Planning Search

by Miguel Ángel Martínez Fernández

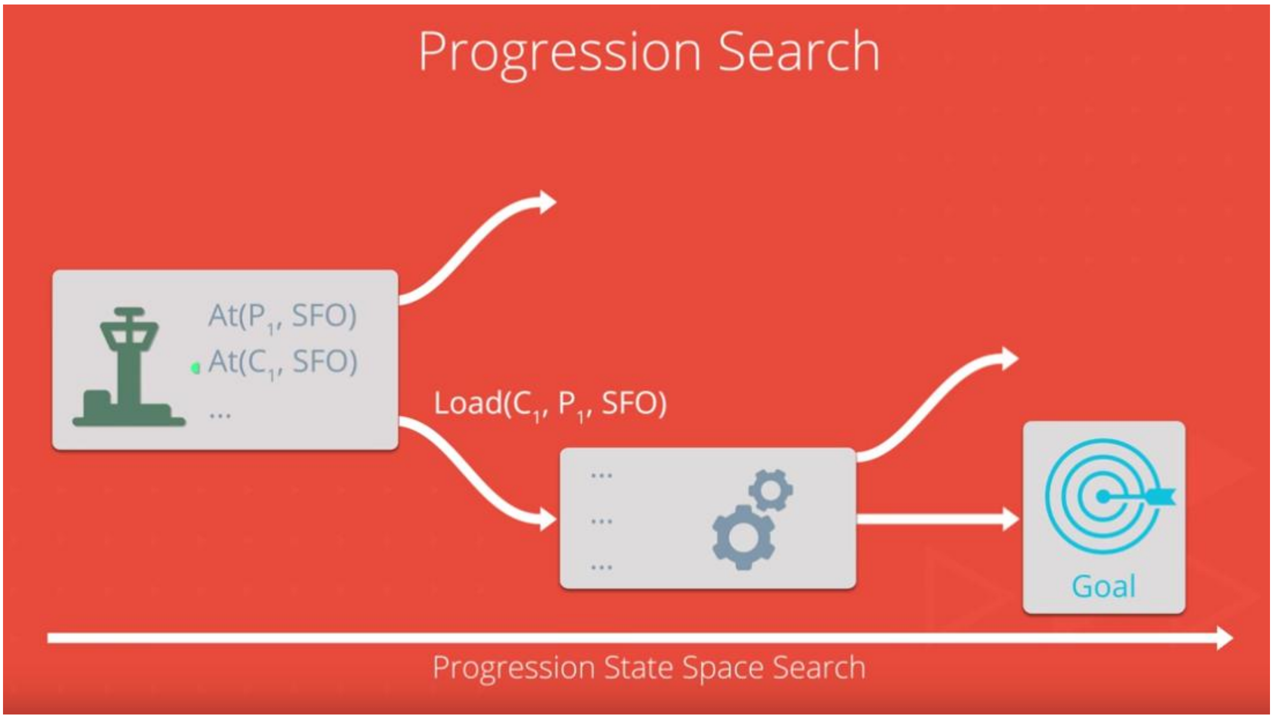
www.github.com/miguelangel

The project

This project will cover a group of problems in classical PDDL (Planning Domain Definition Language) for the air cargo domain discussed in the lectures.

Different tasks will be performed in this project:

- set up the problems for search,
- experiment with various automatically generated heuristics to solve the problems
- provide an analysis of the results



The challenge

A set of problems will be encoded and resolved by the student partial implementation (some code is reuse of companion code from the Stuart Russel/Norvig AIMA book).

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))
```

Air Cargo 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))
```

Air Cargo 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))
```

Air Cargo Problem #3:

```
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))
```

The Results

A few hundreds of hours later, the moment of truth is upon us. The solution is coded and it is time to test it.

Three problems have been tested against different algorithms, aborting the execution if no result is found after 25 minutes.

A few executions later...

Air Cargo Problem #1 Results

	Air Cargo Problem #1				
	Expan sions	Goal Tests	New Nodes	Time elapsed (sec)	Plan length
1. Breadth first search	43	56	180	0.063270470	6
2. Breadth first tree search	1458	1459	5960	0.761046078	6
3. Depth first graph search	21	22	84	0.032917761	20
4. Depth limited search	101	271	414	0.148747783	50
5. Uniform cost search	55	57	224	0.067270886	6
6. Recursive best first search with h 1	4229	230	17023	1.902893294	6
7. Greedy best first graph search with h 1	7	9	28	0.014840714	6
8. A* search with h 1	55	57	224	0.070309599	6
9. A* search with h ignore preconditions	41	43	170	0.086801050	6
10. A* search with h pg levelsum	11	13	50	0.541952483	6

Optimal plan:

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

Air Cargo Problem #2 Results

	Air Cargo Problem #2				
	Expansions	Goal Tests	New Nodes	Time elapsed (sec)	Plan length
1. Breadth first search	3343	4609	30509	3.112878162	9
2. Breadth first tree search	Execution interrupted after 25 min.				
3. Depth first graph search	624	625	5602	0.627360680	619
4. Depth limited search	222719	2053741	2054119	639.611617878	50
5. Uniform cost search	4852	4854	44030	3.693827908	9
6. Recursive best first search with h 1	Execution interrupted after 25 min.				
7. Greedy best first graph search with h 1	990	992	8910	0.923688966	15
8. A* search with h 1	4852	4854	44030	3.617807142	9
9. A* search with h ignore preconditions	1450	1452	13303	2.359241932	9
10. A* search with h pg levelsum	86	88	841	7.715594813	9

Optimal plan:

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)

Air Cargo Problem #3 Results

	Air Cargo Problem #2				
	Expansions	Goal Tests	New Nodes	Time elapsed (sec)	Plan length
1. Breadth first search	14663	18098	129631	14.801872248	12
2. Breadth first tree search	Execution interrupted after 25 min.				
3. Depth first graph search	408	409	3364	0.427939078	392
4. Depth limited search	Execution interrupted after 25 min.				
5. Uniform cost search	18235	18237	159716	12.945869346	12
6. Recursive best first search with h 1	Execution interrupted after 25 min.				
7. Greedy best first graph search with h 1	5614	5616	49429	4.814608156	22
8. A* search with h 1	18235	18237	159716	13.271433661	12
9. A* search with h ignore preconditions	5040	5042	44944	7.346515556	12
10. A* search with h pg levelsum	315	317	2902	34.059256274	12

Optimal plan:

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)

The Conclusion

Testing indicates that the following algorithms reach the optimal solution:

- Breadth first search
- Uniform cost search
- A* search with h 1
- A* search with h ignore preconditions
- A* search with h pg levelsum

Before discussing the algorithms performance, let's groups the algorithms in **uniform search strategies** (1-7) and **heuristic search strategies** (8-10).

Uniform search strategies:

Uniform search methods have access only to the problem definition. That behaviour differs from heuristic search strategies, which uses a heuristic function to estimate the cost of the solution.

- *Depth first graph search* is the fastest and less memory consuming algorithm **but** it does not reach to an optimal path length.
- *Breadth first search* would be the winner if finding the optimal path length is mandatory. BFS is complete and optimal algorithm.

AIMA (section 3.4.7, Figure 3.21) contains a table comparing different uniform search strategies with different criteria (completeness, time, space, optimality). Our results performs as described in that table.

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes^{a}	$\text{Yes}^{a,b}$	No	No	Yes^{a}	$\text{Yes}^{a,d}$
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes^{c}	Yes	No	No	Yes^{c}	$\text{Yes}^{c,d}$

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: ^{a} complete if b is finite; ^{b} complete if step costs $\geq \epsilon$ for positive ϵ; ^{c} optimal if step costs are all identical; ^{d} if both directions use breadth-first search.

Heuristic search strategies

Heuristic search strategies uses specific-domain knowledge to improve its performance by estimating the cost of the solution. AIMA book explains how *the performance of heuristic search algorithms depends on the quality of the heuristic function*.

Based on the results, it can be concluded that:

- *A* search with h ignore preconditions* is the fastest algorithm if memory is not a restriction.
- *A* search with h pg levelsum* is the less memory consuming algorithm if time is not a restriction.

Both solutions are complete.

I sincerely hope you enjoyed this reading as much as I did writing it for you.

Miguel Ángel