Implementing a Planning Search

The goal of this project is to solve deterministic logistics planning problems for an Air Cargo Transport System using a planning search agent.

The first step was to implement the different actions, initial states and goals of the problem described bellow then use **uninformed planning search algorithms (non-heuristic)** and compare results obtained. The algorithms used were:

- **Breadth First Search**: Expands the shallowest nodes first
- **Breadth First Tree Search**: Same as above but with a depth boundary
- **Depth First Graph Search**: Expands the deepest unexpanded node first
- **Depth Limited Search**: Same as above but with a depth boundary
- **Uniform Cost Search**: Expands the node with the lowest path cost

We also used **informed planning search algorithms (heuristic)**, however with a *bad* heuristic `h = 1`. The algorithms used were:

- **Recursive Best first Search**: Expands the node that is closest to the goal while keeping track of the best *alternative path*
- **Greedy Best First Graph Search**: Expands the node that is closest to the goal

Then we developed **domain-independent Heuristics** and **Mutual Exclusion Relations (Mutex)** with a **planning graph** and used **A* Search** in an attempt to find more efficient methodologies.

- **A* Search**: Evaluates nodes by combining the cost to reach the node and the cost to get from the node to the goal

The heuristics and Mutex implemented were:

- **Ignore Preconditions Heuristic**: Drops all preconditions from actions
- **Inconsistent Effects Mutex**: Check if one action negates an effect of another action
- **Interference Mutex**: Check if an effect of one action is the negation of a precondition of another action
- **Competing Needs Mutex**: Check if a precondition of an action is mutually exclusive with a precondition of another action
- **Negation Mutex**: Check if a node (state) is the negation of another node.
- **Inconsistent Support Mutex**: Check if two nodes are in conflict, in other words if they are at the same place at the same time
- **Level Sum Heuristic**: The sum of the level costs of the goals

Finally, we compared the results between each methodologies in order to determine optimal solutions for each problems. See the **Result Analysis** section below.

Table of Contents

Problem Description

The Air Transport System follows the action schema below:

```
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))
```

However, each problem follow different initial states and goals highlighted below:

- Problem 1:

```
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))
```

- Problem 2:

```
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))
```

- Problem 3:

```
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 Analysis

Optimality

Optimality: Refers to the minimum number of actions to reach the goal for each problem. We know that by definition, **Uniform Cost Search** leads to an optimal solution because it always expands the *best* node (smallest cost).

The effectiveness of each algorithm is based on two limiting factors, **Space**, and **Time** and whether they obtain an optimal solution (the smallest path to the solution).**Time** is measured in seconds while **Space** is measured in terms of expanded nodes (constrained by memory).

The results were as follows:

- The **optimal path** for problem 1, 2, and 3 was **6, 9, and 12 actions** respectively with the following paths:

Problem 1	Problem 2	Problem 3
Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C1, P1, SFO)
Load(C2, P2, JFK)	Load(C2, P2, JFK)	Load(C2, P2, JFK)
Fly(P1, SFO, JFK)	Load(C3, P3, ATL)	Fly(P1, SFO, ATL)
Fly(P2, JFK, SFO)	Fly(P1, SFO, JFK)	Load(C3, P1, ATL)
Unload(C1, P1, JFK)	Fly(P2, JFK, SFO)	Fly(P2, JFK, ORD)
Unload(C2, P2, SFO)	Fly(P3, ATL, SFO)	Load(C4, P2, ORD)
	Unload(C3, P3, SFO)	Fly(P2, ORD, SFO)
	Unload(C1, P1, JFK)	Fly(P1, ATL, JFK)
	Unload(C2, P2, SFO)	Unload(C4, P2, SFO)
		Unload(C3, P1, JFK)
		Unload(C1, P1, JFK)
		Unload(C2, P2, SFO)

Uninformed Planning Search (Non-Heuristic) Methodologies

These algorithms are *uninformed* in a sense that they have no additional information about each state beyond the problem description. All they can do is develop successors at each node and distinguish between a goal state from a non-goal state.

Note: For the following analysis

- If ~ then search time elapsed over 30min.
- If **Bold** then best search method for the current analysis
- If * then optimal path length

Problem 1 Analysis:

Search Method	Space (Expensions)	Time (s)	Path Length
Breadth First Search	43	0.030	6*
Breadth First Tree Search	1458	0.922	6*
Depth First Graph Search	12	0.009	12
Depth Limited Search	101	0.088	50
Uniform Cost Search	55	0.040	6*
Recursive Best first Search	4229	2.890	6*
Greedy Best First Graph Search	**7**	**0.006**	**6***

Problem 2 Analysis:

Search Method	Space (Expensions)	Time (s)	Path Length
Breadth First Search	**3343**	**7.885**	**9***
Breadth First Tree Search	~	~	~
Depth First Graph Search	582	2.970	575
Depth Limited Search	222719	950.341	50
Uniform Cost Search	4852	13.575	9*
Recursive Best first Search	~	~	~
Greedy Best First Graph Search	990	2.340	21

Problem 3 Analysis:

Search Method	Space (Expensions)	Time (s)	Path Length
Breadth First Search	**14663**	**42.180**	**12***
Breadth First Tree Search	~	~	~
Depth First Graph Search	627	3.098	596
Depth Limited Search	~	~	~
Uniform Cost Search	18235	57.988	12*
Recursive Best first Search	~	~	~
Greedy Best First Graph Search	5614	16.242	22

Summary:

If optimality is a requirement, we can easily tell that **Breadth First Search** is a superior method as it uses the least amount of time and memory to generate and optimal solution. However, **Depth First Graph Search** is much faster and uses a lot less memory to reach the goal but is not optimal.

Breadth First Search expands the *shallowest* node and therefore will seak the optimal solution but needs to keep all nodes in a previous state in memory and in order to do so **Space** grows exponentially $O(b^{d})$ (with *b*: branching factor and *d*: depth).

Depth First Graph Search expands the *deepest* unexpanded node therefore does not need to keep all previous states in memory and can just discard a *dead-end* branch which leads to **Space** growing linearly $O(bm)$ (with *b*: branching factor and *d*: depth) and vastly reducing memory requirements. It will end on the first solution found rather than the optimal solution therefore it is complete but not optimal.

Informed Planning A* Search (Heuristic) Methodologies

Algorithms are *informed* because they have problem-specific knowledge beyond the problem description. This additional knowledge is interpreted as a *Heuristic Function* which can be either manually written knowing specific information about the workings of a problem or can be generated by **relaxing** a problem. A **relaxed problem** is simply a problem that has *fewer* restriction on the *actions* than the problem description.

The algorithm chosen for this task was **A* Search** because it is a widely known best-first search algorithm that is both *complete* and *optimal* provided that the **heuristic function** is *admissible* (a heuristic that never overestimates the cost to reach the goal).

A **planning graph** was used in order to provide better heursitic estimates. It constrains the initial state with a set of preconditions and asks whether a goal state can be reached from the initial state. It does so by using **mutual exclusion relations** which prevents the creation of a new *branch* unless it validates those relations. This prevents unecessary branching.

Note: For the following analysis

- If ~ then search time elapsed over 30min.
- If **Bold** then best search method for the current analysis
- If * then optimal path length

Problem 1 Analysis:

Search Method	Space (Expensions)	Time (s)	Path Length
Uninformed A* Search	55	0.039	6*
A* Search with Ignore Preconditions Heuristic	**41**	**0.040**	**6***
A* Search with Level Sum Heuristic	11	0.934	6*

Problem 2 Analysis:

Search Method	Space (Expensions)	Time (s)	Path Length

Search Method	Space (Expensions)	Time (s)	Path Length
Uninformed A* Search	4852	11.681	9*
A* Search with Ignore Preconditions Heuristic	**1450**	**4.267**	**9***
A* Search with Level Sum Heuristic	86	148.773	9*

Problem 3 Analysis:

Search Method	Space (Expensions)	Time (s)	Path Length
Uninformed A* Search	18235	52.066	12*
A* Search with Ignore Preconditions Heuristic	**5040**	**16.679**	**12***
A* Search with Level Sum Heuristic	318	916.156	12*

Summary:

All three methodologies lead to optimal solutions, therefore the remaining criteria were **Space** and **Time**. For our purposes, **Time** was the limiting factor was memory usage wasn't an issue. The **A* Search with Ignore Preconditions Heuristic (A*IPH)** lead to significantly faster solutions but did not have the least memory usage. If memory was the limiting factor then the **A* Search with Level Sum Heuristic (A*LSH)** would have been the more adequate methodology.

This is expected as the **A*IPH** is not constrained by preconditions and seeks the lowest cost solution. However, needs to check more nodes to do so. The *A\LSH** method discards more nodes than the previous method but seem to take a lot of time to do so. I wonder if a better implementation of the methodology could reduce the computing time.

Informed or Uniformed Planning Search?

Note: For the following analysis

- If ~ then search time elapsed over 30min.
- If **Bold** then best search method for the current analysis
- If * then optimal path length

Problem 1 Analysis:

Search Method	Space (Expensions)	Time (s)	Path Length
Greedy Best First Graph Search	**7**	**0.006**	**6***
A* Search with Ignore Preconditions Heuristic	41	0.040	6*

Problem 2 Analysis:

Search Method	Space (Expensions)	Time (s)	Path Length
Breadth First Search	3343	7.885	9*

Search Method	Space (Expensions)	Time (s)	Path Length
A* Search with Ignore Preconditions Heuristic	**1450**	**4.267**	**9***

Problem 3 Analysis:

Search Method	Space (Expensions)	Time (s)	Path Length
Breadth First Search	14663	42.180	12*
A* Search with Ignore Preconditions Heuristic	**5040**	**16.679**	**12***

According to the previous analysis, **A* Search with Ignore Preconditions Heuristic** seem to be the superior approach. It is both faster, less memory demanding and optimal. However, for *problem 1*, **Greedy Best First Graph Search** was able to provide even better results while still being optimal. While **Greedy Best First Graph Search** remained faster and less memory intensive than **A* Search with Ignore Preconditions Heuristic**, it no longer provided optimal solution for *problem 2* and *problem 3*.

We can conclude that heuristics are very much worth implementing in order to both reduce the search time and search space of a problem. However, all heuristics are not made equal and **informed planning search methodologies** could lead to worst performance than **uninformed planning search methodologies** if the heuristic function wasn't carefully designed. Nevertheless, **relaxed problems** that allows for automatically generated heuristics show competitive performance without the need for heuristics designed based on prior human knowledge of the workings of a problem.

Result Metrics for Different Search Algorithms

Search type: *breadth_first_search*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes
43	56	180

Plan length: 6 Time elapsed in seconds: 0.029817941814027055

- Problem: *Air Cargo Problem 2*

Expansions	Goal Tests	New Nodes
3343	4609	30509

Plan length: 9 Time elapsed in seconds: 7.884906920642688

- Problem: *Air Cargo Problem 3*

Expansions	Goal Tests	New Nodes
14663	18098	129631

Plan length: 12 Time elapsed in seconds: 42.180175567570174

Search type: *breadth_first_tree_search*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes
1458	1459	5960

Plan length: 6 Time elapsed in seconds: 0.9222024747407813

- Problem: *Air Cargo Problem 2*

over 10 min

- Problem: *Air Cargo Problem 3*

over 10 min

Search type: *depth_first_graph_search*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes
12	13	48

Plan length: 12 Time elapsed in seconds: 0.008644714867890105

- Problem: *Air Cargo Problem 2*

Expansions	Goal Tests	New Nodes
582	583	5211

Plan length: 575 Time elapsed in seconds: 2.9701220228217453

- Problem: *Air Cargo Problem 3*

Expansions	Goal Tests	New Nodes
627	628	5176

Plan length: 596 Time elapsed in seconds: 3.097800628974275

Search type: *depth_limited_search*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes
101	271	414

Plan length: 50 Time elapsed in seconds: 0.08760518932095472

- Problem: *Air Cargo Problem 2*

Expansions	Goal Tests	New Nodes
222719	2053741	2054119

Plan length: 50 Time elapsed in seconds: 950.3409759484738

- Problem: *Air Cargo Problem 3*

over 10 min

Search type: *uniform_cost_search*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes
55	57	224

Plan length: 6 Time elapsed in seconds: 0.03984384764006269

- Problem: *Air Cargo Problem 2*

Expansions	Goal Tests	New Nodes
4852	4854	44030

Plan length: 9 Time elapsed in seconds: 13.575494848097314

- Problem: *Air Cargo Problem 3*

Expansions	Goal Tests	New Nodes
18235	18237	159716

Plan length: 12 Time elapsed in seconds: 57.987703700179615

Search type: *recursive_best_first_search h_1*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes

Expansions	Goal Tests	New Nodes
4229	4230	17029

Plan length: 6 Time elapsed in seconds: 2.889719566884493

- Problem: *Air Cargo Problem 2*
- Problem: *Air Cargo Problem 3*

Search type: *greedy_best_first_graph_search h_1*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes
7	9	28

Plan length: 6 Time elapsed in seconds: 0.005889129407926707

- Problem: *Air Cargo Problem 2*

Expansions	Goal Tests	New Nodes
990	992	8910

Plan length: 21 Time elapsed in seconds: 2.3396260967309424

- Problem: *Air Cargo Problem 3*

Expansions	Goal Tests	New Nodes
5614	5616	49429

Plan length: 22 Time elapsed in seconds: 16.242247980791703

Search type: *astar_search h_1*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes
55	57	224

Plan length: 6 Time elapsed in seconds: 0.038655600525441584

- Problem: *Air Cargo Problem 2*

Expansions	Goal Tests	New Nodes

Expansions	Goal Tests	New Nodes
4852	4854	44030

Plan length: 9 Time elapsed in seconds: 11.681007189518052

- Problem: *Air Cargo Problem 3*

Expansions	Goal Tests	New Nodes
18235	18237	159716

Plan length: 12 Time elapsed in seconds: 52.06644631044929

Search type: *astar_search h_ignore_preconditions*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes
41	1459	170

Plan length: 6 Time elapsed in seconds: 0.039120780202979946

- Problem: *Air Cargo Problem 2*

Expansions	Goal Tests	New Nodes
1450	1452	13303

Plan length: 9 Time elapsed in seconds: 4.267498326372629

- Problem: *Air Cargo Problem 3*

Expansions	Goal Tests	New Nodes
5040	5042	44944

Plan length: 12 Time elapsed in seconds: 16.678899910475568

Search type: *astar_search h_pg_levelsum*

- Problem: *Air Cargo Problem 1*

Expansions	Goal Tests	New Nodes
11	13	50

Plan length: 6 Time elapsed in seconds: 0.9345308689382689

- Problem: *Air Cargo Problem 2*

Expansions	Goal Tests	New Nodes
86	88	841

Plan length: 9 Time elapsed in seconds: 148.77331024048505

- Problem: *Air Cargo Problem 3*

Expansions	Goal Tests	New Nodes
318	320	2934

Plan length: 12 Time elapsed in seconds: 916.1556471566836