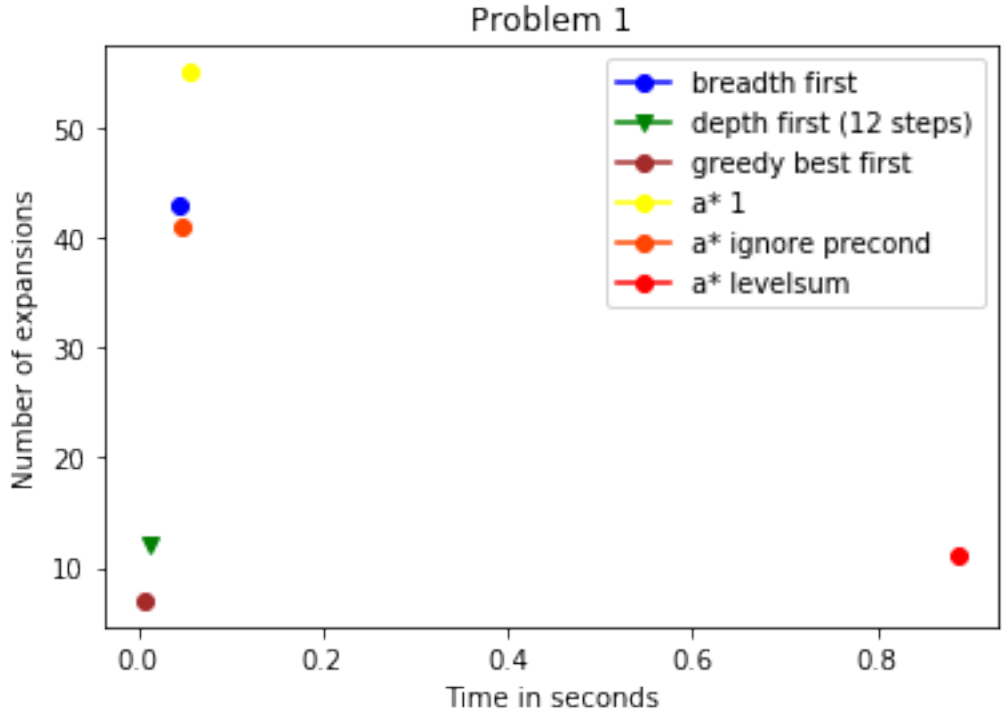
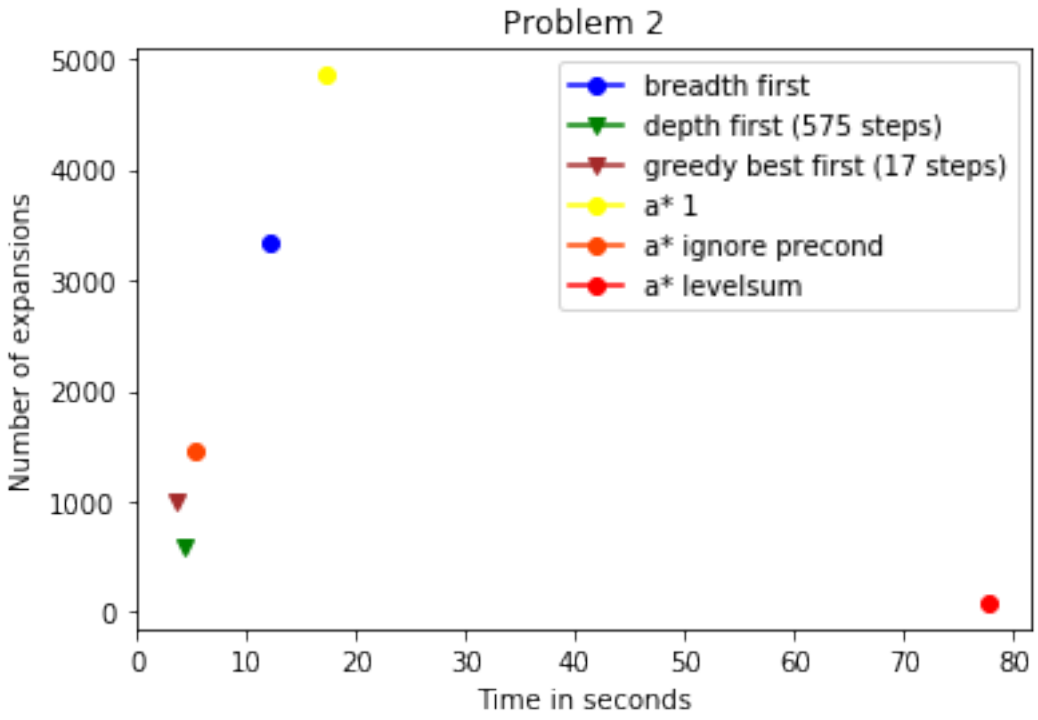
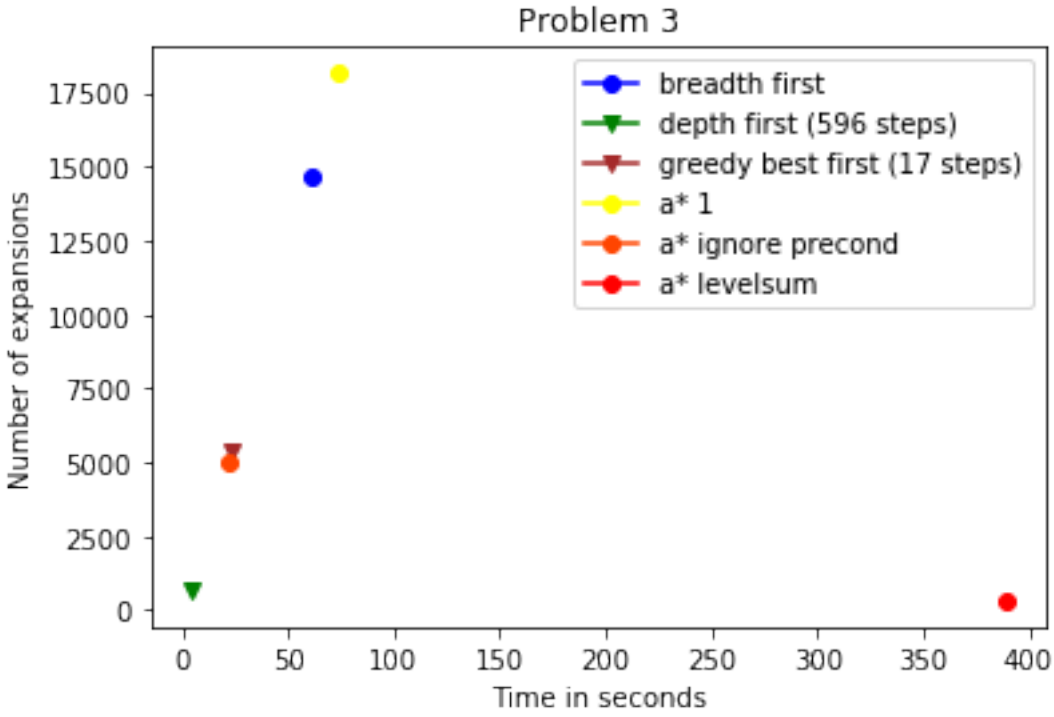
Comparison of search algorithms

Here is the result for a subset of the algorithms. Dots are used when the solution found is optimal. If the solution is not optimal, a triangle is used. The program output that was used to create these diagrams can be found in Appendix 2 at the end of the document.







The optimal solutions for problems 1, 2, and 3 contain 6, 9, and 12 steps, respectively. depth_first and greedy_best_first find solutions that are not optimal for problems 2 and 3.

Interpretation

Breadth first tree search

By definition, breadth first yields optimal solutions. There are many expansions because all states with path length < solution length have to be created. The speed is in the medium range. According to the AIMA book (Section 3.4.1), execution time can be a problem for breadth first. Here this is no problem because the search space is small.

Depth first graph search

Depth first goes all the way down the graph and backtracks only if no more nodes exist. Therefore the path length is almost as high as the number of expansions, and the solutions are far from optimal. As this is depth first *graph* search, the algorithm does not select the same state twice - it just keeps exploring new states until a solution is found. Therefore the algorithm is fast and needs relatively few expansions.

Greedy best first graph search with dummy heuristic

As the heuristic function returns the same value for every node, this algorithm selects nodes randomly. This works surprinsingly well: the algorithm is fast, and the solutions for problem 2 and 3 are not optimal, but much better than those from depth_first.

A* search with dummy heuristic

The A* heuristic estimates total path length as the sum of the path so far and the value of the heuristic function. As the heuristic here always returns 1, this algorithm is equivalent to uniform_cost (see section 3.5.2 in the AIMA book). This means that solutions are optimal. Speed is the second lowest of the algorithms considered, and the number of expansions is the highest. The reason that speed and number of expansions are worse than for breadth first is pointed out in section 3.4.2 of the AIMA book: as uniform cost guarantees to find an optimal solution, the goal test is applied to a node only before it is expanded, not directly after it has been created.

A* search with ignore_preconditions heuristic

As pointed out in section 10.2.3 of the AIMA book, this heuristic does not guarantee to find an optimal solution. But in our case the solutions are optimal. The number of expansions is medium, and the speed is good. Given the data, this algorithm works best overall.

A* search with levelsum heuristic

As mentioned in section 10.3.1 of the AIMA book, this algorithm also does not guarantee optimal solutions. But like ignore_preconditions, optimal solutions are found for our problems. This algorithm is four times slower than the slowest of the others. It is so slow because the planning graph is constructed in each call of the heuristic function.

As the planning graph has polynomial size, this heuristic should work comparatively well for large search spaces. The problems considered here are so simple that the additional effort does not pay off.

Final thoughts

It is not surprising that using an informed heuristic like ignore_preconditions or levelsum yields better results than uninformed searches. But as a general note, I am surprised by the fact that automated planning is so difficult. The problems that we considered here look very simple – a human can solve them in a few seconds. The algorithms we looked at are not better than that. In contrast, the algorithms we implemented for the Sudoku project and the Isolation project outperformed humans.

Appendix 1: optimal plans

Problem 1

```
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 2

```
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)
```

Problem 3

```
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(C1, P1, JFK)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)
```

Appendix 2: program output

```
Problem 1:

1) Solving Air Cargo Problem 1 using breadth_first_search...

Expansions   Goal Tests   New Nodes
    43          56         180

Plan length: 6  Time elapsed in seconds: 0.045388131432586076
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) Solving Air Cargo Problem 1 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
    12          13          48

Plan length: 12  Time elapsed in seconds: 0.011834418901481625
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)

7) Solving Air Cargo Problem 1 using greedy_best_first_graph_search with h_1...

Expansions   Goal Tests   New Nodes
    7           9           28

Plan length: 6  Time elapsed in seconds: 0.007335399038430168
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

8) Solving Air Cargo Problem 1 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
    55          57         224

Plan length: 6  Time elapsed in seconds: 0.05687500058313688
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

9) Solving Air Cargo Problem 1 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
    41          43         170

Plan length: 6  Time elapsed in seconds: 0.04603751298063331
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

10) Solving Air Cargo Problem 1 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
    11          13          50
```

```
Plan length: 6  Time elapsed in seconds: 0.8866269386978214
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

----------------------------------------------------------------
Problem 2:

$ python run_search.py -p 2 -s 1

Solving Air Cargo Problem 2 using breadth_first_search...

Expansions   Goal Tests   New Nodes
   3343        4609       30509

Plan length: 9  Time elapsed in seconds: 12.090223915341547
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)

$ python run_search.py -p 2 -s 3

Solving Air Cargo Problem 2 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
   582         583         5211

Plan length: 575  Time elapsed in seconds: 4.3784089032449485

$ python run_search.py -p 2 -s 7

Solving Air Cargo Problem 2 using greedy_best_first_graph_search with h_1...

Expansions   Goal Tests   New Nodes
   998         1000        8986

Plan length: 17  Time elapsed in seconds: 3.576133921882012
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C2, P2, JFK)
Fly(P2, JFK, ATL)
```

```
Load(C3, P3, ATL)
Fly(P3, ATL, JFK)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Load(C1, P3, JFK)
Fly(P1, JFK, ATL)
Fly(P2, ATL, SFO)
Unload(C2, P2, SFO)
Fly(P2, SFO, ATL)
Fly(P3, JFK, SFO)
Unload(C3, P3, SFO)
Fly(P3, SFO, JFK)
Unload(C1, P3, JFK)

$ python run_search.py -p 2 -s 8

Solving Air Cargo Problem 2 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
   4853        4855       44041

Plan length: 9  Time elapsed in seconds: 17.222562463320674
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(C1, P1, JFK)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)

Solving Air Cargo Problem 2 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
   1450        1452       13303

Plan length: 9  Time elapsed in seconds: 5.394788526473028
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

$ python run_search.py -p 2 -s 10

Solving Air Cargo Problem 2 using astar_search with h_pg_levelsum...
```

```
Expansions   Goal Tests   New Nodes
    86          88         841

Plan length: 9  Time elapsed in seconds: 77.78207557644275
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(C1, P1, JFK)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)

----------------------------------
Problem 3:

$ python run_search.py -p 3 -s 1

Solving Air Cargo Problem 3 using breadth_first_search...

Expansions   Goal Tests   New Nodes
  14663       18098       129631

Plan length: 12  Time elapsed in seconds: 60.87180639099674
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)

$ python run_search.py -p 3 -s 3

Solving Air Cargo Problem 3 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
   627         628         5176

Plan length: 596  Time elapsed in seconds: 4.597075077277337

$ python run_search.py -p 3 -s 7

Solving Air Cargo Problem 3 using greedy_best_first_graph_search with h_1...
```

```
Expansions   Goal Tests   New Nodes
   5399        5401       47691

Plan length: 17  Time elapsed in seconds: 22.71502973066527
Load(C1, P1, SFO)
Fly(P1, SFO, ORD)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, ATL)
Load(C3, 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)

$ python run_search.py -p 3 -s 8

Solving Air Cargo Problem 3 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
  18164       18166       159147

Plan length: 12  Time elapsed in seconds: 73.68106735560823
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C3, P1, ATL)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Unload(C4, P2, SFO)
Unload(C2, P2, SFO)

Solving Air Cargo Problem 3 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
   5038        5040       44924

Plan length: 12  Time elapsed in seconds: 21.994492386094652
Load(C1, P1, SFO)
```

```
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Unload(C2, P2, SFO)

python run_search.py -p 3 -s 10

Expansions   Goal Tests   New Nodes
   316         318         2913

Plan length: 12  Time elapsed in seconds: 388.75686865605076
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(C3, P1, JFK)
Unload(C1, P1, JFK)
Unload(C4, P2, SFO)
Unload(C2, P2, SFO)
```