Dawn Coulter

Foundations of AI

Project 3

Implement a Planning Search

Due Date 2018-02-09

In this project I defined a group of problems in classical Planning Domain Definition Language for the air cargo domain discussed in the lectures. I set up the problems for search and experimented with various automatically generated heuristics. This is my analysis of the results.

**Given three problems in the Air Cargo Action Schema:**  
Action(Load(c, p, a),  
 PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)  
 EFFECT: ¬ At(c, a) ∧ In(c, p))  
Action(Unload(c, p, a),  
 PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)  
 EFFECT: At(c, a) ∧ ¬ In(c, p))  
Action(Fly(p, from, to),  
 PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)  
 EFFECT: ¬ At(p, from) ∧ At(p, to))

**- Problem 1 initial state and goal:**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))

**- Problem 2 initial state and goal:**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))

**- Problem 3 initial state and goal:**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))

**Uninformed Search Strategies Analysis:**

I chose to try 4 different strategies for each of the three given problems. See the results below.

**For P1**  
python run\_search.py -p 1 -s 1  
python run\_search.py -p 1 -s 3  
python run\_search.py -p 1 -s 5  
python run\_search.py -p 1 -s 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Search Strategy | Node Expansions | Goal Tests | New Nodes | Time Elapsed | Optimal |
| breadth first search | 43 | 56 | 180 | 0.0369 | Yes |
| depth first search | 12 | 13 | 48 | 0.009 | no |
| uniform cost search | 55 | 57 | 224 | 0.044 | yes |
| greedy best first | 7 | 9 | 28 | 0.006 | yes |

**For P2**python run\_search.py -p 2 -s 1  
python run\_search.py -p 2 -s 3  
python run\_search.py -p 2 -s 5  
python run\_search.py -p 2 -s 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Search Strategy | Node Expansions | Goal Tests | New Nodes | Time Elapsed | Optimal |
| breadth first search | 3401 | 4672 | 31049 | 9.802 | yes |
| depth first search | 350 | 351 | 3142 | 1.766 | no |
| uniform cost search | 4761 | 4763 | 43206 | 13.254 | yes |
| greedy best first | 550 | 552 | 4950 | 1.523 | yes |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **For P3** |  |  |  |  |

python run\_search.py -p 3 -s 1  
python run\_search.py -p 3 -s 3  
python run\_search.py -p 3 -s 5  
python run\_search.py -p 3 -s 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Search Strategy | Node Expansions | Goal Tests | New Nodes | Time Elapsed | Optimal |
| breadth first search | 14491 | 17947 | 128184 | 48.966 | yes |
| depth first search | 1948 | 1949 | 16253 | 22.734 | no |
| uniform cost search | 17783 | 17785 | 155920 | 56.652 | yes |
| greedy best first | 4031 | 4033 | 35794 | 12.947 | yes |

**Heuristic Search Strategies Analysis:  
For P1**python run\_search.py -p 1 -s 8  
python run\_search.py -p 1 -s 9  
python run\_search.py -p 1 -s 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Search Strategy | Node Expansions | Goal Tests | New Nodes | Time Elapsed | Optimal |
| a\* search h\_1 | 55 | 57 | 224 | 0.05 | yes |
| a\* search h\_ignore\_preconditions | 41 | 43 | 170 | 0.05 | yes |
| a\* searchh\_pg\_levelsum | 55 | 57 | 224 | 3.98 | no |

**For P2**python run\_search.py -p 2 -s 8  
python run\_search.py -p 2 -s 9  
python run\_search.py -p 2 -s 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Search Strategy | Node Expansions | Goal Tests | New Nodes | Time Elapsed | Optimal |
| a\* search h\_1 | 4761 | 4763 | 43206 | 13.24 | yes |
| a\* search h\_ignore\_preconditions | 1450 | 1452 | 13303 | 6.382 | yes |
| a\* searchh\_pg\_levelsum | -- | -- | -- | > 10 minutes | no |

**For P3**python run\_search.py -p 3 -s 8  
python run\_search.py -p 3 -s 9  
python run\_search.py -p 3 -s 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Search Strategy | Node Expansions | Goal Tests | New Nodes | Time Elapsed | Optimal |
| a\* search h\_1 | 17783 | 17785 | 155920 | 57.329 | yes |
| a\* search h\_ignore\_preconditions | 5003 | 5005 | 44586 | 23.854 | yes |
| a\* searchh\_pg\_levelsum | -- | -- | -- | > 10 minutes | no |

**Analysis**

For uninformed searches it appears that breadth first search works best for smaller data sets, however when you look at larger data sets as in the P3 problem it shifts to looking like greedy best first is faster and uses less memory. Just looking at the tables it would appear that depth first search looks to perform the best for small and larger data sets, however I learned from the lectures specifically in Lesson 10 Search the lecture that stated this was 21. Search Comparison 1, that depth first is not an optimal search strategy so I excluded it from my choice of optimal search strategies.

In considering the heuristic search strategies, I would exclude a\* search\_pg\_levelsum as for P2 and P3 it failed to meet the specified criteria of running in less than ten minutes. I am thinking this is most likely because I still have more to learn and my implementation was subpar. With that exclusion you can see from the tables provided that a\* search h\_ignore\_preconditions performed better than a\* search h\_1. For my implementation of these heuristic searches then I would definitely choose a\* search h\_ignore\_preconditions as the optimal search strategy.

If you were to compare between the two types of searches, uninformed vs heuristic, you can see that breadth first still performs the best for smaller data sets and greedy best first is the best for larger data sets. I would have expected a heuristic search to perform better than an uninformed search strategy, so I again am left to wonder and speculate that it may be a lack of my implementation and could be perfected with more study and time.