Description Project:

In this project, we implemented a planning search agent to solve deterministic logistics planning problems for an Air Cargo Transport System.

Problems:

The project contained 3 problems, this reporter is divided the analysis for each problem. The problems are:

- *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))

- *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))

- *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 Actions:

The goal in every problem can be reached using different plans, the tables below show optimal planning for every problem.

Solution Problem 1

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

Solution Problem 2

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

Fly(P2,JFK,SFO)
Unload(C2,P2,SFO)
Load(C3,P3,ATL)
Fly(P3,ATL,SFO)
Unload(C3,P3,SFO)

Solution Problem 3

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

The previous tables show the optimal plan lengths for the **problem 1, 2 and 3 are 6, 9, 12 actions**; respectively.

Uninformed and Informed Search

The following tables contain the results of experiments with different strategy, for the problem 1 used all strategies. But for the problem 2 and 3, in **Uninformed** only I used *breadth_first_search, depth_first_graph_search, uniform_cost_search and greedy_best_first_graph_search with h_1*.

In the tables I considered compare performance of the strategies in terms of speed (time elapsed in seconds), memory (Node Expansions and Goal Test) and Optimality (plan length).

Problem 1

Uninformed Strategy	Time Elapsed(s)	Plan Lenght	Node Expansions	Goal Test
breadth_first_search	0.057	6	43	56
breadth_first_tree_search	1.935	6	1458	1459
depth_first_graph_search	0.016	12	12	13
depth_limited_search	0.175	50	101	271
uniform_cost_search	0.066	6	55	57
recursive_best_first_search with h_1	5.341	6	4229	4230
greedy_best_first_graph_search with h_1	0.066	6	55	57

Informed Strategy	Time Elapsed (s)	Plan Lenght	Node Expansions	Goal Test
astar_search with h_1	0.070	6	55	57
astar_search with h_ignore_preconditions	0.079	6	41	43
astar_search with h_pg_levelsum	2.126	6	11	13

For the problem 1 we observed the all uninformed and informed strategies are optimal; plan length equal 6, except ***depth_first_graph_search*** *and* ***depth_limited_search****.*

Problem 2

Uninformed Strategy	Time Elapsed(s)	Plan Length	Node Expansions	Goal Test
breadth_first_search	27.205	9	3343	4609
depth_first_graph_search	7.654	575	582	583
uniform_cost_search	23.364	9	4761	4763
greedy_best_first_graph_search with h_1	3.004	9	550	552

Informed Strategy	Time Elapsed (s)	Plan Lenght	Node Expansions	Goal Test
astar_search with h_1	16.970	9	4761	4763
astar_search with h_ignore_preconditions	6.339	9	1450	1450
astar_search with h_pg_levelsum	286.476	9	86	88

For the problem2 we observed the all uninformed and informed strategies used are optimal: plan length equal 9, except ***depth_first_graph_search***.

Problem 3

Uninformed Strategy	Time Elapsed(s)	Plan Length	Node Expansions	Goal Test
breadth_first_search	198.658	12	14491	17947
depth_first_graph_search	39.711	1878	1948	1949
uniform_cost_search	90.973	12	17783	17785
greedy_best_first_graph_search with h_1	23.291	22	4031	4033

Informed Strategy	Time Elapsed (s)	Plan Length	Node Expansions	Goal Test
astar_search with h_1	77.682	12	17783	17785

astar_search with h_ignore_preconditions	25.887	12	5003	5005
astar_search with h_pg_levelsum	1513.546	12	311	313

For the problem 3 we observed these all uninformed and informed strategies used are optimal; plan Length equal 12, except ***depth_first_graph_search*** and ***greedy_best_first_graph_search with h_1***.

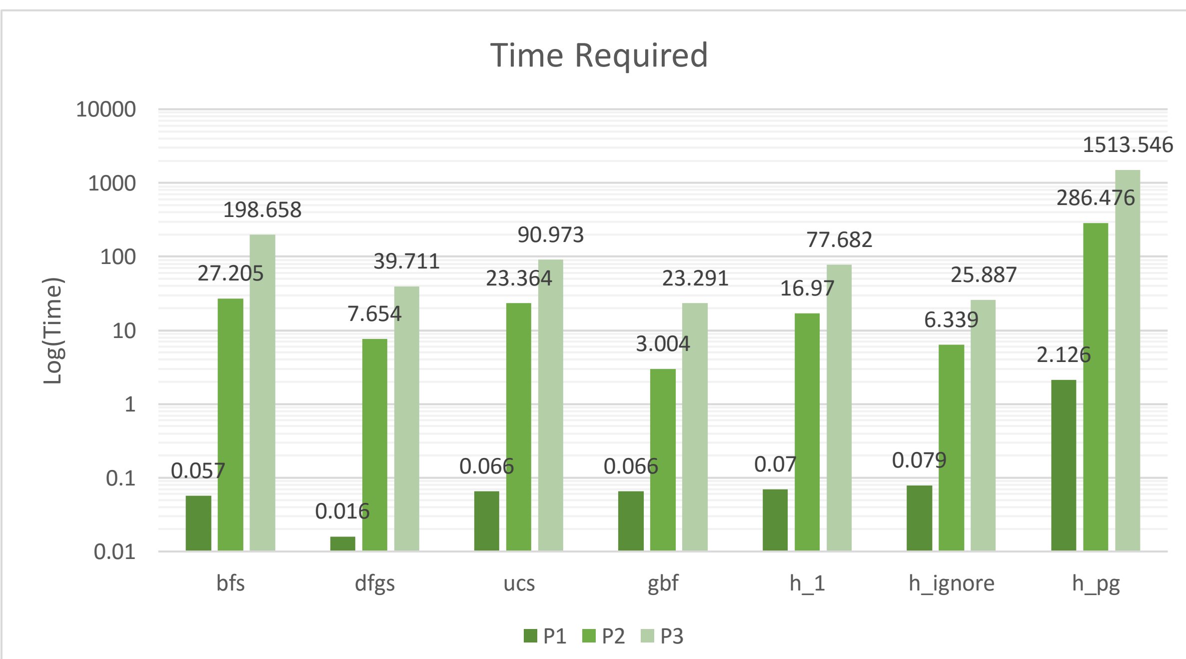
Analysis of Results

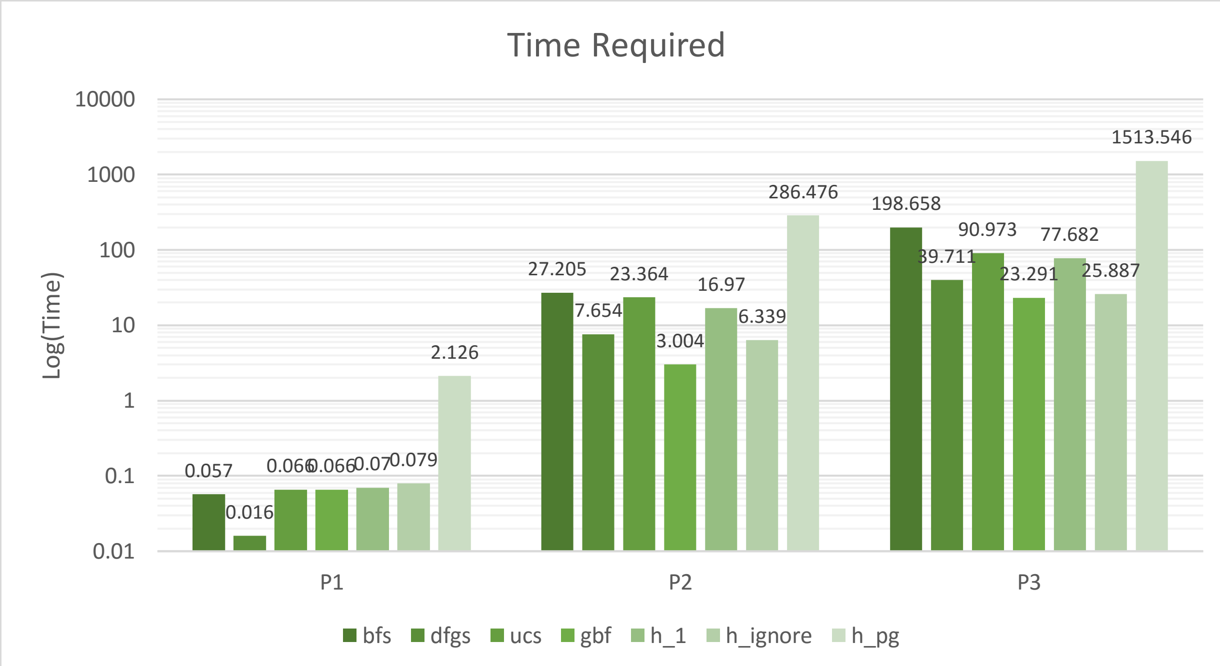
For the analysis only I considered the results in the table for the uninformed strategies: **breadth_first_search**, **depth_first_graph_search**, **uniform_cost_search** and greedy**_best_first_graph_search with h_1**. For the informed strategies, I used all strategies with heuristics.

For the graphics, I used the nomenclature for each strategy:

- Breadth first search =bfs
- Depth first graph search = dfgs
- Uniform cost search = ucs
- Greedy best first graph search=gbf
- Search with h_1 =h_1
- search with h_ignore_preconditions= h_ignore
- search with h_pg_levelsum= h_pg

For Execution Time





The graphs have the scale in logarithms for "times elapsed". The first graph shows the strategies vs problems, it shows different behavior in problems. In the seconds graph show ***log(time)*** for every strategy in every problem.

In uninformed strategies show in the problem 2 and 3, the order in strategies is:

- **bfg>ucs>dfgs>gbf**

Thus, **gbf** is the fastest planning search among bfg, ucs, dfgs and gbf.

In informed strategies show in the problem 2 and 3, the order in strategies is:

- **h_pg>h_1>h_ignore**

Thus, **h_ignore** is the fastest planning search. Comparing the two types of strategies; uninformed and informed, the **gbf** required less time in h_ignore in the 3 problems.

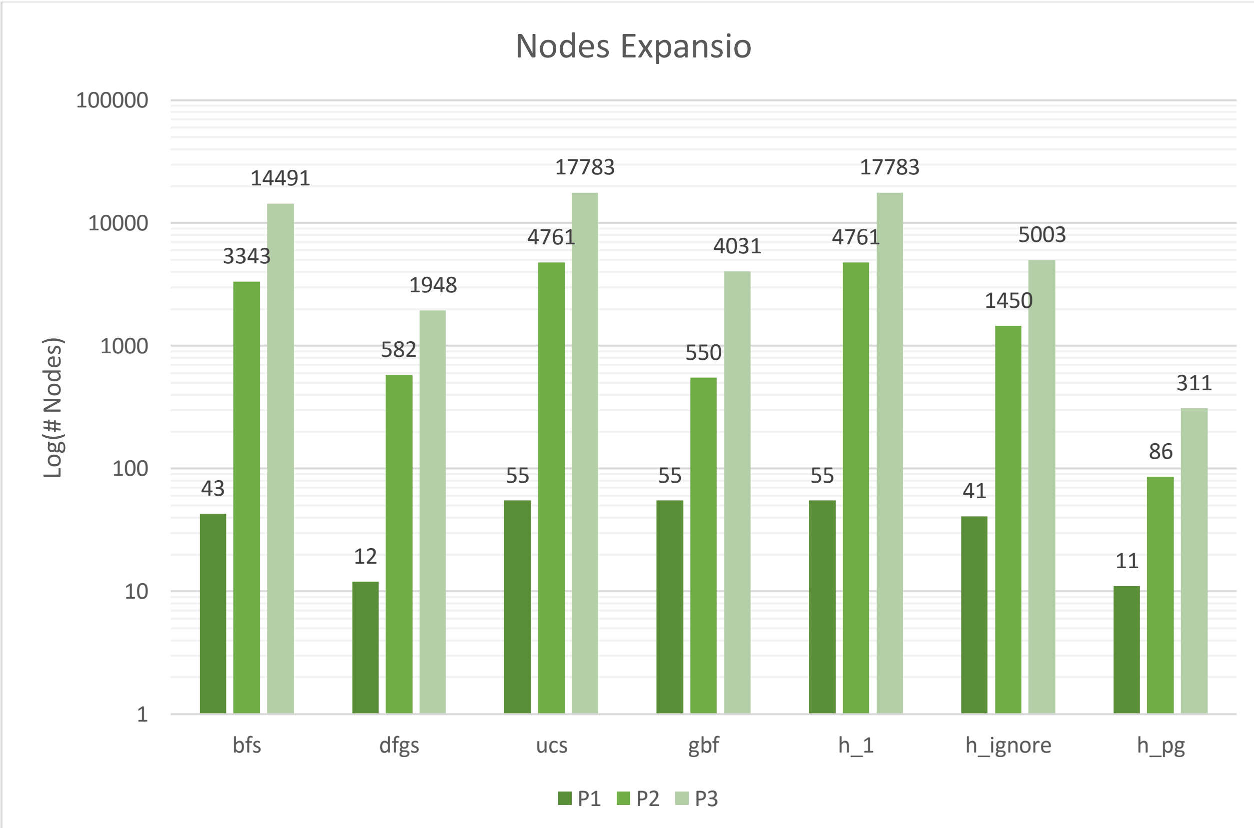
For Node Expansion

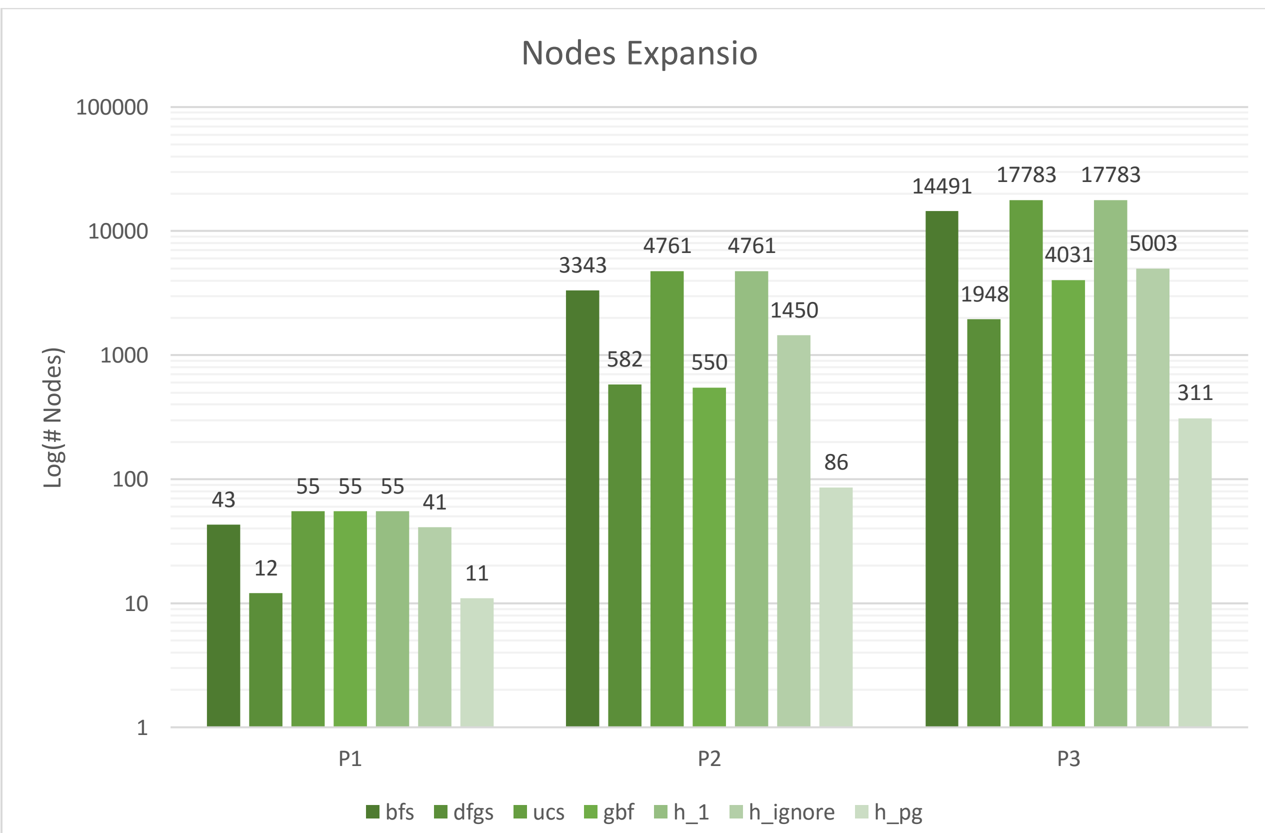
The first graph show node expansions strategies vs problems, it shows different behavior in problems. No is clear the relation between uninformed strategies vs problems, but **dfgs** and **gbf** require fewer nodes expansion.

In informed strategies show in the all problems, the order in strategies is:

- **h_1>h_ignore>h_pg**

Thus, the number of nodes expanded is least by **h_pg** heuristics.

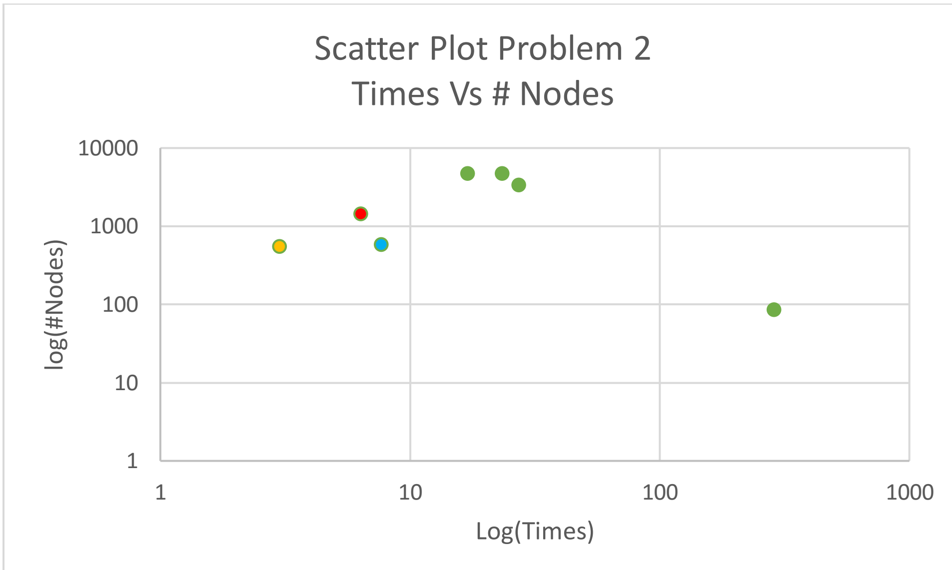

Nodes Expansio
Log(# Nodes)
100000
10000
1000
100
10
1
14491
3343
43
1948
582
12
17783
4761
55
4031
550
55
17783
4761
55
5003
1450
41
311
86
11
bfs
dfgs
ucs
gbf
h_1
h_ignore
h_pg
P1
P2
P3

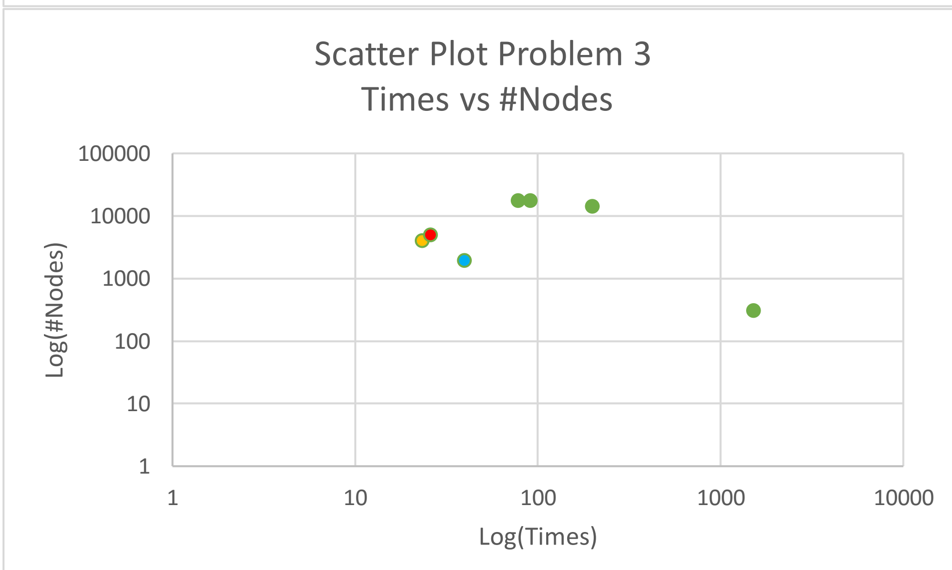

Nodes Expansio
Log(# Nodes)
100000
10000
1000
100
10
1
43
12
55
55
55
41
11
3343
582
4761
550
4761
1450
86
14491
1948
17783
4031
17783
5003
311
P1
P2
P3
bfs
dfgs
ucs
gbf
h_1
h_ignore
h_pg

Conclusion

The optimality of each of the heuristics is guaranteed as all of them are admissible. All the heuristics under consideration provide optimal solution for each of the problems.

Comparing the strategies between the balance of #nodes vs times for problems 2 and 3 for each strategy we see the following:





The isolated point of the graph corresponds to the strategy **h_pg**, it is appreciated that this have a low number of nodes but it requires a lot of time.

The following 3 strategies show a low number of nodes and a smaller amount of time, so a better balance is in red, blue and yellow. These correspond to the strategies h_ingnore, dfgs and gbf, respectively. Of those 3 strategies, the only one that is optimal in all problems is h_ignore, therefore it is the best strategy of both the uninformed and the informed.

Optimal Solution the A*-Search using h_ignore_preconditions:

Problem 1

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

Problem 2

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)

Problem 3

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)