Mauro Restuccia May 15, 2017

Optimal Solution Search Analysis

Problem 1

Init(At(C1, SFO) ∧ At(C2, JFK)
∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO))

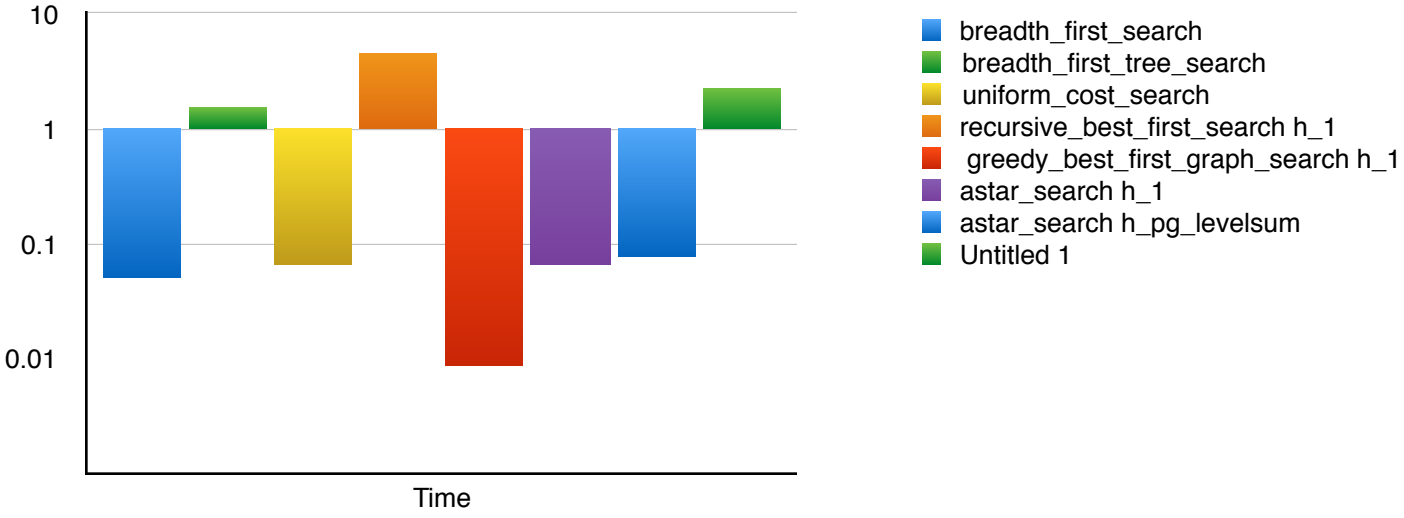
Goal(At(C1, JFK) ∧ At(C2, SFO))

Complexity

- 2 Planes
- 2 Airports
- 2 Cargo

(2P * 2A) + (2C * 2P) + (2C * 2A) = 12 ==> 2^{12}

Results



Problem 1

	Expansions	Goal Test	New Nodes	Length	Time
breadth_first_search	43	56	180	6	0.0502
breadth_first_tree_search	1458	1459	5960	6	1.5630
depth_first_graph_search	21	22	84	20	0.0233
depth_limited_search	101	271	414	50	0.1500
uniform_cost_search	55	57	224	6	0.0640
recursive_best_first_search h_1	4229	4230	17023	6	4.5225
greedy_best_first_graph_search h_1	7	9	28	6	0.0085
astar_search h_1	55	57	224	6	0.0646
astar_search h_ignore_preconditions	41	43	170	6	0.0753
astar_search h_pg_levelsum	11	13	50	6	2.2419

Conclusion

The optimal solution sequence is:

1. Load(C1, P1, SFO)
2. Load(C2, P2, JFK)
3. Fly(P2, JFK, SFO)
4. Unload(C2, P2, SFO)
5. Fly(P1, SFO, JFK)
6. Unload(C1, P1, JFK)

- The optimal search in this case was performed by “greedy_best_first_graph_search h_1”. It was the best search in time and space.
- When the search space is relative small it is possible to get results in a relative small amount of time.
- “Depth first search” and “Depth limited search” didn’t find the optimal solution.
- “Breath first search” and “recursive best first search h1” were the most intensive on Space.

Mauro Restuccia May 15, 2017

Problem 2

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))

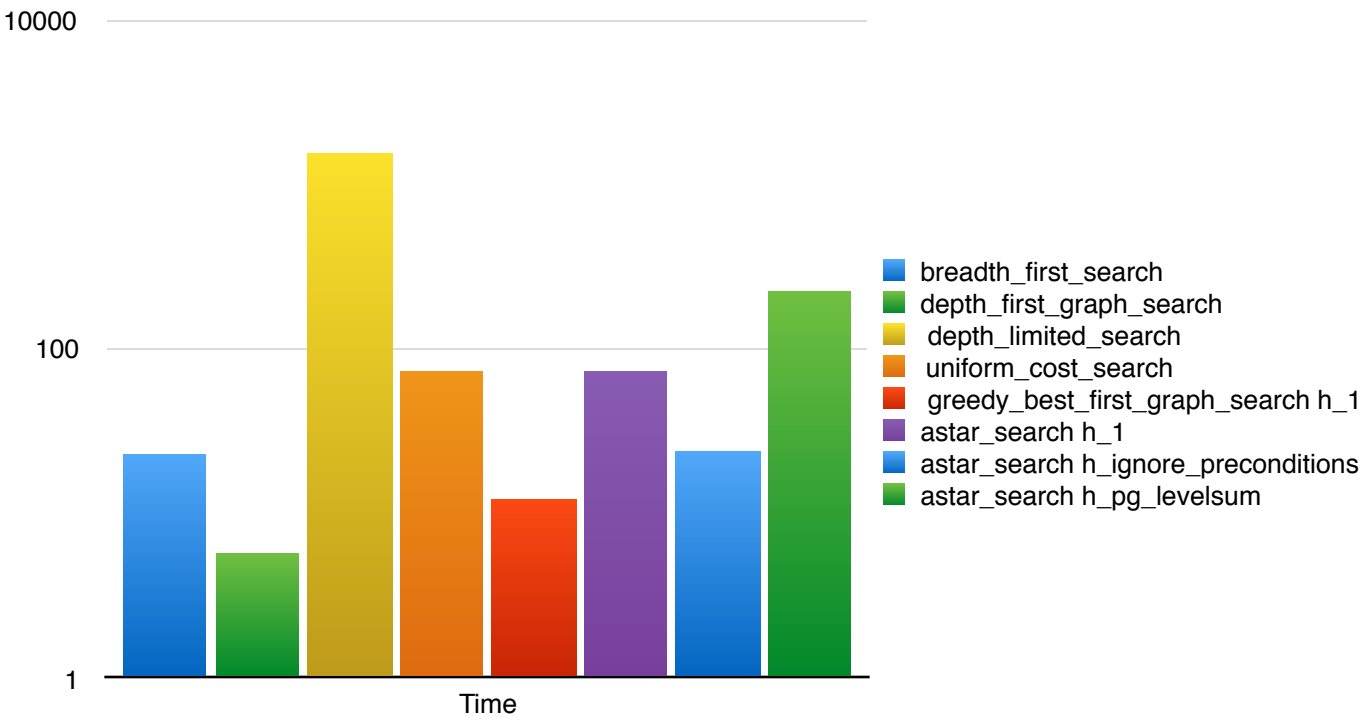
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))

Complexity

- 3 Planes
- 3 Airports
- 3 Cargo

(3P * 3A) + (3C * 3P) + (3C * 3A) = 27 ==> 2^{27}

Results



Problem 2

	Expansions	Goal Test	New Nodes	Length	Time
breadth_first_search	3343	4609	30509	9	22.5893
breadth_first_tree_search	-	-	-	-	-
depth_first_graph_search	624	625	5602	619	5.6684
depth_limited_search	222719	2053741	2054119	50	1544.7755
uniform_cost_search	4853	4855	44041	9	73.1868
recursive_best_first_search h_1	-	-	-	-	-
greedy_best_first_graph_search h_1	998	1000	8982	21	11.9762
astar_search h_1	4853	4855	44041	9	74.0186
astar_search h_ignore_preconditions	1506	1508	13820	9	23.5423
astar_search h_pg_levelsum	86	88	841	9	224.4064

Conclusion

The optimal solutions is:

1. Load(C1, P1, SFO)
2. Fly(P1, SFO, JFK)
3. Unload(C1, P1, JFK)
4. Load(C3, P3, ATL)
5. Fly(P3, ATL, SFO)
6. Unload(C3, P3, SFO)
7. Load(C2, P2, JFK)
8. Fly(P2, JFK, SFO)
9. Unload(C2, P2, SFO)

- “depth_first_graph_search” was the fastest search follow by “greedy_best_first_graph_search h_1” and “breadth_first_search” and later “astar_search h_ignore_preconditions”
- “depth_first_graph_search” and “greedy_best_first_graph_search h_1” didn’t find the optimal solution.
- “breadth_first_search” and “astar_search h_ignore_preconditions” took almost the same time to resolve the optimal solution.
- In conclusion, the optimal search was performed by “astar_search h_ignore_preconditions” because was among the fastest search that found the optimal solution using the less space.

Mauro Restuccia May 15, 2017

Problem 3

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))

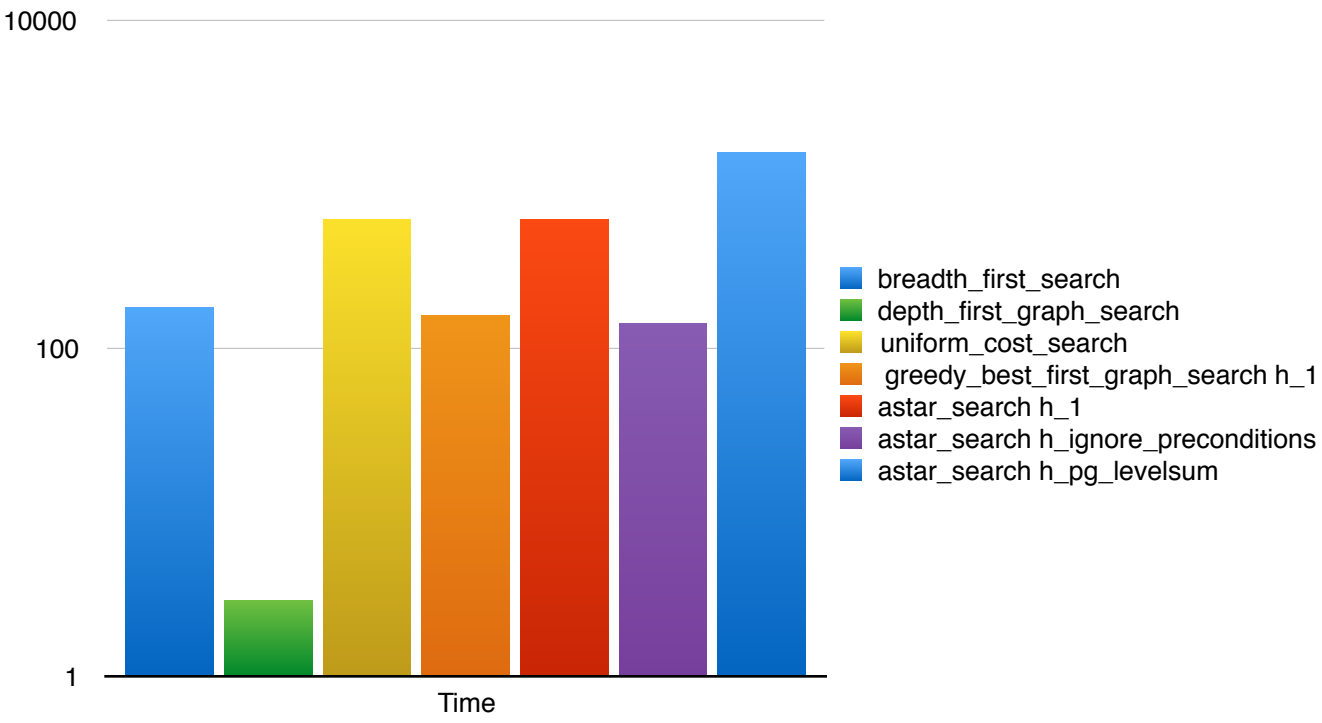
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))

Complexity

- 2 Planes
- 4 Airports
- 4 Cargo

(2P * 4A) + (4C * 2P) + (4C * 4A) = 27 ==> 2^{32}

Results



Mauro Restuccia May 15, 2017

Problem 3

	Expansions	Goal Test	New Nodes	Length	Time
breadth_first_search	14663	18098	129631	12	178.82775
breadth_first_tree_search	-	-	-	-	-
depth_first_graph_search	408	409	3364	392	2.91407
depth_limited_search	-	-	-	-	-
uniform_cost_search	18235	18237	159716	12	609.18520
recursive_best_first_search h_1	-	-	-	-	-
greedy_best_first_graph_search h_1	5614	5616	49429	22	160.04783
astar_search h_1	18235	18237	159716	12	617.16031
astar_search h_ignore_preconditions	5118	5120	45650	12	141.75068
astar_search h_pg_levelsum	414	416	3818	12	1541.22972

Conclusion

The optimal solutions is:

1. Load(C1, P1, SFO)
2. Fly(P1, SFO, ATL)
3. Load(C3, P1, ATL)
4. Fly(P1, ATL, JFK)
5. Unload(C1, P1, JFK)
6. Unload(C3, P1, JFK)
7. Load(C2, P2, JFK)
8. Fly(P2, JFK, ORD)
9. Load(C4, P2, ORD)
10. Fly(P2, ORD, SFO)
11. Unload(C2, P2, SFO)
12. Unload(C4, P2, SFO)

- “depth_first_graph_search”, “astar_search h_ignore_preconditions” and “greedy_best_first_graph_search h_1” where the most efficient in time.
- But only “astar_search h_ignore_preconditions” found the optimal solution.
- “astar_search h_pg_levelsum” found the solution too, but took too long.

Conclusion

- Depth first search does not seems to be adequate for this type of problem unless the complexity is low.
- Breath first search will take too long to find a solution to the goal.
- Uniform cost perform well, but expands node too much as continue searching for a more optimal solution when already have the optimal solution.
- Algorithms using heuristics seems to perform better.
- A* is the one performing better in more complex search spaces.
- A* ignore preconditions seem to be faster than all the other A* alternatives.
- Relaxing the problem is important to find the optimal solution.