Tony Chan
Udacity AI Nano-degree
Friday, 8 September 2017

Game Isolation heuristic analysis

Comparing effects of alpha-beta pruning towards winning chances

1. Optimal Plan for Problems 1, 2, and 3

	Problem 1	Problem 2	Problem 3
Best Path Length	6	9	12
Optimal Path	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)	Load(C3, P3, ATL) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Unload(C4, P2, SFO) Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C3, P1, JFK) Unload(C2, P2, SFO) Unload(C1, P1, JFK)

2. Non-heuristic search result

Generally depth-first search get the result quicker with non-optimal path given it prioritise depth than length of the path during search. Breadth-first guarantee shortest paths given it expands shortest path first, however given lots of combination, it operates slower. Greedy_best_first_graph_search works well with prioritising expanding the nodes which has highest hope to get the answer yet it cannot reach optimal path once problem get complicated **(Please read table in appendix)**

3. Heuristic search result

h_1 works the best in time for Problem 1 given the heuristic function is so simple so it can expand more node to find the answer than other 2 A* search within same time. However, once problems get complicated, h_ignore_preconditions & h_pg_levelsum works better in Problem 2 and 3. Within these 2 heuristic, h_ignore_preconditions wins in easier problem like Problem 2 for time given its heuristic is easy to compute (but of coz less accurate than h_pg_levelsum so performance drops once it becomes Problem 3)

(**Please read table in appendix**)

4. Heuristic vs Non-Heuristic

Heuristic not always win over non-heuristic. It only wins when the problem is complicated enough that expanding node is more computational expensive than calculating the heuristic. Which means we shouldn't overkill with too complicated heuristic if the path length is not too long or too complex.

		Problem 1	Problem 2	Problem 3
Breadth-first search	**Optimality**	Yes	Yes	Yes
	Time elapsed	0.0254	12.7275	90.4109
	Number of nodes expansion	43	3346	14120
Depth-first search	**Optimality**	No	No	No
	Time elapsed	0.0075	7.7857	3.4076
	Number of nodes expansion	12	1124	677
Greedy_best_first_graph_search	**Optimality**	Yes	No	No
	Time elapsed	0.0073	2.2620	15.2407
	Number of nodes expansion	7	998	5578
A* (h_1)	**Optimality**	Yes	Yes	Yes

		Problem 1	Problem 2	Problem 3
	Time elapsed	0.0341	10.8410	47.9334
	Number of nodes expansion	55	4853	18223
A* (h_ignore_preconditions)	**Optimality**	Yes	Yes	Yes
	Time elapsed	0.03589	3.9120	15.6846
	Number of nodes expansion	41	1450	5040
A* (h_pg_levelsum)	**Optimality**	Yes	Yes	Yes
	Time elapsed	0.0354	3.9554	15.2966
	Number of nodes expansion	41	1450	5040