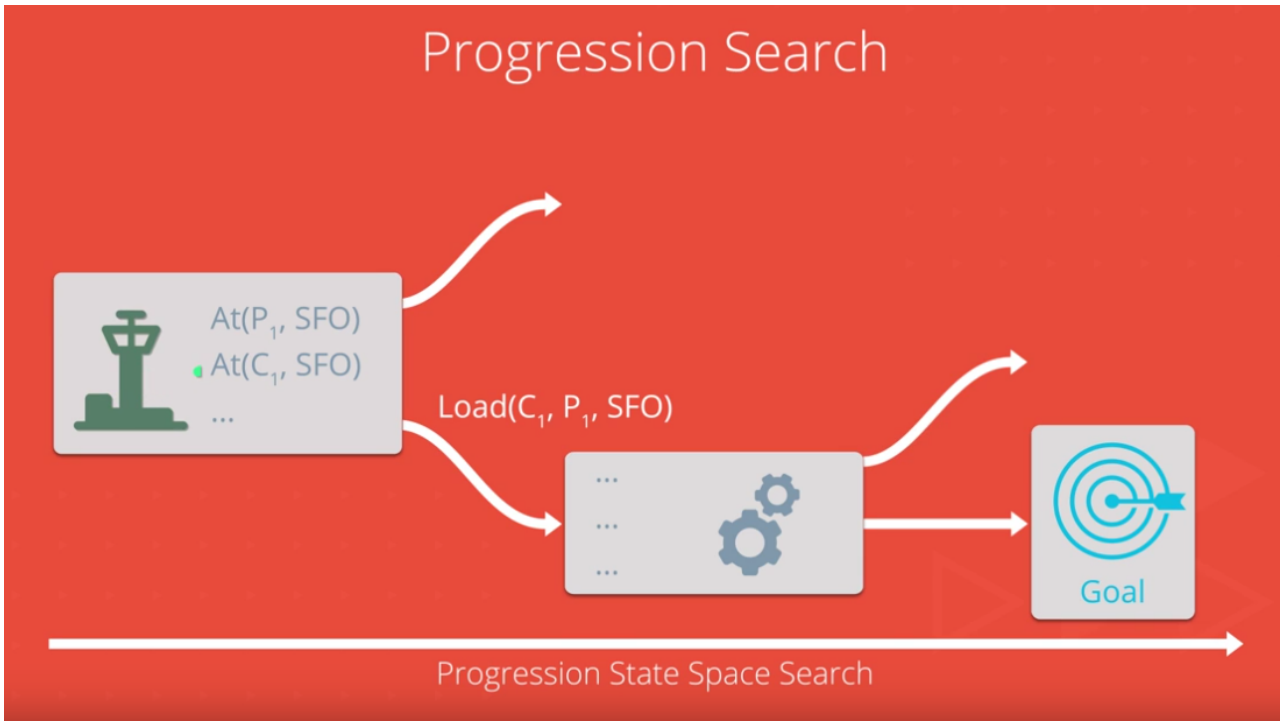
Heuristic Analysis

Air Cargo Planning

Dave Fang / dave.fang@outlook.com



Overview

After finishing the code, I tried to use different heuristic function aka. search algorithms. The problem we need to settle as follow.

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

```
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

```
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

```
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))
```

The homework provided by course contains 10 different search algorithms in `run_search.py`, and following are the results collected from the script.

Problem #1

ALGORITHM	TIME(S)	EXPANSIONS	TESTS	NODES	LENGTH	OPTIMAL
BFS	0.03	43	56	180	6	Yes
BFTS	1.04	1458	1459	5960	6	Yes
DFGS	0.01	21	22	84	20	No
DLS	0.10	101	271	414	50	No
UCS	0.04	55	57	224	6	Yes
RBFS	3.03	4229	4230	17023	6	Yes
GBFGS	0.008	7	9	28	6	Yes
A* (H1)	0.04	55	57	224	6	Yes
A*(H PRE-)	0.04	41	43	170	6	Yes
A*(H PG)	1.14	11	13	50	6	Yes

To sum up, taking time consuming and final result into account, I do believe that GBFGS (Greedy Best First Graph) is the best algorithm under this conditions. And the optimal solution is:

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

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

Problem #2

ALGORITHM	TIME(S)	EXPANSIONS	TESTS	NODES	LENGTH	OPTIMAL
BFS	15.44	3343	4609	30509	9	Yes
BFTS	-	-	-	-	-	-
DFGS	3.57	624	625	5602	619	No
DLS	-	-	-	-	-	-
UCS	13.08	4853	4855	44041	9	Yes
RBFS	-	-	-	-	-	-
GBFGS	2.48	998	1000	8982	15	No
A* (H1)	12.79	4852	4854	44030	9	Yes
A*(H PRE-)	4.45	1450	1452	13303	9	Yes
A*(H PG)	195.28	86	66	841	9	Yes

Considering both time consuming and performance, we can say that the A* (h ignore preconditions) is the best solution, and the optimal result is as follow.

```
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	TIME(S)	EXPANSIONS	TESTS	NODES	LENGTH	OPTIMAL
BFS	115.01	14491	17947	128184	12	Yes
BFTS	-	-	-	-	-	-
DFGS	1.78	408	409	3364	392	No
DLS	-	-	-	-	-	-
UCS	52.21	18223	18225	15916	12	Yes
RBFS	-	-	-	-	-	-
GBFGS	16.33	5579	5581	49159	22	No
A* (H1)	54.80	18224	18225	159618	12	Yes
A*(H PRE-)	18.19	5040	5042	44944	12	Yes
A*(H PG)	-	-	-	-	-	-

As we can see, the performance of A* (h ignore preconditions) is the best, and the time consuming is also ok. The optimal solution is as follow:

```
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)
```

Compare & Conclusion

Non Heuristic Search

As we can see, DFGS (Depth First Graph Search) consumes the minimum time and memory, but the problem is that it could produce the optimal solution.

Besides, DLS (Depth Limit Search) has the worst performance and could not finish the task in limited time, because it just simply expands nodes as deep as it can.

BFTS (Breadth First Tree Search) is an algorithm that can solve easy problem but complex one, it will take a long time to find best solution when facing complex problem.

UCS (Uniform Cost Search) is a bit like BFS, but an optimized version of BFS, which expands the lowest cost branch. UCS could find the optimal solution, but it depends on the complexity of problem. When facing a complex problem, it will try to expand lots of branches which will consume more time and memory.

To sum up, BFS and UCS can perform well facing easy question, but if problem becomes complicated, their performances are unsatisfied.

Heuristic Search

From the source course provided, there are three different A* search functions aka heuristic functions.

As we can see from the charts, we can know A* with PG Level function uses less memory due to expanding less nodes, but it takes more time than others, and it will consume numerous time when facing complex problem. And both A* H1 and A* H Ignore Preconditions can produce the optimal plan.

But we can only have one winner, and the winner is A* H Ignore Preconditions function, which uses less memory and consumes less time. It beats almost other functions, and always can give the optimal plan.

After reviewing the code of heuristics functions, it’s obvious that A* Ignore Preconditions function is the fastest, because it uses a cache mechanism. To the other hand, the A* Level Sum function does tons of computation, which is the reason why it performs so bad.

Reference

[1] Stuart Russell, Peter Norvig, "Artificial Intelligence: A Modern Approach" 3rd edition chapter 10 or 2nd edition Chapter 11

[2] https://en.wikipedia.org/wiki/Search_algorithm