## Introduction

In this project we are given three planning schemes with different initial states and goals as given below, and expected to generate plans to achieve given goals using uninformed and heuristic search strategies.

*Table 1. Problem descriptions*

|  |  |
| --- | --- |
| **Problem** | **Initial State and Goal** |
| **AC1** | **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)) |
| **AC2** | **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)) |
| **AC3** | **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)) |

## Results

Using the provided "run\_search.py" script, we evaluated the performance of different uninformed and heuristic search strategies to achieve the given goal for each problem. Following parameters are investigated to performance evaluation

*Table 2. Parameters for performance evaluation*

|  |  |  |
| --- | --- | --- |
| **Abbr.** | **Parameter** | **Performance criteria for** |
| Exp. | Node Expansions | Memory |
| GT. | Goal Tests | Memory |
| NN. | New Nodes | Memory |
| PL. | Plan length | Optimality |
| Time | Execution time (s) | Speed |

We run the search strategies for

Three uninformed search strategies:

1. Breath first search
2. Depth first graph search
3. Uniform cost search

Two heuristic search strategies:

1. A\* search – Ignore preconditions
2. A\* search – Level sum

Table 3 displays the performance of all runs and Table 4 lists the plans generated by corresponding search strategies. Note that for Air Cargo 2 and 3 depth first graph search algorithm finds very large plans, hard to list in the table. Hence, they are omitted.

*Table 3. Performance results of search strategies*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Uninformed Strategies** | | | **Heuristic Strategies** | |
| Problem | **Breadth first search** | **Depth first graph search** | **Uniform cost search** | **Ignore preconditions** | **Level sum** |
| AC1 | Exp. 43  GT. 56  NN. 180  PL. 6  Time 0.05 | Exp. 12  GT. 13  NN. 48  PL. 12  Time 0.01 | Exp. 55  GT. 57  NN. 224  PL. 6  Time 0.05 | Exp. 41  GT. 43  NN. 170  PL. 6  Time 0.05 | Exp. 11  GT. 13  NN. 50  PL. 6  Time 1.05 |
| AC2 | Exp. 3343  GT. 4609  NN. 30509  PL. 9  Time 10.6 | Exp. 582  GT. 583  NN. 5211  PL. 575  Time 4.3 | Exp. 4852  GT. 4854  NN. 44030  PL. 9  Time 15.4 | Exp. 1450  GT. 1452  NN. 13303  PL. 9  Time 5.6 | Exp. 86  GT. 88  NN. 841  PL. 9  Time 89.2 |
| AC3 | Exp. 14663  GT. 18098  NN. 129631  PL. 12  Time 52.8 | Exp. 627  GT. 628  NN. 5176  PL. 596  Time 4.6 | Exp. 18234  GT. 18236  NN. 159707  PL. 12  Time 63.8 | Exp. 5040  GT. 5042  NN. 44944  PL. 12  Time 21.4 | Exp. 318  GT. 320  NN. 2934  PL. 12  Time 448.2 |

*Table 4. Plans*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Uninformed Strategies** | | | **Heuristic Strategies** | |
| **Problem** | **Breadth first search** | **Depth first graph search** | **Uniform cost search** | **Ignore preconditions** | **Level sum** |
| **AC1** | Load(C2, P2, JFK)  Load(C1, P1, SFO)  Fly(P2, JFK, SFO)  Unload(C2, P2, SFO)  Fly(P1, SFO, JFK)  Unload(C1, P1, JFK) | Fly(P1, SFO, JFK)  Fly(P2, JFK, SFO)  Load(C1, P2, SFO)  Fly(P2, SFO, JFK)  Fly(P1, JFK, SFO)  Unload(C1, P2, JFK)  Fly(P2, JFK, SFO)  Fly(P1, SFO, JFK)  Load(C2, P1, JFK)  Fly(P2, SFO, JFK)  Fly(P1, JFK, SFO)  Unload(C2, P1, SFO) | 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)  Fly(P1, SFO, JFK)  Unload(C1, P1, JFK)  Load(C2, P2, JFK)  Fly(P2, JFK, SFO)  Unload(C2, P2, SFO) | Load(C1, P1, SFO)  Fly(P1, SFO, JFK)  Load(C2, P2, JFK)  Fly(P2, JFK, SFO)  Unload(C1, P1, JFK)  Unload(C2, P2, SFO) |
| **AC2** | Load(C2, P2, JFK)  Load(C1, P1, SFO)  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) | Too long to display | 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(C3, P3, SFO)  Unload(C1, P1, JFK)  Unload(C2, P2, SFO) | Load(C3, P3, ATL)  Fly(P3, ATL, SFO)  Unload(C3, P3, SFO)  Load(C1, P1, SFO)  Fly(P1, SFO, JFK)  Unload(C1, P1, JFK)  Load(C2, P2, JFK)  Fly(P2, JFK, SFO)  Unload(C2, P2, SFO) | Load(C1, P1, SFO)  Fly(P1, SFO, JFK)  Load(C2, P2, JFK)  Fly(P2, JFK, SFO)  Load(C3, P3, ATL)  Fly(P3, ATL, SFO)  Unload(C3, P3, SFO)  Unload(C1, P1, JFK)  Unload(C2, P2, SFO) |
| **AC3** | Load(C2, P2, JFK)  Load(C1, P1, SFO)  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) | Too long to display | 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(C4, P2, SFO)  Unload(C3, P1, JFK)  Unload(C1, P1, JFK)  Unload(C2, P2, 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(C4, P2, SFO)  Unload(C3, P1, JFK)  Unload(C1, P1, JFK)  Unload(C2, P2, SFO) | Load(C2, P2, JFK)  Fly(P2, JFK, ORD)  Load(C4, P2, ORD)  Fly(P2, ORD, SFO)  Load(C1, P1, SFO)  Fly(P1, SFO, ATL)  Load(C3, P1, ATL)  Fly(P1, ATL, JFK)  Unload(C4, P2, SFO)  Unload(C3, P1, JFK)  Unload(C1, P1, JFK)  Unload(C2, P2, SFO) |

## Analysis of Uninformed Search Strategies

Following three uninformed search strategies are selected

* 1. Breath first search
  2. Depth first graph search
  3. Uniform cost search

|  |  |
| --- | --- |
| Optimality | Both breath first search and uniform cost search algorithms produce optimal plans for the three given problems. Depth first graph search, on the other hand, generates non-optimal plans, especially for AC2 & AC3 larger than 550 |
| Memory | Node expansions and new nodes indicate the memory usage of the strategies. As for memory efficiency, depth first graph search algorithm outperforms the other two by using around 20-25% of the memory used by those two algorithms. However this algorithm does not achieve an optimal plan. Breath first search uses less memory than uniform cost search, on the other hand. |
| Execution time | In terms of execution time depth first graph search finishes within 20% of the time of the other two strategies. Breath first graph search executes faster than uniform cost search on the average. |

Even though depth first graph search executes very fast and uses very little memory, it suffers greatly in terms of plan length. Therefore breath first search will be the preferred strategy as it provides optimal solution, executes faster, and uses less memory when compared to uniform cost search.

Table 5 shows optimal plans achieved by applying the breath first search strategy. Other optimal plans for alternative strategies can be seen on Table 4.

*Table 5. Optimal plans for uninformed search strategies*

|  |  |  |
| --- | --- | --- |
| AC1 | AC2 | AC3 |
| Load(C2, P2, JFK)  Load(C1, P1, SFO)  Fly(P2, JFK, SFO)  Unload(C2, P2, SFO)  Fly(P1, SFO, JFK)  Unload(C1, P1, JFK) | Load(C2, P2, JFK)  Load(C1, P1, SFO)  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) | Load(C2, P2, JFK)  Load(C1, P1, SFO)  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) |

## Analysis of Heuristic Search Strategies

Following two heuristic search strategies are investigated

* 1. A\* search with Ignore preconditions
  2. A\* search with level sum

|  |  |
| --- | --- |
| Optimality | Both heuristic search strategies achieve the optimal plan. |
| Memory | Level sum algorithm uses significantly less memory than ignore preconditions heuristic (4-15 times). |
| Execution time | Level sum algorithm executes significantly longer than ignore preconditions heuristics (15-20 times). |

Since both algorithms achieve optimal plans, ignore preconditions heuristic will be chosen if the speed is important, and level sum heuristic will be preferred if the memory consumption is important in making a decision.

Optimal plans from ignore preconditions is given in Table 6. Please check Table 4 for the full list of optimal plans belonging to alternative strategies.

|  |  |  |
| --- | --- | --- |
| AC1 | AC2 | AC3 |
| Load(C1, P1, SFO)  Fly(P1, SFO, JFK)  Unload(C1, P1, JFK)  Load(C2, P2, JFK)  Fly(P2, JFK, SFO)  Unload(C2, P2, SFO) | Load(C3, P3, ATL)  Fly(P3, ATL, SFO)  Unload(C3, P3, SFO)  Load(C1, P1, SFO)  Fly(P1, SFO, JFK)  Unload(C1, P1, JFK)  Load(C2, P2, JFK)  Fly(P2, JFK, SFO)  Unload(C2, P2, 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(C4, P2, SFO)  Unload(C3, P1, JFK)  Unload(C1, P1, JFK)  Unload(C2, P2, SFO) |

## Conclusion

When the performance values listed in Table 3 are compared, we can see that both heuristic and uninformed search strategies achieve the optimal plans. However both heuristic strategies uses significantly less memory to execute (with level sum offering the most reduction) and ignore preconditions heuristic runs the fastest.

Heuristic search algorithms perform significantly better than their uninformed search counterparts. Moreover they offer flexibility of memory vs speed with the selection of different heuristics such as ignore preconditions, level sum, etc. Therefore they have a clear win against uninformed search algorithms.