Planing Searching Heuristic Analysis

By Yutao Liu

This project includes skeletons for the classes and functions needed to solve deterministic logistics planning problems for an Air Cargo transport system using a planning search agent. And with the increase of state and action variables , the complexity of the problem is exponential. In this analysis report , we first make an analysis to the complexity for the problem . Second we compare with different heuristics functions and find which is better.

Problem Analysis

As we can see, All problems are in the Air Cargo domain, They have the same action schema defined, but different initial states and goals.

- **Problem 1**

  It is very simple and has 12 state variables which may contain up to $2^{12} = 4096$ states and $2^3 * 3 = 24$ actions. We can use ordinary problem-solving agent.

  For this problem, the shortest number of actions is **6** and one of the optimal sequences of actions is:

  Load(C1, P1, SFO)
  Fly(P1, SFO, JFK)
  Load(C2, P2, JFK)
  Fly(P2, JFK, SFO)
  Unload(C1, P1, JFK)
  Unload(C2, P2, SFO)

- **Problem 2**

  It is a little complicated and has 27 state variables which may contain up to $2^{27} \approx 100,000,000$ states and $3^3 * 3 = 81$ actions. The problem is a little complicated but still can find a solution.

  For this problem, the shortest number of actions is 9 and one of the optimal sequences of actions is:

  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(C1, P1, JFK)

Unload(C3, P3, SFO)
Unload(C2, P2, SFO)

- **Problem 3**

It is like a real-world problem and very complicated. It has 32 state variables which may contain up to 2^{32} states and $4*2*4*3=96$ actions. Maybe we can find a solution but in the real world problems' solving, we hope to find the efficient and optimistic solution but not just a solution.

For this problem, the shortest number of actions is 12 and one of the optimal sequences of actions is:

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)

Part 1 - Planning problems

In this part, I just followed the guidance of AIMA book and implemented methods and functions in the air cargo problem. Then we just compared five different standard search algorithms.

Problem 1

	Search Algorithms	**Nodes Expansions**	**Goal Tests**	**New Nodes**	**Time elapsed in seconds**	**Number of actions**
1	UCS uniform_cost_search	55	57	224	0.047	6
2	BFS breadth_first_search	**43**	**56**	**180**	**0.034**	**6**
3	BFTS breadth first tree search	1458	1459	5960	1.088	6
4	DFGS depth_first_graph_search	21	22	84	0.021	20
5	DLS depth_limited_search	201	271	414	0.111	50

The problem is very easy, and taking into account the efficiency, the breadth first search is the best search algorithms. The algorithm search every possible state and find the best solution quickly, and number of actions to reach the goal is equal or less than others.

Problem 2

	Search Algorithms	Nodes Expansions	Goal Tests	New Nodes	Time elapsed in seconds	Number of actions
1	UCS uniform_cost_search	4281	4283	34717	11.773	9
2	BFS breadth_first_search	2847	4035	23488	12.938	9
3	BFTS breadth first tree search	-	-	-	-	-
4	DFGS depth_first_graph_search	329	330	1917	0.741	105
5	DLS depth_limited_search	-	-	-	-	-

As we can see in the result, BFTS and DLS algorithms runs too much time and still cant find a solution. BFS has a small number of actions but cost too much time. DFS has a large number of actions. But neither the algorithms can find a solution which has small number of actions and cost less time.

Problem 3

	Search Algorithms	Nodes Expansions	Goal Tests	New Nodes	Time elapsed in seconds	Number of actions
1	UCS uniform_cost_search	17783	17785	155920	60.303	12
2	BFS breadth_first_search	-	-	-	-	-
3	BFTS breadth first tree search	-	-	-	-	-
4	DFGS depth_first_graph_search	292	293	2388	1.434	288
5	DLS depth_limited_search	-	-	-	-	-

Because of CPU computing power constraints, only UCS and DFGS algorithms can find a solution. But DFGS will takes 288 actions. Compare with UCS, there is a better solution than the DFGS takes

Algorithms analysis

- **BFS(Breadth First Search)**

BFS can solve easy problem quickly but can't solve complex problem. Because the complexity of problem grows exponentially.

- **DFGS(Depth First Graph Search)**

DFGS always cost less time and find a solution but not the optimistic solution.

- **BFTS(Breadth First Tree Search)**

BFTS only find a solution in problem 1, This algorithms search almost every states which cost too much time.

- **DLS(Depth Limited Search)**

DLS also only find a solution in problem 1. This algorithms limits the depth of the search but cant know which solution is the best.

Part 2 - Domain-independent heuristics

In this part , I Implement a Planning Graph with automatic heuristics. Using this artificial intelligence function will help us to get a better result.

Problem 1

	Search Algorithms	Nodes Expansions	Goal Tests	New Nodes	Time elapsed in seconds	Number of actions
1	RBFS recursive_best_first_search h_1	4229	4230	17023	3.235	6
2	GBFGS greedy_best_first_graph_search h_1	7	9	28	0.006	6
3	AS h_1 astar_search h_1	55	57	224	0.061	6
4	AS h_i_p astar_search h_ignore_preconditions	41	43	170	0.061	6
5	AS h_pg_levelsum astar_search h_pg_levelsum	11	13	50	1.188	6

In this experiment, we can find out that using heuristic functions will find a better solution.

RBFS function with heuristic always being 1 will search all the states and cost too much time.
GBFS search the best solution and cost minimum time, because the problem is very simple and very easy to search the graph and find the best solution.
As we can see, astar search algorithm would take into account both precondition and heuristic for the forward level. So if we just consider preconditions or forward heuristic will search more state to get a solution.

Problem 2

	Search Algorithms	Nodes Expansions	Goal Tests	New Nodes	Time elapsed in seconds	Number of actions
1	RBFS recursive_best_first_search h_1	-	-	-	-	-
2	GBFGS greedy_best_first_graph_search h_1	486	488	3690	1.298	16
3	AS h_1 astar_search h_1	4281	4283	34717	11.757	9
4	AS h_i_p astar_search h_ignore_preconditions	1269	1271	10728	4.435	9
5	AS h_pg_levelsum astar_search h_pg_levelsum	169	171	1291	235.692	9

This problem is a little complicated but we know that the states of the problem is about 100,000,000 in the previous analysis, its very hard to search every states like RBFS to search every state and find a best one.
And we know that the problem is not very much complicated, using GBFS can easily find a local optimal solution but may not find a global optimal solution.
Compared with third and fourth heuristic function, we can find out the ignore the precondition will expand less states and cost less time to find a solution.
But we also find that AS h_pg_levelsum is not easy to compute and cost us too much time.

Problem 3

	Search Algorithms	Nodes Expansions	Goal Tests	New Nodes	Time elapsed in seconds	Number of actions
1	RBFS recursive_best_first_search h_1	-	-	-	-	-
2	GBFGS greedy_best_first_graph_search h_1	4031	4033	35794	12.763	22
3	AS h_1 astar_search h_1	17783	17785	155920	58.192	12
4	AS h_i_p astar_search h_ignore_preconditions	5003	5005	44586	18.625	12
5	AS h_pg_levelsum astar_search h_pg_levelsum	311	313	2863	1110	12

In this experiment, we can get all the previous conclusion but also AS h_pg_levelsum will search minimum expansions and get a global optimal solution. We can also find that AS h_pg_levelsum would spend a lot computing resources, but in real world problems solutions, a bad solution would cost much money and is not efficient.

Algorithms Analysis

- **RBFS**

Recursive functions will search every state to find a best solution, It will cost too much computing resources. It;s very hard to get a solution from RBFS in exponential problem.

- **GBFS**

It stops when the number of actions wouldn't decrease, may only be a local optimal solution but not the global optimal solution. In problem 2 and 3, there is a proof of this.

- **Astar Search**

There is 3 different types of heuristic functions.

AS h_1 is heuristic always being 1, and it expands more and more states to find a solution.

AS h_i_p ignore the precondition function expands less states and cost less time to find a solution.

AS h_pg_levelsum algorithm would take into account both precondition and heuristic for the forward level. It expands and tests less states but is not easy to compute and cost us too much time.

Conclusion

Compared with all the heuristic functions, I think using heuristic function would find a solution more quickly. Astar search with heuristic function of level sum is the better solution. Because problems are complicated in real world, although it would spend a lot computing resources, but a bad solution will cost much money and is not efficient.

References

[1] Russell S, Norvig P, Intelligence A. A modern approach[J]. Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, 1995, 25: 27.