Sai Tai
https://github.com/saitai0802
http://www.sai-tai.com/blog/

Implement a Planning Search

December 26, 2017

Overview

In this project, we implemented a planning search agent to solve deterministic logistics planning problems for an Air Cargo transport domain presented in the lectures and text (Russell & Norvig, 2010).

In this project, we use a planning graph and automatic domain-independent heuristics with A* search to optimal plans for each problem and uninformed progression search algorithms (breadth-first, depth-first, greedy_best_first search etc.). In this report, we will discuss and compare their results/performance between uninformed and heuristic search.

Content

1. **Planning Problems:** We will talk about the problem we want to solve by using PDDL (Planning Domain Definition Language) and providing an optimal plan for each problem .
2. **Search Strategies Analysis:** Compare and contrast uninformed search and domain-independent heuristics. Then plot result metrics for each searching algorithms .
3. **Non-heuristic vs Heuristic search:** A discussion to talk about what was the best heuristic used in these problems.

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))
```

We represent those below three problems by using PDDL (Planning Domain Definition Language):

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))
```

Optimal Plans

We can use various searching algorithms to reach the goal of our problem, but only some searching algorithms can perform an optimal plan lengths for problems 1,2, and 3:

Problem 1 Plan length: 6	Problem 2 Plan length: 9	Problem 3 Plan length: 12
Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	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)	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK)

Search Strategies Analysis

We used both uninformed search strategies and informed (Heuristic) search strategies to solve our problem. The table below represents all the results for the three problems for comparison. Performance measures were collected using commands provided by run_search.py.
eg: python run_search.py -p 1 -s 1 2 3 4 5 6 7

The searches were run using the run_search.py utility. There are 3 indicators to tell us how a search algorithm performs in terms of **optimality**, **memory usage** and **speed**.

1. Optimality

   If a solution of **optimal length** is found then assign True; No, otherwise

2. Memory usage

   The **plan length** represents the number of actions required to reach the goal as determined by the particular planning search.

   The **node expansions** represent the number of frontier nodes that were expanded in order to find the solution, and represent the complexity of the search.

3. Speed

   Execution time through the process to let our agent to find a goal.

Problem 1 results

Search Strategies	Heuristic	Optimal	Path Length	Node Expansion	Execution time(s)
breadth_first_search		True	6	43	0.03316
breadth_first_tree_search		True	6	1458	0.99929
depth_first_graph_search			20	21	0.01447
depth_limited_search			50	101	0.09613
uniform_cost_search		True	6	55	0.03770
recursive_best_first_search		True	6	4229	2.89834
greedy_best_first_graph_search		True	6	7	0.00501
A* Search	h=1 (null heuristic)	True	6	55	0.04014
A* Search	h_ignore_preconditions	True	6	41	0.04117
A* Search	h_levelsum	True	6	11	1.02478

Problem 2 results

Search Strategies	Heuristic	Optimal	Path Length	Node Expansion	Execution time(s)
breadth_first_search		True	9	3343	8.83151
breadth_first_tree_search					
depth_first_graph_search			619	624	3.51757
depth_limited_search					
uniform_cost_search		True	9	4852	12.3418
recursive_best_first_search					
greedy_best_first_graph_search			15	990	2.49964
A* Search	h=1 (null heuristic)	True	9	4852	13.0171
A* Search	h_ignore_preconditions	True	9	1450	5.25863
A* Search	h_levelsum	True	9	86	174.5849

Problem 3 results

Search Strategies	Heuristic	Optimal	Path Length	Node Expansion	Execution time(s)
breadth_first_search		True	12	14491	42.3380
breadth_first_tree_search					
depth_first_graph_search			2014	2099	20.9777
depth_limited_search					
uniform_cost_search		True	12	17783	51.6178
recursive_best_first_search					
greedy_best_first_graph_search			22	4031	11.7135
A* Search	h=1 (null heuristic)	True	12	17783	54.3666
A* Search	h_ignore_preconditions	True	12	5003	18.2772
A* Search	h_levelsum	True	12	311	1132.77

Compare and contrast non-heuristic(Uninformed) search result

Search strategies that come under the heading of uninformed search (a.k.a., blind search) 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. Thus, these searching algorithm are without heuristics to aid them.

Throughout these 3 problems, only **breadth first search**, **uniform cost search can** lead our agent to find an optimal action plan. Though, **Depth first search** is the fastest and uses the least memory usage, but it can't perform optimal plan when problem becomes more complex.

Assume we always want to find an optimal plan for our problem, and have an agent performs faster and uses less memory. **Breadth First Search** is the recommended search strategy among all the others Uninformed Search Algorithm. **Breadth First Search** is complete and optimal. Its only downside is memory usage, if the problem's branching factor is high, as shown the image below:

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

Compare and contrast Heuristic(Informed) search result

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 project we use A* Search with three different heuristics including h=1, h_ignore_preconditions and h_levelsum .

1. h=1 - Basically this is not a true heuristic.
2. Ignore-preconditions – Based on the assumption of sub-goal independence, as mentioned in the README, this heuristic is simply the number of goal requirements for the problem still not met at any given node in the planning search.
3. Level-sum – The planning graph implemented creates a graph with alternating “levels” of S0, A0, S1, A1, S2, …. where S levels are literal levels and A levels are action levels. The level cost for

Throughout these 3 problems, all 3 heuristic search can lead our us to find an optimal action plan. As for informed search strategies, **A* Search with Ignore Preconditions heuristic** is the fastest and uses the least memory.

Though **h_levelsum** has the best memory usage in term of node expansions, but its execution time is way too long if we compare to others heuristic functions. Thus, **A* Search with Ignore Preconditions heuristic** is the recommended search strategy among all the others Informed Search Algorithm.

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, **Breadth 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.

Our choice is between **Breadth First Graph Search** and **A* Search with Ignore Preconditions heuristic**. Because **A* Search with Ignore Preconditions heuristic** is faster and uses less memory than **Breadth First Graph Search**, it would be the best choice overall for our Air Cargo problem.

Problem 3 results

Search Strategies	**Heuristic**	**Optimal**	**Path Length**	**Node Expansion**	**Execution time(s)**
breadth_first_search		True	12	14491	42.3380
uniform_cost_search		True	12	17783	51.6178
A* Search	h=1 (null heuristic)	True	12	17783	54.3666
A* Search	h_ignore_preconditions	True	12	5003	18.2772
A* Search	h_levelsum	True	12	311	1132.77

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.

References

Russell, S., & Norvig, P. (2010). Artificial Intelligence: a modern approach.