Planning project Report

Udacity Artificial intelligence nanodegree

Abdelrhman-Yasser

15th December, 2018

Introduction

In this report we will evaluate our search techniques and heuristics in Air cargo problem introduced and solved with planning in Udacity AI nanodegree, we will also show difference between different techniques and answer questions introduced by mentor in the project.

Questions to be answered

1. Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

2. Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

3. Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

Problem : Air cargo transport

1. Air cargo transport problem involving loading and unloading cargo onto and off of planes and flying it from place to place.
2. The problem can be defined with three actions: Load, Unload, and Fly.
3. The actions affect two predicates: In(c, p) means that cargo c is inside plane p, and At(x, a) means that object x (either plane or cargo) is at airport a.
4. Note that cargo is not At anywhere when it is In a plane, so At really means “available for use at a given location.”

Schema

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))

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))

Search techniques used

1. Breadth first search
2. Depth first search
3. Uniform cost search
4. Greedy best first search with heuristic :
    a. Unmet goals
    b. Per goal level-sum
    c. Per goal max-level
    d. Per goal set-level
5. A star search with heuristic :
    a. Unmet goals
    b. Per goal level-sum
    c. Per goal max-level
    d. Per goal set-level

Heuristics

I. Unmet goals:

- This 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.

II. Per goal level-sum:

- This heuristic uses a planning graph representation of the problem state space to estimate the sum of the number of actions that must be carried out from the current state in order to satisfy each individual goal condition.

III.Per goal max-level:

- This heuristic uses a planning graph representation of the problem to estimate the maximum level cost out of all the individual goal literals.
- The level cost is the first level where a goal literal appears in the planning graph

IV.Per goal set-level:

- This heuristic uses a planning graph representation of the problem to estimate the level cost in the planning graph to achieve all of the goal literals such that none of them are mutually exclusive.

Evaluation

Now we will evaluate several techniques used from running problems 1, 2, 3, 4 on all searches and all heuristics and that information we get from runs on local machine:

1. Problem 1 :

Search / property	Actions	Expansions	Plan length	Time
Breadth first search	20	43	6	0.02
Depth first search	20	21	20	0.011
Uniform cost search	20	60	6	0.03
Greedy with unmet goals	20	7	6	0.003
Greedy with per goal level-sum	20	6	6	0.58
Greedy with per goal max-level	20	45	8	0.587
Greedy with per goal set-level	20	6	6	0.59
A* with unmet goals	20	50	6	0.012
A* with per goal level-sum	20	28	6	0.324
A* with per goal max-level	20	55	6	0.25
A* with per goal set-level	20	33	6	0.625

2. Problem 2

Search / property	Actions	Expansions	Plan length	Time
Breadth first search	72	3343	9	0.45
Depth first search	72	624	619	0.617
Uniform cost search	72	5154	9	0.769
Greedy with unmet goals	72	17	9	0.045
Greedy with per goal level-sum	72	9	9	1.043
Greedy with per goal max-level	72	3010	15	190
Greedy with per goal set-level	72	9	9	3.943
A* with unmet goals	72	2467	9	1.5
A* with per goal level-sum	72	357	9	44
A* with per goal max-level	72	4742	9	361
A* with per goal set-level	72	1037	9	301

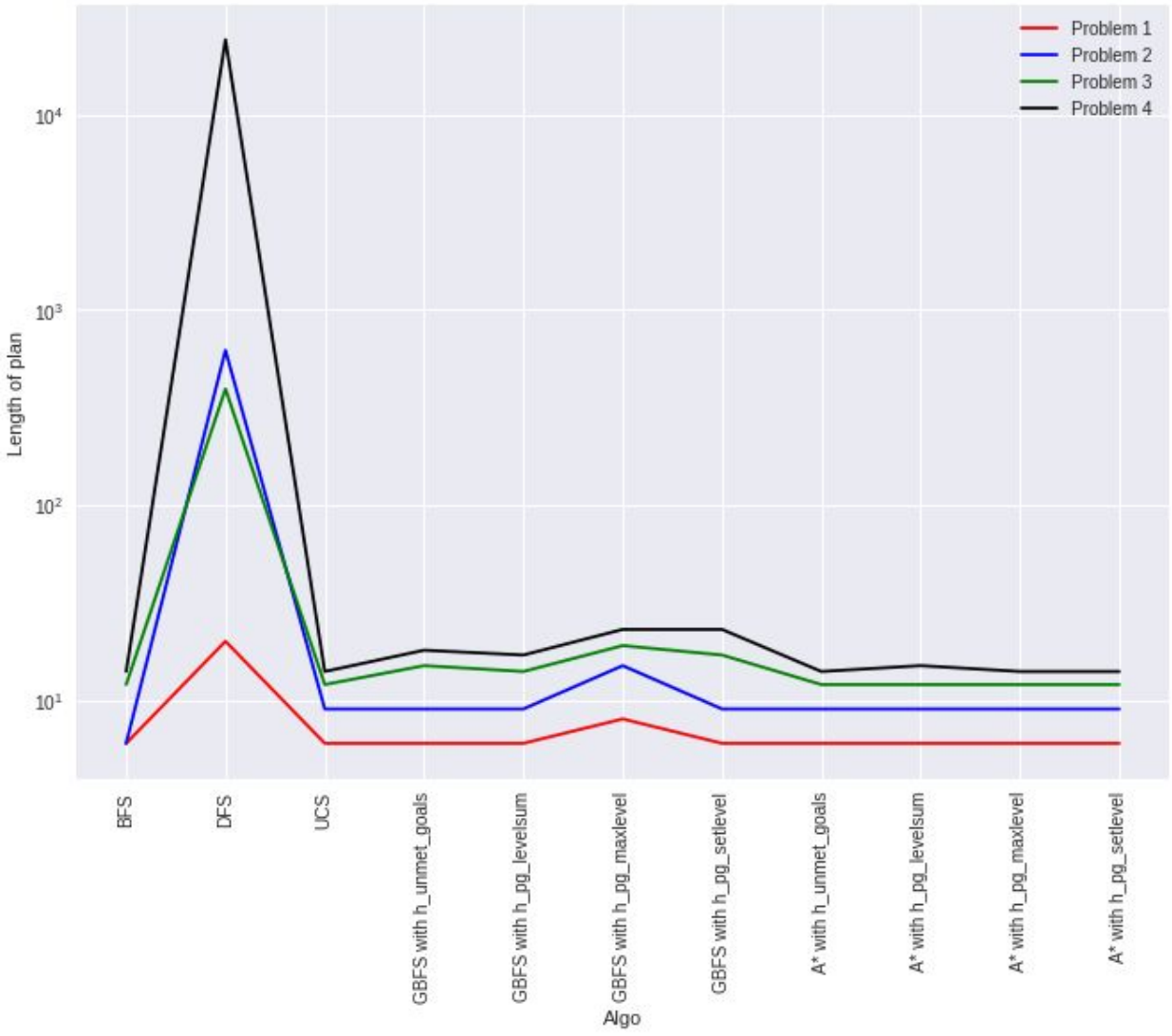
3. Problem 3

Search / property	Actions	Expansions	Plan length	Time
Breadth first search	88	14663	12	1.588
Depth first search	88	408	392	0.37
Uniform cost search	88	18510	12	2.645
Greedy with unmet goals	88	25	15	0.017
Greedy with per goal level-sum	88	14	14	3.792
Greedy with per goal max-level	88	2817	19	506
Greedy with per goal set-level	88	35	17	20
A* with unmet goals	88	7388	12	1.88
A* with per goal level-sum	88	369	12	78.07
A* with per goal max-level	88	15642	12	1979
A* with per goal set-level	88	3423	12	1610

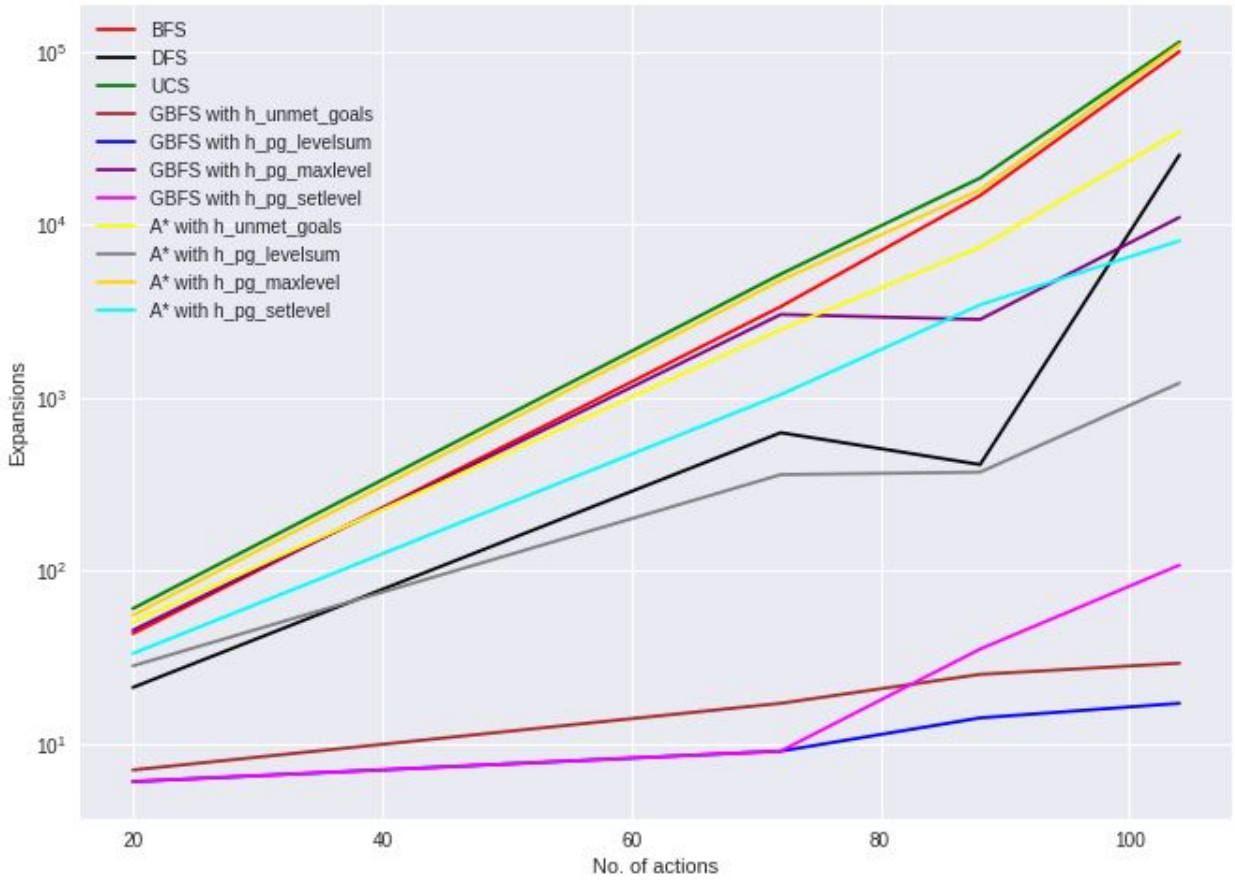
4. Problem 4

Search / property	Actions	Expansions	Plan length	Time
Breadth first search	104	99763	14	12.4
Depth first search	104	25174	24132	2138
Uniform cost search	104	113339	14	17.6
Greedy with unmet goals	104	29	18	0.05
Greedy with per goal level-sum	104	17	17	10
Greedy with per goal max-level	104	10971	23	3357
Greedy with per goal set-level	104	107	23	86
A* with unmet goals	104	34330	14	8.87
A* with per goal level-sum	104	1208	15	409
A* with per goal max-level	104	109321	14	3600
A* with per goal set-level	104	8019	14	2765

Charts

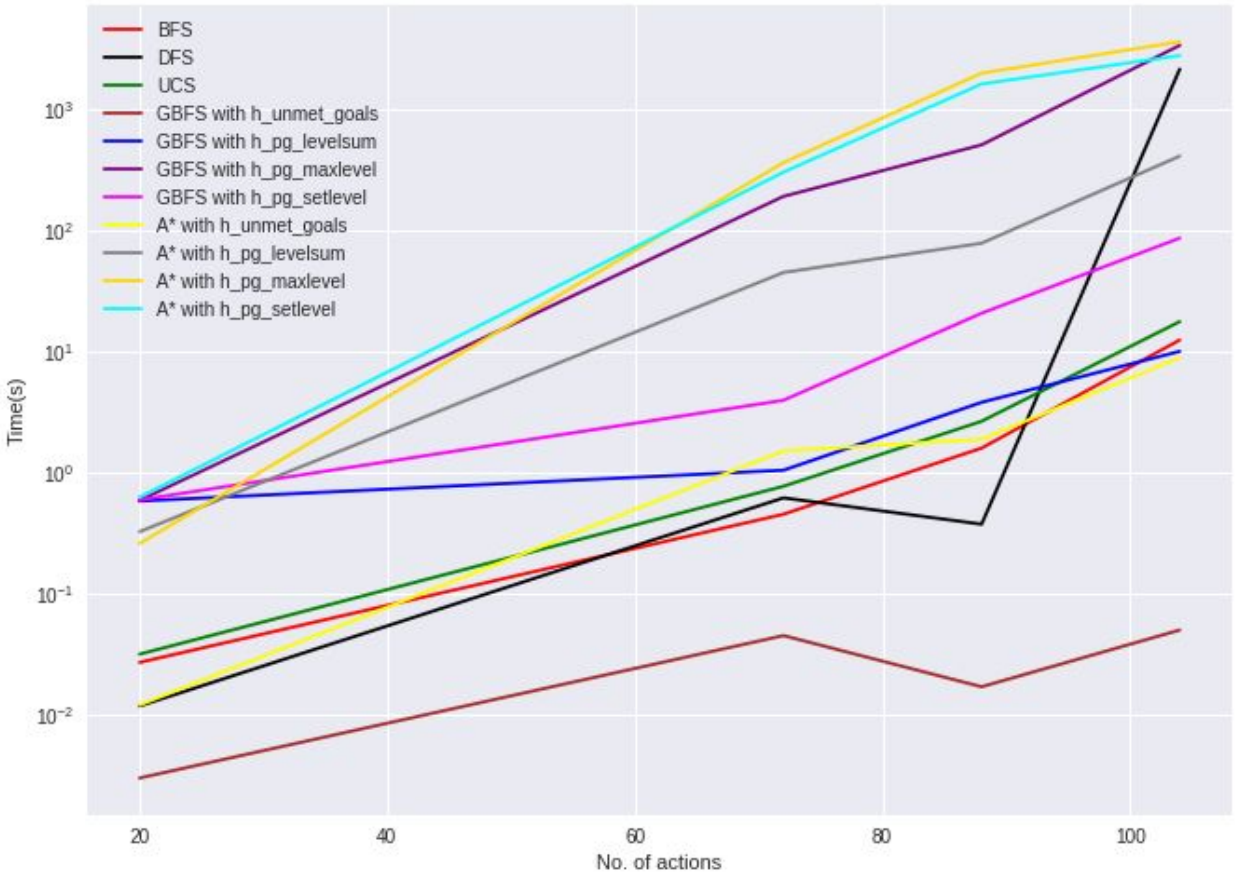


It's clear that A star always get shortest path to goal , and greedy with heuristics gives more shorter path than regular BFS, DFS, UCS



There are several notes about this chart

- **BFS, UCS expansions is growing exponentially with actions**
- **A star with unmet goals is doing very bad probably because of heuristic is not good**
- **Greedy is better from expansions point of view**



There are several notes about this chart

- **A star is taking a lot of time to get shorter path**
- **Greedy with unmet goals is best algo from time point of view**

Conclusion

Question 2

Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

Answer : since actions is few UCS or BFS can be used as time and expansions won't make problem with restricted domain.

Question 2

Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

Answer : I would go with Greedy with unmet goals as it's time ie small relative to other techniques as in evaluation.

Question 3

Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

Answer : I would go with A start as it reaches shortest path whenever the heuristic is admissible which is the case with our heuristics .