Heuristic Analysis of Planning Search Agents for Air Cargo Transport System

By Pankaj Mathur

Problem Details

An air cargo transport problem involves loading and unloading cargo onto and off of planes and flying it from place to place.

The problem can be defined with three actions:

- Load
- Unload
- Fly

The actions affect two predicates:

- In(c, p) means that cargo c is inside plane p
- At(x, a) means that object x (either plane or cargo) is at airport a

Please note that cargo is not At anywhere when it is In a plane, so At really means "available for use at a given location."

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))

Optimal Plan

PROBLEM 1 (Plan length: 6)	PROBLEM 2 (Plan length: 9)	PROBLEM 3 (Plan length: 12)
1. Load(C1, P1, SFO)	1. Load(C1, P1, SFO)	1. Load(C1, P1, SFO)
2. Load(C2, P2, JFK)	2. Load(C2, P2, JFK)	2. Load(C2, P2, JFK)
3. Fly(P2, JFK, SFO)	3. Load(C3, P3, ATL)	3. Fly(P1, SFO, ATL)
4. Unload(C2, P2, SFO)	4. Fly(P1, SFO, JFK)	4. Load(C3, P1, ATL)
5. Fly(P1, SFO, JFK)	5. Unload(C1, P1, JFK)	5. Fly(P2, JFK, ORD)
6. Unload(C1, P1, JFK)	6. Fly(P2, JFK, SFO)	6. Load(C4, P2, ORD)
	7. Unload(C2, P2, SFO)	7. Fly(P1, ATL, JFK)
	8. Fly(P3, ATL, SFO)	8. Unload(C1, P1, JFK)
	9. Unload(C3, P3, SFO)	9. Unload(C3, P1, JFK)
		10. Fly(P2, ORD, SFO)
		11. Unload(C2, P2, SFO)
		12. Unload(C4, P2, SFO)

Comparison of Non-Heuristic Search Results

Optimality

DFS is obviously turn out to non-optimal whereas BFS and UCS where just right on track and were perfectly optimal.

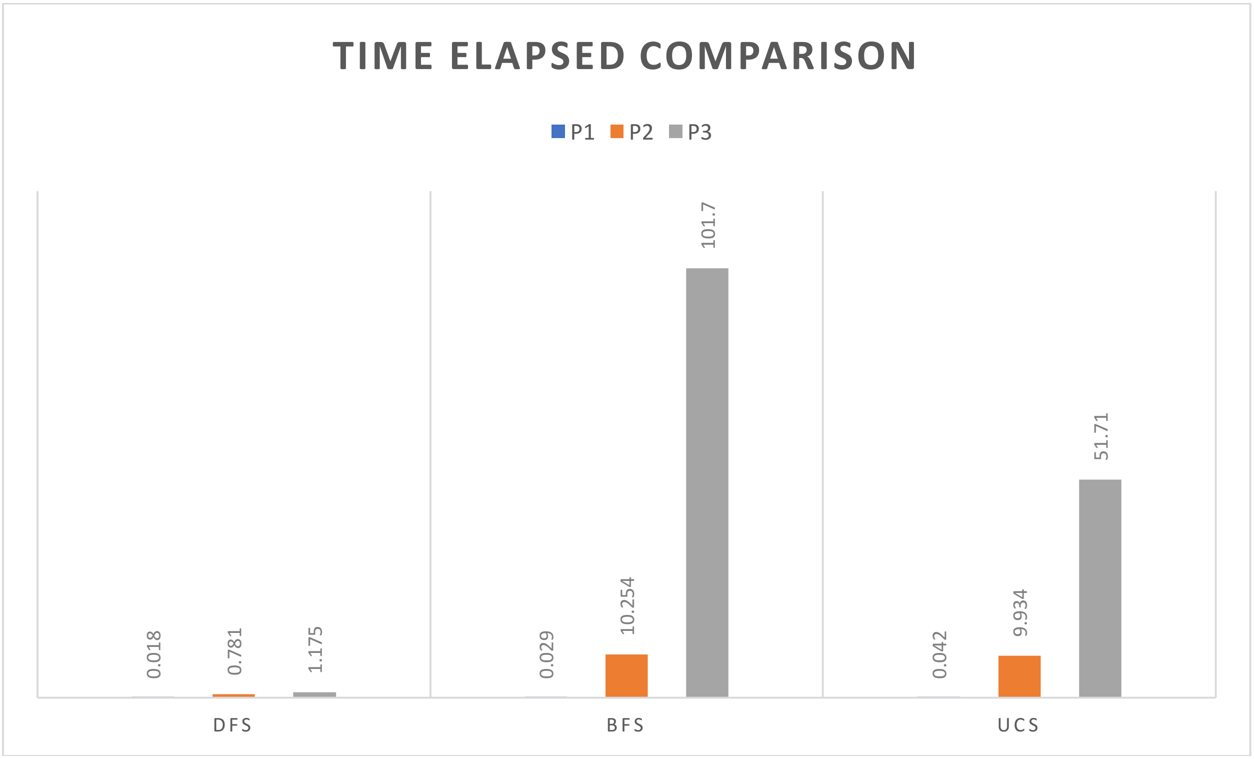
Optimality	**DFS**	**BFS**	**UCS**
P1	NO	YES	YES
P2	NO	YES	YES
P3	NO	YES	YES

Time Elapsed

Despite being non-optimal, DFS was way better in terms of time elapsed by BFS, UCS, which is due to fact that DFS always look for deep down path and in our problem case, the best solution was actually deep below. If we take criteria of Optimal solution then UCS was better than BFS for P2 and P3, but for P1 BFS was better than UCS

Time Elapsed	**DFS**	**BFS**	**UCS**
P1	0.018	0.029	0.042
P2	0.781	10.254	9.934
P3	1.175	101.70	51.710

Here is the comparison graph:

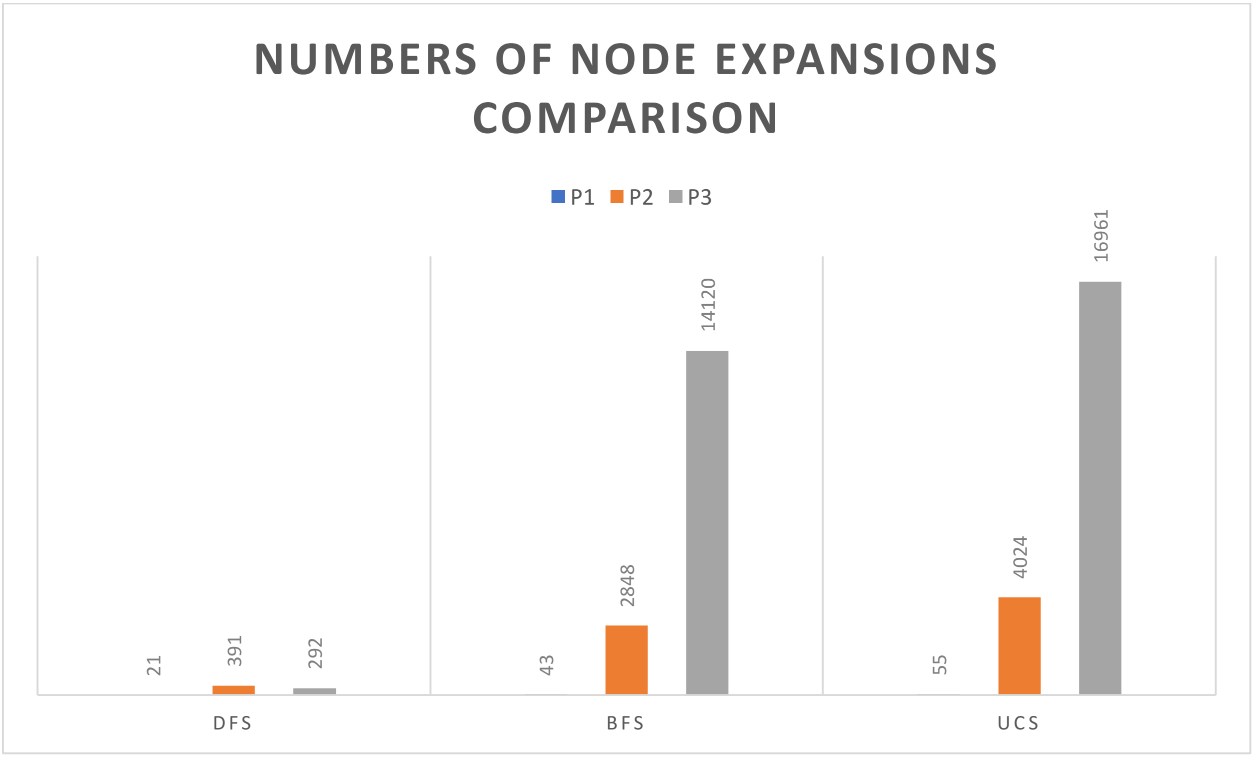


Numbers of Node Expansions

Again, DFS was far better in terms of number of node expansion but in terms of optimal solutions BFS was better than UCS for expanding less nodes.

Numbers of Node Expansions	DFS	BFS	UCS
P1	21	43	55
P2	391	2848	4024
P3	292	14120	16961

Here is the comparison graph



Which Non-Heuristic Search Agent DFS, BFS and UCS?

Since, DFS does not give an optimal solution since each plan is longer then than it is necessary, for example, for P1: DFS plan length is 12 instead of 6, for P2: DFS plan length is 346 instead of 9 and for P3: plan length is 1878 instead of 12. This doesn't sound anywhere realistic for Air Cargo problem and may not be in real world too, where fuel cost is immersive factor.
BFS and UCS does give an optimal solution because each step-in plan has equal cost, for example, the step cost of flying from SFO to ATL is same as flying from JFK to ORD.

References:

- **Chapter 10 Classical Planning from Book Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig**

Comparison of Heuristic Search Results

Optimality

Both A* with levelsum and A* with ignore_preconditions were optimal for Problems P1 and P2, however, A* with levelsum turn out to be non-optimal for P3 with plan length of 13 instead 12.

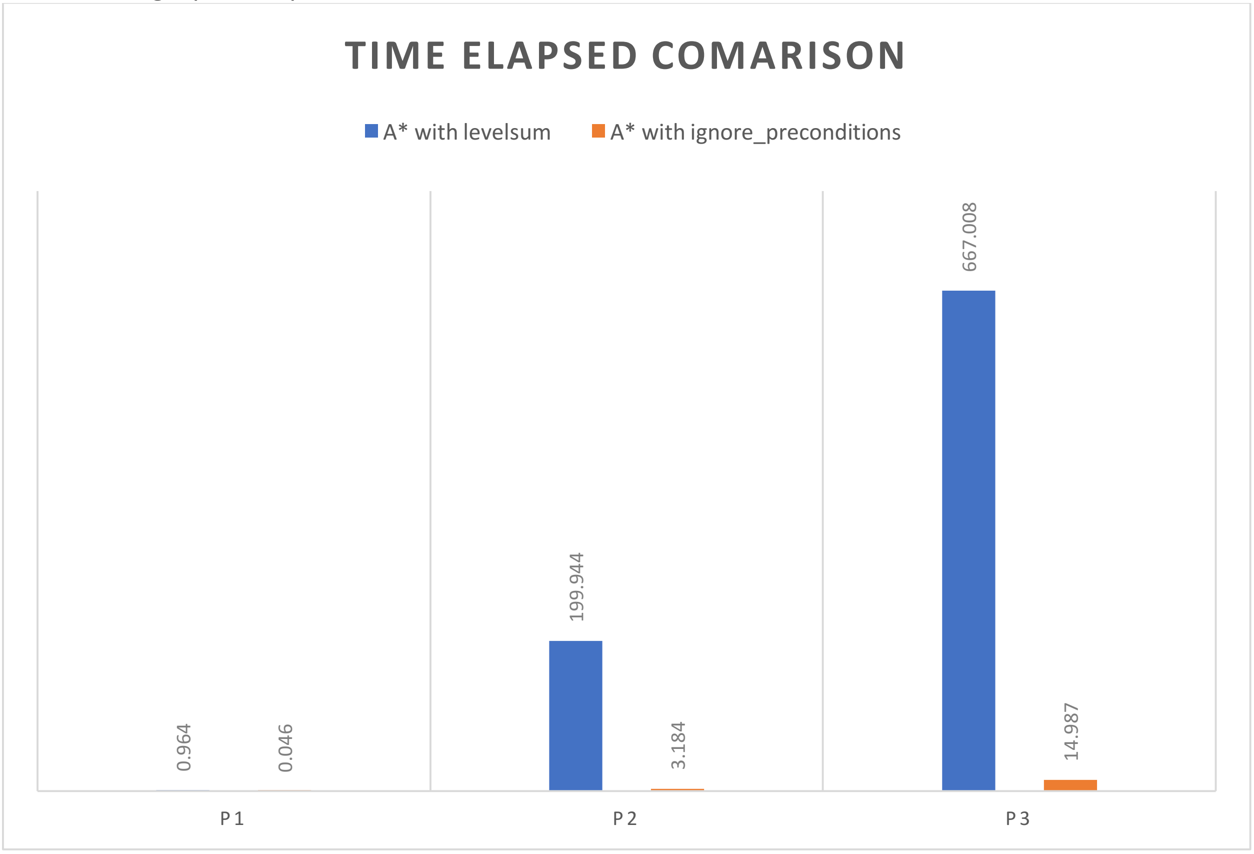
Optimality	**A* with levelsum**	**A* with ignore_preconditions**
P1	YES	YES
P2	YES	YES
P3	NO	YES

Time Elapsed

In terms of comparison of time spent by 2 different Heuristic search functions, clearly A* with ignore_preconditions function was winner, as it was far ahead of A* with levelsum function for each of the problem type, i.e. p1, p2 and p3.

Time Elapsed	A* with levelsum	A* with ignore_preconditions
P1	0.964	0.046
P2	199.944	3.184
P3	667.008	14.987

Here is the graph comparison:



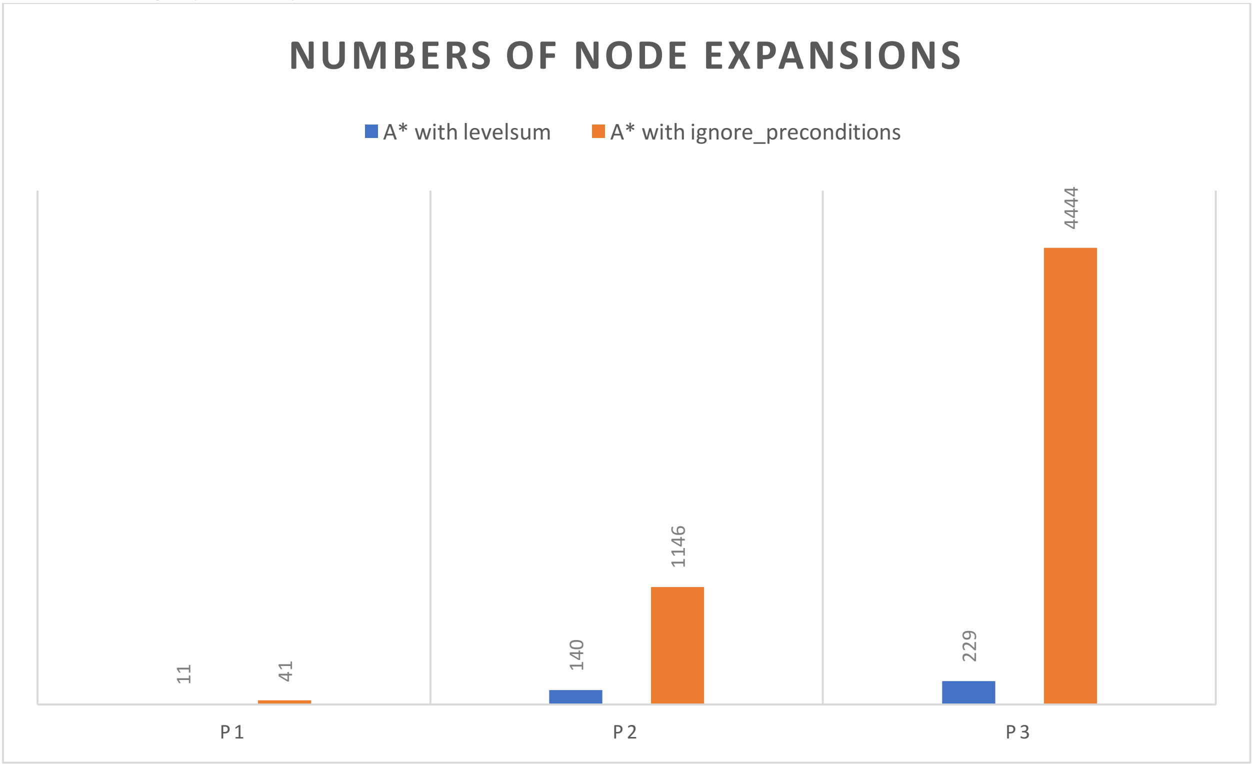
Numbers of Node Expansions

Interestingly A* with ignore_preconditions expanded many more nodes than A* with levelsum function in less amount of time. This was mostly because of neglecting and bypassing all the deep calculations for all the pre-conditions that come on each node. Hence, less computing time for each node and despite large amount of node expansion, the A* with ignore_preconditions was faster than A* with levelsum.

Numbers of Node Expansions	A* with levelsum	A* with ignore_preconditions
P1	11	41
P2	140	1146

P3	229	4444

Here is the graph comparison:



Why A*with ignore_preconditions is faster than A* with pg_levelsum?

Because **Ignore pre- conditions heuristic** drops all preconditions from actions. Every action becomes applicable in every state, and any single goal fluent can be achieved in one step as long as if there is an applicable action—if not, the problem is impossible. The numbers of node expanded in each problem, p1, p2 and p3 is proof of them they are significantly higher for ignore_preconditions then pg_levelsum, in addition to that the time spent by ignore_preconditions literally lowest on bottom compare to pg_levelsum.

References:

- **Chapter 10 Classical Planning from Book Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig**

Comparison between best of Non-Heuristic and Heuristic Results

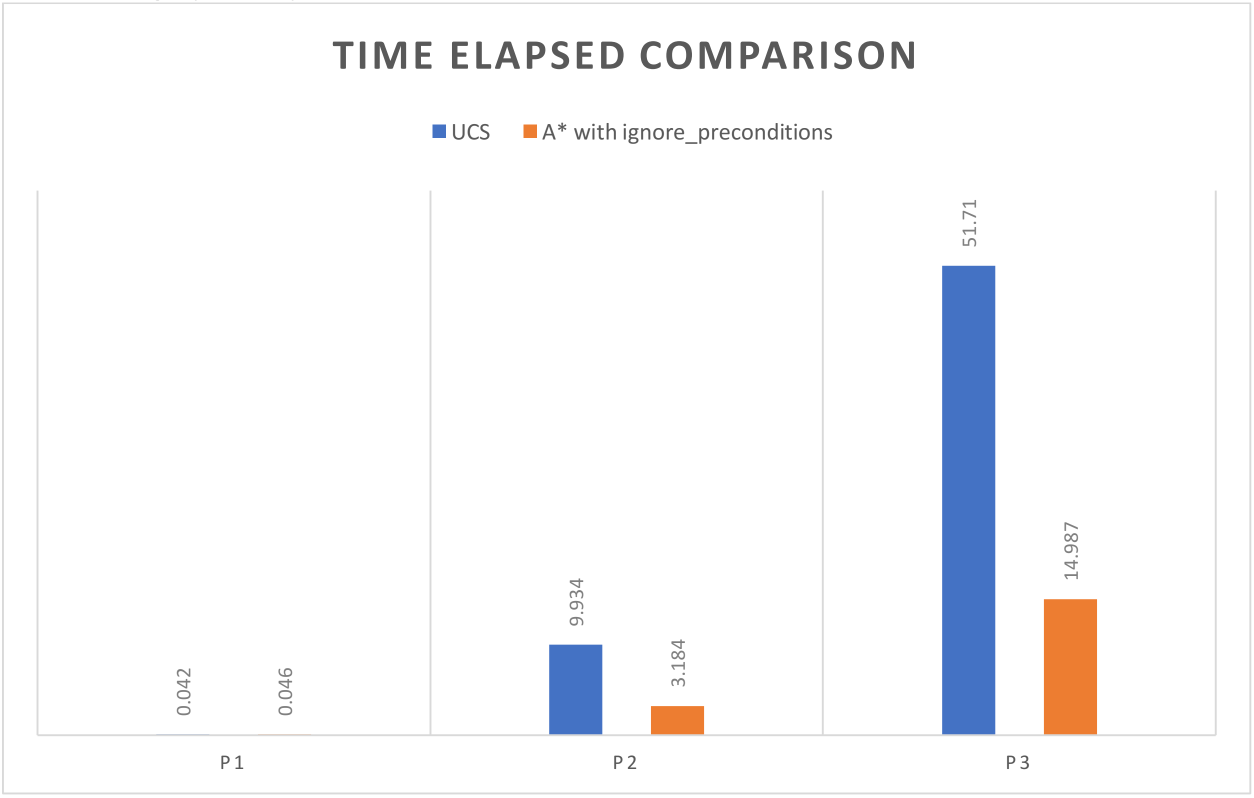
When we compare Non-Heuristic and Heuristic functions, we took following criteria to choose the best:

- First, we took only optimal functions for all problems p1, p2, p3. For Non-Heuristic it was UCS and BFS, for Heuristic it was A* with ignore_preconditions
- Second, we choose the fastest search functions among the results of first criteria.

Time Elapsed

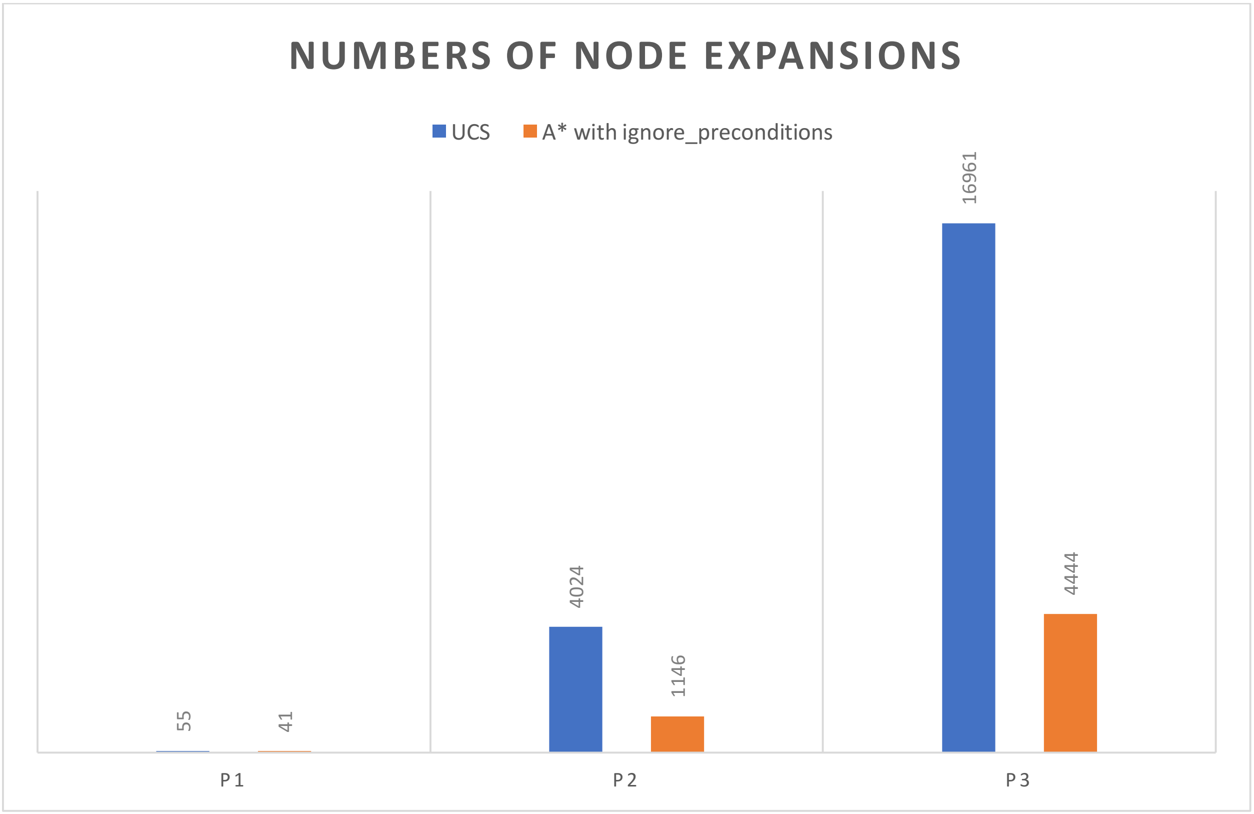
Time Elapsed	UCS	A* with ignore_preconditions
P1	0.042	0.046
P2	9.934	3.184
P3	51.710	14.987

Here is the graph comparison:



Numbers of Node Expansions	UCS	A* with ignore_preconditions
P1	55	41
P2	4024	1146
P3	16961	4444

Here is the graph comparison:



Final Verdict

Based upon our search criteria, Time elapsed between UCS and A* with ignore_preconditions and being optimal functions, we can say that, A* with ignore_preconditions heuristic is the best search planning Agent for Air Cargo Transport problem!