AUTOMATIC PLANNING - REPORT - LAB 1

Manu S. Joseph(mansj125)

Lab 1.1. Emergency Services Logistics, Initial Version

The domain was created using the following types:

1. *crate* - Used to denote a crate of supplies
2. *person* - Used to denote an injured person
3. *helicopter* - Used to denote an UAV
4. *content* - Used to denote the content that is present in the crate
5. *location* - Used to denote the location of person, crate or helicopter

The domain used the following predicates

1. *obj_at* - associate an object to a location(object can be a helicopter, crate or person).
2. *person_crate* - associate a crate contents to a person.
3. *crate_content* - associate a crate to its contents.
4. *heli_crate* - associate a crate to helicopter when it is carrying it.
5. *heli_free* - denotes if helicopter is not carrying any crates and is free to carry a crate.

The domain has the following actions defined

1. *pickup_crate* - pick a crate from a location using helicopter. Associates helicopter with the crate. (Preconditions added so that action can be done only if helicopter and crate are in the same location and the helicopter is free).
2. *fly_to* - fly helicopter from one location to another. Can be done with or without a crate loaded in the helicopter
3. *deliver* - deliver a crate from the helicopter to an injured person. (Preconditions added so that the action can be done only if the helicopter and person is in the same location).

The domain definition file is named - *domain.pddl*

Two problem were created to verify the domain using ff planner they are named -
- *problem_small_1.pddl* (which is the problem with 1 UAV, 1 person and 1 crate)
- *problem_small_2.pddl* (which is the problem with 1 UAV, 2 persons and 3 crates).

The outputs of running these problems in the domain are in *"/Output_Lab1.1/terminal_out_ff_problem_small_1"* and *"Output_Lab1.1/terminal_out_ff_problem_small_2"*.

One difficulty that I faced during the domain definition was on not using negative preconditions to check if helicopter is already carrying a crate. This was solved by introducing another precondition(*heli_free*).

Another difficulty that I faced while trying to make ff planner run was based on the problem initialisation. I had missed to include the predicate *heli_free* at init in the problem definition which caused ff planner to not find any solution. This was fixed after some debugging.

Lab 1.2. Early State of The Art Planners

Question 1)

The small problem that was defined earlier were run using IPP. The problem was run in very little time, the time in seconds was less than 0.00 second for both the problems. The output of running the problems using IPP can be found at *"Output_Lab1.2/ terminal_out_ipp_problem_small_1"* and *"Output_Lab1.2/ terminal_out_ipp_problem_small_2"*.

On testing it was found that IPP can solve the following problem in almost one minute, 1 - UAV, 10 - Location, 15 - people, 20 - crates and 6 goals. The plan for this problem was found by IPP in 66 seconds.

The problem definition is named,
Problem_Max1min_ipp_uav_problem_u1_r0_l10_p15_c20_g6_ct2.pddl

The outputs from running this problem in IPP can be found at, *"Output_Lab1.2/ terminal_out_uav_problem_max1min_ipp"*

Question 2)

It could be seen that on increasing the number of people to 30 without change in anything else the run time increased to 108 seconds. On increasing the number of crates to 30 from the base problem the run time increased to 223 second.

The problem definition with increased number of crates and people are present in the repository they are named,
Problem_IncreasedCrates_uav_problem_u1_r0_l10_p15_c30_g6_ct2.pddl and *Problem_IncreasedPeople_uav_problem_u1_r0_l10_p30_c20_g6_ct2.pddl.* Their outputs from running are also present in the folder Output_Lab1.2 with similar names.

The run time increase when the number of people are increased is less than the run time increase when the number of crates are increased. This might be because of the fact that there are more precondition, effects and predicates that are using crates when compared to people. So an increase in crates corresponds to a larger increase in the search space.

One more fact that was noticed was that an increase in the number of UAV's caused a huge increase in the search time. This might also be because of the above mentioned reason.

Lab 1.3. Newer Planners

Question 1)

The problem that was run in 1 minute by IPP was run using the modern planners and it had the following run times before the first solution was generated

1. *lama-2011* - 0 seconds (I think less the search time is in between 0 - 0.1 seconds)
2. *madagascar-p* - 0.05 seconds

3. *jasper (ipc 2014)* - 0 seconds (I think less the search time is in between 0 - 0.1 seconds)
4. *cerburs (ipc 2018)* - 0.0222303 seconds

The outputs and plans from running these planners on the old problem can be found in the folder Output_Lab1.3 with the name containing *"…old_uav_problem…"* and the planner name.

Question 2)

Larger problems were created and tested for the newer planners and it was found that the newer planners could solve the following problems around 1 minute

1. *lama-2011* - Uav - 2, location - 99, people - 70, crates - 80 and goals - 70
2. *madagascar-p* - Uav - 2, location - 112, people - 82, crates - 82 and goals - 82
3. *jasper (ipc 2014)* - Uav - 2, location - 117, people - 97, crates - 97 and goals - 97
4. *cerburs (ipc 2018)* - Uav - 2, location - 68, people - 60, crates - 60 and goals - 60

The problem definitions and outputs are named with "…Max1min…" along with the planner name and are present in the repository and in the Output_lab1.3 folders.

Lab 1.4. Theory: State Space and Branching Factor

Question 1)

There are five objects in the problem instance. Since the domain in untyped the predicates parameters can be assigned to any of the objects to obtain a state. Hence the following combinations are possible for the predicates

- *at* - has $2^{5*5} = 2^{25}$ combinations
- *has-photo-of* - has 2^5 combinations
- *photo-possible* - has $2^{5*5} = 2^{25}$ combinations
- *uav* - has 2^5 combinations
- *location* - has 2^5 combinations
- *target* - has 2^5 combinations

So the total number of combinations gives the total number of state = 2^{70} states

Question 2)

Generalising given U UAV's, T targets and L locations. O = U+T+L. Then there are possibly the following combinations

- *at* - has 2^{O*O} combinations
- *has-photo-of* - has 2^O combinations
- *photo-possible* - has 2^{O*O} combinations
- *uav* - has 2^O combinations
- *location* - has 2^O combinations
- *target* - has 2^O combinations

Then the total number of state = $2^{2(O*O) + 4O}$

Question 3)

Since the domain is typed we can only assign the specific type of objects to the parameters in the predicates. Hence the following combinations are possible for the small problem with 1 UAV, 2 locations and 2 target,

- *at* - has $2^{1*2} = 2^2$ combinations (Since we have 1 UAV and 2 locations)
- *has-photo-of* - has 2^2 combinations (Since we have 2 target)
- *photo-possible* - has $2^{2*2} = 2^4$ combinations (Since we have 2 locations and 2 target)

Then the total number of possible states = 2^8 states

Generalising to U UAV's, T targets and L locations. We can see that we have the following possible combinations,

- *at* - has 2^{U*L} combinations (Since we have U UAV's and L locations)
- *has-photo-of* - has 2^T combinations (Since we have T target's)
- *photo-possible* - has 2^{L*T} combinations (Since we have L locations and T target)

Then the total number of possible states = $2^{U*L + T + L*T}$

Question 4)

Assuming that the domain is typed and there are U UAV's, L locations and T targets. Assuming the current state is valid the following possible actions are possible,

- *Move* - U * L * 1 possible combinations (All U UAV's can be moved, to L - 1 locations from 1 locations. It can only be moved to L -1 locations because the UAV is already at a location. It is assumed that movement from a location to the same location is not considered a valid action).

- *take-photo* - U * 1 * T possible combinations (Assuming an optimistic assumption to photo-possible it can be through that a photo of any target can be taken from any location. Hence U uavs can take photos from the locations they are at currently to T targets. U * 1 is done because take-photo action can only done from the location at the UAV is currently and the UAV can only be at one location at a time.)

Therefore the total number of possible actions are = (U * L * 1) + (U * 1 * T)