Challenge Introduction

In this chapter, we will briefly introduce the defined problems and state one of possibly many optimal solutions.

Problem Briefing

In this project, you will define a group of problems in classical PDDL (Planning Domain Definition Language) for the air cargo domain. The task is to transport cargo located at defined airports, over planes, located at defined airports, to defined airports. In total, three tasks (problems) are defined that are:

```
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))
```

Figure 1: Project 1 Definition

```
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))
```

Figure 2: Project 2 Definition

```
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))
```

Figure 3: Project 3 Definition

Optimal Path

The optimal paths for the three aforementioned problems are stated here. Please mind that there exist many possible combinations that all constitute an optimal path:

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

Figure 4: Project 1 Solution

```
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)
```

Figure 5: Project 2 Solution

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

Figure 6: Project 3 Solution

Benchmark Comparison

We compared ten different algorithms on the three defined problems. We defined a timeout threshold of ten minutes, after which we stopped the search. We compared algorithm performance based on elapsed time, number of expansions, number of goals tests, number of new nodes and length of solution.

Problem 1

On problem 1, all algorithms can find a solution and all algorithms but the depth first approaches can find an optimal path length. All algorithms but the recursive best first search converge in less than one second. With regards the expansions, goal tests and new nodes, the a-star search versions and greedy best first graph search are the most efficient.

Table 1: Algorithm Performance on Problem 1

Algorithm	**Elapsed Time**	**Expansion**	**Goal Test**	**New Nodes**	**Solution Length**
A-Star Search (H1)	0.04	55	57	224	6
A-Star Search (Ignore Preconditions)	0.03	41	43	170	6
A-Star Search (Level Sum)	0.52	11	13	50	6
Breadth First Search	0.03	43	56	180	6
Breadth First Tree Search	0.87	1458	1459	5960	6
Depth First Graph Search	0.01	21	22	84	20
Depth Limited Search	0.09	101	271	414	50
Greedy Best First Graph Search	0.01	7	9	28	6
Recursive Best First Search	2.45	4229	4230	17023	6
Uniform Cost Search	0.04	55	57	224	6

Problem 2

On problem 2, only 70% of all algorithms can solve the problem, where all failures are caused by timeout exceptions. Amongst the succeeding algorithms, only five are able to find an optimal path, namely a-start search approaches, breadth first search and uniform cost search. With regards to processing time, all algorithms that did not timeout converge in less than 140 seconds with breadth first search being the fastest (7 seconds). Speaking of expansions, goal tests and new nodes, a-star search with level sum heuristic is the most efficient.

Table 2: Algorithm Performance on Problem 2

Algorithm	**Elapsed Time**	**Expansion**	**Goal Test**	**New Nodes**	**Solution Length**
A-Star Search (H1)	34.39	4852	4854	44030	9
A-Star Search (Ignore Preconditions)	11.75	1450	1452	13303	9
A-Star Search (Level Sum)	136.50	86	88	841	9
Breadth First Search	7.11	3343	4609	30509	9
Breadth First Tree Search	600.00	198697	198698	1791415	
Depth First Graph Search	12.06	624	625	5602	619
Depth Limited Search	600.00	50329	461397	461766	
Greedy Best First Graph Search	7.63	990	992	8910	17
Recursive Best First Search	600.00	90514	90514	802658	
Uniform Cost Search	26.95	4852	4854	44030	9

Problem 3

Finally, only six algorithms are able to solve problem 3, four of which with an optimal solution length. The fastest converging algorithm is the depth first graph search. However, the fastest algorithm that returns an optimal solution is a-star search with ignore preconditions heuristic. It is also a-star search with ignore preconditions heuristic that is the most efficient when it comes to expansions, goal tests and new nodes.

Table 3: Algorithm Performance on Problem 3

Algorithm	**Elapsed Time**	**Expansion**	**Goal Test**	**New Nodes**	**Solution Length**
A-Star Search (H1)	212.59	18235	18237	159716	12
A-Star Search (Ignore Preconditions)	76.56	5040	5042	44944	12
A-Star Search (Level Sum)	600.00	172	173	1567	
Breadth First Search	136.37	14663	18098	129631	12
Breadth First Tree Search	600.00	73946	73946	623618	
Depth First Graph Search	6.27	408	409	3364	392
Depth Limited Search	600.00	35217	311305	311635	
Greedy Best First Graph Search	80.34	5614	5616	49429	22
Recursive Best First Search	600.00	57789	57789	476432	
Uniform Cost Search	207.94	18235	18237	159716	12

Conclusion

Amongst the uninformed search algorithms, breadth first search does a solid job in finding the optimal path in considerable time. Also, it is apparent that intelligent heuristics can improve search efficiency by a wide margin, as illustrated with the a-star search approaches. However, calculating the heuristics can take substantial time, for example the level sum heuristic. It is important to carefully compare the reduction in terms of expansions, goal tests and new nodes with the increase in processing time per evaluation. In our examples, the a-star search with ignore preconditions heuristic shows the perfect balance with regards to search efficiency and evaluation cost.