Heuristic Analysis

Optimal Plans

I worked out following planning solutions

Problem #	Plan
1	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)
2	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Load(C3, P3, ATL) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)
3	Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C4, P2, SFO) Unload(C3, P1, JFK) Unload(C2, P2, SFO) Unload(C1, P1, JFK)

Jörg Jenni, The Old Curates House, Horton Street, Heckmondwike, WF16 0LL, United Kingdom
Tel: +44 (0) 7872 463615
Mail: jjenni@gmx.ch

Analysis

The performance of the following searches/heuristics were analysed:

Name	Search	Heuristic
breadth_first	Breadth First	None
depth_first_graph	Depth First Graph	None
uniform_cost_search	Uniform Cost Search	None
h_ignore_preconditions	A*-Search	Ignore-Preconditions
h_pg_levelsum	A*-Search	Levelsum

Following tables compare the performance of these searches once by calculating the "optimality" using "Nr Node Expansions" and once by calculating the optimality using "Time Elapsed".

The optimality is calculated by comparing each individual search with the hypothetical "best search". The best search combines the best result of each individual search into once hypothetical best search.

The Optimality is defined by

$$\frac{1}{N \cdot P^2}$$

whereas N is "Nr node expansion" or "Time Elapsed", and P is "Plan Length". P is to the power of 2 to give a small plan length a higher weight.

Search / Problem	Problem 1				Problem 2				Problem 3			
	Nr node expansions	Plan Length	Optimality [Relative]	**Optimality [%]**	Nr node expansions	Plan Length	Optimality [Relative]	**Optimality [%]**	Nr node expansions	Plan Length	Optimality [Relative]	**Optimality [%]**
breadth_first	43	6	0.0006	**25.6%**	3343	9	0.0000	**2.6%**	14663	12	0.0000	**2.2%**
depth_first_graph	21	20	0.0001	**4.7%**	624	619	0.0000	**0.0%**	408	392	0.0000	**0.1%**
uniform_cost_search	55	6	0.0005	**20.0%**	4853	9	0.0000	**1.8%**	18223	12	0.0000	**1.7%**
h_ignore_preconditions	41	6	0.0007	**26.8%**	1450	9	0.0000	**5.9%**	5040	12	0.0000	**6.3%**
h_pg_levelsum	11	6	0.0025	**100.0%**	86	9	0.0001	**100.0%**	316	12	0.0000	**100.0%**
Best Result	11	6	0.0025	100.0%	86	9	0.0001	100.0%	316	12	0.0000	100.0%

Figure 1: Optimality based on "nr nodes expansion" (ideal for memory constraint environments)

Jörg Jenni, The Old Curates House, Horton Street, Heckmondwike, WF16 0LL, United Kingdom
Tel: +44 (0) 7872 463615
Mail: jjenni@gmx.ch

Search / Problem	Problem 1				Problem 2				Problem 3			
	Time elapsed [s]	Plan Length	Optimality [Relative]	**Optimality [%]**	Time elapsed [s]	Plan Length	Optimality [Relative]	**Optimality [%]**	Time elapsed [s]	Plan Length	Optimality [Relative]	**Optimality [%]**
breadth_first	0.014	6	1.9273	**49.5%**	5.067	9	0.0024	**37.8%**	25.887	12	0.0003	**4.0%**
depth_first_graph	0.007	20	0.3503	**9.0%**	1.915	619	0.0000	**0.0%**	1.041	392	0.0000	**0.1%**
uniform_cost_search	0.017	6	1.6155	**41.5%**	7.139	9	0.0017	**26.8%**	32.591	12	0.0002	**3.2%**
h_ignore_preconditions	0.016	6	1.7082	**43.9%**	3.007	9	0.0041	**63.7%**	13.789	12	0.0005	**7.5%**
h_pg_levelsum	0.506	6	0.0549	**1.4%**	45.939	9	0.0003	**4.2%**	234.348	12	0.0000	**0.4%**
Best Result	0.007	6.000	3.892	100.0%	1.915	9	0.0064	100.0%	1.041	12	0.0067	100.0%

Figure 2: Optimality based on "time elapsed" (ideal for CPU constraint environments)

Discussion

The tables above show that depending on whether we care about "nr node expansions" or "time elapsed" different searches are optimal. A*-Search with Levelsum heuristics is optimal for a memory constraint environment as it expands a minimal number of nodes which saves memory. A*-Search with Ignore Preconditions heuristics is the optimal search for a CPU constraint environment as it requires minimal CPU time. Note: Though time elapsed is the wall time I expect the CPU time not to be hugely different. Measuring CPU-cycles instead wall time would further clarify that claim.

It also must be said that the Levelsum heuristic is a bad choice if we are not constraint by memory. Even though the Planning Graph makes us expand into a minimal number of nodes its construction costs a lot of CPU-time. I doubt that further code optimization would improve that significantly.

For easy problems (see Problem 1) non-heuristic searches, except Depth First Graph Search, is also a good option on CPU-constraint environments. But as complexity increases (see Problem 2 and 3) search with a heuristic is always better.

I conclude:

1. Non-heuristic searches are ideal for easy problems as they are fast.
2. Searches with heuristics are better when problems become harder.
3. A*-search with Levelsum heuristic is exceptionally good given we must minimize the number of expanded nodes.
4. A*-search with Ignore Preconditions heuristic is under normal conditions (no memory constraint) the best choice as it is fast and can solve complex problems efficiently.

Jörg Jenni, The Old Curates House, Horton Street, Heckmondwike, WF16 0LL, United Kingdom
Tel: +44 (0) 7872 463615
Mail: jjenni@gmx.ch