Lab 3 matge373

3.1

The initial runs will be made using the following problem instance:
UAVs: 4
Carriers: 1
Locations: 5
Persons: 5
Crates: 6
Goals: 5
Numbers: 4

- IPP

The IPP does not work for added cost, so this planner will not be run.

- Lama2011

This planner makes use of multiple UAVs, looking at both its first and second solution. It's not optimal (obviously), but it tries to run multiple UAVs "concurrently".

- madagascar-p

Madagascar also uses multiple UAVs, whilst also doing a lot of weird and unecessary actions, for instance taking the carrier to a location and then back, without ever loading it.

- YAHSP3

The problem instance above took longer than 60 seconds to complete with this planner and was therefore aborted.

Running it with:
UAVs: 4
Carriers: 1
Locations: 2
Persons: 2
Crates: 3
Goals: 2
Numbers: 4

Generated a plan which uses two UAVs to load the carrier, and then one to fly the carrier to the persons in need.

- ipc2018 (Cerberus-gl)

Cerberus also uses multiple UAVs for the initial problem instance.

- General thoughts

Overall, it seems like all of the sequential planners tested uses multiple UAVs if given, whilst all except madagascar seems to be consistent in doing so. This is judged by the fact that madagascar seems to just do a bunch of random actions (not really, but it looks like it) to reach the goal, meaning the use of multiple UAVs could simply be a coincidence. There is

most definitely room for improvement if the planners were aware of the parallel execution being available. However, many of the planners started with loading crates on multiple UAVs, then flying them out to people, indicating that their initial plans were quite good in the sense of running multiple agents in parallel.

3.2

Which actions can/cannot be executed in parallel?:

The actions available in my domain currently is:

- LOAD-UAV
- UNLOAD-UAV-TO-PERSON
- FLY-UAV
- FLY-CARRIER
- LOAD-CRATE-ON-CARRIER
- TAKE-CRATE-FROM-CARRIER

My idea of the *world* is that there would be space at a location for multiple UAVs to deliver crates to people. However, I am unsure if there is a way to limit this. For instance in an extreme case, it would be odd to have 10 UAVs unloading their crates to the same person at the same time. While two UAVs, in my opinion, would still be reasonable. But to avoid weird instances like this, **UNLOAD-UAV-TO-PERSON** should also be unable to be run concurrently.

What we want to avoid is having UAVs trying to take the same actions with the same objects, for instance, if uav1 and uav2 simultaneously try to pick up crate1. This means that **LOAD-UAV** should be unable to run concurrently, to avoid issues like that.

Flying a UAV will be completely fine to run in parallel.

All of the actions involved with the carrier should not be able to run in parallel. This would however be good, given that one could guarantee that UAVs do not try to fly the same carrier to different locations simultaneously, or if they were to unload the same crate simultaneously. As for loading the carrier, this should not be able to be done concurrently either, since that could potentially mean that the carriers' maximum capacity is reached and ignored if two UAVs load, and therefore count the carrier capacity up once, at the same time.

All in all, it seems like all actions except FLY-UAV should be blocked from running concurrently. If there is a good way of ensuring that UAVs does not try taking actions with the same objects, this can be revisited.

3.3

To ensure that the rovers cannot send multiple pieces of data at the same time I added a "ready" condition to the domain. This works similarly to the "at" condition for the drive action. Essentially, it forces the rover to wait until it is read to send a message, when it is sending a message, this is changed to false (so that the rover is not ready, i.e. it is transmitting a message). Then again, when the data has been sent, the rover is again put in ready-mode.

3.4

To get a general idea of the planners, I started with a smaller problem instance:

UAVs: 4
Carriers: 1
Locations: 4
Persons: 4
Crates: 5
Goals: 4
Numbers: 4

- ITSAT

The initial problem took close to 30 seconds to finish. Plan looks alright, no odd/unnecessary actions taken.

- YAHSP3

The initial problem is taking a lot of time to find a decently good solution. It could find five different ones within 20 seconds, but many of them contained weird actions such as loading and unloading a carrier with the same crate multiple times.

- TFD (Temporal-fast-downward)

The initial two were really fast, but further optimization is taking >>1 minute. Plan looks alright, does not seem to take any unnecessary or weird actions.

Running the slightly larger problem instance:
UAVs: 4
Carriers: 2
Locations: 8
Persons: 8
Crates: 9
Goals: 8
Numbers: 4

Gave the following:

- ITSAT

Found its first solution within 50 seconds and the plan looks alright, no odd actions. The generated plan gave a result of 930s.

- YAHSP3

This planner finds an initial plan very quickly, even though it's an extremely bad one (~3250s). After roughly 80 seconds it managed to find a solution that “only” costs 1666 seconds to complete, which is still worse than ITSAT.

- TFD

Found a decent solution within 40-50 seconds (hard to interpret the output in the terminal). This came out with a cost of 1319 seconds.

Discussion:

Generally, changing the number of crates and UAVs will make the problem harder to solve for all the planners, almost exponentially. Giving the planner more UAVs would give it the ability to solve the problems quicker, as more UAVs can carry crates to people in need, but it also increases the search space significantly. The same thing happens with the amount of crates, increasing them without changing the number of goals gives the planner more options of crates to take to a location, making it more time-consuming.