Heuristic Analysis of Planning Search

Synopsis:

The project defines a group of problems in classical PDDL (Planning Domain Definition Language) for which students were required to implement a planning search agent to solve the deterministic logistics planning problems for an Air Cargo transport system. The project required students to implement **uninformed non-heuristic search methods**, then implement **domain-independent heuristic search methods**, and finally compare the performance results.

Planning Problems:

Three planning problems were given in the Air Cargo domain that utilize the same **action schema**.

• 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))
```

• 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))
```

• 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))
```

Uninformed Non-Heuristic Search Strategies:

According to [1] section 3.4, search strategies that are considered to be an **uninformed search** (also known as **blind search**) are strategies that have no additional information about states beyond that provided in the problem definition. All they can do is generate successors and distinguish a goal state from a non-goal state.

The uninformed non-heuristic planning part of the experiment involved using ***Breadth-First Search***, ***Breadth-First Tree Search, Depth-First Graph Search***, ***Depth-Limited Search,*** *and* ***Uniform-Cost Search***. The performance of these strategies are judged on the basis of **optimality** (an optimal solution has the lowest path cost among all solutions), **time complexity** (execution time; elapsed time), and **space complexity** (memory usage; measured in search node expansions). The amount of goal tests and new nodes, while they do not change the results of the analysis, are reported for the purpose of stating all results obtained during experimentation. For Problems 2 and 3, data collection was stopped for Breadth-First Tree Search and Depth-Limited Search because their execution times exceeded 10 minutes (time limit imposed).

Problem 1 Results:

Search Type	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
Breadth-First Search	43	56	180	6	0.028	Yes
Breadth-First Tree Search	1458	1459	5960	6	0.931	Yes
Depth-First Graph Search	12	13	48	12	0.008	No
Depth-Limited Search	101	271	414	50	0.086	No
Uniform-Cost Search	55	57	224	6	0.035	Yes

Problem 2 Results:

Search Type	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
Breadth-First Search	3343	4609	30509	9	13.438	Yes
Depth-First Graph Search	582	583	5211	575	3.067	No
Uniform-Cost Search	4761	4763	43206	9	10.654	Yes

Problem 3 Results:

Search Type	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
Breadth-First Search	14491	17947	128184	12	117.049	Yes
Depth-First Graph Search	1948	1949	16253	1878	19.326	No
Uniform-Cost Search	17783	17785	155920	12	45.914	Yes

Analysis:

In all three problems, ***Breadth-First Search*** and ***Uniform-Cost Search*** are the only uninformed non-heuristic strategies that yield an optimal action plan that was executed below the 10-minute time limit. Regarding the memory usage and execution time, ***Depth-First Graph Search*** uses the least amount of memory and in the shortest amount of time. However, it does not yield an optimal action plan in any of the given problems since its *plan length* has consistently been much longer than the others.

Comparing the ratios of *execution time* and *search nodes expanded* for ***Breadth-First Search*** vs ***Uniform-Cost Search*** for each of the problems:

		BFS : UCS	Percentage
Execution Time (seconds)	Problem 1	0.028 : 0.035	80%
	Problem 2	13.438 : 10.654	126%
	Problem 3	117.049 : 45.914	255%

		BFS : UCS	Percentage
Search Nodes Expanded	Problem 1	43 : 55	78%
	Problem 2	3343 : 4761	70%
	Problem 3	14491 : 17783	81%

As can be seen from the tables above, the **execution time** for ***Breadth-First Search*** rises much more drastically when compared to ***Uniform-Cost Search*** as the **space complexity** increases in size. And given that the ratio of node expansion is consistently around 80%, I would say that the ***Uniform-Cost Search*** is the best of these uninformed non-heuristic search strategies.

Informed Heuristic Search Strategies:

According to [1] section 3.5, search strategies that are considered to be an **informed search** are ones that use problem-specific knowledge beyond the definition of the problem itself, and can find solutions more efficiently than an uninformed strategy.

The informed heuristic planning part of the experiment involved using three different variations of the A* Search algorithm, ***A* Search h_1***, ***A* Search h_ignore_preconditions***, and ***A* Search - h_pg_levelsum***. The same performance metrics that were used for evaluating the uninformed searches will be used for evaluate the performance of these informed heuristic searches.

Problem 1 Results:

Search Type	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
A* Search h_1	55	57	224	6	0.043	Yes
A* Search h_ignore_preconditions	41	43	170	6	0.044	Yes
A* Search h_pg_levelsum	11	13	50	6	0.857	Yes

Problem 2 Results:

Search Type	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
A* Search h_1	4761	4763	43206	9	10.649	Yes
A* Search h_ignore_preconditions	1450	1452	13303	9	3.943	Yes
A* Search h_pg_levelsum	86	88	841	9	153.428	Yes

Problem 3 Results:

Search Type	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
A* Search h_1	17783	17785	155920	12	47.269	Yes
A* Search h_ignore_preconditions	5003	5005	44586	12	15.540	Yes
A* Search h_pg_levelsum	311	313	2863	12	861.196	Yes

Analysis:

All three of the informed heuristic searches yield an optimal action plan, however, only h_1 and h_ignore_preconditions heuristics returned results within 10 minutes. If A* Search h_pg_levelsum is allowed to run to completion (in this case, just over 14 minutes), it would yield a correct result with the least amount of memory usage.

Comparing the ratios of *execution time* and *search nodes expanded* for ***A* Search h_1*** vs ***A* Search h_ignore_preconditions*** for each of the problems:

		A* h1 : A* h_ignore	Percentage
Execution Time (seconds)	Problem 1	0.043 : 0.044	98%
	Problem 2	10.649 : 3.943	270%
	Problem 3	47.269 : 15.540	304%

		A* h1 : A* h_ignore	Percentage
Search Nodes Expanded	Problem 1	55 : 41	134%
	Problem 2	4761 : 1450	328%
	Problem 3	17783 : 5003	355%

As can be seen from the tables above, the **execution time** for ***A* Search h_1*** rises much more drastically when compared to ***A* Search h_ignore_preconditions*** as the **space complexity** increases in size. I would say that the ***A* Search h_ignore_preconditions*** is the best of these informed heuristic search strategies as it is the fastest while also expanding the least number of search nodes.

Uninformed vs Informed Search Strategies:

As seen from the data presented earlier, the best uninformed non-heuristic search method was determined to be ***Uniform-Cost Search***, while the best informed heuristic search method was determined to be ***A* Search h_ignore_preconditions***, but how do these compare in performance?

Problem 1 Results:

Search Type	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
Uniform-Cost Search	55	57	224	6	**0.035**	Yes
A* Search h_ignore_preconditions	**41**	43	170	6	0.044	Yes

Problem 2 Results:

Search Type	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
Uniform-Cost Search	4761	4763	43206	9	10.654	Yes
A* Search h_ignore_preconditions	**1450**	1452	13303	9	**3.943**	Yes

Problem 3 Results:

Search Type	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
Uniform-Cost Search	17783	17785	155920	12	45.914	Yes
A* Search h_ignore_preconditions	**5003**	5005	44586	12	**15.540**	Yes

From the above results, ***A* Search h_ignore_preconditions*** expands the fewest nodes and is the fastest, aside from Problem 1 in which ***Uniform-Cost Search*** is *slightly* faster. Overall, the best choice for our Air Cargo problem is ***A* Search h_ignore_preconditions***.

Optimal Sequence of Actions:

The following table desribes the optimal sequence of actions to solve the Air Cargo problems presented above:

Problem	Search Type	Optimal Sequence
Air Cargo Problem 1	A* Search h_ignore_preconditions	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)
Air Cargo Problem 2	A* Search h_ignore_preconditions	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)
Air Cargo Problem 3	A* Search h_ignore_preconditions	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)

Search Strategies Discussion:

The above results clearly demonstrate the benefits of using informed search strategies with custom heuristics as opposed to uninformed search strategies when trying to obtain an optimal plan. The benefits are substantial with regards to both speed and memory usage. One other benefit of using informed strategies is the customizable trade-off between execution speed and memory usage by creating different heuristics, and this is simply not possible with uninformed search strategies.

References:

1. Stuart J. Russell, Peter Norvig (2010), Artificial Intelligence: A Modern Approach (3rd Edition)