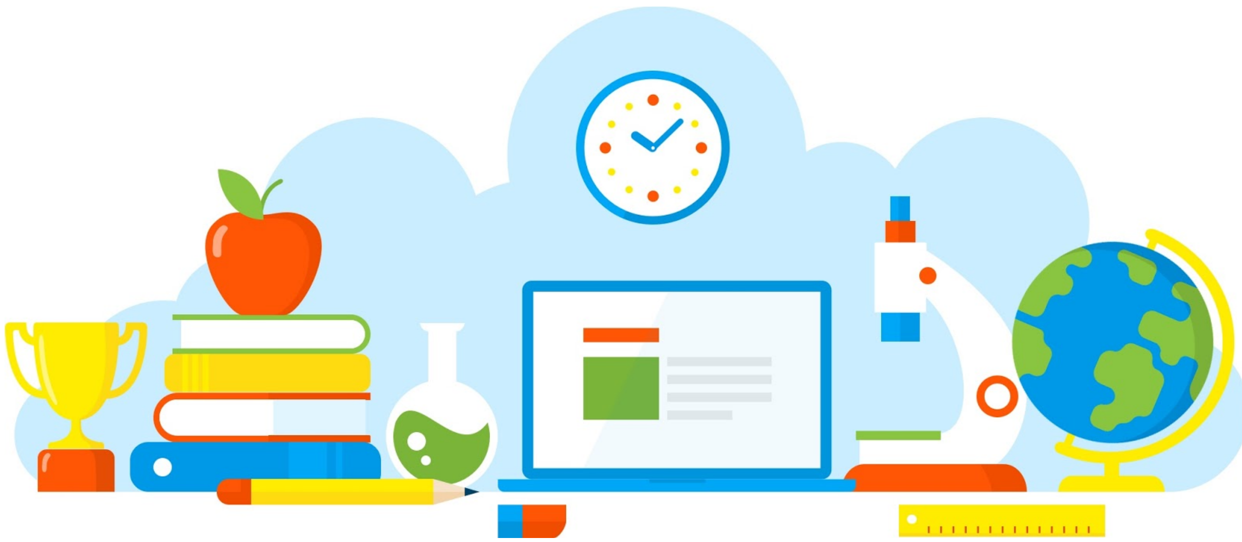

# Planning Search Heuristic Analysis

*By Mohamed Bakr ALi*

# Introduction:

For this project, we implemented a planning search agent to solve deterministic logistics planning problems for an Air Cargo transport system. We use a planning graph and automatic domain-independent heuristics with A* search and compare their results/performance against several uninformed non-heuristic search methods (breadth-first, depth-first, etc.).

# Planning problems:

- 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(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:

All they can do is generate successors and distinguish a goal state from a non-goal state. In this section, we compare the performance of seven such strategies in terms of **speed** (execution time, measured in seconds), **memory usage** (measured in search node expansions) and **optimality** (Yes, if a solution of optimal length is found; No, otherwise). The number of goal tests and number of new nodes are not reported in the tables below since they do not change the results of our analysis below.

- **Problem 1 result**:

| Search Strategy | Optimal | Path Length | Execution Time (s) | Node Expansions |
|---|---|---|---|---|
| Breadth First Search | Yes | 6 | 0.0464 | 43 |
| Breadth First Tree Search | Yes | 6 | 0.7367 | 1458 |
| Depth First Graph Search | No | 12 | 0.0066 | 12 |
| Depth Limited Search | No | 50 | 0.0705 | 101 |
| Uniform Cost Search | Yes | 6 | 0.0288 | 55 |
| Recursive Best First Search | Yes | 6 | 2.2037 | 4229 |
| Greedy Best First Graph Search | Yes | 6 | 0.00417 | 7 |

- **Problem 2 result:**

| Search Strategy | Optimal | Path Length | Execution Time (s) | Node Expansions |
|---|---|---|---|---|
| Breadth First Search | Yes | 9 | 7.2539 | 3401 |
| Breadth First Tree Search | --- | --- | --- | --- |
| Depth First Graph Search | No | 346 | 1.2379 | 350 |
| Depth Limited Search | --- | --- | --- | --- |
| Uniform Cost Search | Yes | 9 | 9.7369 | 4761 |
| Recursive Best First Search | --- | --- | --- | --- |
| Greedy Best First Graph Search | Yes | 9 | 1.01222 | 550 |

- **Problem 3 result:**

| Search Strategy | Optimal | Path Length | Execution Time (s) | Node Expansions |
|---|---|---|---|---|
| Breadth First Search | Yes | 12 | 33.384 | 14491 |
| Breadth First Tree Search | --- | --- | --- | --- |
| Depth First Graph Search | No | 1878 | 17.4739 | 1948 |
| Depth Limited Search | --- | --- | --- | --- |
| Uniform Cost Search | Yes | 12 | 42.8779 | 17783 |
| Recursive Best First Search | --- | --- | --- | --- |
| Greedy Best First Graph Search | No | 22 | 9.5244 | 4031 |

### ***Analysis*:**

With this 3-problem set, **Breadth First Search** and **Uniform Cost Search** are the only two uninformed search strategies that **yield an optimal action plan** under the 10mn time limit. When it comes to execution speed and memory usage, **Depth First Graph Search** is the **fastest and uses the least memory**. However, it does not generate an optimal action plan (problem 1: plan length of 12 instead of

6, problem 2: plan length of 346 instead of 9, problem 3: plan length of 1878 instead of 12).

If *finding the optimal path length is critical*, what strategy should we use? Because it performs **faster and uses less memory** than Uniform Cost Search, **Breadth First Search** is the recommended search strategy. This isn't much of a surprise, as BFS is complete and optimal. Its only downside is memory usage, if the problem's branching factor is high, as shown in [1] section 3.4.7:

| Criterion | Breadth-First | Uniform-Cost | Depth-First | Depth-Limited | Iterative Deepening | Bidirectional (if applicable) |
|---|---|---|---|---|---|---|
| Complete? | Yes[a] | Yes[a,b] | No | No | Yes[a] | 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] | 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 $\epsilon$; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

Which search strategy should we use, if *having an optimal path length is not the primary criteria*? For problems 2 and 3, the Depth First Graph Search plan lengths are so much longer than the optimal path length that it wouldn't make sense to use this search strategy. **Greedy Best First Graph Search is the best alternative**. In problems 1 and 2, it manages to find the optimal path. In problem 3, it does not find the optimal path but the path length it generates is 22 instead of 10, which is much better than Depth First Graph Search (1878 path length!). Moreover, it still provides execution time savings and uses less memory than the best search strategy for an optimal solution (Breadth First Search).

***

# informed Search Strategies Analysis:

informed search strategy — one that uses problem-specific knowledge beyond the definition of the problem itself — can find solutions more efficiently than can an uninformed strategy. In this section, we compare the performance of **A* Search using three different heuristics**. Here again, we evaluate these strategies in terms of **speed**, **memory usage** and **optimality**.

- **Problem 1 result**:

| Search Strategy | Optimal | Path Length | Execution Time (s) | Node Expansions |
|---|---|---|---|---|
| A* Search with h1 heuristic | Yes | 6 | 0.0657 | 55 |
| A* Search with Ignore Preconditions heuristic | Yes | 6 | 0.0495 | 41 |
| A* Search with Level Sum heuristic | Yes | 6 | 0.9787 | 11 |

- **Problem 2 result**:

| Search Strategy | Optimal | Path Length | Execution Time (s) | Node Expansions |
|---|---|---|---|---|
| A* Search with h1 heuristic | Yes | 9 | 10.361 | 4761 |
| A* Search with Ignore Preconditions heuristic | Yes | 9 | 4.9108 | 1450 |
| A* Search with Level Sum heuristic | Yes | 9 | 206.562 | 86 |

**Problem 3 result**:

| Search Strategy | Optimal | Path Length | Execution Time (s) | Node Expansions |
|---|---|---|---|---|
| A* Search with h1 heuristic | Yes | 12 | 46.780 | 17783 |
| A* Search with Ignore Preconditions heuristic | Yes | 12 | 20.504 | 5003 |
| A* Search with Level Sum heuristic | --- | --- | --- | --- |

***Analysis***:

While all heuristics yield an optimal action plan, only the h1 and Ignore Preconditions heuristics return results within the 10mn max execution time set by the Udacity staff. Which heuristic should we use? Of the two strategies mentioned above, **A* Search with Ignore Preconditions heuristic is the fastest**. If we let search run to completion on our machine, **A* Search with Level Sum heuristic uses the least memory**, but its execution time is much slower (26 mn for problem 2!).

## informed Search vs Uninformed Search :

The search strategies that generate optimal plans are Breadth First Search, Uniform Cost Search, and A* Search with all three heuristics.

As we saw earlier, when it comes to execution speed and memory usage of uninformed search strategies, **Depth First Graph Search** is faster and uses less memory than Uniform Cost Search. As for informed search strategies, **A* Search with Ignore Preconditions heuristic** is the fastest and uses the least memory. So, really, the choice is between Depth First Graph Search and A* Search with Ignore Preconditions heuristic. Here we compare their results against our 3-problem set.

From the results above, because it is faster and uses less memory, **A* Search with Ignore Preconditions heuristic** would be the best choice overall for our Air Cargo problem.