How Good is Almost Perfect?

Malte Helmert   Gabriele Röger

Albert-Ludwigs-Universität Freiburg, Germany

AAAI 2008

Outline

Optimal sequential planning

Optimal sequential planning
$= \mathrm{A}^*$ (or similar)
$+$ admissible heuristic

(mostly)

Folklore

Everybody knows:

If a heuristic has constant absolute error,
A^* requires a linear number of node expansions.

Comparison: Heuristic vs. breadth-first search

Actually, state-of-the art optimal sequential planners are not much better than breadth-first search.

Experiments of Helmert, Haslum & Hoffmann (2007)

- BFHSP solved **37** tasks
- $\mathrm{A}^* + h^{\max}$ solved **46** tasks
- $\mathrm{A}^* + h^{\mathsf{PDB}}$ solved **54** tasks
- blind search solved **42** tasks

Are our heuristics bad?

Two possible explanations:

- Our heuristics aren't that good.
- There is something fishy going on.

(Or both.)

Folklore + fine print

Everybody knows:

If a heuristic has constant absolute error,
A^* requires a linear number of node expansions.

But. . .

This relies on several assumptions:

- fixed branching factor
- only a single goal state
- no transpositions

These assumptions do not hold in any common planning task!

Almost perfect heuristics

Almost perfect heuristics differ from the perfect heuristic h^* only by an additive constant:

Definition

Define heuristic $h^* - c$ (for $c \in \mathbb{N}_1$) as

$$(h^* - c)(s) := \max(h^*(s) - c, 0)$$

$\rightarrow$ unlikely to be obtainable in practice

The topic of this work

How many nodes must A^* expand for a planning task $\mathcal{T}$, given an almost perfect heuristic $h^* - c$?

Definition

$$N^c(\mathcal{T}) := \text{number of states } s \text{ with } g(s) + (h^* - c)(s) < h^*(\mathcal{T})$$

$\rightsquigarrow$ If this number grows fast with scaling task size, we have a problem.

Objective

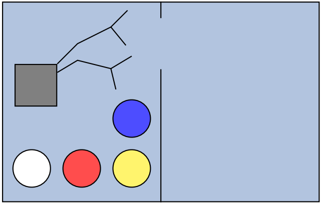
Results for $N^c(\mathcal{T})$ for IPC domains
$\longrightarrow$ Focus on domains in APX

Outline

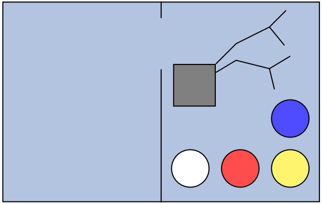
Our goal

Find sequence $(\mathcal{T}_n)$ of scaling tasks for which $N^c(\mathcal{T}_n)$ grows exponentially, even for small values of c.

Gripper



initial state



goal state

- $\mathcal{T}_n$: Task with n balls
- S_n: Total number of reachable states of $\mathcal{T}_n$

$$S_n = 2 \cdot (2^n + 2n2^{n-1} + n(n-1)2^{n-2})$$

Gripper
Theorem

Theorem

Let $n \in \mathbb{N}_0$ *with* $n \geq 3$.
If n *is even, then*

- $N^1(\mathcal{T}_n) = N^2(\mathcal{T}_n) = \frac{1}{2}S_n - 3$
- $N^c(\mathcal{T}_n) = S_n - 2n - 2$ *for all* $c \geq 3$.

If n *is odd, then*

- $N^1(\mathcal{T}_n) = N^2(\mathcal{T}_n) = S_n - 3$
- $N^c(\mathcal{T}_n) = S_n - 2$ *for all* $c \geq 3$.

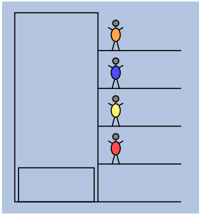
Gripper

Proof

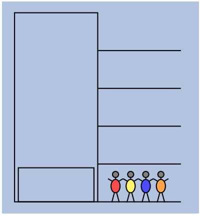
Proof sketch

- n is even
    - states with an even number of balls in each room
        - basically all are part of an optimal plan
    - states with an odd number of balls in each room
        - all are part of plans of length $h^*(\mathcal{T}_n) + 2$
- n is odd
    - basically all states are part of an optimal plan

Miconic-Simple-Adl



initial state



goal state

- $\mathcal{T}_n$: Task with n passengers (and $n+1$ floors)
- S_n: Total number of reachable states of $\mathcal{T}_n$

$$S_n = 3^n(n+1)$$

Miconic-Simple-Adl

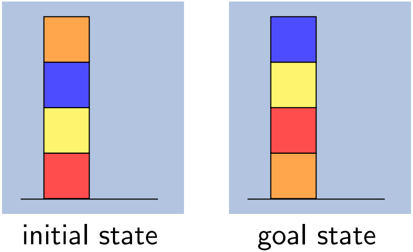
Theorem

Theorem

For all $c \geq 4$*:*

$$N^c(\mathcal{T}_n) = S_n - (2^n - 1)(n + 1).$$

Blocksworld



$\mathcal{T}_n$: Task with n blocks $(n \geq 2)$

Blocksworld

Theorem

Theorem

$$N^1(\mathcal{T}_n) = 4 \cdot \sum_{k=0}^{n-3} B_k + 3B_{n-2} + 1$$

n	$N^1(\mathcal{T}_n)$	n	$N^1(\mathcal{T}_n)$
2	4	9	3748
3	8	10	17045
4	15	11	84626
5	32	12	453698
6	82	13	2605383
7	253	14	15924744
8	914	15	103071652

Outline

Question

Theoretical results

There exist task families for which the number of states expanded by $h^* - c$ grows exponentially, even for small c.

Interesting question

Can we observe this behaviour in practice?

$\longrightarrow$ Experiments with IPC tasks

Problem

Problem

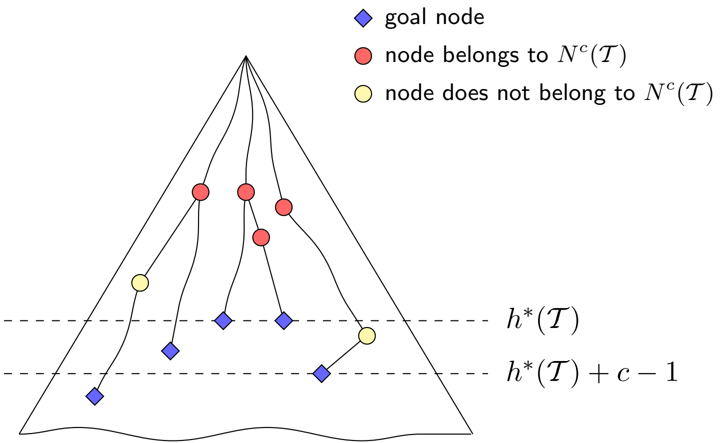
- Values $N^c(\mathcal{T})$ are defined in terms of h^*.
- Usually h^* cannot be determined efficiently.

Naive way of computing $N^c(\mathcal{T})$

- Completely explore the state space of $\mathcal{T}$.
- Search backwards from the goals to determine the $h^*(s)$ values.

$\longrightarrow$ Observation: Generating all states is not necessary.

Search space



$\rightsquigarrow$ Poster session: today, 6:00-9:30 PM

Results

Blocksworld

task	$h^*(\mathcal{T})$	$N^1(\mathcal{T})$	$N^2(\mathcal{T})$	$N^3(\mathcal{T})$	$N^4(\mathcal{T})$	$N^5(\mathcal{T})$
04-1	10	10	10	16	16	29
05-2	16	28	28	72	72	162
06-2	20	27	27	144	144	476
07-1	22	106	106	606	606	2244
08-1	20	66	66	503	503	2440
09-0	30	411	411	3961	3961	21135

Results
Gripper

task	$h^*(\mathcal{T})$	$N^1(\mathcal{T})$	$N^2(\mathcal{T})$	$N^3(\mathcal{T})$	$N^4(\mathcal{T})$
01	11	125	125	246	246
02	17	925	925	1842	1842
03	23	5885	5885	11758	11758
04	29	34301	34301	68586	68586
05	35	188413	188413	376806	376806
06	41	991229	991229	1982434	1982434
07	47	5046269	5046269	10092510	10092510

Results

Logistics (IPC 2)

task	$h^*(\mathcal{T})$	$N^1(\mathcal{T})$	$N^2(\mathcal{T})$	$N^3(\mathcal{T})$	$N^4(\mathcal{T})$	$N^5(\mathcal{T})$
4-0	20	159	408	1126	1780	2936
5-0	27	459	2391	5693	14370	21124
6-0	25	411	2160	5712	14485	23967
7-1	44	17617	111756	427944	1173096	
8-1	44	4843	27396	157645	558869	
9-0	36	2778	15878	61507	183826	460737
10-0	45	10847				
11-0	48	10495				

Results

Miconic-Simple-Adl

task	$h^*(\mathcal{T})$	$N^1(\mathcal{T})$	$N^2(\mathcal{T})$	$N^3(\mathcal{T})$	$N^4(\mathcal{T})$	$N^5(\mathcal{T})$
1-0	4	4	4	4	4	4
2-1	6	6	22	26	26	26
3-1	10	58	102	102	102	102
4-2	14	148	280	470	560	560
5-1	15	209	759	1136	1326	1399
6-4	18	397	948	1936	2844	3436
7-4	23	3236	7654	11961	15780	16968
8-3	24	1292	5870	15188	25914	34315
9-3	28	20891	39348	39348	39348	39348
10-3	28	6476	16180	65477	129400	224495
11-3	32	58268	130658	258977	399850	497030
12-4	34	83694	181416	541517	970632	1640974
13-2	40	461691	947674	2203931	3443154	4546823

Results

Miconic-Strips

task	$h^*(\mathcal{T})$	$N^1(\mathcal{T})$	$N^2(\mathcal{T})$	$N^3(\mathcal{T})$	$N^4(\mathcal{T})$	$N^5(\mathcal{T})$
1-0	4	4	4	4	4	4
2-1	7	18	29	34	37	37
3-1	11	70	138	195	241	251
4-4	15	166	507	814	1182	1348
5-4	18	341	1305	2708	4472	5933
6-4	21	509	2690	7086	13657	21177
7-4	25	3668	13918	32836	61852	95548
8-3	28	4532	35529	97529	205009	349491
9-3	32	25265	114840	321202	700640	1239599
10-3	34	8150	97043	423641	1151402	2505892

Dismal prospects

Depressing theoretical and experimental results

- Other (similar) search techniques cannot perform better than A^*.
- With other (real) heuristics it gets worse.

What is the cause of this behaviour?

Main problem

- many independently solvable subproblems
  which can be arbitrarily permuted
- many possible orders

Why is this not common knowledge?
$\longrightarrow$ does not happen in 15-Puzzle, Rubik's Cube, etc.

What do the results mean for us?

Some possible conclusions:

Conclusion?

We need heuristics that are better than almost perfect.
How feasible is this?

Conclusion?

We need more search enhancements.
Look to domain-dependent search for guidance?

What can the search community offer us?

Domain-specific search enhancements for Sokoban (Junghanns and Schaeffer, 2001):

- transposition table $\rightsquigarrow$ irrelevant to analysis
- move ordering $\rightsquigarrow$ irrelevant to analysis
- deadlock tables $\rightsquigarrow$ irrelevant to analysis
- tunnel macros $\rightsquigarrow$ generalizable?
- goal macros $\rightsquigarrow$ incomplete
- goal cuts $\rightsquigarrow$ incomplete
- pattern search $\rightsquigarrow$ heuristic improvement
- relevance cuts $\rightsquigarrow$ incomplete
- overestimation $\rightsquigarrow$ suboptimal
- rapid random restart $\rightsquigarrow$ irrelevant to analysis

$\rightsquigarrow$ Poster session: today, 6:00-9:30 PM

General search enhancements

Some techniques that might work in general:

- partial-order reduction
- symmetry elimination
- problem simplification

What do the results mean for us?

Some alternative conclusions:

Conclusion?

Heuristic search doesn't cut it.

What about more global reasoning methods, such as SAT planning, or symbolic exploration techniques like breadth-first search with BDDs?

Conclusion?

Optimal planning, beyond a certain point, is too hard.

We can hope to scale a bit better than blind search, but not very far. Maybe study near-optimal planning in a more principled way instead?

The end

Thank you for your attention!