# Planning Search Analysis

Bhinesh Patel 20 May 2017

## Planning Problems

We were given three classical PDDL problems in the Air Cargo domain. They have the same action schema defined, but different initial states and goals as shown below.

| 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)) |

## Search Strategies

The Planning problems were solved using 10 strategies as shown in the table below. These were separated in two groups; uninformed (blind) search and informed (heuristic) search. Uninformed strategies have no additional information about states beyond that provided in the problem definition. All they can do is generate successors and test them if the goal state has been reached. Informed strategies know whether one state is more promising than another so that the search path can be directed appropriately.

| Index | Search Strategy | Search Type |
|---|---|---|
| 1 | breadth_first_search | Uninformed |
| 2 | breadth_first_tree_search | Uninformed |
| 3 | depth_first_graph_search | Uninformed |
| 4 | depth_limited_search | Uninformed |
| 5 | uniform_cost_search | Uninformed |
| 6 | recursive_best_first_search with h_1 | Hueristic |
| 7 | greedy_best_first_graph_search with h_1 | Hueristic |
| 8 | astar_search with h_1 | Hueristic |
| 9 | astar_search with h_ignore_preconditions | Hueristic |
| 10 | astar_search with h_pg_levelsum | Hueristic |

## Optimal Plans

The optimal plan for each problem along with the length were obtained as shown below.

| Optimal Plans and Length | | |
|---|---|---|
| **Problem 1** | **Problem 2** | **Problem 3** |
| **Length 6** | **Length 9** | **Length 12** |
| Load(C1, P1, SFO) | Load(C1, P1, SFO) | Load(C1, P1, SFO) |
| Load(C2, P2, JFK) | Load(C2, P2, JFK) | Load(C2, P2, JFK) |
| Fly(P1, SFO, JFK) | Load(C3, P3, ATL) | Fly(P1, SFO, ATL) |
| Fly(P2, JFK, SFO) | Fly(P1, SFO, JFK) | Load(C3, P1, ATL) |
| Unload(C1, P1, JFK) | Fly(P2, JFK, SFO) | Fly(P2, JFK, ORD) |
| Unload(C2, P2, SFO) | Fly(P3, ATL, SFO) | Load(C4, P2, ORD) |
| | Unload(C1, P1, JFK) | Fly(P2, ORD, SFO) |
| | Unload(C2, P2, SFO) | Fly(P1, ATL, JFK) |
| | Unload(C3, P3, SFO) | Unload(C1, P1, JFK) |
| | | Unload(C2, P2, SFO) |
| | | Unload(C3, P1, JFK) |
| | | Unload(C4, P2, SFO) |
| | | |

## Uninformed Search Strategies Comparison

In this section the results of solving each of the three problems by the five **uninformed** search strategies are discussed.

### Problem 1

| Search Strategy | Plan Length | Expansions | Goal Test | New Nodes | Time(ms) |
|---|---|---|---|---|---|
| breadth_first_search | 6 | 43 | 56 | 180 | 39 |
| breadth_first_tree_search | 6 | 1458 | 1459 | 5960 | 1,157 |
| depth_first_graph_search | 20 | 21 | 22 | 84 | 17 |
| depth_limited_search | 50 | 101 | 271 | 414 | 113 |
| uniform_cost_search | 6 | 55 | 57 | 224 | 45 |

### Problem 2

| Search Strategy | Plan Length | Expansions | Goal Test | New Nodes | Time(s) |
|---|---|---|---|---|---|
| breadth_first_search | 9 | 3343 | 4609 | 30509 | 16 |
| breadth_first_tree_search | - | - | - | - | - |
| depth_first_graph_search | 619 | 624 | 625 | 5602 | 4 |
| depth_limited_search | 50 | 222719 | 2053741 | 2054119 | 1,791 |
| uniform_cost_search | 9 | 4780 | 4782 | 43381 | 15 |

### Problem 3

| Search Strategy | Plan Length | Expansions | Goal Test | New Nodes | Time(s) |
|---|---|---|---|---|---|
| breadth_first_search | 12 | 14663 | 18098 | 129631 | 117 |
| breadth_first_tree_search | - | - | - | - | - |
| depth_first_graph_search | 392 | 408 | 409 | 3364 | 2 |
| depth_limited_search | - | - | - | - | - |
| uniform_cost_search | 12 | 17882 | 17884 | 156769 | 62 |

In all three problems, the quickest to produce a solution was ***depth_first_graph*** search, it also had the least nodes expanded thus consuming the least memory but the plan was far from optimal. ***breadth_first*** and ***uniform_cost*** search were close in performance and number of nodes expanded. Both produced optimal length plans. Producing an optimal plan is more important for practical considerations hence ***uniform_cost*** search is recommended if one needs to use an uninformed search. However if speed and memory usage is a priority then ***depth_first_graph*** search is recommended at the cost of a sub optimal plan length.

In Problem 2 and Problem 3, some of the search strategies failed to come back with a solution even after running for several hours. These have no data in the table.

## Informed Search Strategies Comparison

In this section the results of solving each of the three problems by the five **informed** search strategies are discussed.

### Problem 1

| Search Strategy | Plan Length | Expansions | Goal Test | New Nodes | Time(ms) |
|---|---|---|---|---|---|
| recursive_best_first_search with h_1 | 6 | 4229 | 4230 | 17023 | 3,426 |
| greedy_best_first_graph_search with h_1 | 6 | 7 | 9 | 28 | 6 |
| astar_search with h_1 | 6 | 55 | 57 | 224 | 56 |
| astar_search with h_ignore_preconditions | 6 | 41 | 43 | 170 | 52 |
| astar_search with h_pg_levelsum | 6 | 11 | 13 | 50 | 1,238 |

### Problem 2

| Search Strategy | Plan Length | Expansions | Goal Test | New Nodes | Time(s) |
|---|---|---|---|---|---|
| recursive_best_first_search with h_1 | - | - | - | - | - |
| greedy_best_first_graph_search with h_1 | 17 | 598 | 600 | 5382 | 2 |
| astar_search with h_1 | 9 | 4780 | 4782 | 43381 | 15 |
| astar_search with h_ignore_preconditions | 9 | 1450 | 1452 | 13303 | 6 |
| astar_search with h_pg_levelsum | 9 | 86 | 88 | 841 | 247 |

### Problem 3

| Search Strategy | Plan Length | Expansions | Goal Test | New Nodes | Time(ms) |
|---|---|---|---|---|---|
| recursive_best_first_search with h_1 | - | - | - | - | - |
| greedy_best_first_graph_search with h_1 | 26 | 4498 | 4500 | 39970 | 16 |
| astar_search with h_1 | 12 | 17882 | 17884 | 156769 | 86 |
| astar_search with h_ignore_preconditions | 12 | 5034 | 5036 | 44886 | 26 |
| astar_search with h_pg_levelsum | 12 | 314 | 316 | 2894 | 1,598 |

In Problem 1, ***greedy_best_first_graph_search_with_h_1*** heuristic was quickest and had the least expansions thus using the least memory. It also produced an optimal plan. However in Problems 2 and 3, while still the quickest; it failed to find an optimal solution. ***astar_search_with_ h_ignore_preconditions*** was the second quickest and found the optimal plan in all three problems. It also had a modest number of expansions hence memory consumption. It would be the informed search strategy that is recommended. The least expansions were by ***astar_search_with_h_pg_levelsum*** but at the cost of a very time consuming search. It also produced an optimal plan in all three problems.

***recursive_best_first_search_with_h_1*** did not find a solution in Problems 2 and 3 after running for several hours at which point the search was halted.

## Uniformed vs Informed Comparison

| Problem | Search Strategy | Plan Length | Expansions | Goal Test | New Nodes | Time(s) |
|---|---|---|---|---|---|---|
| 1 | uniform_cost_search | 6 | 55 | 57 | 224 | 0.045 |
| | astar_search with h_ignore_preconditions | 6 | 41 | 43 | 170 | 0.052 |
| 2 | uniform_cost_search | 9 | 4780 | 4782 | 43381 | 15 |
| | astar_search with h_ignore_preconditions | 9 | 1450 | 1452 | 13303 | 6 |
| 3 | uniform_cost_search | 12 | 17882 | 17884 | 156769 | 62 |
| | astar_search with h_ignore_preconditions | 12 | 5034 | 5036 | 44886 | 26 |

***uniform_cost*** search was the recommended uninformed search strategy and ***astr_search_with_h_ignore_precondtions*** was the recommended informed search strategy. Comparing the two, ***astr_search_with_h_ignore_precondtions*** was quicker, led to fewer expansions thus memory usage in all three problems, hence it is the recommended search strategy for the problem given.

In Chapter 3.4.2 of AIMA ($3^{rd}$ edition Russell & Norvig) the complexity of ***uniform_cost*** search is of the order of $O(b^{d+1})$ and we see that the number of **Expansions** increase greatly with the complexity of the problems. **astar_search_with_h_ignore_preconditions** has similar exponential expansion but the targeted search can lead to fewer expansions.

In Chapter 10.3 it is shown that Planning Graphs reduce the exponential complexity of the search to polynomial size. This can be seen in the low number of expansions by ***astar_search_with_h_pg_levelsum*** search.