Implementing a Planning

By Jorge Onieva

The following experiments were conducted as part of the project “Implementing a Planning ” in the Udacity Artificial Intelligence course, after implementing the required components.

Non-heuristic approaches

These are the results of the experiments run using a non-heuristic strategy:

Problem 1

Algorithm	Expansions	Goal tests	New nodes	Plan length	Time (seconds)
Breadth first	**43**	**56**	**180**	**6**	**0.0251925**
Breadth first tree	**1458**	**1459**	**5960**	**6**	**0.77994256**
depth first graph	21	22	84	20	0.01102053
depth limited	101	271	414	50	0.08473138
uniform cost	**55**	**57**	**224**	**6**	**0.03080901**
recursive best first with h1	**4429**	**4230**	**17023**	**6**	**2.19419049**
greedy best first graph with h1	**7**	**9**	**28**	**6**	**0.00468474**

Possible optimal solution:

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

Problem 2

Algorithm	Expansions	Goal tests	New nodes	Plan length	Time (seconds)
breadth first	**3343**	**4609**	**30509**	**9**	**7.11377812**
breadth first tree	-	-	-	-	> 15 mins.
depth first graph	624	625	5602	619	3.03387539
depth limited	222719	2053741	2054119	50	764.007937
uniform cost	**4852**	**4854**	**44030**	**9**	**10.2461465**
recursive best first with h1	-	-	-	-	> 15 mins.
greedy best first graph with h1	990	992	8910	21	2.12531059

Possible optimal solution:

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)

Problem 3

Algorithm	Expansions	Goal tests	New nodes	Plan length	Time (seconds)
breadth first	**14663**	**18098**	**129631**	**12**	**36.7739571**
breadth first tree	-	-	-	-	> 15 mins.
depth first graph	408	409	3364	392	1.55486933
depth limited	-	-	-	-	> 15 mins.
uniform cost	**18235**	**18237**	**159716**	**12**	**44.644808**
recursive best first with h1	-	-	-	-	> 15 mins.
greedy best first graph with h1	5614	5616	49429	22	16.359228

Possible optimal solution:

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)

Based on these results, we can conclude that the Depth First Graph approach is clearly much more efficient than the rest of the techniques in terms of the number of nodes analyzed and the time to reach a solution. Breadth First Tree (Problems 2 and 3) and Depth Limited (Problem 3) experiments were aborted after 15 minutes of execution with no solution. This is a clear sign of how a relatively simple problem may not be feasible to solve using these techniques.

However, it is also clear than the obtained solution in Depth First Graph is way less optimal than the other approaches in terms of the number of actions needed to get to the solution (plan length). Please note that the strategies that reached optimal solutions are displayed in bold.

Therefore, it seems reasonable to recommend Breadth First or Uniform Cost when we are looking for optimal solutions (in terms of the plan length), but Depth First Graph would probably be our best shot in terms of the time needed to find a solution. A compromise between the both approaches would be Greedy Best First Graph, which seems to offer a good balance between the plan length and the execution time.

Heuristic approaches

These are the results of the same problems, but this time using heuristic approaches:

Problem 1

Algorithm	Expansions	Goal tests	New Nodes	Plan length	Time (seconds)
A* with h1	55	57	224	6	0.0343885
A* with ignore preconditions	41	43	170	6	0.0293103
A* with levelsum	11	13	50	6	0.8233210

Problem 2

Algorithm	Expansions	Goal tests	New Nodes	Plan length	Time (seconds)
A* with h1	4825	2854	44030	9	10.153669
A* with ignore preconditions	1450	1452	13303	9	3.6046294
A* with levelsum	86	88	841	9	164.48902

Problem 3

Algorithm	Expansions	Goal tests	New Nodes	Plan length	Time (seconds)
A* with h1	18235	18237	159716	12	45.380569
A* with ignore preconditions	5040	5042	44944	12	14.672169
A* with levelsum	316	318	2912	12	837.29360

In general, heuristic searches seem to find optimal results in terms of the plan length (at least as good as the best we got with non-heuristic approaches). However, we observe a huge difference regarding the execution time and the number of nodes analyzed depending on the heuristic that we use.

In terms of number of nodes analyzed, Levelsum shows the best results, reducing dramatically the number of nodes required to reach an optimal solution. However, the price that is paid in terms of execution time compared to other approaches can make this solution unfeasible in many scenarios. The justification is that levelsum is a more complex heuristic that needs to perform many more operations in each step of the planning in order to detect inconsistencies [1].

However, we can see how Ignore Preconditions heuristic seems to work much better in A* search, reaching an optimal solution faster than the non-heuristic approaches that also reached an optimal solution to the problem. Therefore, in this problem, a very simple heuristic (also explained in [1]) reaches the same optimal solutions that a more complicated one, and at the same time outperforms non-heuristic approaches in the execution time and number of visited nodes.

Conclusions

Considering all the strategies analyzed, A* using an ignore preconditions heuristic seems to reach an optimal balance in terms of optimal solution – execution time, and it would be the recommended technique unless there is a strong need to minimize the number of visited nodes (for example in very reduced memory scenarios).

References

[1] Russell, S.J., Norvig, P., Canny, J.F., Malik, J.M. and Edwards, D.D., 2003. *Artificial intelligence: a modern approach* (Vol. 2, No. 11). Upper Saddle River: Prentice hall.