AIND Project 3 : Implement a Planning Search
Shan He

Plan Search Heuristic Analysis

In this project, we implemented a planing search agent to solve a deterministic logical planning problems for an Air Cargo System. We use a planning graph and domain-independent heuristic with A star search and compare their performances against several uninformed non- heuristic search methods .

• Planning Problems

We were given three planning problems in the Air Cargo domain that use the same **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))
```

The optimal plan lengths is 6 :
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 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))
```

The optimal plan lengths is 9 :

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(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

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))
```

The optimal plan lengths is 12 :

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

• Uninformed Search Strategies Analysis

From the book "Artificial Intelligence: A Modern Approach by Norvig and Russell" section 3.4: uniformed search is known as blind search, which has 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.

In this section, we compare the performance of uninformed strategies in terms of **speed** (execution time, measured in seconds), **memory usage** (measured in search node expansions) and **optimality** (Yes, if a solution of optimal length is found; No, otherwise). The number of goal tests and number of new nodes are not reported in the tables below since they do not change the results of our analysis below.

Performance measures were collected using the following commands:

```
python run_search.py -p 1 -s 1 2 3 4 5 6 7 >> run_uninformed_search_results_p1.txt
python run_search.py -p 2 -s 1 3 5 7 >> run_uninformed_search_results_p2.txt python
run_search.py -p 3 -s 1 3 5 7 >> run_uninformed_search_results_p3.txt
```

For Problem 2 and 3 , because their execution time exceeded 10 minutes, we cancelled data collection for Breadth First Tree Search, Depth Limited Search, and Recursive Best First Search

Problem 1 Result

Search Strategy	Optimal	Path Length	Execution Time(s)	Node Expansions
Breadth First Search	Yes	6	0.047	43
Breadth First Tree Search	Yes	6	1.24	1458
Depth First Graph Search	No	20	0.18	21
Depth Limited Search	No	50	0.127	101
Uniform Cost Search	Yes	6	0.053	55
Recursive Best First Search	Yes	6	3.54	4229
Greedy Best First Graph Search	Yes	6	0.008	7

Problem 2 Result

Search Strategy	Optimal	Path Length	Execution Time(s)	Node Expansions
Breadth First Search	Yes	9	20.00	3343
Breadth First Tree Search	-	-	-	-
Depth First Graph Search	No	619	5.18	624
Depth Limited Search	-	-	-	-
Uniform Cost Search	Yes	9	19.10	4853
Recursive Best First Search	-	-	-	-
Greedy Best First Graph Search	No	15	3.54	998

Problem 3 Result

Search Strategy	Optimal	Path Length	Execution Time(s)	Node Expansions
Breadth First Search	Yes	12	146.29	14663
Breadth First Tree Search	-	-	-	-
Depth First Graph Search	No	392	2.58	408
Depth Limited Search	-	-	-	-
Uniform Cost Search	Yes	12	74.00	18223
Recursive Best First Search	-	-	-	-
Greedy Best First Graph Search	No	22	22.2	5578

Analysis

With this 3-problem set, **Breadth First Search** and **Uniform Cost Search** are the only two uninformed search strategies that **yield an optimal action plan** under the 10mn time limit. When it comes to execution speed and memory usage, **Depth First Graph Search** is the **fastest and uses the least memory**. However, it does not generate an optimal action plan .

Which search strategy should we use, if *having an optimal path length is not the primary criteria*? For problems 2 and 3, the Depth First Graph Search plan lengths are so much longer than the optimal path length that it wouldn't make sense to use this search strategy. **Greedy Best First Graph Search is the best alternative**. In problems 1 and 2, it manages to find the optimal path. In problem 3, it does not find the optimal path but the path length it generates is 22 instead of 10, which is much better than Depth First Graph Search. Moreover, it still provides execution time savings and uses less memory than the best search strategy for an optimal solution (Breadth First Search).

• Informed (Heuristic) Search Strategies Analysis

From the book **"Artificial Intelligence: A Modern Approach by Norvig and Russell" section 3.5** : Informed search strategy — one that uses problem−specific knowledge beyond the definition of the problem itself — can find solutions more efficiently than can an uninformed strategy. In this section, we compare the performance of **A* Search using three different heuristics**. Here again, we evaluate these strategies in terms of **speed**, **memory usage** and **optimality**.

Performance measures were collected using the following commands:

```
python run_search.py -p 1 -s 8 9 10 >> run_informed_search_results_p1.txt python
run_search.py -p 2 -s 8 9 10 >> run_informed_search_results_p2.txt python
run_search.py -p 3 -s 8 9 >> run_informed_search_results_p3.txt
```

For Problems 3, because its execution time exceeded 10 minutes, we did not collect data for A* Search with Level Sum heuristic.

Problem 1 Results

Search Strategy	Optimal	Path Length	Execution Time(s)	Node Expansions
A* Search with h1 heuristic	Yes	6	0.054	55
A* Search with Ignore Preconditions heuristic	Yes	6	0.046	41
A* Search with Level Sum heuristic	Yes	6	1.47	11

Problem 2 Results

Search Strategy	Optimal	Path Length	Execution Time(s)	Node Expansions
A* Search with h1 heuristic	Yes	9	14.83	4853
A* Search with Ignore Preconditions heuristic	Yes	9	5.36	1450

Search Strategy	Optimal	Path Length	Execution Time(s)	Node Expansions
A* Search with Level Sum heuristic	Yes	9	263.075	9

Problem 3 Results

Search Strategy	Optimal	Path Length	Execution Time(s)	Node Expansions
A* Search with h1 heuristic	Yes	12	73.58	18223
A* Search with Ignore Preconditions heuristic	Yes	12	23.38	5040
A* Search with Level Sum heuristic	-	-	-	-

Analysis

While all heuristics yield an optimal action plan, only the h1 and Ignore Preconditions heuristics return results within the 10mn max execution time set by the Udacity staff.

Which heuristic should we use? Of the two strategies mentioned above, **A* Search with Ignore Preconditions heuristic is the fastest**. If we let search run to completion on our machine, **A* Search with Level Sum heuristic uses the least memory**, but its execution time is much slower

• Informed vs Uninformed Search Strategies

The search strategies that generate optimal plans are Breadth First Search, Uniform Cost Search, and A* Search with all three heuristics.

As we saw earlier, when it comes to execution speed and memory usage of uninformed search strategies, **Depth First Graph Search** is faster and uses less memory than Uniform Cost Search. As for informed search strategies, **A* Search with Ignore Preconditions heuristic** is the fastest and uses the least memory. So, really, the choice is between Depth First Graph Search and A* Search with Ignore Preconditions heuristic. After comparing that, we find that **A* Search with Ignore Preconditions heuristic** would be the best choice overall for our Air Cargo problem for faster and less memory usage

• Conclusion

The results above clearly illustrate the benefits of using informed search strategies with custom heuristics over uninformed search techniques when searching for an optimal plan. The benefits are significant both in terms of speed and memory usage. Another, more subtle, benefit is that one can customize the trade–off between speed and memory usage by using different heuristics, which is simply not possible with uninformed search strategies.