heuristic_analysis

August 12, 2017

1 Analysis

1.1 Introduction

We solved the Air Cargo Problem using several search algorithms, which we are going to compare here.
The problems were given as follows:
All problems are in the Air Cargo domain. They have the same action schema defined, but different initial states and goals.

1.1.1 Air Cargo Action Schema:

Action(Load(c, p, a),
PRECOND: At(c, a) , At(p, a) , Cargo(c) , Plane(p) , Airport(a)
EFFECT: not 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) not In(c, p))
Action(Fly(p, from, to),
PRECOND: At(p, from) , Plane(p) , Airport(from) , Airport(to)
EFFECT: not At(p, from) , At(p, to))

1.1.2 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))

1.1.3 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))

1.1.4 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))

1.2 Optimal Solutions

The optimal solutions two problems 1 to 3 are as follows:

1.2.1 Air Cargo Problem One

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

1.2.2 Air Cargo Problem Two

Load(C1, P1, SFO)
Load(C2, P2, JFK)
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)

1.2.3 Air Cargo Problem Three

Load(C1, P1, SFO)
Load(C2, P2, JFK)
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)

1.3 Uninformed Searches

Doing breadth-first-search, depth-first-search and depth-limited-search with a limit of 50 steps, we found the following parameters.

1.3.1 Breadth First Search

Solving Air Cargo Problem 1 using breadth_first_search ...
Expansions 43
Goal Tests 56
New Nodes 180
Plan length: 6 Time elapsed in seconds: 0.08521554700564593

Solving Air Cargo Problem 2 using breadth_first_search ...
Expansions 3343
Goal Tests 4609
New Nodes 30509
Plan length: 9 Time elapsed in seconds: 17.348794581994298

Solving Air Cargo Problem 3 using breadth_first_search ...
Expansions 14663
Goal Tests 18098
New Nodes 129631
Plan length: 12 Time elapsed in seconds: 124.13952059501025

1.3.2 Depth First Search

Solving Air Cargo Problem 1 using depth_first_graph_search ...
Expansions 21
Goal Tests 22
New Nodes 84
Plan length: 20 Time elapsed in seconds: 0.043530315975658596

Solving Air Cargo Problem 2 using depth_first_graph_search ...
Expansions 624
Goal Tests 625
New Nodes 5602
Plan length: 619 Time elapsed in seconds: 4.235358421996352

Solving Air Cargo Problem 3 using depth_first_graph_search ...
Expansions 408
Goal Tests 409
New Nodes 3364
Plan length: 392 Time elapsed in seconds: 2.2638161049981136

1.3.3 Depth Limited Search

Solving Air Cargo Problem 1 using depth_limited_search ...
Expansions 101
Goal Tests 271
New Nodes 414
Plan length: 50 Time elapsed in seconds: 0.19056284299585968

Solving Air Cargo Problem 2 using depth_limited_search ...
Expansions 222719
Goal Tests 2053741
New Nodes 2054119
Plan length: 50 Time elapsed in seconds: 1283.8764354590094

Solving Air Cargo Problem 3 using depth_limited_search ...
... took longer than 45 minutes so we did not wait till the end...

1.3.4 Comment

We see that Breadth First Search is optimal while Depth First and Depth Limited Search are not. It is true that Breadth First Search is generally optimal, while Depth First and Depth Limited search are not, see Artificial Intelligence, A Modern Approach, Russel, Norvig, 3rd Edition Chapter 3.4.7. Depth First Search runs a lot faster than Breadth First Search and expands less nodes. Depth Limited Search has trouble finding a solution as it has to find a path that is at most the length of its limit, here 50 steps. It takes longer than and expands more nodes than Depth First Search yet finds a shorter plan for Problem 2. Still it takes longer to run than Breadth First Search for problem 2. It took longer than 45 minutes for Problem 3 so we did not run it to the end.
We conclude that Depth Limited search was not really useful for this group of problems, while Breadth First Search is the choice if you want an optimal solution and Depth First Search is the choice if you want a fast solution.

1.4 Informed Searches

We are going to conduct A-Star-Searches using to different heuristics. First, the ignore-preconditions heuristic and then the level-sum heuristic from a planning graph.

1.4.1 A-Star-Search with ignore-preconditions heuristic

Solving Air Cargo Problem 1 using astar_search with h_ignore_preconditions ...
Expansions 41
Goal Tests 43
New Nodes 170
Plan length: 6 Time elapsed in seconds: 0.10398646301473491

Solving Air Cargo Problem 2 using astar_search with h_ignore_preconditions ...
Expansions 1450
Goal Tests 1452

New Nodes 13303
Plan length: 9 Time elapsed in seconds: 5.797789709002245

Solving Air Cargo Problem 3 using astar_search with h_ignore_preconditions ...
Expansions 5040
Goal Tests 5042
New Nodes 44944
Plan length: 12 Time elapsed in seconds: 22.53121719000046

1.4.2 A-Star-Stearch with level-sum-heuristic

Solving Air Cargo Problem 1 using astar_search with h_pg_levelsum ...
Expansions 11
Goal Tests 13
New Nodes 50
Plan length: 6 Time elapsed in seconds: 0.7509568399982527

Solving Air Cargo Problem 2 using astar_search with h_pg_levelsum ...
Expansions 86
Goal Tests 88
New Nodes 841
Plan length: 9 Time elapsed in seconds: 57.99230600602459

Solving Air Cargo Problem 3 using astar_search with h_pg_levelsum ...
Expansions 316
Goal Tests 318
New Nodes 2912
Plan length: 12 Time elapsed in seconds: 295.33291399199516

1.5 Conclusion

We are now going to compare Breadth First Search and Depth First Search to the informed A-Star searches. We exclude Depth Limited Search as we had found it to be not useful above. We also note that Depth First Search is non-optimal.

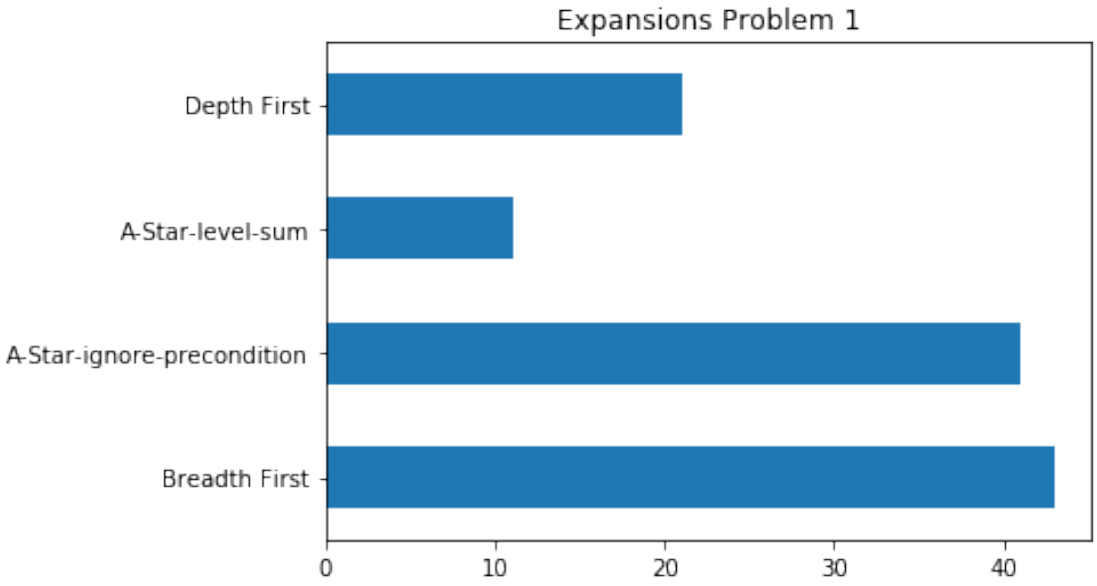
```
In [1]: import pandas as pd
        from matplotlib import pyplot as plt
        %matplotlib inline

        df1=pd.DataFrame(
            data=[[43,0.09],
                  [41,0.10],
                  [11,0.75],
                  [21,0.04]],
            index=['Breadth First','A-Star-ignore-precondition','A-Star-level-sum','Depth First'
            columns=['Expansions','Time Elapsed'])
        df2=pd.DataFrame(
            data=[[3343,17.35],
```

```
                [1450,5.80],
                [86,58.00],
                [624,4.24]],
            index=['Breadth First','A-Star-ignore-precondition','A-Star-level-sum','Depth First'
            columns=['Expansions','Time Elapsed'])
        df3=pd.DataFrame(
            data=[[14663,124.14],
                [5040,22.53],
                [316,295.33],
                [408,2.26]],
            index=['Breadth First','A-Star-ignore-precondition','A-Star-level-sum','Depth First'
            columns=['Expansions','Time Elapsed'])
```

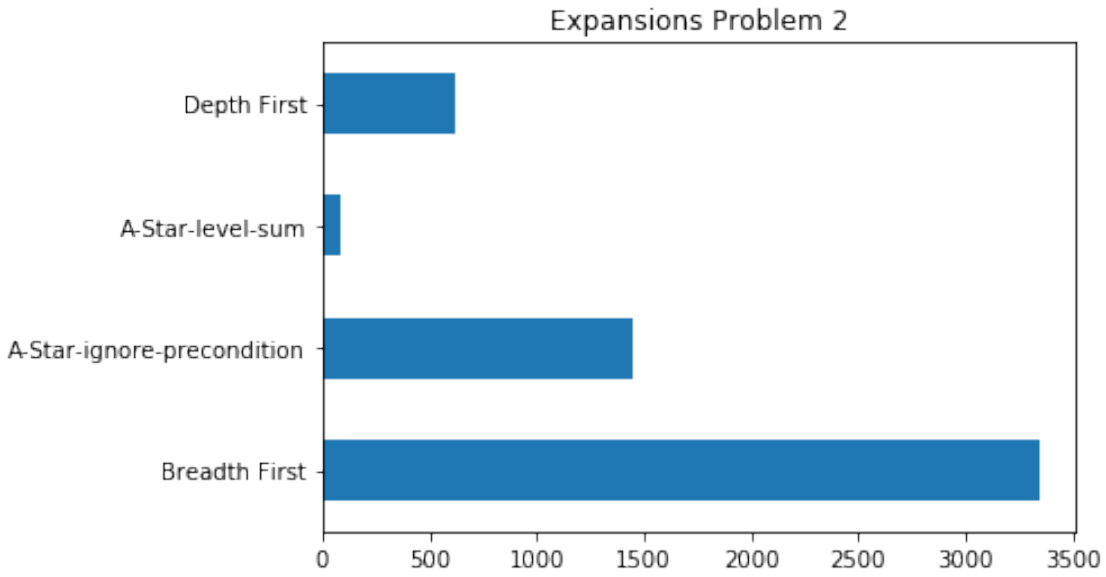
```
In [2]: df1['Expansions'].plot(kind='barh',title='Expansions Problem 1')
```

```
Out[2]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb563ac6828>
```



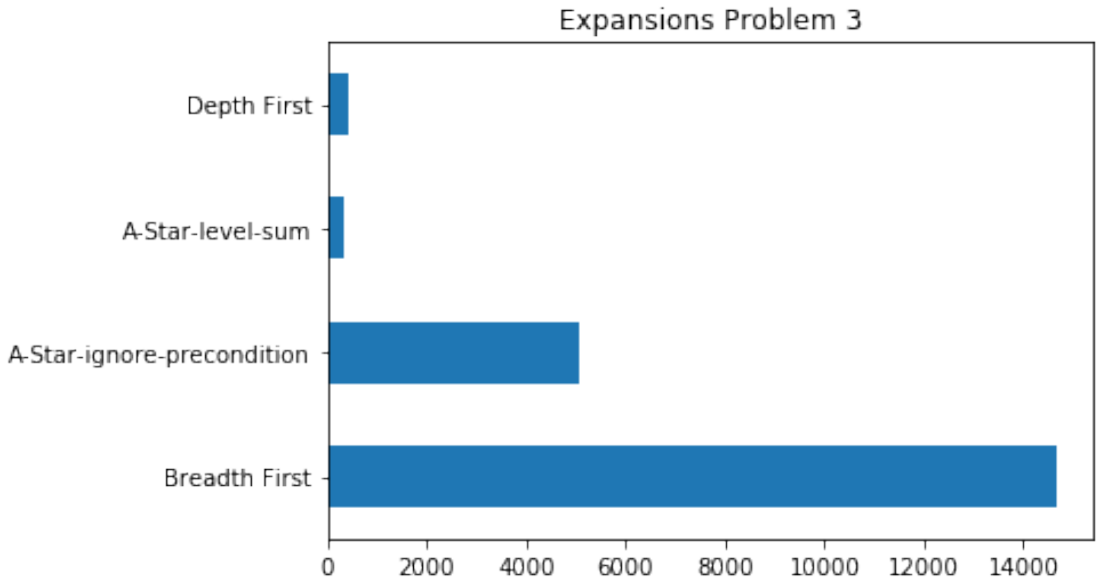

```
In [3]: df2['Expansions'].plot(kind='barh',title='Expansions Problem 2')
```

```
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5884d9160>
```



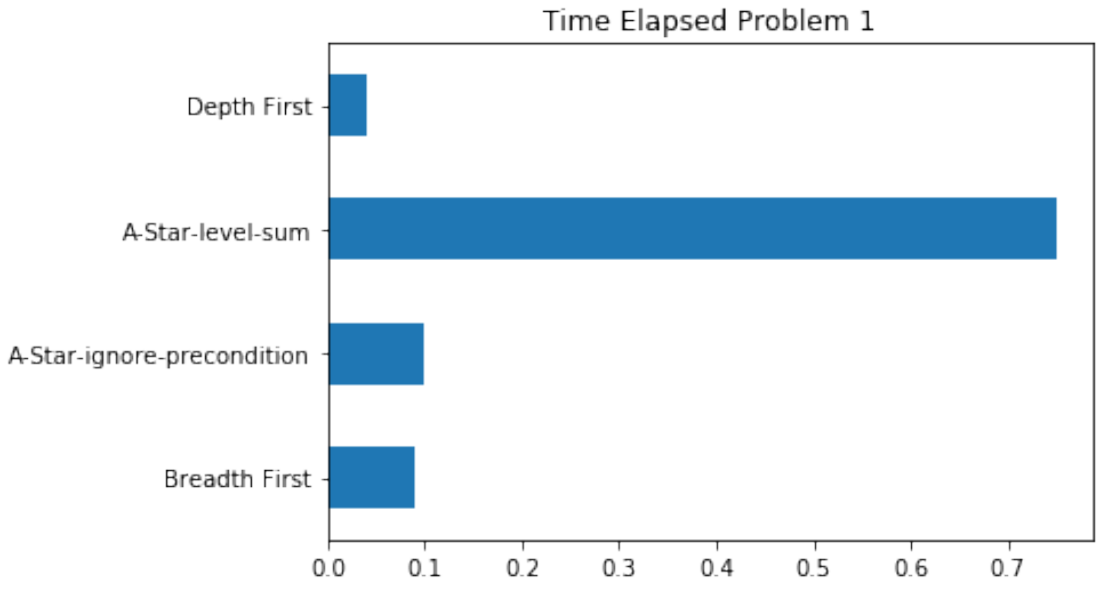

```
In [4]: df3['Expansions'].plot(kind='barh',title='Expansions Problem 3')
```

```
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5613905c0>
```



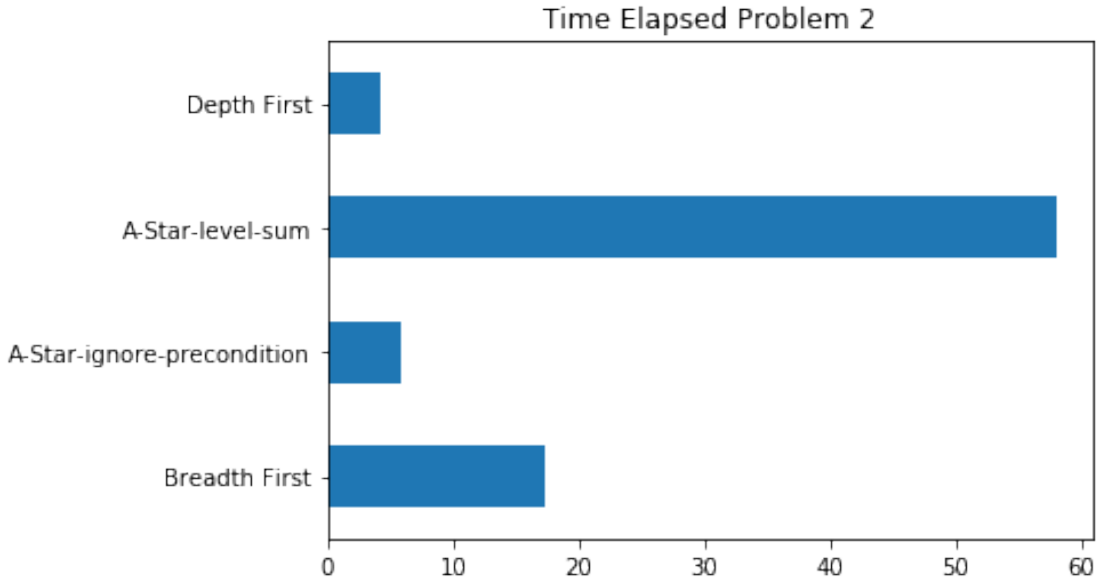

```
In [5]: df1['Time Elapsed'].plot(kind='barh',title='Time Elapsed Problem 1')
```

```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5612b97f0>
```



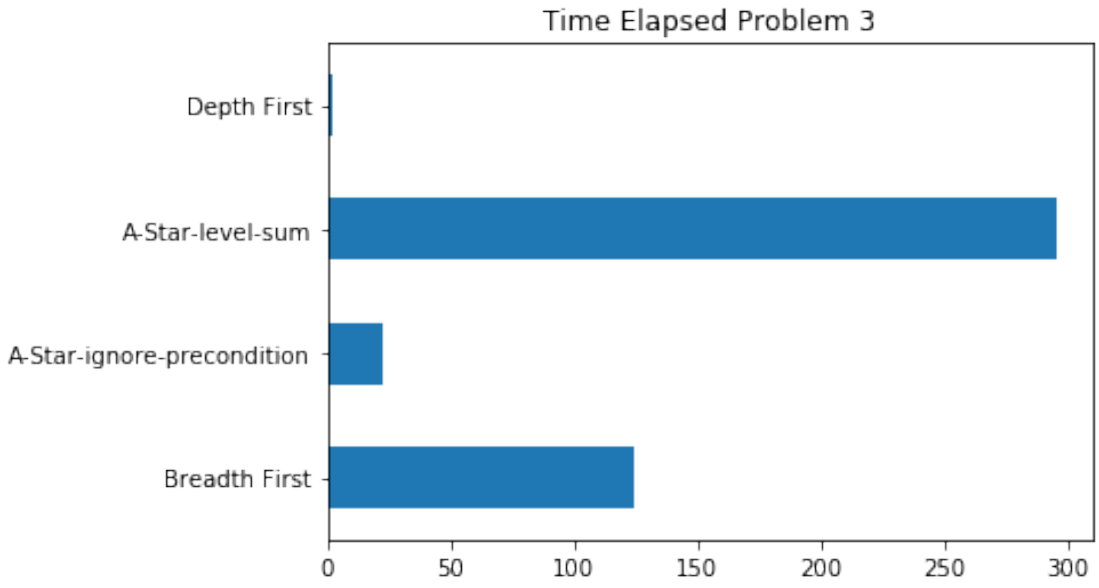

```
In [6]: df2['Time Elapsed'].plot(kind='barh',title='Time Elapsed Problem 2')
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb56124f6d8>
```




```
In [7]: df3['Time Elapsed'].plot(kind='barh',title='Time Elapsed Problem 3')
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb56116b860>
```



We can see that Depth First Search generally is the fastest of the algorithms, yet as stated above it is not optimal while Breadth First Search is. The ignore-precondition heuristic is admissible, see Artificial Intelligence, A Modern Approach, Russel, Norvig, 3rd Edition (AIMA) Chapter 10.2.3., thus the tree version of A-Star Search with the ignore preconditions heuristic is optimal, see AIMA Chapter 3.5.2. The level-sum heuristic can in general be inadmissible yet "works well in practice for problems that are largely decomposable", see AIMA Chapter 10.3.1.
In our case both heuristics lead to optimal results.
Concerning nodes expanded A-Star Search with the level-sum heuristic expands the least nodes yet it is the slowest of the algorithms because it has to compute the planning graph in every step. A-Star Search with the ignore-preconditions heuristic still expands less nodes than Breadth First Search and it is faster than Breadth First Search as the heuristic is easy to compute. Therefore we conclude that A-Star Search with the ignore-preconditions heuristic is the best option we have for our problem.