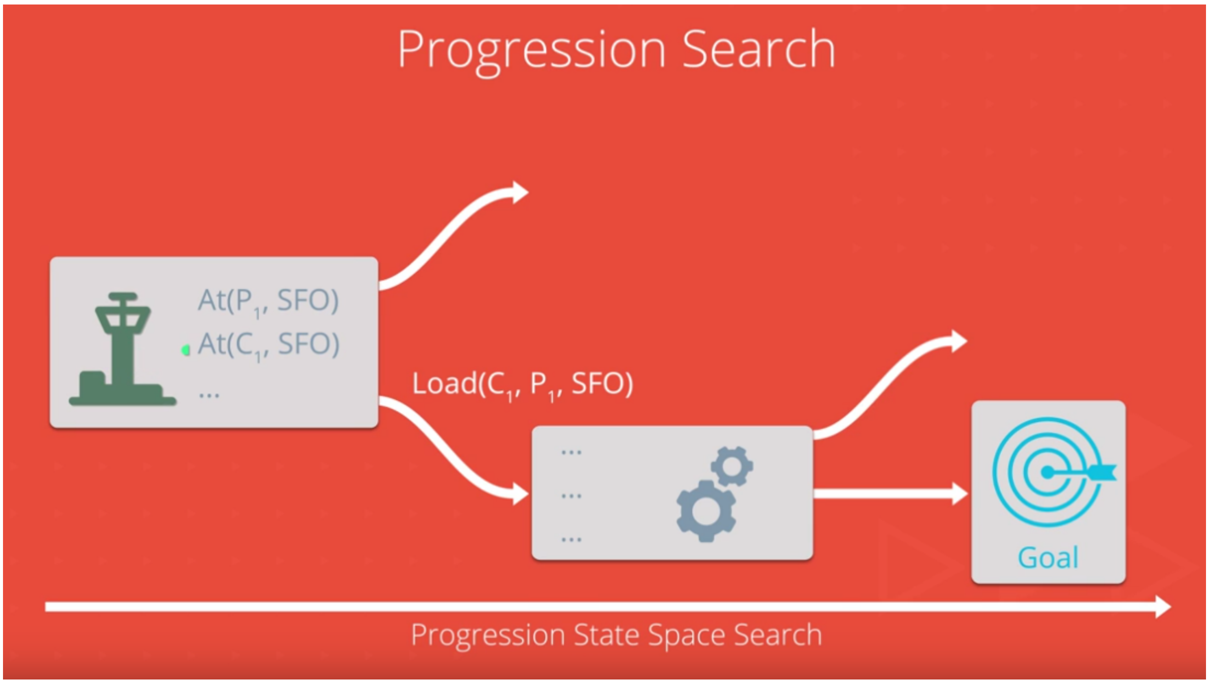
Planning Search Heuristic Analysis

By Pablo Mateo



Experimentation and documentation metrics for non-heuristic planning solution searches

In this project, we implemented a planning search agent to solve deterministic logistics planning problems for an **Air Cargo transport System**. We start with several problems, all of them in the Air Cargo domain. They all have the same action schema defined, but different initial states and goals.

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))
```

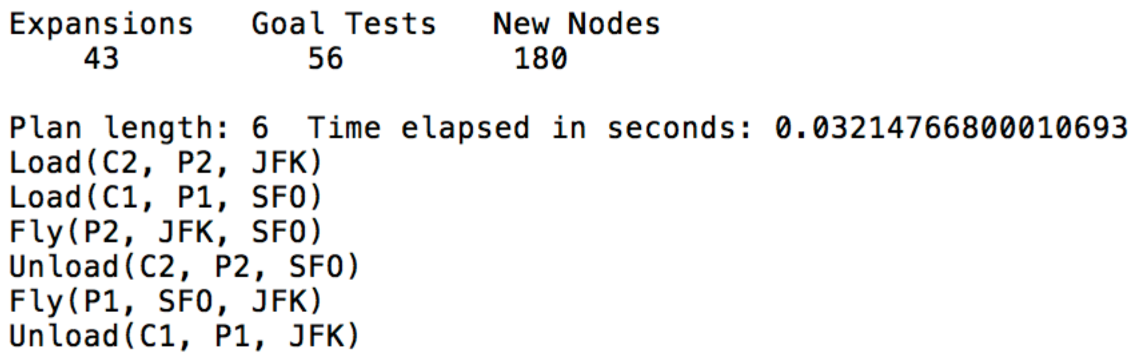
Problem 1: Initial State and Goal

```
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))
```

For the first problem, we are asked to move Cargo 1 from SFO airport to JFK and Cargo 2 the other way round (from JFK to SFO). From the 3 analysed problems, this is the simpler one, so we were able to use all of the required search functions in a reasonable computing time.

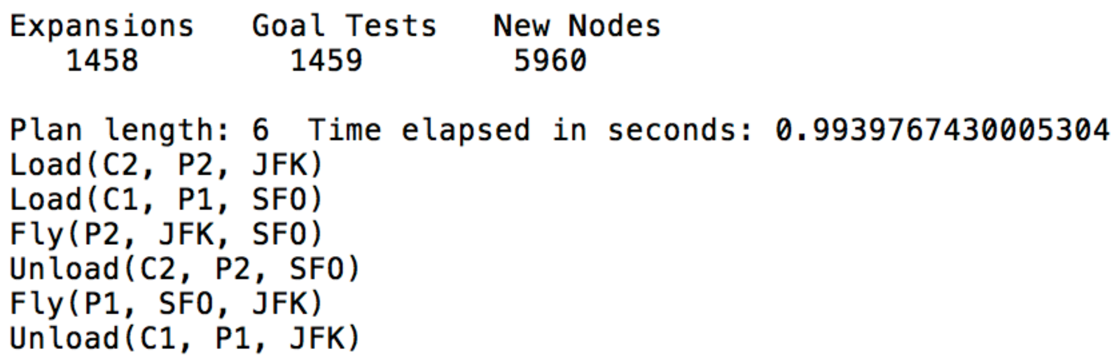
Search Functions Results

1. Breadth-First Search



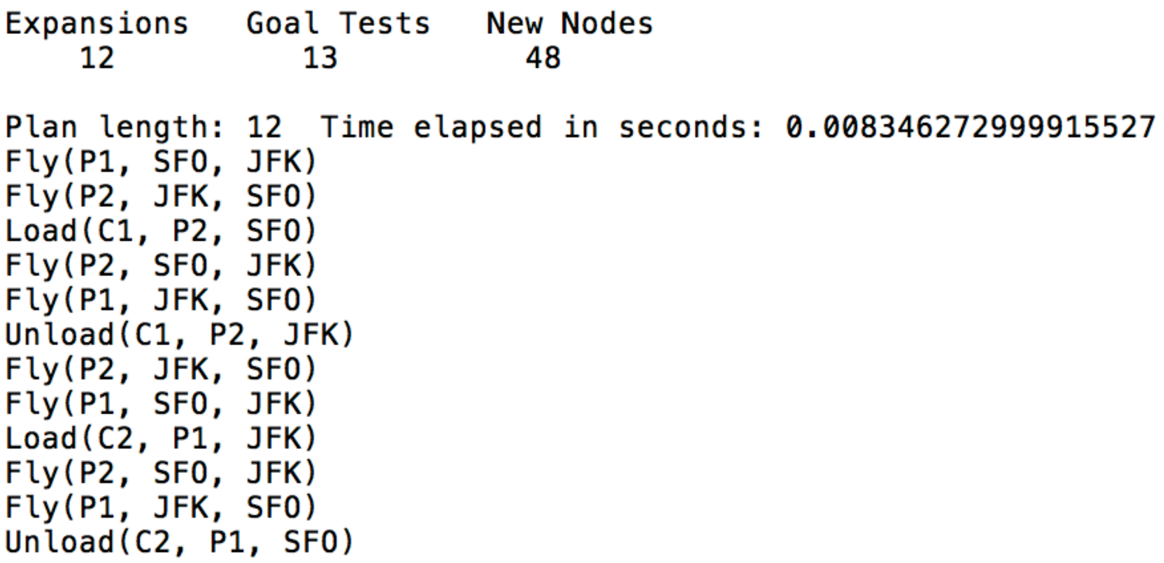
```
Expansions   Goal Tests   New Nodes
    43           56          180

Plan length: 6  Time elapsed in seconds: 0.03214766800010693
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

2. Breadth-First Tree Search



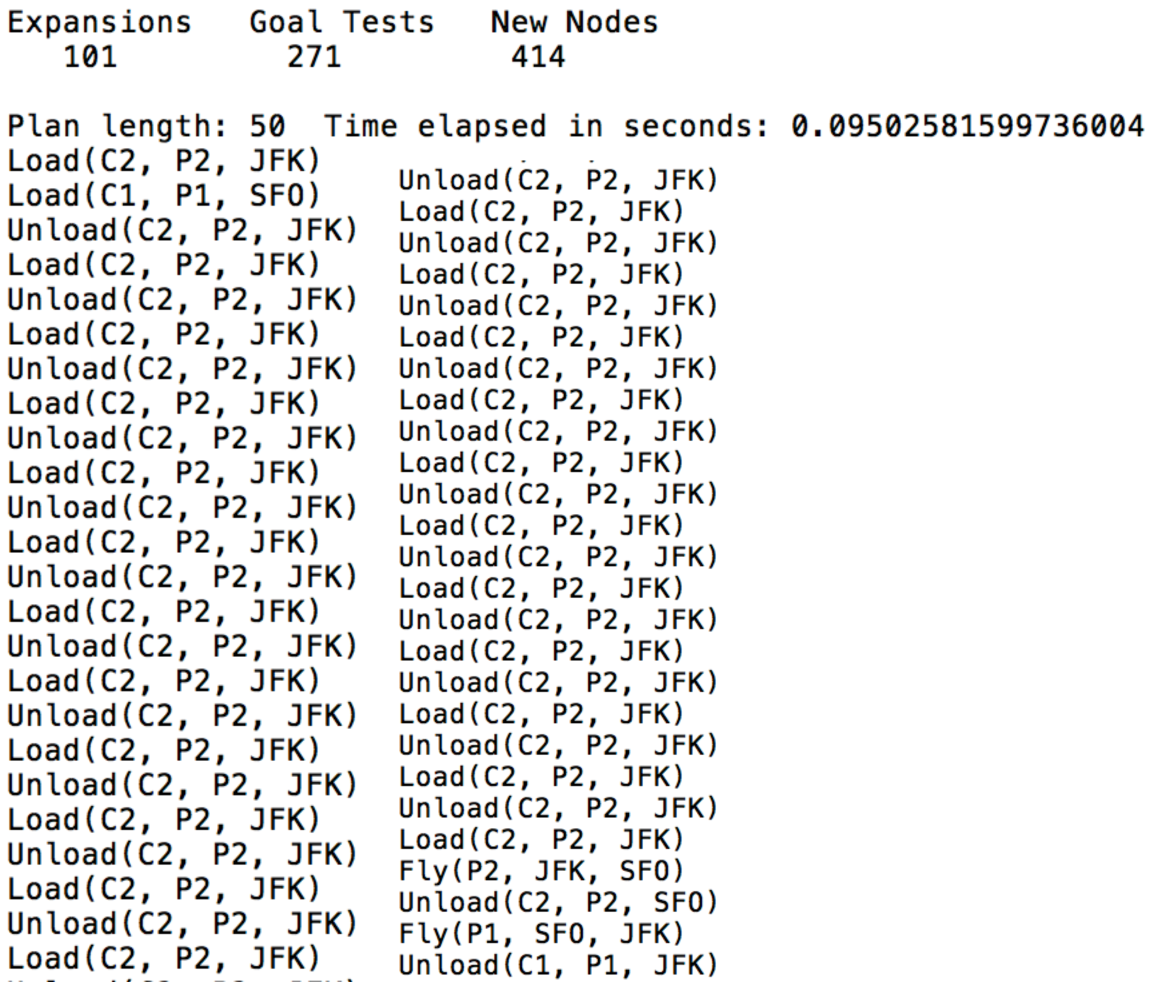
```
Expansions   Goal Tests   New Nodes
   1458         1459        5960

Plan length: 6  Time elapsed in seconds: 0.9939767430005304
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

3. Depth-First Graph Search



```
Expansions   Goal Tests   New Nodes
    12           13          48

Plan length: 12  Time elapsed in seconds: 0.008346272999915527
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)
```

4. Depth Limited Search

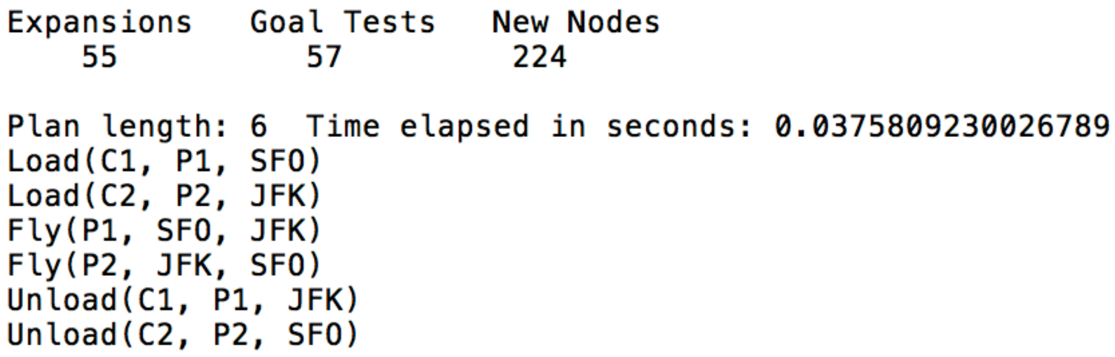


```
Expansions   Goal Tests   New Nodes
   101          271         414

Plan length: 50  Time elapsed in seconds: 0.09502581599736004
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
Unload(C2, P2, JFK)
Load(C2, P2, JFK)
```

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

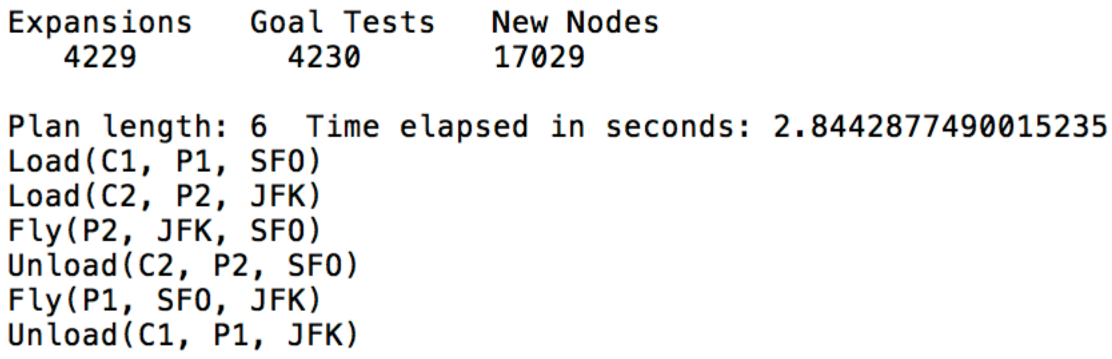
[…]

5. Uniform Cost Search



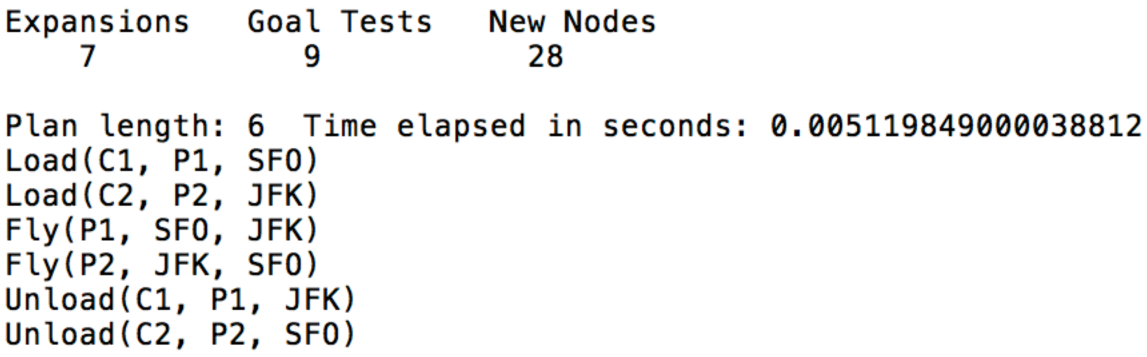
```
Expansions   Goal Tests   New Nodes
    55           57         224

Plan length: 6  Time elapsed in seconds: 0.0375809230026789
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

6. Recursive Best-First Tree Search h_1



```
Expansions   Goal Tests   New Nodes
   4229         4230       17029

Plan length: 6  Time elapsed in seconds: 2.8442877490015235
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

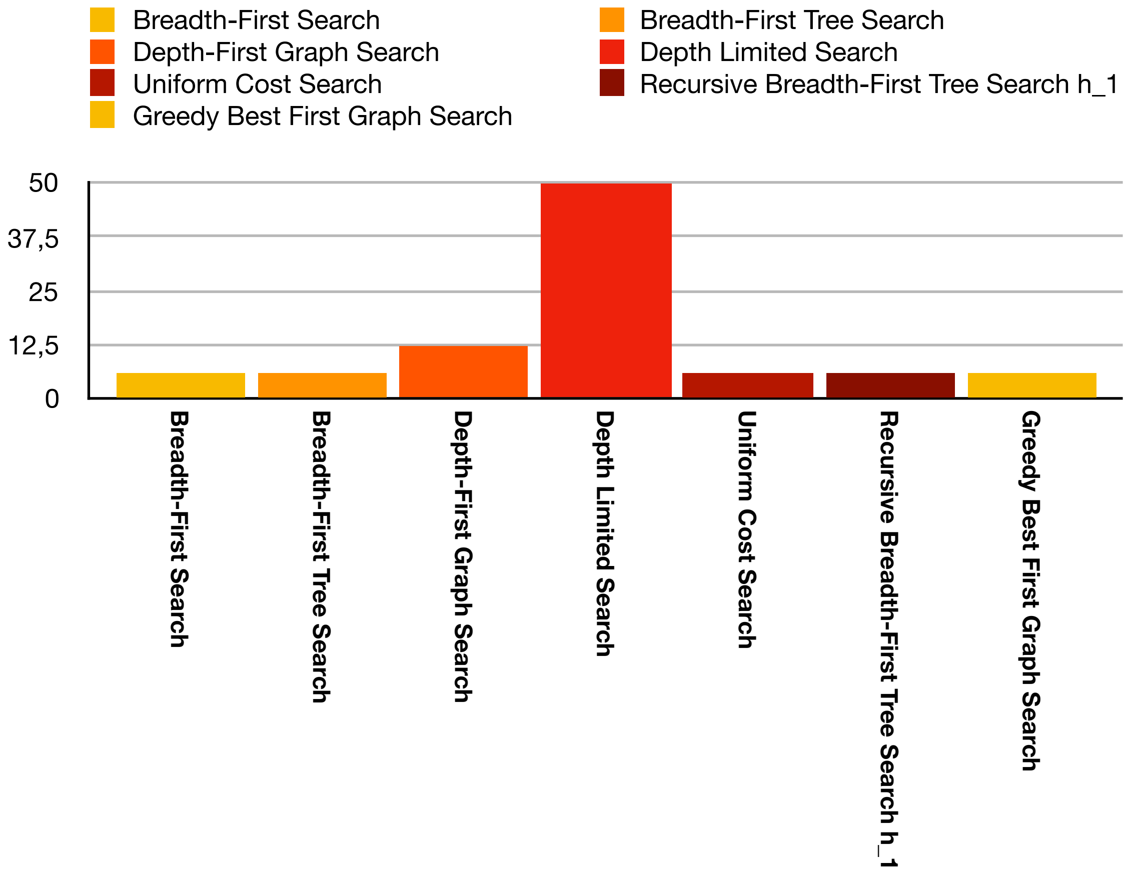
7. Greedy Best First Graph Search



```
Expansions   Goal Tests   New Nodes
    7            9          28

Plan length: 6  Time elapsed in seconds: 0.005119849000038812
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

Problem 1 Non-Heuristics Solutions Comparison

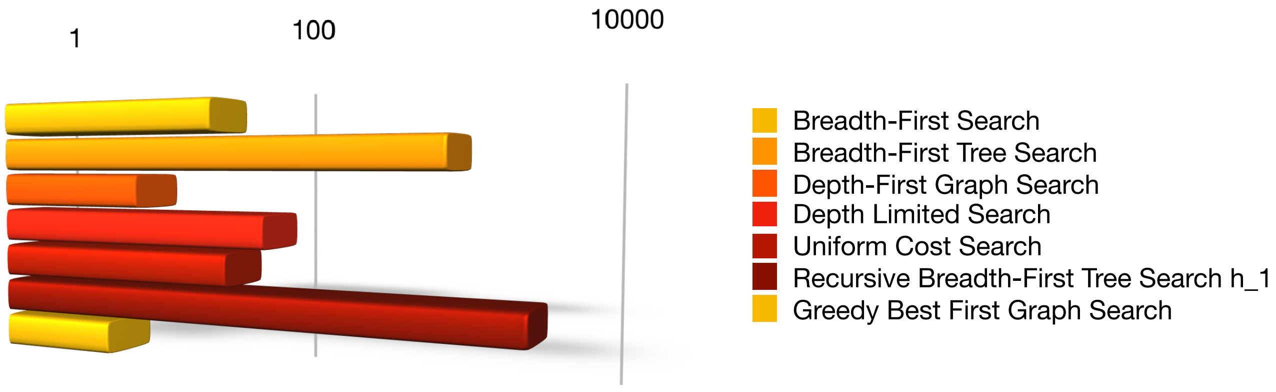
Plan Length



As we can see in the following graph, most of the search functions returned an optimal plan of 6 actions to achieve our goal. **Depth Limited Search** was the most inefficient one as its solution proposed 50 different actions. **Depth-First Graph Search** was also not optimal (12 actions) which is not far away from the solution but in a real-world environment it will mean to double the effort.

**Note: Some of the charts shown use a logarithmic scale.*

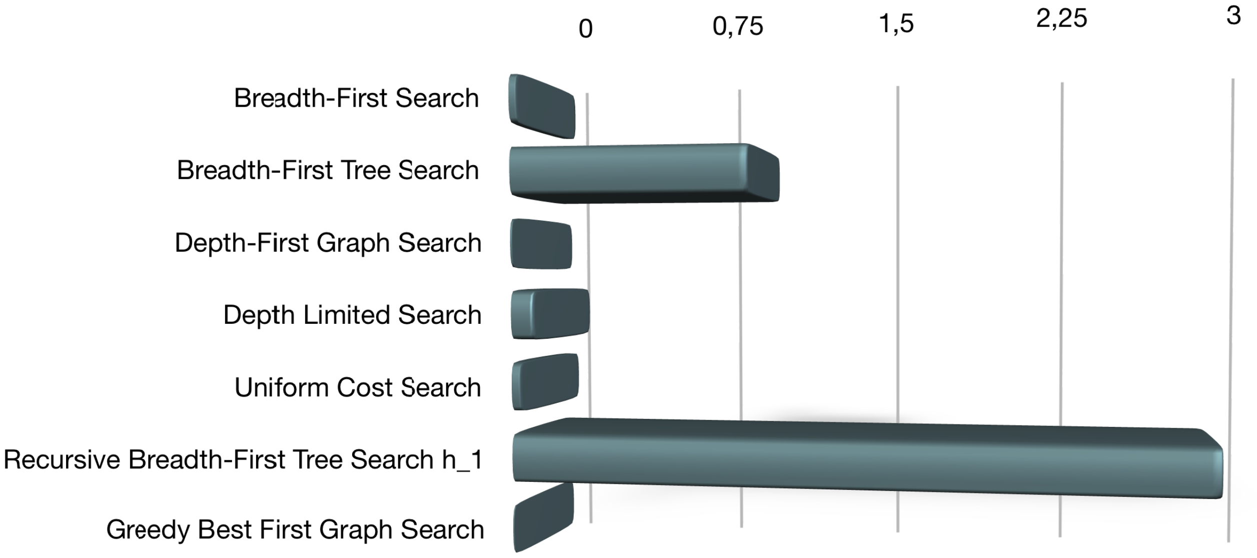
Expansions



The functions with least expansions was the **Greedy Best First Graph Search** with only 7, followed by the **Depth-First Graph Search**. **Recursive Breadth-First Tree Search** was the one with most expansions (4.229) and therefore, as we will see later, the one which took more time to analyse. The rest of them ranged between 12 to 101 expansions with the exception of **Breadth-First Tree Search** (1.458 expansions).

Chart scale is logarithmic as function n.6 is way higher than the others

Time Elapsed



As Problem 1 is the simpler of the 3 problems analysed, the time it took to the functions to perform the search was very low. All of the algorithms proved extremely fast in this environment as they all took less than a second to find the optimal solution. Only **Recursive Breadth First Tree Search** took more time (2,8442877 seconds). However, it was also the algorithm that more Goal Tests and new nodes were analysed or created to find the solution.

Problem 1 - Comparison Table.

Search Function	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
Breadth-First Search	6	43	56	180	0,032147
Breadth-First Tree Search	6	1458	1459	5960	0,993976
Depth-First Graph Search	12	12	13	48	0,0083462
Depth Limited Search	50	101	271	414	0,0950258
Uniform Cost Search	6	55	57	224	0,0375809
Recursive Breadth-First Tree Search h_1	6	4229	4230	17029	2,8442877
Greedy Best First Graph Search	6	7	9	28	0,00511984

As we can see in this table, 5 out of 7 search functions find an optimal solution with 6 actions in a very short period of time. Comparing all of them, **Greedy Best First Graph Search** looks like the one to choose as it has the lowest values in all 5 analysed parameters, followed by Breadth First Search.

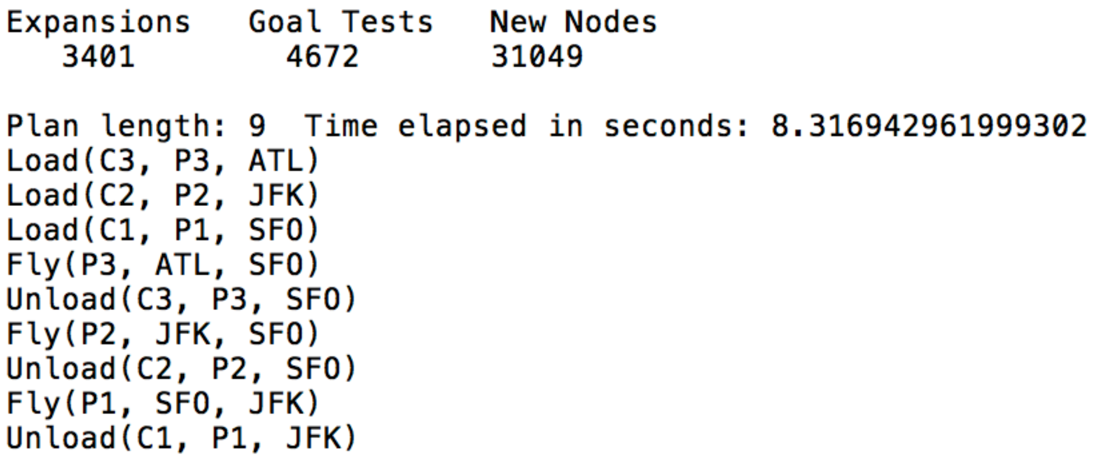
Problem 2: Initial State and Goal

```
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))
```

For the second problem, more parameters are introduced. We have three planes and cargos in SFO, JFK and ATL airports to start. Cargo 1 should end at JFK and both 2 and 3 at ATL.

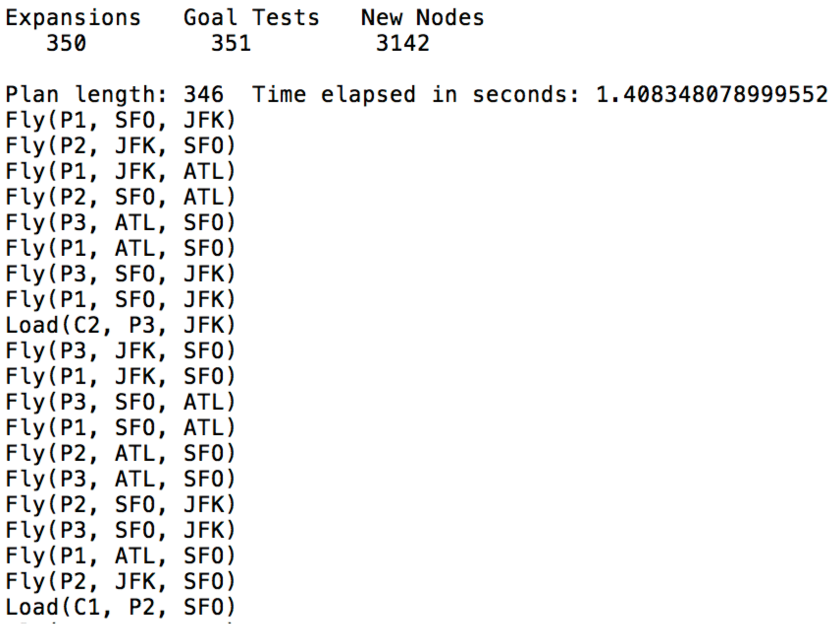
Search Functions Results

1. Breadth-First Search



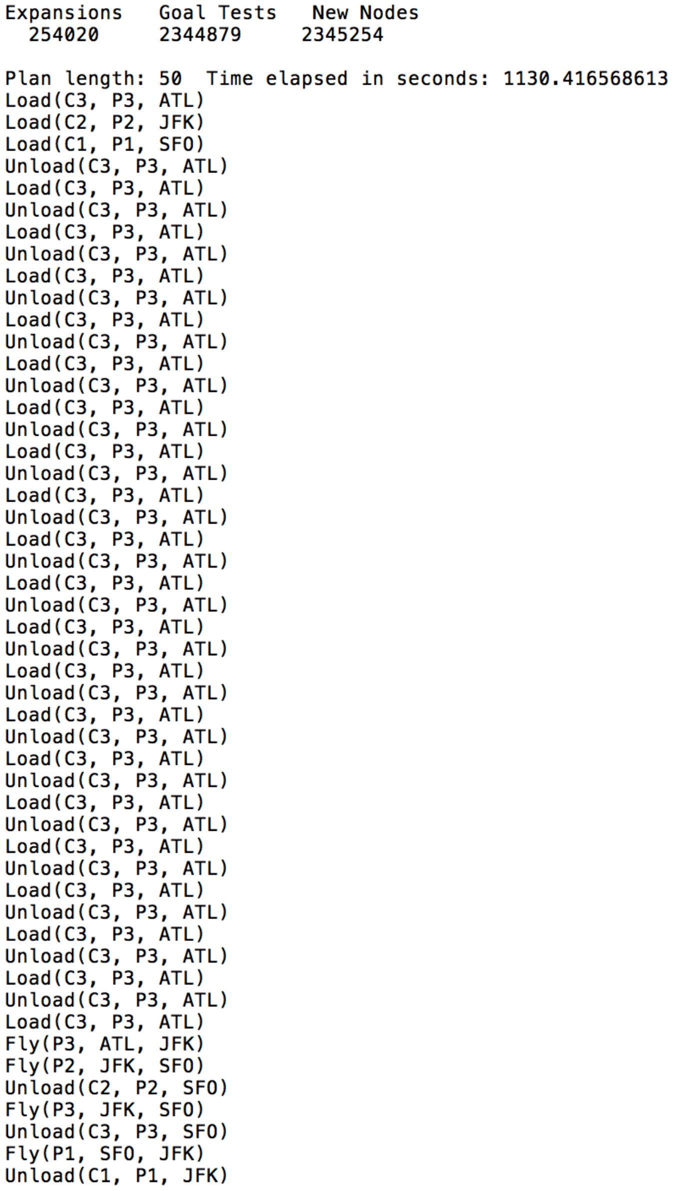
```
Expansions   Goal Tests   New Nodes
   3401         4672        31049

Plan length: 9  Time elapsed in seconds: 8.316942961999302
Load(C3, P3, ATL)
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

3. Depth-First Graph Search



```
Expansions   Goal Tests   New Nodes
   350          351          3142

Plan length: 346  Time elapsed in seconds: 1.408348078999552
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P1, JFK, ATL)
Fly(P2, SFO, ATL)
Fly(P3, ATL, SFO)
Fly(P1, ATL, SFO)
Fly(P3, SFO, JFK)
Fly(P1, SFO, JFK)
Load(C2, P3, JFK)
Fly(P3, JFK, SFO)
Fly(P1, JFK, SFO)
Fly(P3, SFO, ATL)
Fly(P1, SFO, ATL)
Fly(P2, ATL, SFO)
Fly(P3, ATL, SFO)
Fly(P2, SFO, JFK)
Fly(P3, SFO, JFK)
Fly(P1, ATL, SFO)
Fly(P2, JFK, SFO)
Load(C1, P2, SFO)
```

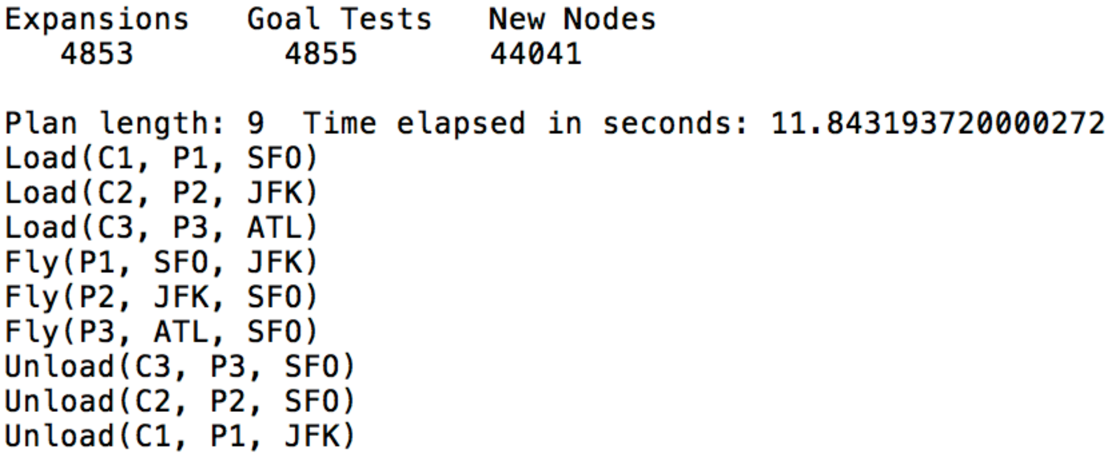
4. Uniform Cost Search



```
Expansions   Goal Tests   New Nodes
  254020     2344879     2345254

Plan length: 50  Time elapsed in seconds: 1130.416568613
Load(C3, P3, ATL)
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Unload(C3, P3, ATL)
Load(C3, P3, ATL)
Fly(P3, ATL, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P3, JFK, SFO)
Unload(C3, P3, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

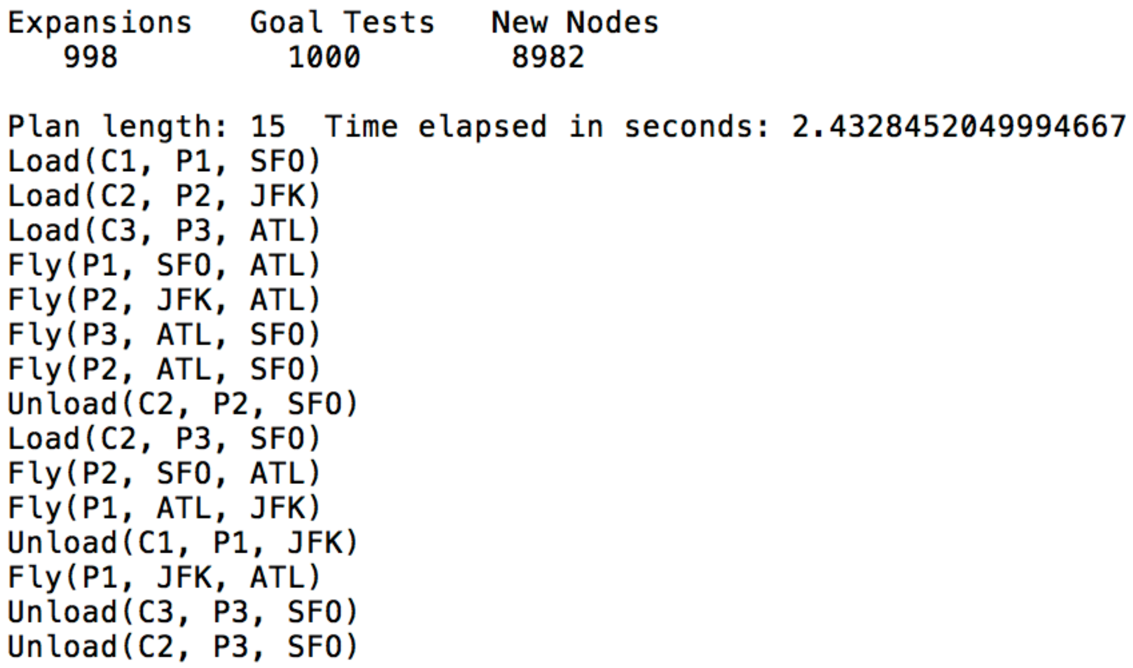
[…]

5. Uniform Cost Search



```
Expansions   Goal Tests   New Nodes
   4853        4855        44041

Plan length: 9  Time elapsed in seconds: 11.843193720000272
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(C2, P2, SFO)
Unload(C1, P1, JFK)
```

7. Breadth-First Search



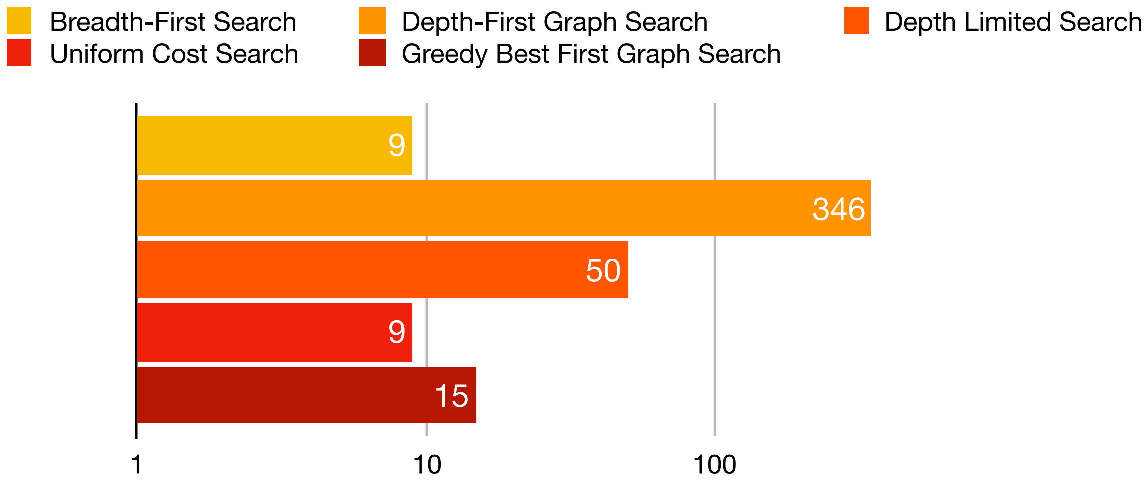
```
Expansions   Goal Tests   New Nodes
   998         1000        8982

Plan length: 15  Time elapsed in seconds: 2.4328452049994667
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, ATL)
Fly(P2, JFK, ATL)
Fly(P3, ATL, SFO)
Fly(P2, ATL, SFO)
Unload(C2, P2, SFO)
Load(C2, P3, SFO)
Fly(P2, SFO, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Fly(P1, JFK, ATL)
Unload(C3, P3, SFO)
Unload(C2, P3, SFO)
```

For this problem, we have not included results for **Breadth First Tree Search** nor **Recursive Best First Search** as they were taking too long to process.

However, we can take very interesting conclusions form the ones analysed.

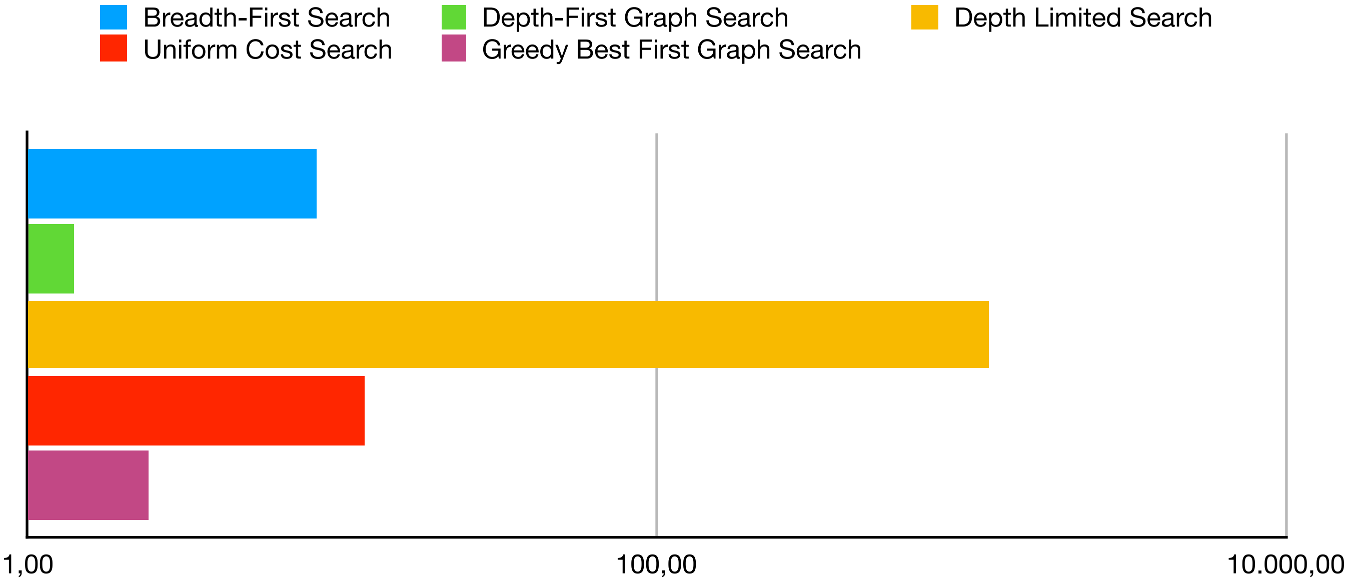
Problem 2 Non-Heuristics Solutions Comparison

Plan Length



The optimal solution for problem number 2 are 9 actions. We can see taking a look at the chart on top, that only 2 algorithms have found the best solution, those being **Breadth First Search** and **Uniforms Cost Search**.

Plan Elapsed Time



Checking the elapsed time, we found that the optimal solutions are those with average elapsed time. Both **Breadth First** and **Uniform Cost** get to the solution in a very similar time. However, we can see that those that take less time are not as good as these two.

Problem 2 - Comparison Table

Search Function	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
Breadth-First Search	9	3401	4672	31049	8,3169
Depth-First Graph Search	346	350	351	3142	1,4083
Depth Limited Search	50	254020	2344879	2345254	1130,4165
Uniform Cost Search	9	4853	4855	44041	11,8431
Greedy Best First Graph Search	15	998	1000	8982	2,4328

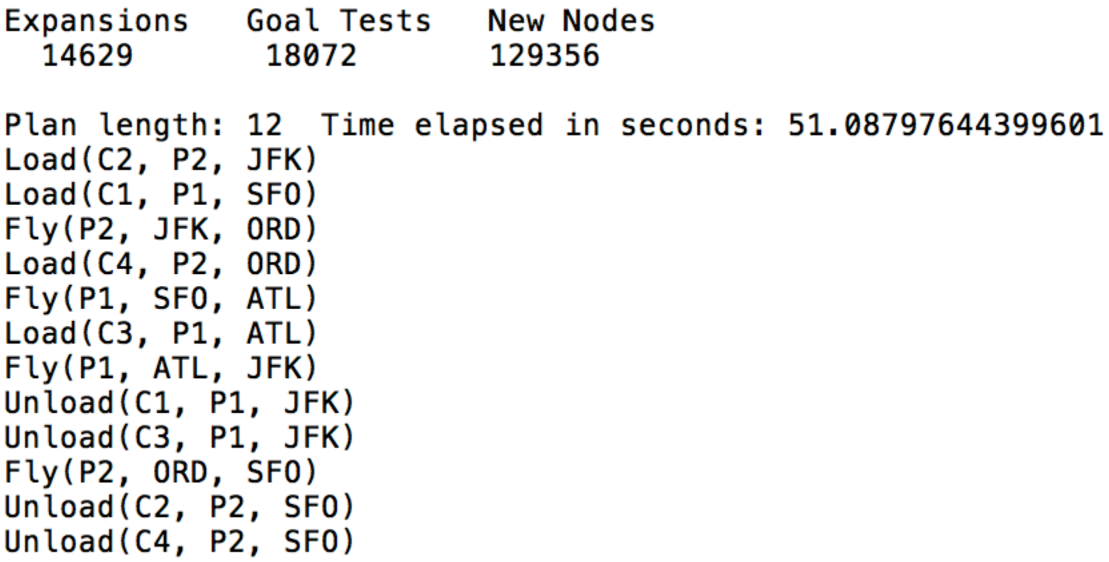
Problem 3: Initial State and Goal

```
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))
```

For the third problem, we have 4 airports (SFO, JFK, ATL & ORD), 4 cargos and 2 available airplanes.

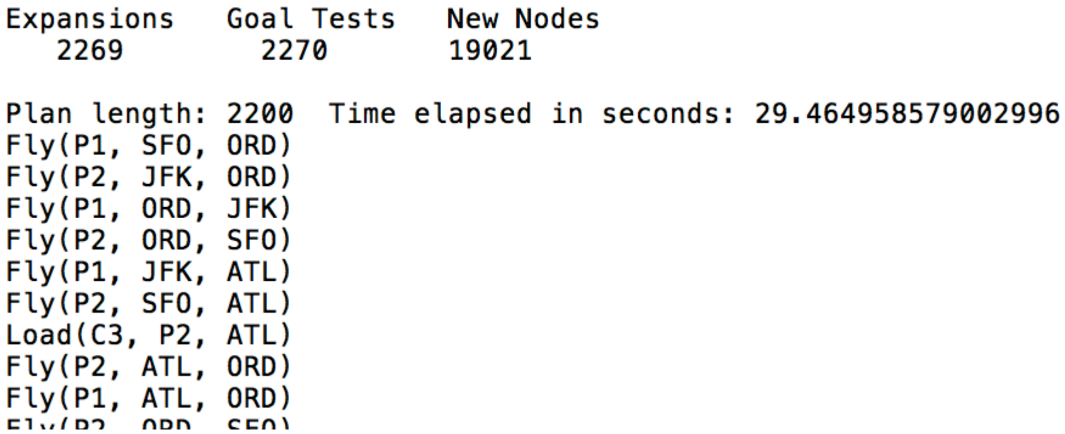
Search Functions Results

1. Breadth-First Search



```
Expansions   Goal Tests   New Nodes
  14629        18072      129356

Plan length: 12  Time elapsed in seconds: 51.08797644399601
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)
```

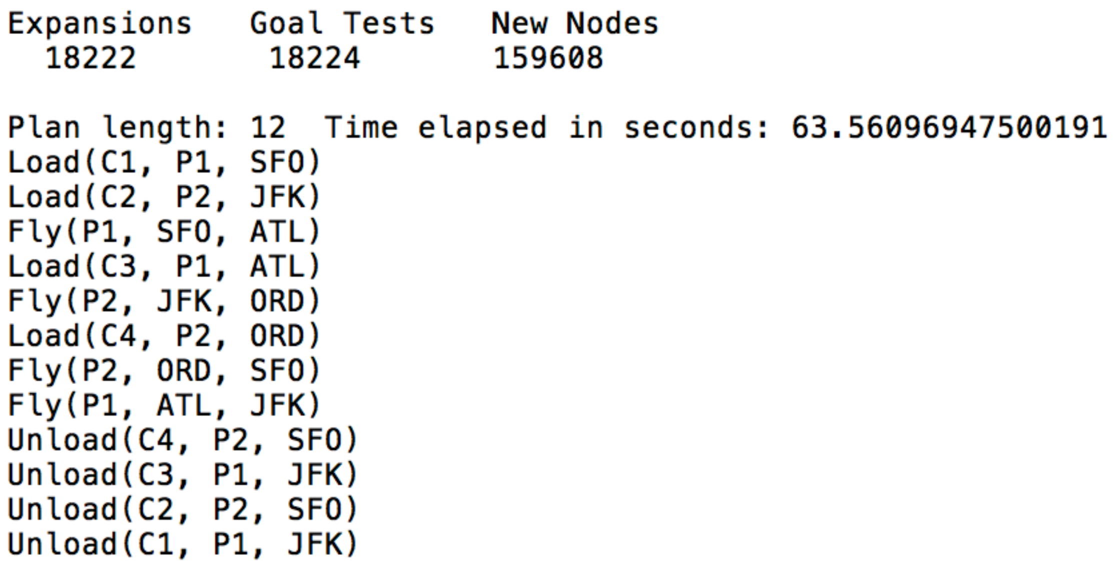
3. Depth-First Graph Search



```
Expansions   Goal Tests   New Nodes
   2269        2270       19021

Plan length: 2200  Time elapsed in seconds: 29.464958579002996
Fly(P1, SFO, ORD)
Fly(P2, JFK, ORD)
Fly(P1, ORD, JFK)
Fly(P2, ORD, SFO)
Fly(P1, JFK, ATL)
Fly(P2, SFO, ATL)
Load(C3, P2, ATL)
Fly(P2, ATL, ORD)
Fly(P1, ATL, ORD)
Fly(P2, ORD, SFO)
```

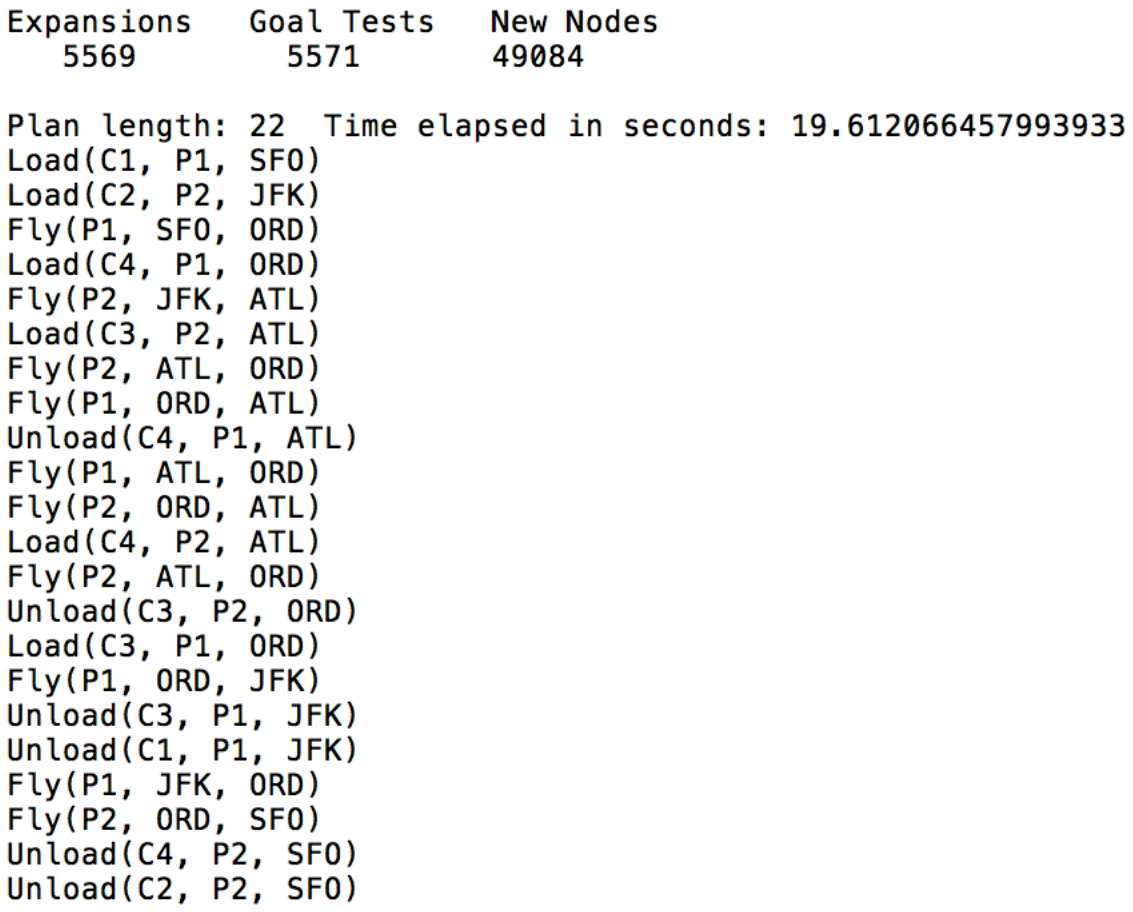
[...]

5. Uniform Cost Search



```
Expansions   Goal Tests   New Nodes
  18222        18224      159608

Plan length: 12  Time elapsed in seconds: 63.56096947500191
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(C2, P2, SFO)
Unload(C1, P1, JFK)
```

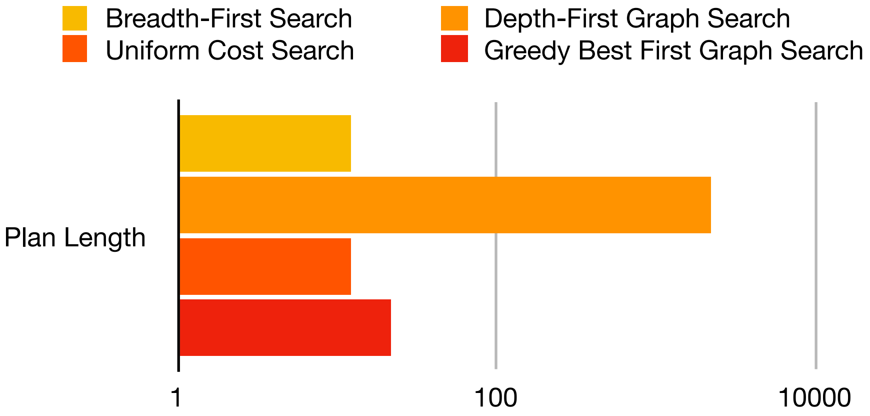
7. Greedy Best First Graph Search



```
Expansions   Goal Tests   New Nodes
   5569        5571       49084

Plan length: 22  Time elapsed in seconds: 19.612066457993933
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ORD)
Load(C4, P1, ORD)
Fly(P2, JFK, ATL)
Load(C3, P2, ATL)
Fly(P2, ATL, ORD)
Fly(P1, ORD, ATL)
Unload(C4, P1, ATL)
Fly(P1, ATL, ORD)
Fly(P2, ORD, ATL)
Load(C4, P2, ATL)
Fly(P2, ATL, ORD)
Unload(C3, P2, ORD)
Load(C3, P1, ORD)
Fly(P1, ORD, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Fly(P1, JFK, ORD)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Unload(C2, P2, SFO)
```

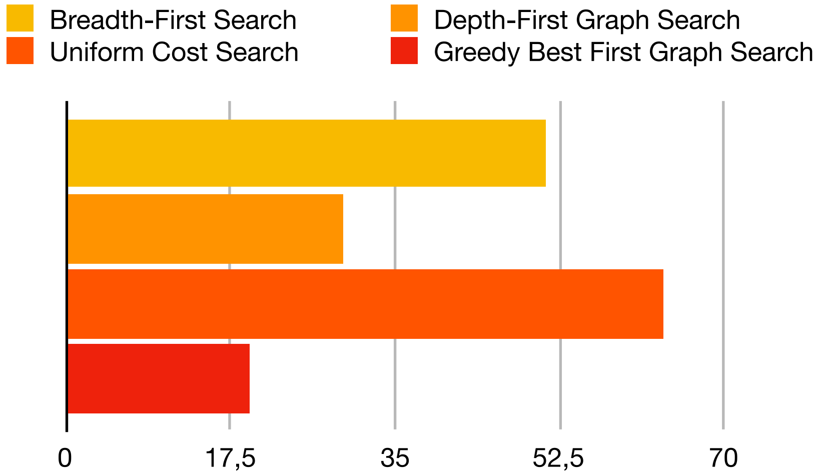
Problem 3 Non-Heuristics Solutions Comparison

Plan Length



With Problem 3 we can see a huge difference in actions required between **Depth-First** and the rest of algorithms. **Breadth-First** and **Uniform Cost** once more obtain the optimal solution (12 actions required), while **Greedy Best First Graph Search** gets close but with 22 actions.

Plan Elapsed Time



In this table we can observe that less time can be traduced into worst solutions. Although **Depth First** is slower than **Greedy Best** and even though it returns a poorer solution. **Breadth First** and **Uniform Cost** both get to the optimal solution in similar times (close to a minute). We must indicate that ***pypy*** has been used to increase the performing of the algorithm and make them a bit faster computing the possible solutions.

Problem 3 - Comparison Table

Search Function	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
Breadth-First Search	12	14629	18072	129356	51,0879
Depth-First Graph Search	2200	2269	2270	19021	29,46495
Uniform Cost Search	12	18222	18224	159608	63,56096
Greedy Best First Graph Search	22	5569	5571	49084	19,6120664

Once more and because of the complexity of the problem, some algorithms haven’t been analysed as they were taking too much time to compute.

We can see that in these cases, **Breadth First Search** and **Uniform Cost search** are the best algorithms to use. In the first problem, **Greedy Best First Graph Search** turned to be quicker, more efficient and the best option to get to an optimal solution. However, when the problem gets more complicated, it gets close to the optimal solution, but it is not able to provide it. Comparing **Breadth First** with **Uniform Cost**, both algorithms have a similar output. However, as the parameters increase and the problem requires a bigger amount of data to be analysed, **Breadth First Search** seems to perform better as the difficulty of the problem rises.

Experimentation and documentation metrics for Heuristic planning solution searches

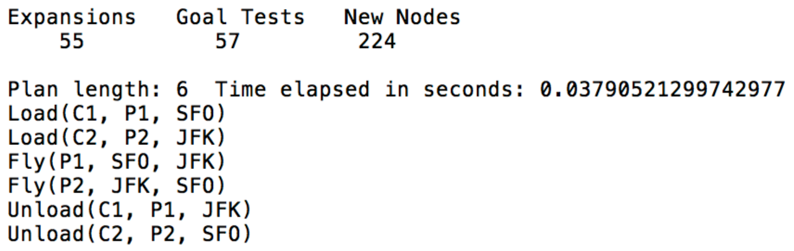
The 3 problems were also analysed using an A* algorithm with 3 different Heuristics. Being those:

- **A* Search**
- **A* Search with Ignore Preconditions**
- **A* Search with LevelSum**

Problem 1

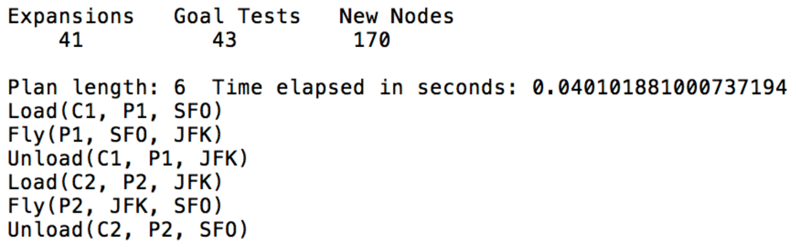
Heuristics Functions Results

1. A-Star Search



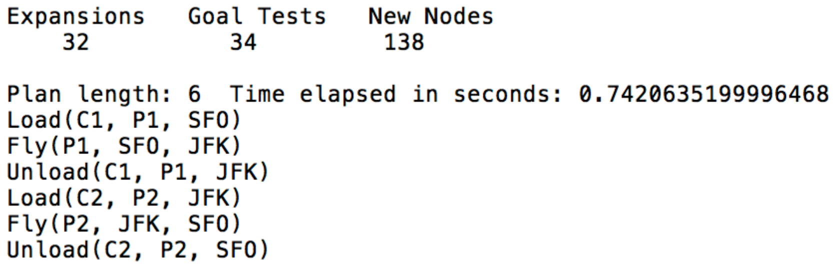
```
Expansions   Goal Tests   New Nodes
    55           57          224

Plan length: 6  Time elapsed in seconds: 0.03790521299742977
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

2. A-Star Search w. Ignore Preconditions



```
Expansions   Goal Tests   New Nodes
    41           43          170

Plan length: 6  Time elapsed in seconds: 0.040101881000737194
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
```

3. A-Star Search w. LevelSum

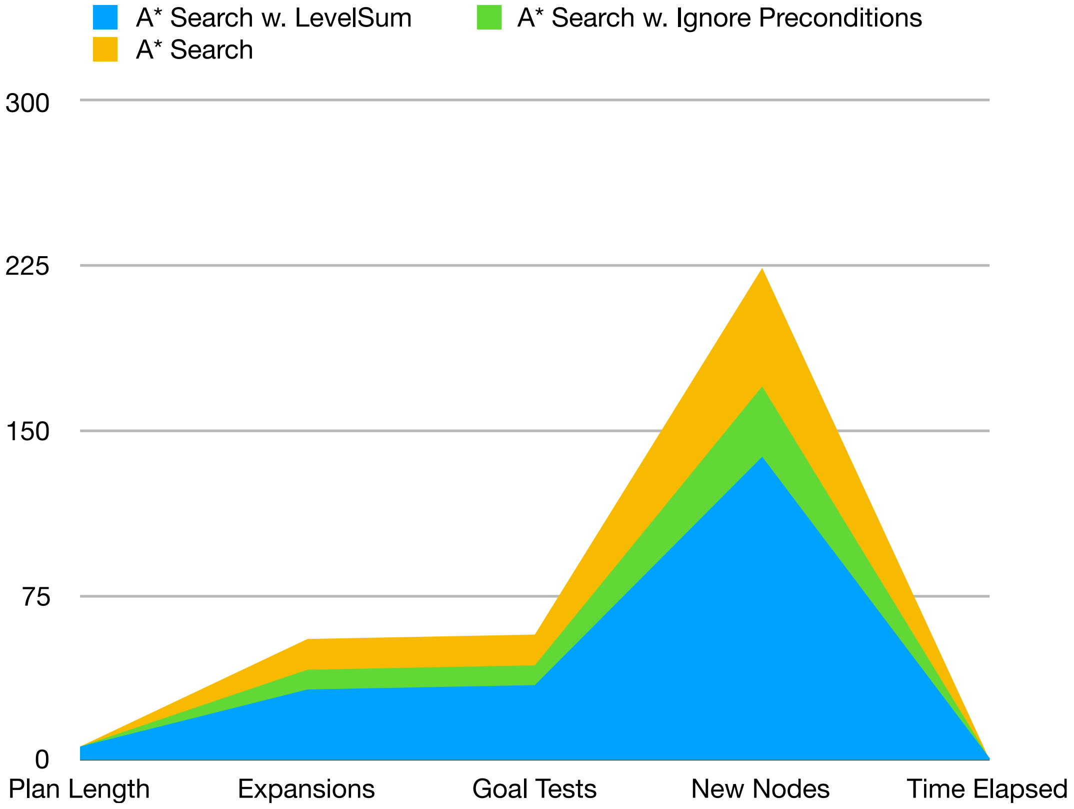


```
Expansions   Goal Tests   New Nodes
    32           34          138

Plan length: 6  Time elapsed in seconds: 0.7420635199996468
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
```

Problem 1 Heuristics Solutions Comparison

Comparison Table

Search Function	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
A* Search	6	55	57	224	0,0379
A* Search IP	6	41	43	170	0,4010
A* Search LS	6	32	34	138	0,7421

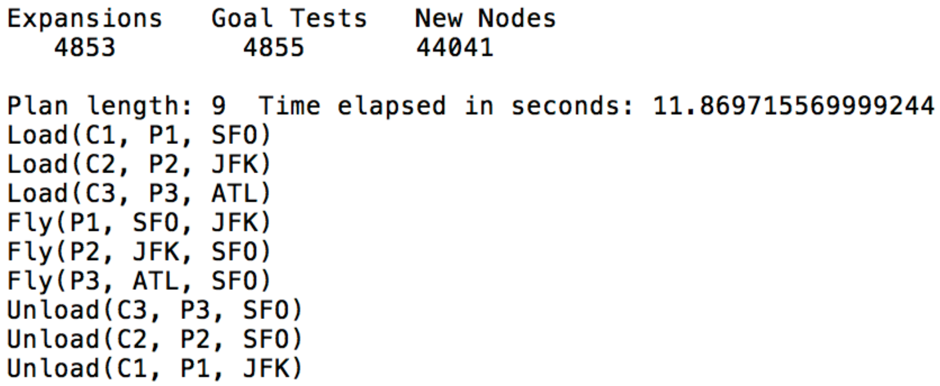


With the first problem, we can find out that the differences between our strategies are really small. The 3 algorithms get to the optimal solution (6 actions) in less than a second and there are just slightly differences in expansions, goal tests and new nodes. However, **A* Search** proves to be the fastest in comparison to the other 2 heuristics used.

Problem 2

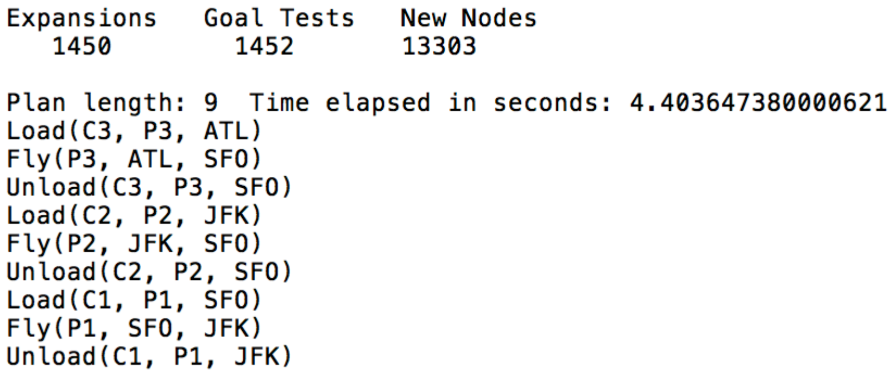
Heuristics Functions Results

1. A-Star Search



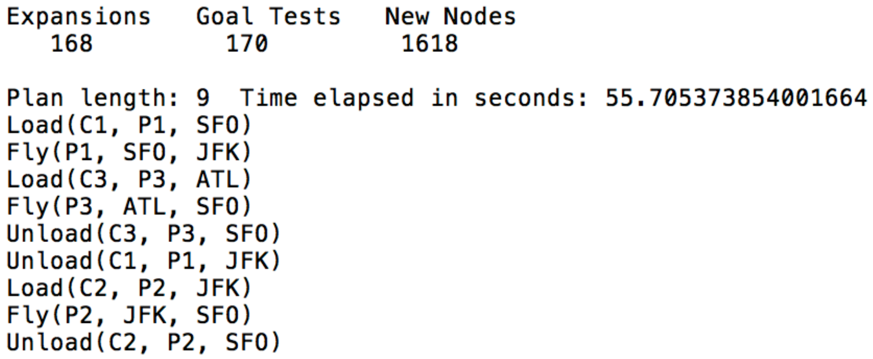
```
Expansions    Goal Tests   New Nodes
   4853         4855       44041

Plan length: 9  Time elapsed in seconds: 11.869715569999244
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(C2, P2, SFO)
Unload(C1, P1, JFK)
```

2. A-Star Search w. Ignore Preconditions



```
Expansions   Goal Tests   New Nodes
   1450         1452       13303

Plan length: 9  Time elapsed in seconds: 4.403647380000621
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

3. A-Star Search w. LevelSum

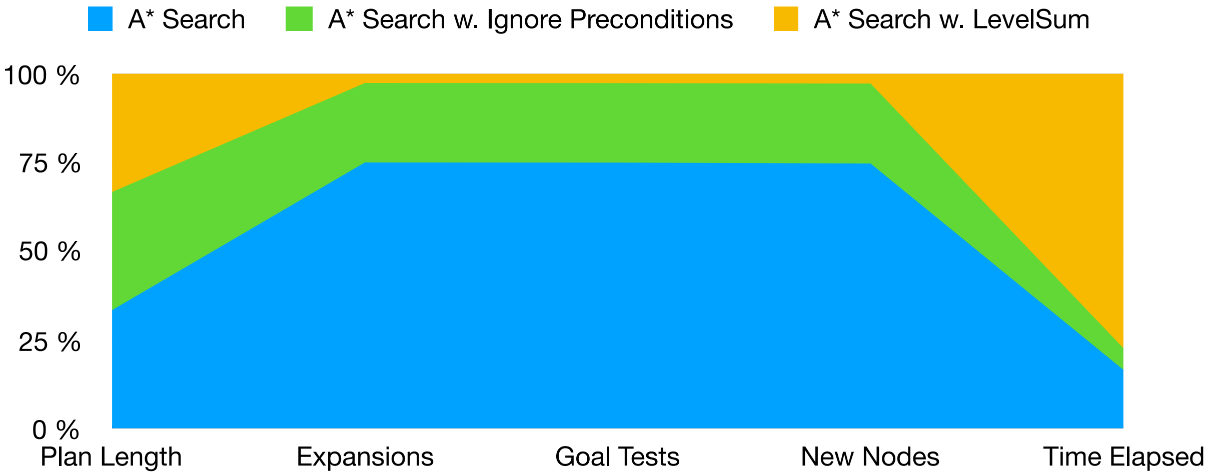


```
Expansions   Goal Tests   New Nodes
   168          170         1618

Plan length: 9  Time elapsed in seconds: 55.705373854001664
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
```

Problem 2 Heuristics Solutions Comparison

Comparison Table

Search Function	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
A* Search	9	4853	4855	44041	11,8697
A* Search IP	9	1450	1452	13303	4,4036
A* Search LS	9	168	170	1618	55,7054



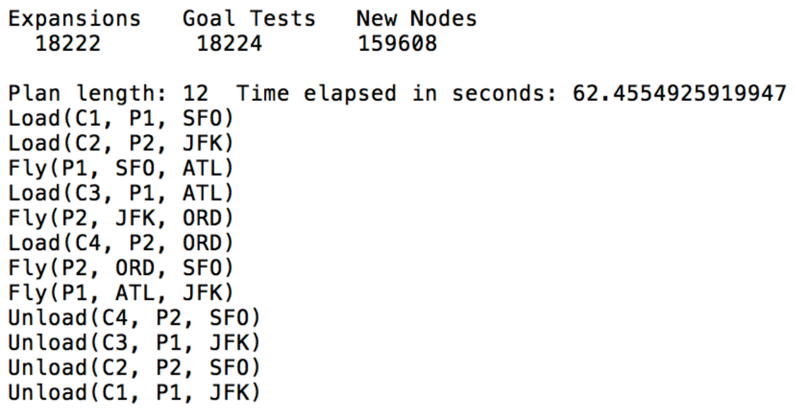
For the second problem, the differences are slightly bigger. **A* Search IP** is much faster that the other 2 options. Although **A* Search LS** is much more efficient than the other 2. **A* Search** is like the middle option if you cannot decide wether if you prefer memory optimisation or speed.

The graph is in percentage mode for a more visual representation of the data.

Problem 3

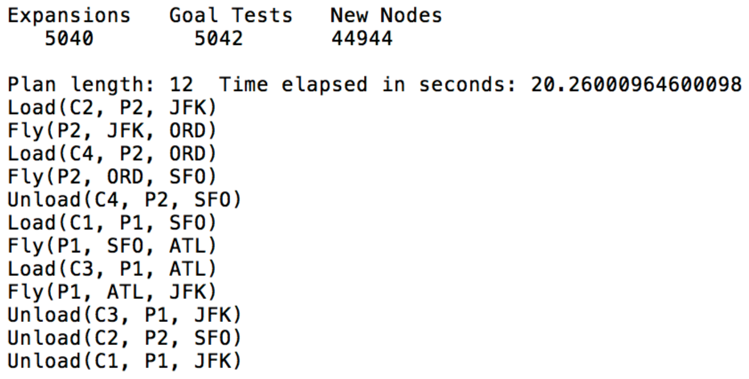
Heuristics Functions Results

1. A-Star Search



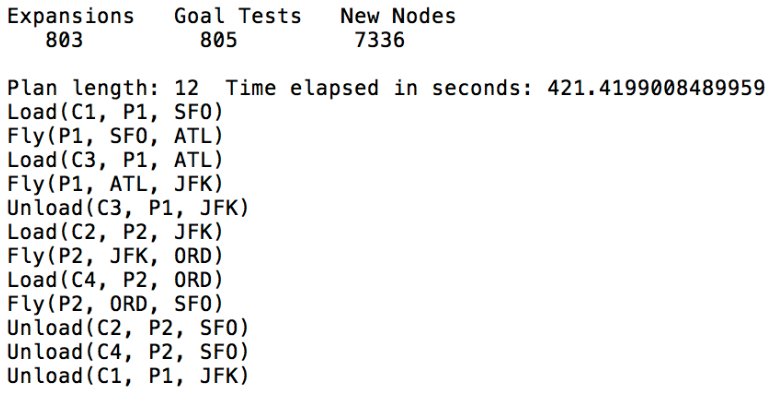
```
Expansions   Goal Tests   New Nodes
  18222        18224       159608

Plan length: 12  Time elapsed in seconds: 62.4554925919947
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(C2, P2, SFO)
Unload(C1, P1, JFK)
```

2. A-Star Search w. Ignore Preconditions



```
Expansions   Goal Tests   New Nodes
   5040         5042        44944

Plan length: 12  Time elapsed in seconds: 20.26000964600098
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

3. A-Star Search w. LevelSum

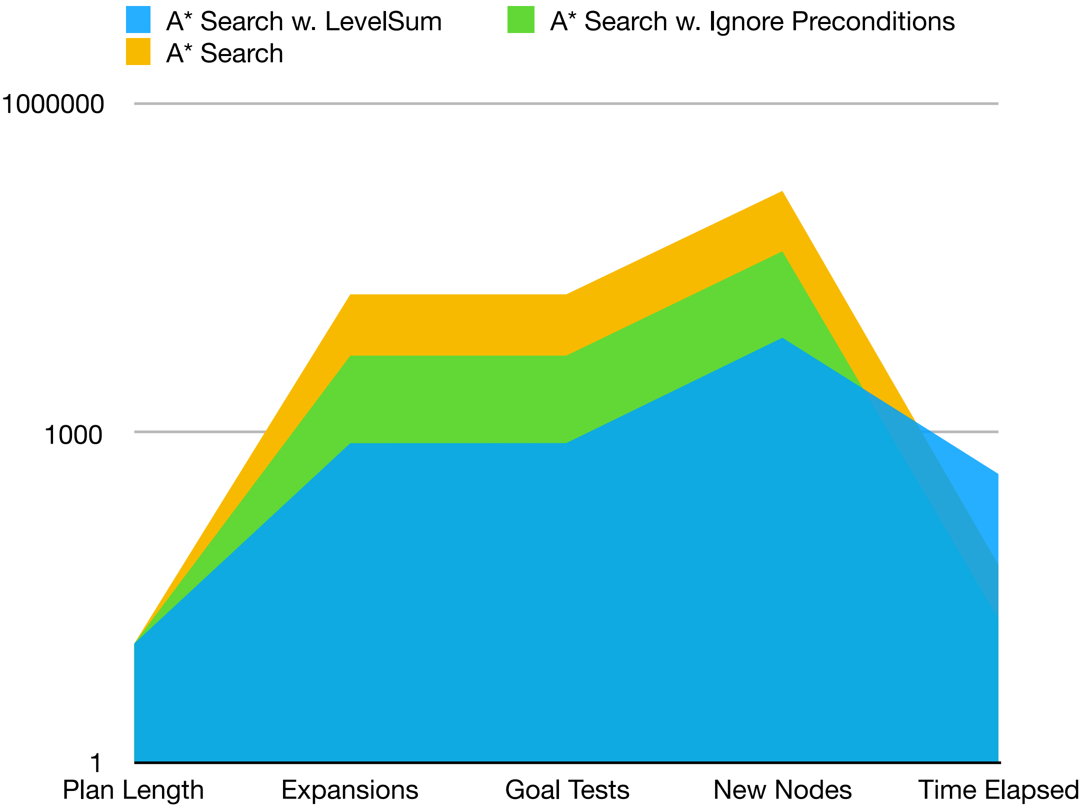


```
Expansions   Goal Tests   New Nodes
   803          805         7336

Plan length: 12  Time elapsed in seconds: 421.4199008489959
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)
Unload(C1, P1, JFK)
```

Problem 2 Heuristics Solutions Comparison

Comparison Table

Search Function	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
A* Search	12	18222	18224	159608	62,4555
A* Search IP	12	5040	5042	44944	20,2600
A* Search LS	12	803	805	7336	421,4199



For the third problem, once again the three search algorithms found the optimal solution (12 actions in this scenario). Time increased considerably due to the increased complexity. **A* Search IP** was able to find a solution in a third of a minute and **A* Search** took a few seconds over a minute to solve it. **A* Search LS** however required over 7 minutes to find a solution, although it required much fewer nodes to do so.

In conclusion, **A* Search with Ignore Preconditions** seems to be the best overall algorithm to solve the three problems taking into account both speed and memory usage.