Written Analysis **Pedro Felipe Gardeazábal Rodríguez**
Project: Implement Planning Search
Course: AIND - Udacity Date: 02/10/2018

Implement Planning Search

Problems to solve

The cargo problems to solve and their action schema are:

- 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))
```

Optimal solutions

After running the ten proposed search types for problems 1, 2 and 3 (where the complexity and number of goals increases as the problem number increases) the optimal solution, stated as the minimal number of steps needed to achieve the goals, for each one of the problems was obtained only by the same five algorithms in each case. These algorithms are breadth_first_search, uniform_cost_search, $A^\star$ with h1, $A^\star$ with h_ignore_preconditions and $A^\star$ with h_pg_levelsum. The optimal planning solutions are listed on table 1 below (the ordering of the solution steps depending on the algorithm, the order presented here corresponds to $A^\star$ with h_pg_levelsum's solutions).

Problem	Problem 1 (air_cargo_p1)	Problem 2 (air_cargo_p2)	Problem 3 (air_cargo_p3)
Solution Steps	• Load(C1, P1, SFO) • Load(C2, P2, JFK) • Fly(P1, SFO, JFK) • Fly(P2, JFK, SFO) • Unload(C1, P1, JFK) • Unload(C2, P2, SFO)	• Load(C1, P1, SFO) • Load(C2, P2, JFK) • Load(C3, P3, ATL) • Fly(P1, SFO, JFK) • Fly(P2, JFK, SFO) • Fly(P3, ATL, SFO) • Unload(C1, P1, JFK) • Unload(C2, P2, SFO) • Unload(C3, P3, SFO)	• Load(C1, P1, SFO) • Load(C2, P2, JFK) • Fly(P1, SFO, ATL) • Load(C3, P1, ATL) • Fly(P2, JFK, ORD) • Load(C4, P2, ORD) • Fly(P2, ORD, SFO) • Fly(P1, ATL, JFK) • Unload(C1, P1, JFK) • Unload(C2, P2, SFO) • Unload(C3, P1, JFK) • Unload(C4, P2, SFO)

Table 1: Optimal planning solutions for problems 1, 2 and 3 from $A^\star$ with h_pg_levelsum.

Uninformed planning searches

Of the proposed uninformed searches to solve problems air_cargo_p1, air_cargo_p2 and air_cargo_p3, the three searches chosen to compare their metrics and giving a solution are breadth_first_search, depth_first_graph_search and uniform_cost_search (see table 2).

From table 2 it can be seen that the optimal solution for the three problems was obtained only by breadth_first_search and uniform_cost_search, but they took longer than depth_first_graph_search to arrive to a solution. They also realized more expansions than depth_first_graph_search. All this is consistent with the fact that depth_first_graph_search avoids repeated states but unfortunately outputs the first solution it encounters [1].

On the other hand, uniform_cost_search being similar to breadth_first_search, except by the fact that the former examines all the nodes at the goal's depth while still searching for a lower cost [2], their metrics are along the same order of magnitude but slightly higher for uniform_cost_search. These differences become more evident as the problem becomes more complex.

Domain-independent heuristics

Of the domain-independent heuristic searches available to solve problems air_cargo_p1, air_cargo_p2 and air_cargo_p3, the resulting metrics for searches $A^\star$ with h_ignore_preconditions and $A^\star$ with h_pg_levelsum are presented here (see table 3).

Problem	Search Type	Metrics				
		Expansions	Goal Tests	New Nodes	Plan Length	Time (s)
air_cargo_p1	breadth_first_search	43	56	180	6	0.021
	depth_first_graph_search	21	22	84	20	0.009
	uniform_cost_search	55	57	224	6	0.026
air_cargo_p2	breadth_first_search	3343	4609	30509	9	5.618
	depth_first_graph_search	624	625	5602	619	2.363
	uniform_cost_search	4853	4855	44041	9	8.047
air_cargo_p3	breadth_first_search	14663	18098	129631	12	35.315
	depth_first_graph_search	408	409	3364	392	1.212
	uniform_cost_search	18151	18153	159038	12	42.028

Table 2: Metrics for three uninformed search methods.

$A^\star$ with h_ignore_preconditions, assuming that each goal can be carried out in one step, takes as its cost the number of goals that haven't been achieved at the evaluated node [3]. On the other hand, $A^\star$ with h_pg_levelsum, assuming as well each goal as independent, defines its cost as the sum of the number of levels necessary to reach each goal from the evaluated node [4].

Even if both methods output the same solution, $A^\star$ with h_ignore_preconditions is always faster and expands less nodes than $A^\star$ with h_pg_levelsum. This seems reasonable from the definition of the heuristics were we can see that $A^\star$ with h_ignore_preconditions doesn't need to look forward to calculate a cost while $A^\star$ with h_pg_levelsum needs at each node to evaluate further nodes until it finds an achieved goal, and this for all the goals!

As the table shows for the three problems, even if $A^\star$ with h_pg_levelsum is more complete than $A^\star$ with h_ignore_preconditions [4], it carries an overload on number of evaluations and then on timing. This effect becomes very time consuming as the complexity or number of goals in a problem increases (compare both searches in table 3).

Problem	$A^\star$ Type Search	Metrics				
		Expansions	Goal Tests	New Nodes	Plan Length	Time (s)
air_cargo_p1	h_ignore_preconditions	41	43	170	6	0.025
	h_pg_levelsum	55	57	224	6	0.637
air_cargo_p2	h_ignore_preconditions	1450	1452	13303	9	2.964
	h_pg_levelsum	4853	4855	44041	9	358.462
air_cargo_p3	h_ignore_preconditions	5038	5040	44926	12	11.307
	h_pg_levelsum	18151	18153	159038	12	2091.064

Table 3: Metrics for two $A^\star$ search methods.

Conclusion

Given the time necessary to find a solution and the number of nodes expanded the best heuristic to solve problems 1, 2 and 3 is $A^\star$ with h_ignore_preconditions.

For simple problems like problem 1, in terms of time and node expansions the performance of $A^\star$ with h_ignore_preconditions is similar to an uninformed search giving an optimal solution. As the problem complexity slightly increases, like problem 2, $A^\star$ with h_ignore_preconditions starts to show a real advantage in time and appears to become a method of choice when the problem complexity is high and an optimal solution is required.

The method depth_first_graph_search outperforms significantly all the methods presented here in time and number of expansions but unfortunately loses its appeal when looking for an optimal solution given that outputs the first solution it encounters.

References

[1] Stuart J. Rusell, Peter Norvig. Section 3.4.3. *Artificial Intelligence a Modern Approach*. 3^rd Edition. Pearson Education, 2015. 86-87. Print.

[2] Section 3.4.2. *Ibid.*, 84-85.

[3] Section 10.2.3. *Ibid.*, 382.

[4] Section 10.3.1. *Ibid.*, 388.