution is also a real solution, it is optimal.
3. Else, use hints from the failure of the relaxed solution to strengthen the relaxation, and repeat from 1.

- ...with many instances:
  - Incremental generation of valid cuts in MIP.
  - Counterexample-guided abtraction refinement.
  - **Here:** Delete relaxation of planning problem.

The Delete Relaxation

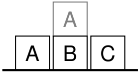
- The delete relaxation, P^+, of a planning problem P is exactly like P except del(a) $= \emptyset$ for each a.
- Relaxation: any plan for P is also valid for P^+.
  - Actions (and goal) require atoms to be *true*.
  - $h^+(s) = h^\star(P^+, s) \leq h^\star(P, s)$.
- Cost-optimal delete relaxed planning is "only" NP-hard (and often feasible in practice).
- No negative interactions in P^+:
  - Combining plans for separate goals always yields a valid plan for their conjunction.
  - This is not true in P.

The Delete Relaxation (example)



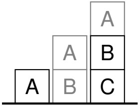
(on-table A)
(on-table B)
(on-table C)
(on A B)
(on A C)
(on B A)
(on B C)
(on C A)
(on C B)
(clear A)
(clear B)
(clear C)

(MoveToT$^+$ A B)
pre: (on A B),
(clear A)
add: (on-table A),
(clear B)



(on-table A)
(on-table B)
(on-table C)
(on A B)
(on A C)
(on B A)
(on B C)
(on C A)
(on C B)
(clear A)
(clear B)
(clear C)

(MoveFromT$^+$ B C)
pre: (on-table B),
(clear B), (clear C)
add: (on B C)



(on-table A)
(on-table B)
(on-table C)
(on A B)
(on A C)
(on B A)
(on B C)
(on C A)
(on C B)
(clear A)
(clear B)
(clear C)

Goal: (on A B),
(on B C)

Strengthening the Relaxation

- The P^C construction:
  - Represent $c = \{p_1, \ldots, p_k\}$ with a new atom π_c
  - Modify problem so π_c is true iff c.
- **Theorem**: $h^\star(P^C) = h^\star(P)$.
- **Theorem**: Let S be an optimal plan for P^+. If S is not valid for P, there is an (efficiently findable) $C = \{c_1, \ldots, c_n\}$ such that S is not valid for $(P^C)^+$.
- **Corollary**: $h^+(P^C) = h^\star(P)$ for large enough C.

The P^C Construction

- Let C be a set of sets of atoms (conjunctions).
- Atoms in P^C: atoms in P and $\{\pi_c \mid c \in C\}$.
- Notation:

$$x^C = x \cup \{\pi_c \mid c \subseteq x\}.$$
$$C^t(a) = \{c \in C \mid c \subseteq (\text{add}(a) \cup \text{pre}(a)) - \text{del}(a)\};$$
$$C^f(a) = \{c \in C \mid c \cap \text{del}(a) \neq \emptyset\};$$
$$C^p(a) = \{c \in C - C^t(a) \mid c \cap \text{add}(a) \neq \emptyset, c \cap \text{del}(a) = \emptyset\};$$
$$C^n(a) = \text{the rest.}$$

- Initial state: s_0^C
- Goal: G^C

The P^C Construction (cont'd)

- Actions in P^C: $\alpha_{a,X}$ with

$$\begin{aligned}
\text{pre}(\alpha_{a,X}) &= \Big(\text{pre}(a) \cup \bigcup_{c \in X} (c - \text{add}(a))\Big)^C \\
\text{add}(\alpha_{a,X}) &= \text{add}(a) \cup \{\pi_c \mid c \in C^t(a) \cup X\} \\
\text{del}(\alpha_{a,X}) &= \text{del}(a) \cup \{\pi_c \mid c \in C^f(a)\} \\
\text{cost}(\alpha_{a,X}) &= \text{cost}(a)
\end{aligned}$$

  for each action a and each $X \subseteq C^p(a)$ (downward closed).
- $|P^C|$ can be exponential in $|C|$.

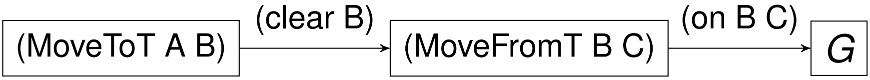
The P^C Construction (example)

- $c_1 = \{(\text{on A B}), (\text{on B C})\}$, $c_2 = \{(\text{clear B}), (\text{on B C})\}$
- $C = \{c_1, c_2\}$
- $\alpha_{(\text{MoveFromT A B}),\emptyset} \equiv (\text{MoveFromT A B}) + \text{del: } \pi_{c_2}$
- $\alpha_{(\text{MoveFromT A B}),\{c_1\}}$:
  - pre: (on-table A), (clear A), (clear B),
    (on B C), π_{c_2}
  - add: (on A B), π_{c_1}
  - del: (on-table A), (clear B), π_{c_2}

Conflict Extraction: What C?

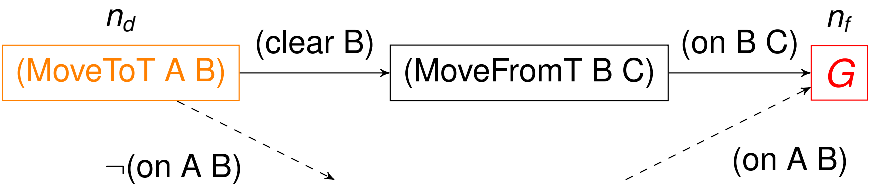
NICTA

- S is a valid plan for P^+.
- RPDG(S):
  - Graph over $\{n_a \mid a \in S\} \cup \{n_G\}$;
  - $n_a \xrightarrow{l} n'$ iff $l = \text{pre}(n') - R^+(S - \{a\}) \neq \emptyset$.
  - Transitively reduced.
- Example:



Conflict Extraction: What C?

- S is not a valid plan for P:
  - Some $p \in \text{pre}(n_f)$ fails to hold for some n_f.
  - p must have been deleted by some action (associated with n_d) before n_f.
- Example:



Conflict Extraction: What C?

- A **simple dependency path** from n to n' in RPDG(S) is a path with one (arbitrarily chosen) atom p from each edge label.
- A **dependency closure** D from n to n' in RPDG(S) is a minimal (w.r.t. $\subset$) union of paths such that:
  - D contains a simple dependency path from n to n'.
  - For all $n_a \xrightarrow{q} \in D$ and $b \in S$ such that $b \neq a$ and $q \in \text{add}(b)$, D contains a simple dependency path from n to n_b.

Conflict Extraction: What C?

- Case 1: Path in RPDG(S) from n_d to n_f.
  - Let D be a dependency closure from n_d to n_f.
  - $C = \{\{p, q\} \mid q$ labels an edge in $D\}$.
- Case 2: No path in RPDG(S) from n_d to n_f.
  - Let n_c be the first-in-S common descendant of n_d and n_f.
  - Let D_{n_d}, D_{n_f} be dependency closures from n_d to n_c and n_f to n_c.
  - $C = \{\{q, q'\} \mid q$ labels an edge in D_{n_d}, $q' = p$ or q' labels an edge in $D_{n_f}\}$.

Case 1:

$$n_d \xrightarrow[q_1]{} n_1 \xrightarrow[q_2]{} \cdots \xrightarrow[q_{m-1}]{} n_{m-1} \xrightarrow{q_m} n_f \xleftarrow{p}$$

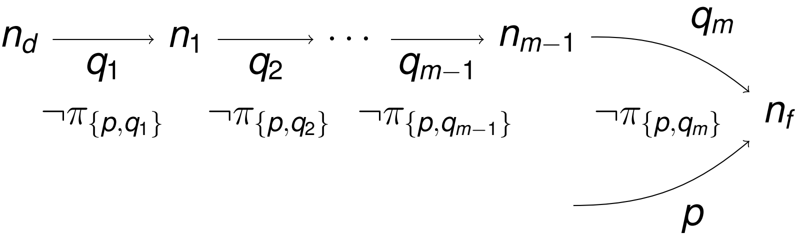
Case 2:

$$n_d \xrightarrow[q_1]{} n_1 \xrightarrow[q_2]{} \cdots \xrightarrow[q_{m-1}]{} n_{m-1} \xrightarrow{q_m} n_c$$

$$\xrightarrow[p]{} n_f \xrightarrow[q'_1]{} n'_1 \xrightarrow[q'_2]{} \cdots \longrightarrow n'_{l-1} \xrightarrow{q'_l} n_c$$

Why It Works

Case 1:



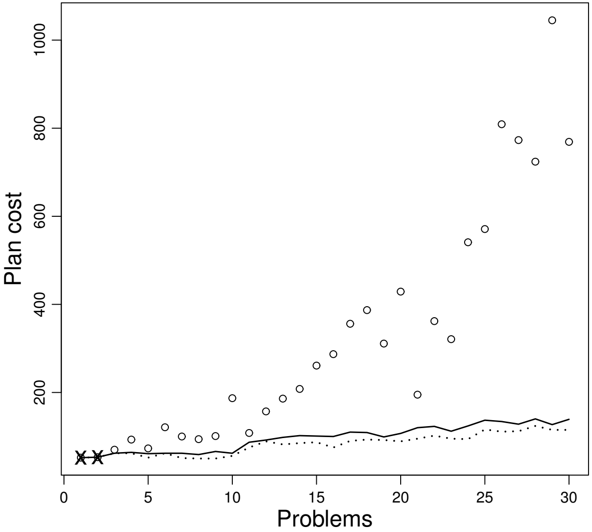
- No representative of a_d adds $\pi_{\{p,q_1\}}$ because $p \in \text{del}(a_d)$.
- Any representative of a_i that adds $\pi_{\{p,q_i\}}$ requires $\pi_{\{p,q_{i-1}\}}$.
- $\pi_{\{p,q_m\}} \in \text{pre}(n_f)$ cannot hold.

h^{++}

1. Compute an optimal plan S for P^+.
2. If S is valid for P, done (optimal plan).
3. If S is not valid for P, find C as above, set $P \coloneqq P^C$ and repeat from 1.

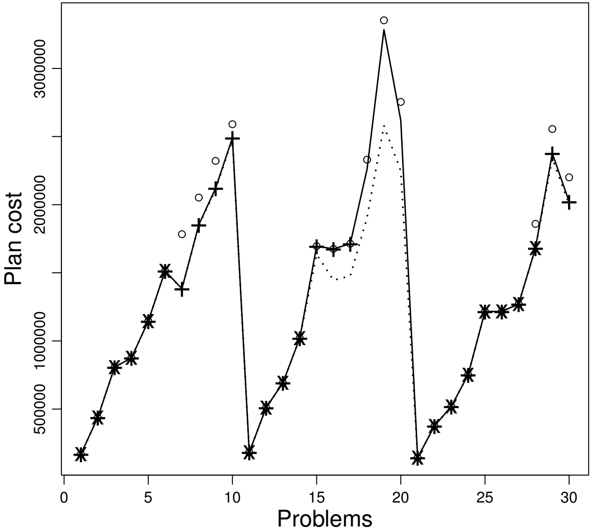
- How to compute S?
  - Iterative landmark-based algorithm.
    - Advantage: Anytime lower bound on h^+.
  - Reduction to weighted MaxSAT.
  - Specialisations of (heuristic) search.

Results



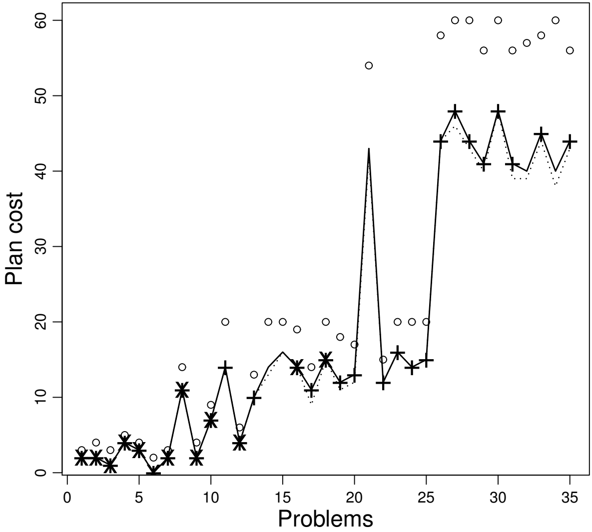
Elevators domain, IPC 2008

Results



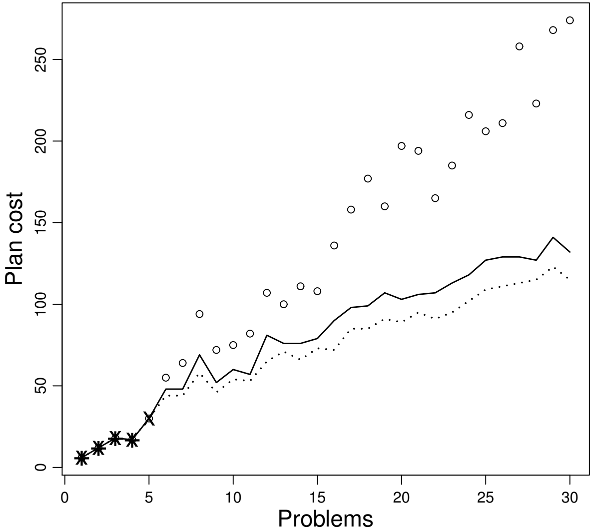
ParcPrinter domain, IPC 2008

Results



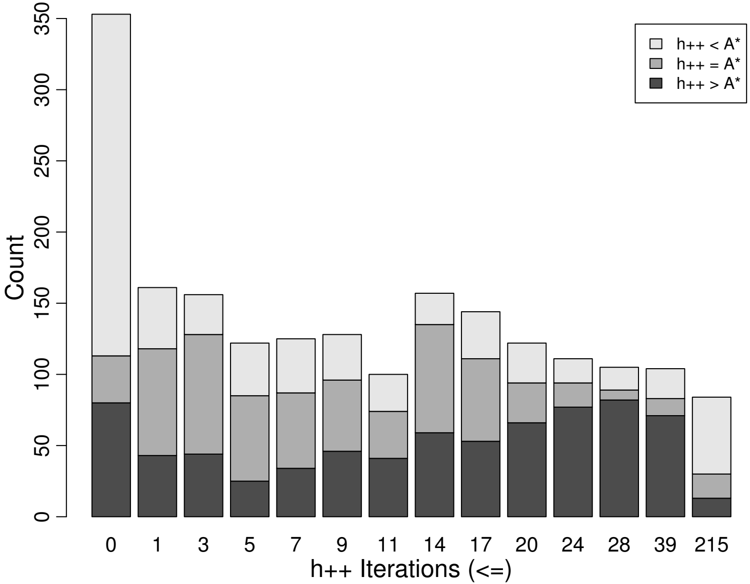
Power Network Alarm Processing domain

Results



Pathways Domain, IPC 2006

Results



h++ < A*
h++ = A*
h++ > A*
Count
0
50
100
150
200
250
300
350
0
1
3
5
7
9
11
14
17
20
24
28
39
215
h++ Iterations (<=)

Conclusions

- Finding good plans and proving good lower bounds are different problems – and should be attacked with different methods.
- The gap remains.
  - Current & future work: Finding better plans.
  - Apply iterative strengthening to abstractions.
- Planning can learn from other areas of optimisation.