Lab 1 matge373

1.1

- Please tell us whether you had any specific difficulties in modeling the problem domain or getting FF to generate a plan.

To me, the main struggle was getting to know PDDL since I've only done very basic tasks in one course before. Especially figuring out how to use typing was more difficult than I thought, but that makes sense to me now.

The first time I ran the planner after finishing writing my domain and problem, I got a segmentation fault. This was difficult to understand as it didn't specify anything, and the lab assistant didn't really understand why either. The lab assistant then tried changing my typing of "xxx - obj" to "xxx - object", which worked. So FF was in some way sensitive to strictly defining my types as objects, rather than a generic thing called "obj".

1.2 IPP

/courses/TDDD48/planners/ipp/plan domain1.pddl **p1_1.pddl** tempout, where p1_1 is the first problem, the one with only one person. It took 4 time steps to reach the goal, which in reality is a fraction of a second.

/courses/TDDD48/planners/ipp/plan domain1.pddl **p1_2.pddl** tempout. This took 7 time steps to complete, which also is a very short amount of time (less than a second).

1.2.3

IPP can still do simple tasks very quickly, for instance a problem with 1 uav, 5 locations, 5 persons, 6 crates and 6 goals. Increasing the amount of crates to 15 increases runtime by about 20x, so increasing the number of crates results in a more complicated plan to solve. Increasing the amount of people does increase the runtime a little bit, but it's barely noticeable (0.012 vs. 0.015 seconds for 5 vs. 15 people). So it does not have the same impact.

Running the IPP with:
UAVs: 1
Carriers: 0
Locations: 5
Persons: 5
Crates: 10
Goals: 9

Increased the runtime, until finding a goal, to 76 seconds.

1.3

These planners will initially run with:
UAVs: 1
Carriers: 0
Locations: 5
Persons: 5
Crates: 10
Goals: 9

- Lama-2011

It took approximately 0.05 seconds to find the first solution.

Creating a problem instance with:
UAVs: 1
Carriers: 0
Locations: 100
Persons: 90
Crates: 200
Goals: 150

Lama-2011 managed to get a first result after 18+7+38 = 63 seconds.

- madagascar-p

The initial problem took 0.02 seconds to complete.

Creating a problem instance with:
UAVs: 1
Carriers: 0
Locations: 30
Persons: 20
Crates: 50
Goals: 40

Madagascar-p managed to get a first result after 49 seconds.

- YAHSP3

This planner didn't finish within the time limit for the first given problem instance. I cancelled it after 77 seconds (it got to bound 21). Testing it on 6 crates and 6 goals instead it finished within a second.

- ipc2018

I decided to try Cerberus-gl which took .011 seconds to find the first solution.

Creating a problem instance with:

UAVs: 1
Carriers: 0
Locations: 30
Persons: 30
Crates: 100
Goals: 85

Cerberus-gl found its first solution after 44.5 seconds (I'm counting that as 45 seconds, which was acceptable).

1.4

1.

5 objects, since we're counting impossible states there should be 5 combinations available for each predicate.

2^(5*5) combinations of "at"
2^5 combinations of "has-photo-of"
2^(5*5) combinations of "photo-possible"

2^5 uav
2^5 loc
2^5 target

⇒ 2^(25+25+20) = 2^70 = very large

2.

O = U+T+L objects in total. Since we're still counting impossible, sensible and meaningless states we should have the following:

2^(O^2 + O^2 + 4*O)

3.

Now in the small instance we'll have the following:
2^(1*2)
2^2
2^(2*2)

⇒ 2^8 = 256

In the general case we would have the following:
2^(U*L)
2^(T)
2^(L*T)

⇒ 2^(UL+T+LT)

This is now because we know only locations that are instanced where the ?loc variable is, for instance. This drastically lowers the state space mainly because of this, and also since we've removed the type predicates.

4.

Starting off, we'd want to move somewhere and take a photo there, if possible. In this case, we can assume that photo-possible is always true for each location we visit. Then the number of unique actions instances should be:

- Move:
  - U*(L-1)
  - So each UAV can go to another location, but not the location it's already on (since it would be meaningless).
- Take-photo
  - Since we're "optimistic" about photo-possible, we're assuming that it's always possible to take a photo when we want to.
  - This would then result in T instances.

In total, this would give us U*(L-1) + T as the highest number of unique applicable actions.

<u>Extra part:</u>

1. **What was wrong with the original domain?**

I found an error in the initial domain where the UAV was allowed to use the same crate to satisfy multiple people. For instance, if persons p1 and p2 both needed water but were at different locations, l1 and l2, the UAV could load a crate that contained water, and fly to p1 at l1, satisfying their "need" by giving them the crate. Afterwards, it would simply pick the same crate back up and fly with that to p2 at l2, and unload it there.

All in all, there was nothing to prevent the same crates being used multiple times.

2. **How might this have affected the results?**

I don't think this affected the results *too much*, as the UAV still had to pick up the correct crate, fly it to a person, unload it, and then pick it back up and fly to the next (assuming there were multiple people in need of the same contents). So what the planned did was it realized that it was already at the same location at a crate (conveniently enough), so it did not have to go back to the depot to get another one.

I'm uncertain whether this actually affected the planner in a positive or negative way timewise. It could be that it was now trying to optimize the use of a single crate as much as possible, instead of just picking up a crate at the depot and flying it out over and over again. My hunch is that it probably affected it negatively, but not by a lot. In lab 2, when I realized I had this issue, I compared the two on one of the planners and it differed around 2-3 seconds of planning time (in favor of the "correct" domain").