Air Cargo Transport

This is an analysis of the performance of a forward planning agent in formulating a plan to move cargo between airports. The agent is given an initial start state with the locations of cargoes and planes at different airports. The agent is given a set of potential actions it can take, and the agent is given a goal state to reach. More background regarding forward planning agents and the air cargo transport problem can be found in the textbook **Artificial Intelligence: A Modern Approach (Russell/Norvig)** in chapter 10 of the 3rd edition (chapter 11 of the 2nd edition: http://aima.cs.berkeley.edu/2nd-ed/newchap11.pdf). The agent was tested with four problems of increasing complexity using 11 different search algorithms. Details are given below.

Problems:

1. Air Cargo Problem #1 (2 Cargoes, 2 Planes, 2 Airports, 2 Goals)

   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))

2. Air Cargo Problem #2 (3 Cargoes, 3 Planes, 3 Airports, 3 Goals)

   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))

3. Air Cargo Problem #3 (4 Cargoes, 2 Planes, 4 Airports, 4 Goals)

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

4. Air Cargo Problem #4 (5 Cargoes, 2 Planes, 4 Airports, 5 Goals)

   Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD) ∧ At(C5, ORD)
   ∧ At(P1, SFO) ∧ At(P2, JFK)
   ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4) ∧ Cargo(C5)
   ∧ Plane(P1) ∧ Plane(P2)
   ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))

   Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, JFK) ∧ At(C4, SFO) ∧ At(C5, JFK))

Potential Actions:

Action(Load(c, p, a),
   Precondition: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
   Effect: ¬ At(c, a) ∧ In(c, p))

Action(Unload(c, p, a),
   Precondition: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
   Effect: At(c, a) ∧ ¬ In(c, p))

Action(Fly(p, from, to),
   Precondition: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
   Effect: ¬ At(p, from) ∧ At(p, to))

Search Algorithms Tested:

1. Breadth First Search
2. Depth First Graph Search
3. Uniform Cost Search
4. Greedy Best First Graph Search With Unmet Goals Heuristic
5. Greedy Best First Graph Search With Level Sum Heuristic
6. Greedy Best First Graph Search With Max Level Heuristic
7. Greedy Best First Graph Search With Set Level Heuristic
8. A* Search With Unmet Goals Heuristic
9. A* Search With Level Sum Heuristic
10. A* Search With Max Level Heuristic
11. A* Search With Set Level Heuristic

Results

Problem One:

Search Algorithm	Actions	Expansions	Goal Tests	New Nodes	Path Length	Time (Seconds)
Breadth First	20	43	56	178	6	0.039
Depth First	20	21	22	84	20	0.013
Uniform Cost	20	60	62	240	6	0.038
Greedy Best First with Unmet Goals Heuristic	20	7	9	29	6	0.004
Greedy Best First with Level Sum Heuristic	20	6	8	28	6	0.435
Greedy Best First with Max Level Heuristic	20	6	8	24	6	0.194
Greedy Best First with Set Level Heuristic	20	6	8	28	6	1.009
A* Search With Unmet Goals Heuristic	20	50	52	206	6	0.026
A* Search With Level Sum Heuristic	20	28	30	122	6	0.436
A* Search With Max Level Heuristic	20	43	45	180	6	0.217
A* Search With Set Level Heuristic	20	33	35	138	6	0.607

Problem Two:

Search Algorithm	Actions	Expansions	Goal Tests	New Nodes	Path Length	Time (Seconds)
Breadth First	72	3343	4609	30503	9	0.596
Depth First	72	624	625	5602	619	0.801
Uniform Cost	72	5154	5156	46618	9	1.201
Greedy Best First with Unmet Goals Heuristic	72	17	19	170	9	0.044
Greedy Best First with Level Sum Heuristic	72	9	11	86	9	0.480
Greedy Best First with Max Level Heuristic	72	27	29	249	9	0.755
Greedy Best First with Set Level Heuristic	72	9	11	84	9	1.996
A* Search With Unmet Goals Heuristic	72	2467	2469	22522	9	1.177
A* Search With Level Sum Heuristic	72	357	359	3426	9	11.382
A* Search With Max Level Heuristic	72	2887	2889	26594	9	65.078
A* Search With Set Level Heuristic	72	1037	1039	9605	9	161.149

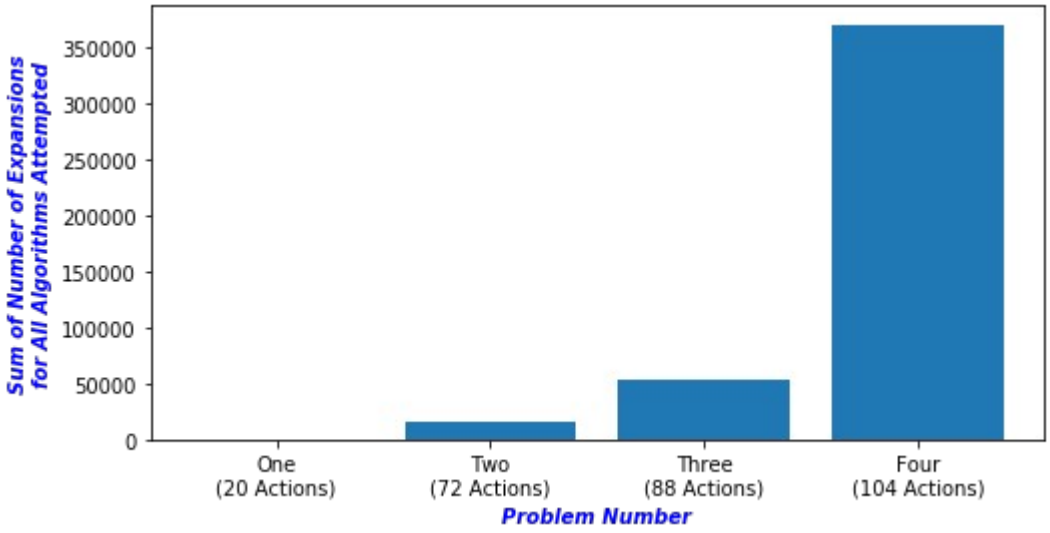
Problem Three:

Search Algorithm	Actions	Expansions	Goal Tests	New Nodes	Path Length	Time (Seconds)
Breadth First	88	14663	18098	129625	12	1.708
Depth First	88	408	409	3364	392	0.410
Uniform Cost	88	18510	18512	161936	12	2.842
Greedy Best First with Unmet Goals Heuristic	88	25	27	230	15	0.087
Greedy Best First with Level Sum Heuristic	88	14	16	126	14	2.172
Greedy Best First with Max Level Heuristic	88	21	23	195	13	1.097
Greedy Best First with Set Level Heuristic	88	35	37	345	17	11.383
A* Search With Unmet Goals Heuristic	88	7388	7390	65711	12	2.421
A* Search With Level Sum Heuristic	88	369	371	3403	12	21.490
A* Search With Max Level Heuristic	88	9580	9582	86312	12	355.126
A* Search With Set Level Heuristic	88	3423	3425	31596	12	846.396

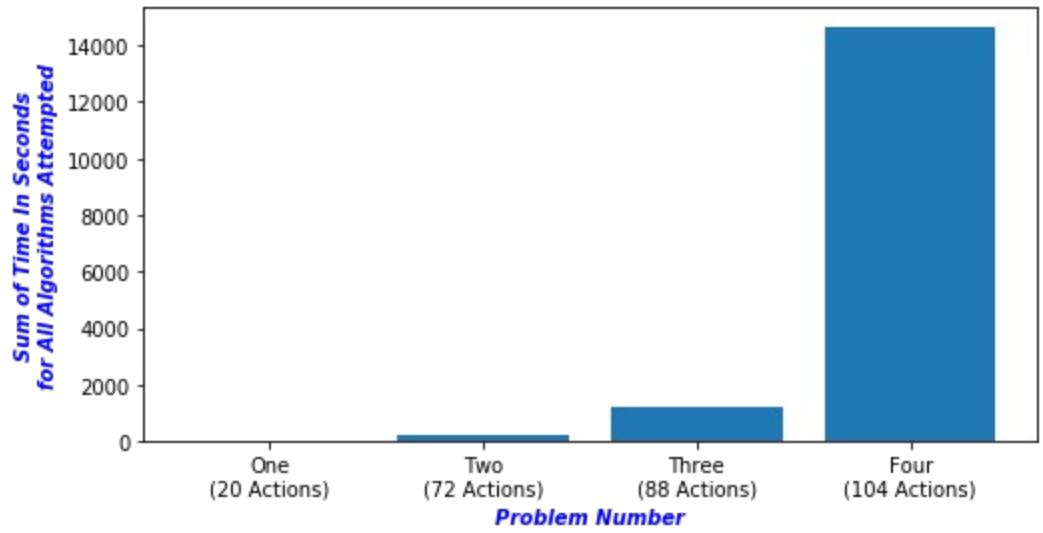
Problem Four:

Search Algorithm	Actions	Expansions	Goal Tests	New Nodes	Path Length	Time (Seconds)
Breadth First	104	99736	114953	944130	14	10.872
Depth First	104	35174	25175	228849	24132	1511.397
Uniform Cost	104	11339	113341	1066413	14	15.650
Greedy Best First with Unmet Goals Heuristic	104	29	31	280	18	0.099
Greedy Best First with Level Sum Heuristic	104	17	19	165	17	2.843
Greedy Best First with Max Level Heuristic	104	56	58	580	17	4.332
Greedy Best First with Set Level Heuristic	104	107	109	1164	23	45.807
A* Search With Unmet Goals Heuristic	104	34330	34332	328509	14	8.523
A* Search With Level Sum Heuristic	104	1208	1210	12210	15	106.505
A* Search With Max Level Heuristic	104	62077	62079	599376	14	1601.940
A* Search With Set Level Heuristic	104	22606	22608	224229	14	9314.945

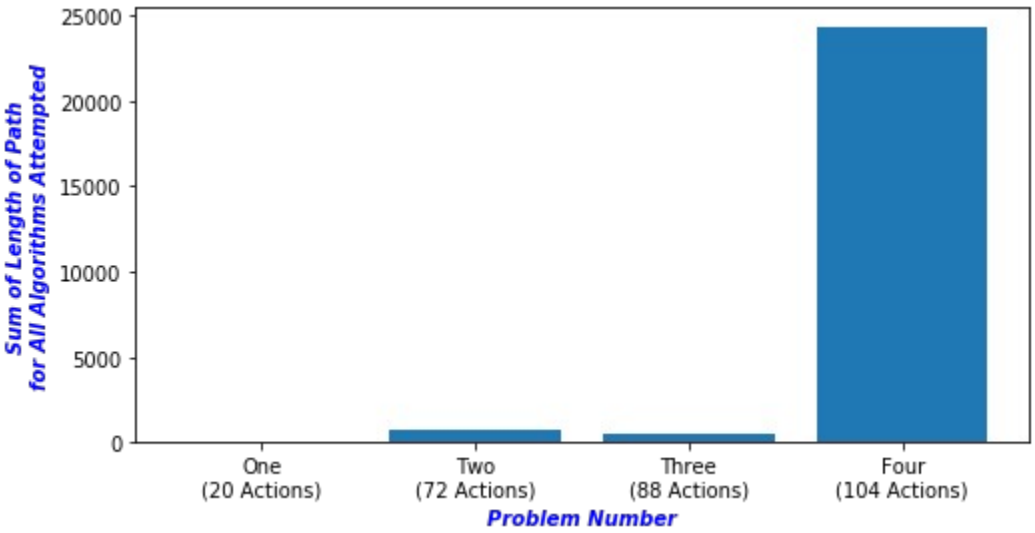
The results show that as the number of actions in each problems increased, the number of expansions increased considerably. This is illustrated in the chart below.



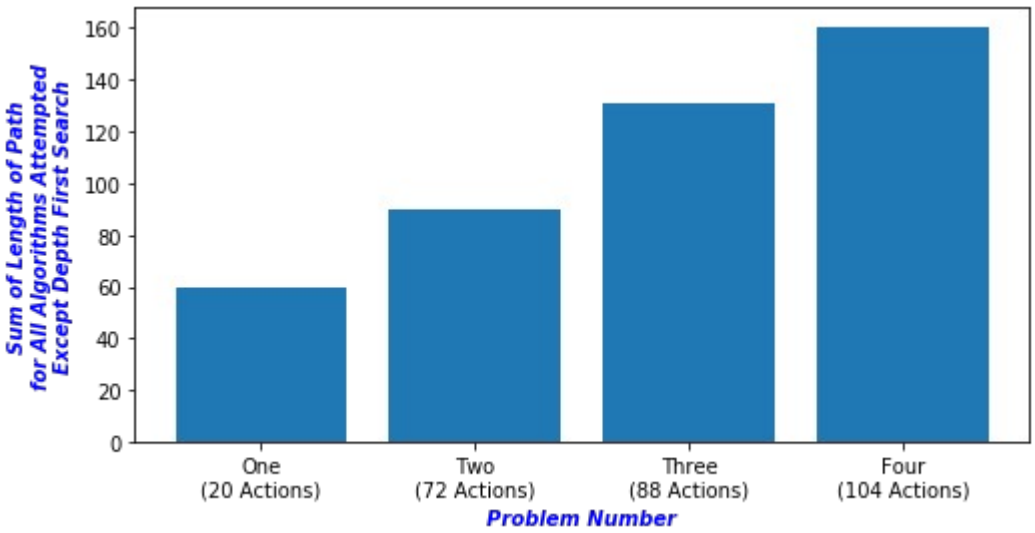
The results also show that the search time increases considerably as the number of actions increases.



An initial comparison of the length of path with respect to the number of actions in each problem shows similar results to those above.



But, it is obvious in looking at the tables that with respect to path length, the Depth First Search algorithm gives path lengths that markedly skew the results. When the Depth First Search results are removed, the path length show a more steady increase with respect to the number of actions in each problem.



Q&A

Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

If the the planning needs to be operated in real time, the agent must formulate a plan in a very short amount of time. In the first problem, which had the least amount of actions, almost all the algorithms gave an optimal path in less than a second, so it is possible that all of those algorithms would be acceptable depending on the specifics of the problem. The choice of algorithm would likely come down to a trade off between the speed of the algorithm vs. the optimal path always being found. Greedy Best First Search with Unmet Goals Heuristic was the fastest of the Algorithms tested, while Breadth First Search will always return the optimal path.

Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

As the size the domain increases, finding an optimal path becomes unlikely or impossible within reasonable time constraints. Of the algorithms tested, Greedy Best First Search with Unmet Goals Heuristic was the only one to return a path in under one second for the fourth problem with the largest domain. It would likely be the best choice for even larger domains.

Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

If the only consideration is finding an optimal plan, Breadth First Search would be the most appropriate algorithm. It is guaranteed to find an optimal plan.