AIND Project #3
- Planning Search -
Analysis

Non Heuristic Planning solution searches (Uninformed Searches)

Air Cargo Problem 1

Algorithms	#Node expansions	#goal tests	Plan length	Elapsed time (sec)	Solution Optimality
breadth first search	43	56	6	0,0802	yes
breadth first tree search	1 458	1 459	6	1,0046	yes
depth first graph search	12	13	12	0,0234	no
depth limited search	101	271	50	0,2079	no
uniform cost search	55	57	6	0,0746	yes

Air Cargo Problem 2

Algorithms	#Node expansions	#goal tests	Plan length	Elapsed time (sec)	Solution Optimality
breadth first search	3 343	4 609	9	3,3415	yes
breadth first tree search	-	-	-	timeout (>15m)	yes
depth first graph search	582	583	575	0,7422	no
depth limited search	222 719	2 053 741	50	731,7332	no
uniform cost search	4 853	4 855	9	4,2705	yes

Air Cargo Problem 3

Algorithms	#Node expansions	#goal tests	Plan length	Elapsed time (sec)	Solution Optimality
breadth first search	14 663	18 098	12	12,9318	yes
breadth first tree search	-	-	-	timeout (>15m)	yes
depth first graph search	4 339	4 340	856	3,6475	no
depth limited search	-	-	-	timeout (>15m)	no
uniform cost search	18 223	18 225	12	14,7316	yes

Results

Regarding uninformed searches, base on these results, we can conclude that :

- if the primary concern is to find an **optimal solution**, then **Breadth first search** and **Uniform cost search** perform quite similarly, and give the best performance speed while keeping memory requirements reasonable (#Node expansion)

- if **speed of execution** and/or **memory constraints** are the main concern, then **Depth first graph search** is the best choice as it's the fastest solution while expanding the less nodes. But this is *at the cost of the path length which is not optimal.*

- compared to the other options, Breadth first tree search and Depth limited search does not look viable solutions for medium to larger problems, as the computing time exceeded 10 minutes (no results provided in some of the tests)

Heuristic Planning solution searches (Informed Searches)

I took the liberty to include Recursive best first search and Greedy best first search with the A* searches results.

Also see details about the used heuristics

- 'h_1' is not a true heuristics : it returns a constant value of 1.
- 'h_Ignore preconditions' heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.
- 'h_pg_levelsum' heuristic uses a planning graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition. (admissible if goals independent)

Air Cargo Problem 1

Algorithms	#Node expansions	#goal tests	Plan length	Elapsed time (sec)	Solution Optimality
recursive best first search (h_1)	4 229	4 230	6	2,3531	yes
greedy best first graph search (h_1)	7	9	6	0,0234	no
astar search (h_1)	55	57	6	0,0784	yes
astar search (h_ignore_preconditions)	41	43	6	0,1010	yes
astar search (h_pg_levelsum)	11	13	6	0,5230	yes

Air Cargo Problem 2

Algorithms	#Node expansions	#goal tests	Plan length	Elapsed time (sec)	Solution Optimality
recursive best first search (h_1)	-	-	-	timeout (>15m)	yes
greedy best first graph search (h_1)	998	1 000	21	1,0601	no
astar search (h_1)	4 853	4 855	9	4,3751	yes
astar search (h_ignore_preconditions)	1 450	1 452	9	2,7362	yes
astar search (h_pg_levelsum)	86	88	9	9,6632	yes

Air Cargo Problem 3

Algorithms	#Node expansions	#goal tests	Plan length	Elapsed time (sec)	Solution Optimality
recursive best first search (h_1)	-	-	-	timeout (>15m)	yes
greedy best first graph search (h_1)	5 578	5 580	22	4,7462	no
astar search (h_1)	18 223	18 225	12	14,8456	yes
astar search (h_ignore_preconditions)	5 040	5 042	12	8,3402	yes
astar search (h_pg_levelsum)	366	368	12	55,9755	yes

Results

Regarding uninformed searches, base on these results, we can conclude that :

- if the primary concern is to find an **optimal solution**, then **A* search with the ignore preconditions heuristic** perform the best in terms of execution time, while keeping memory requirements reasonable (#Node expansion)

- if **speed of execution** is the main concern, then **greedy best first graph search with the h_1 heuristic** is the best choice as it's the fastest solution while expanding among the less nodes. But this is *at the cost of the path length which is not optimal.*
- if **memory constraints** is the main concern, then **A* search with the h_pg_levelsum heuristic** is the best choice as it is expanding the fewer nodes. All the more, the solution it provides is **optimal.**

- compared to the other options, Recursive best first search with the h_1 heuristic, does not look viable solution for medium to larger problems, as the computing time exceeded 10 minutes (no results provided in some of the tests

Conclusion

if we now compare both non-heuristic search methods (Uninformed search) and heuristic search methods (Informed search), let's focus on the Air Cargo Problem 3 as it is the largest/complex of the 3 problems.

Algorithms	#Node expansions	#goal tests	Plan length	Elapsed time (sec)	Solution Optimality
breadth first search (Uninformed Search)	14 663	18 098	12	12,9318	yes
astar search (h_ignore_preconditions) (Informed Search)	5 040	5 042	12	8,3402	yes

By putting the results side by side, it is obvious that **A* search with the h_pg_levelsum heuristic** is the overall best search algorithm to use, as it is the fastest one to find an optimal solution, while expanding the less nodes and thus keeping the memory requirements lower. Probably a « go-to » solution for average needs.

Relaxing the problem definition turns out to be a very effective heuristic. It is also good to remember that « the performance of heuristic search algorithms depends on the quality of the heuristic function » as stated in the AIMA book. All the more, carefully crafting the heuristics function and fine tuning its performance, eventually by storing precomputed computations (for example), can have a strong impact on the final search performance, which is a strong advantage of Informed search methods over Uninformed search methods.

Selection of Optimal Plan for the problems

Air Cargo Problem 1

This optimal plan of length 6 was produced by the uniform_cost_search algorithm in 0.08s

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

=> Goal(At(C1, JFK) ∧ At(C2, SFO)) is reached

Air Cargo Problem 2

This optimal plan of length 9 was produced by the astar_search with h_ignore_preconditions in 2.7s

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)

=> Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO)) is reached

Air Cargo Problem 3

This optimal plan of length 12 was produced by the astar_search with h_ignore_preconditions in 8.3s

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)

=> Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO)) is reached

Annexes :

Setup and remark on the execution time

Execution time has been improved by using pypy3 (Python JIT). All tests has been carried out by parallelising 4 pypy3 jobs on a Xeon E5-1620v4 CPU using GNU parallel, and by constraining the jobs duration to 15 minutes

Prerequisites:
- Setup AIND conda environment
- install pypy3 (https://bitbucket.org/pypy/pypy/downloads/pypy3-v5.10.1-linux64.tar.bz2)
- install GNU parallel (`sudo apt-get install -y parallel` on Ubuntu)

Config file : jobs.conf (paths to be adjusted)

```
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 8 > p1_informed-8.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 9 > p1_informed-9.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 10 > p1_informed-10.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 1 > p1_uninformed-1.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 2 > p1_uninformed-2.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 3 > p1_uninformed-3.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 4 > p1_uninformed-4.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 5 > p1_uninformed-5.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 6 > p1_informed-6.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 1 -s 7 > p1_informed-7.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 8 > p2_informed-8.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 9 > p2_informed-9.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 10 > p2_informed-10.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 1 > p2_uninformed-1.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 2 > p2_uninformed-2.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 3 > p2_uninformed-3.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 4 > p2_uninformed-4.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 5 > p2_uninformed-5.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 6 > p2_informed-6.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 2 -s 7 > p2_informed-7.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 8 > p3_informed-8.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 9 > p3_informed-9.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 10 > p3_informed-10.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 1 > p3_uninformed-1.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 2 > p3_uninformed-2.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 3 > p3_uninformed-3.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 4 > p3_uninformed-4.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 5 > p3_uninformed-5.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 6 > p3_informed-6.txt
timeout 15m /media/shared/aind-planning/pypy3-v5.10.1-linux64/bin/pypy3  run_search.py -p 3 -s 7 > p3_informed-7.txt
```

Execution

```
source activate aind
cd /media/shared/aind-planning/
parallel -j4 -a jobs.conf --no-notice --joblog jobs.log
```