REPORT

1. INTRODUCTION

Report of the first assignment of AIRO2 in the Robotics Engineering course.

We were asked to model a scenario in which three robots plan their actions for moving the crates from their locations towards the conveyor belt located in a warehouse.

2. METODOLOGY

After having read the specification file, we reasoned about the best planning engine to use. Among many engines we selected LPG because supports PDDL 2.1, particularly we were interested on:

- *Durative action*, used for modelling temporal relationships in the actions of the domain.
- *Fluents*, numeric variables shared between the actions, its value is changed on-the-fly by actions.

Every member of the group worked on his domain and after having presented it we discussed the:

- Feasibility of the solution.
- Level of abstraction.
- Optimality.
- Number of states generated.

We have decided the best plan, completed the work with additional extensions and thought about possible improvements (see final paragraph).

3. DOMAIN

In the domain, we declared types, predicates and actions for defining, distinguish and represent partially the space we need for plan the problem

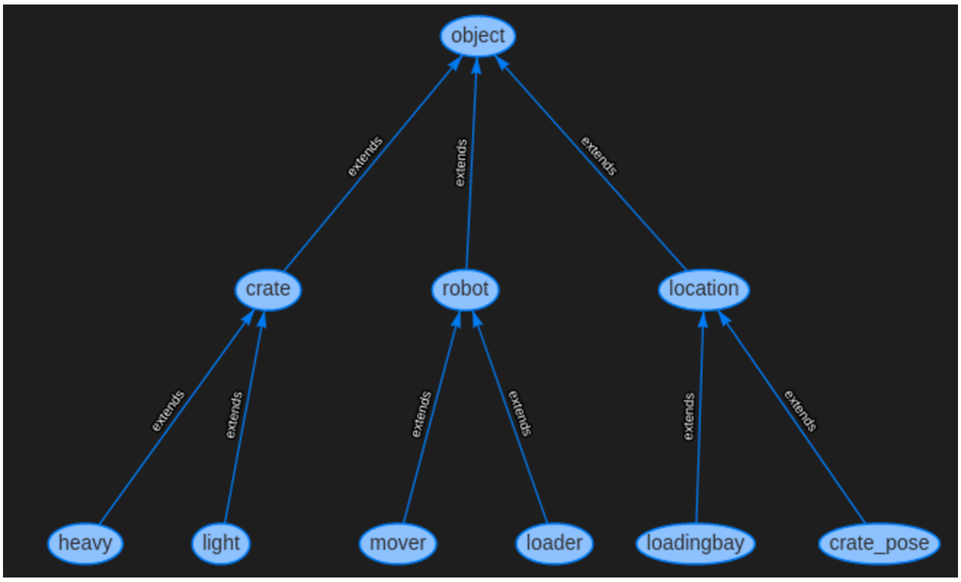

FIGURE 1 – TYPES TREE

3.1 TYPES

- Agents: movers and loaders.
- Locations: loading bay and the other crates positions.
- Crates: heavy or light

3.2 PREDICATES

- For locating the agents and objects in the environment: robot_at, crate_at, crate-at-conveyor.
- For expressing the state: empty-robot, holding-robot, free, busy, crate-carried, carry-2carried
- For distinguish the objects: fragil, not-fragil, group-a, group-b, active-a, active-b, active-c

3.3 FUNCTION

- Weight of the crates.
- extra-time: allow to fit problems with fragile crates with the others, its value modify the duration formula of some actions.
- loader-capability: for distinguish the liftable weight of loaders.
- count: number of crates that belong to a specific group.
- (i) (j) (k): counters for groups.
- timeunit : The cost (time of motion) that will be used in a metric to be minimized and getting optimal solution.
- mover_battery: for define battery charge

3.4 ACTIONS

	Duration	**Precondition**	**Effect**
move_empty	$\frac{distance}{10}$	- Mover at loading bay -Enough Battery	- Mover in crate pose - Update timeunit
move_light_1r	$\frac{distance\ weight}{100}$	- Mover at crate pose - The crate is light - The mover is not empty -Enough Battery	- Mover at loading bay - Update timeunit -decrease battery
move_light_2r	$\frac{distance\ weight}{150}$	- Movers at crane pose - The crate is light - The movers share the same crane - Movers are different - The mover is not empty -Enough Battery	- Movers at loading bay - Update timeunit -decrease battery
move_heavy	$\frac{distance\ weight}{100}$	- Movers at crate pose - The crane is heavy - The movers share the same crane - Movers are different - The mover is not empty -Enough Battery	- Movers at loading bay - Update timeunit -decrease battery

- Actions for moving the movers:

- Actions for picking the crates:

	Precondition	Effect
pickup_light	- Movers at crane pose - Mover is empty - Crate is light - Crate doesn't belong to any group	- Mover hold crane - The crate is not in crate pose Increase the counter "k" of 1 unite
pickup_light_a	- Movers at crate pose - The crate is light - Movers are empty - The crate belongs to group a	- Movers hold the same crane - The crate is not in crane pose - Increase the counter "i" of 1 unite
pickup_light_b	- Movers at crate pose - The crate is light - Movers are empty - The crate belongs to group b	- Movers hold the same crane - The crate is not in crane pose - Increase the counter "j" of 1 unite
pickup_2light	- Movers at crate pose - Movers are empty - The crate is light and fragile	- Movers hold the same crate - The crate is not in crate-pose - Increase the counter "k" of 1 unite
pickup_2light_B	- Movers at crate pose - Movers are empty - The crate is light and fragile - Crate belongs to group B	- Movers hold the same crate - The crate is not in crate-pose - Increase the counter "j" of 1 unite
pickup_2light_A	- Movers at crate pose - Movers are empty - The crate is light and fragile - Crate of group A	- Movers hold the same crate - The crate is not in crate-pose - Increase the counter "i" of 1 unite
pickup_heavy	- Movers at crate pose - The crate is heavy - Movers are empty - The movers share the same crate	- Movers hold the same crane - The crate is not in crane pose - Increase the counter "k" of 1 unite
pickup_heavy_A	- Movers at crate pose - The crate is heavy - Movers are empty - The movers share the same crate - Crate of group A	- Movers hold the same crane - The crate is not in crane pose Increase the counter "i" of 1 unite
pickup_heavy_B	- Movers at crate pose - The crate is heavy - Movers are empty - The movers share the same crate - Crate of group B	- Movers hold the same crane - The crate is not in crane pose Increase the counter "j" of 1 unite

- Actions for drop the crates:

	Precondition	Effect
Drop_1	- Mover at loading bay - Mover carry a crate	- Crate at loading bay - Mover is empty
Drop_2	- Movers at loading bay - Movers are empty - Movers are different	- Crate at loading bay - Mover is empty

- Action load:

	Duration	Precondition	Effect
Load	4 + extra-time	- The loader and crate both in loading bay - The crate weight is liftable by the loader	- Crate at the conveyor belt - Crate become busy until the end of the action - Update timeunit

- Actions for monitoring the states:

	Precondition	Effect
Change_group	[i , j, k] > = [count_A, count_b, count_no]	-Assign I,j,k equal to zero - active_a,active_b, active_c equal True

- Action for charging

	Duration	Precondition	Effect
charge	1	- mover at loading bay - mover without the crates	- Increase timeunit of one - At the end mover battery is 22

4. PROBLEM

This paragraph want provide a visual hint for better understanding the problem's codeIn every problem we defined every predicate, objects and function according to the specification in the "init" of the problem files. The goals are always the same: place crates at the conveyor belt. And the metric (or policy) is to minimize timeunit.

4.1 PLAN VISUALIZATION OF PROBLEM 1

Below the objects. We tried to minimize the number of them, for make faster the search.

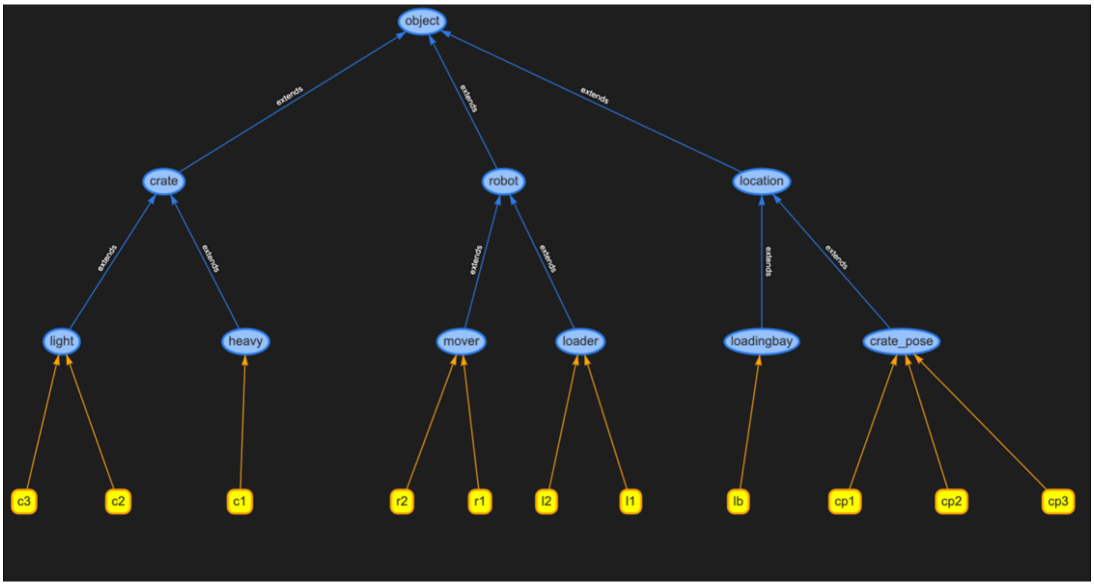

FIGURE 2 – OBJECT TREE

Below the numeric fluents added for the extentions . In general consider also distances and weight for each crate.

	value
count-a	2
count-b	0
count-no	1
extra-time	undefined
i	0
j	0
k	0
timeunit	0

loader	free	busy	loader-capability
l1	true		200
l2	true		50

FIGURE 4 – Loader-capability

FIGURE 3 – Numeric Fluents

Loader capability equal to 50 in figure 4 is the new loader bought by the company, which cannot load heavy crates.

It is important represent location of each object, in our abstraction we consider position of crates and robots. The conveyor belt is not a type as in the other case, but a predicate which is true after the action “load”.

crate \ location	lb	cp1	cp2	cp3
c1		crate-at crate-pose		
c3				crate-at crate-pose
c2			crate-at crate-pose	

FIGURE 5 – Initial position of crates

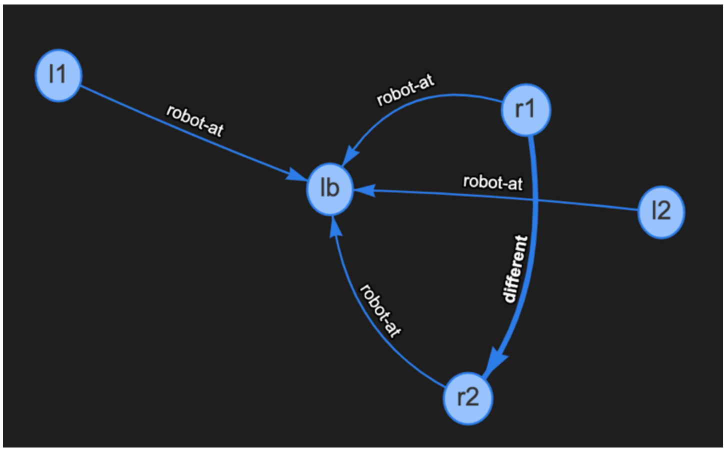

FIGURE 6 – Initial position of robots.

For convenience it is represented the only problem 1. The problems are almost similar, for further info look the problems file.

5. EXTENSIONS

Every action adds to a fluent (timeunit) its duration and in the metric of the domain we tell the planner to follow this policy, make timeunit as little as possible. Unfortunately, this method did not work and we decide to delete the fluent.

5.1 STANDARD VERSION

For being consistent with the change of state of the agents, we solve the standard problem with four types of motions, two types of drops and one load action. In the motion, we consider the case of the mover moving carrying no crates, carrying one crate light, carrying on light crate with two movers, and carry one heavy crate with two movers. The drops actions consider the fact to change the state of only one robot or two robots. And the load only works if the crate is on the loading bay without caring about the weight. Light and heavy are “son-types” of crates, distance and weight are property of each crate so they are fluents initialize in the problem with the right values.

The result is optimal in terms of durations, we wanted to insert “macros” if the planner would support it, because we understood some actions for example move-pick-drop are quite recursive in the plan.

5.2 EXTENSION - TWO LOADERS

For this extension we added another loader which cannot lift heavy crates. For this purpose we defined the fluent “load-capability ?loader” initialized with two different value for the two different loaders. As condition in the load action, the planner control out that the weight of the crate is less than the load-capability, and in this way it worked.

5.3 EXTENSION – FRAGILE CRATE

Predicates fragile and not fragile distinguish crates, we added another fluent “extra time” which it is initialize at the beginning. Extra-time take into account that the duration in load action is different in base of the crates (from 0 to 6), with this fluent we stay on the configuration with one and only load action. The (fragile and not-fragile) are added in the condition of pickup-light-2r and move_light_2r in order to force the planner to pick and move fragile crates with two robots.

5.3 EXTENSION – BATTERY

For solving this extension, we added the action “charge” which increase the fluent “battery -capacity” in the effect. In every motion action, the planner engine makes sure that the battery charge of movers is enough to move from one to other locations and due to the cost of moving the battery-capacity is decreased as specification. If the battery is not enough, the mover will go to the loading bay for charging.

In the problem 3, we notice that 20 as battery-capacity is not enough to move and solve the problem. For this reason, we defined as battery capacity 22.

5.4 EXTENSION – CRATES GROUP

For solving this extension, we added many actions and predicate. Compared with the standard version the actions pass from 6 to 18 actions. We used (group ?crate) ro say if crate belongs to A,B or NO group. We used count_a, count_b to initialize the number of crates and the counters i ,j and k for keeping track of crates moved in each group. The counters are incremented each time we pickup a crate.

(i) -> pickup_A

(j) ->. Pickup_B

(k) ->. Pickup

6. OPTIMALITY

We worked on a excel sheet which allow us to determine by brute force the optimality of the problems. Optimality is in terms of duration of actions. Thanks to this sheets we compared the result with ours and unfortunately they are not optimal.

PROBLEM 1:

	Fragile?	weight	distance	group	durations		
					reach the crate	pick and return	load
c1	no	70	10	none	1	move_heavy=7	4
c2	yes	20	20	A	2	move_2= 2.66	6
c3	no	20	20	A	2	~~move_1= 4~~ ~~or~~ move_2= 2.66	4

Optimal result : 21.32

Our Result : 22.66

PROBLEM 2:

	Fragile?	weight	distance	group	durations		
					reach the crate	pick and return	load
c1	no	70	10	A	1	move_heavy=7	4
c2	yes	80	20	A	2	move_heavy=16	6
c3	no	20	20	B	2	move_1= 4 or move_2= 2.66	4
c4	no	30	10	B	1	move_1= 3 or move_2= 2	4

Optimal result : 35.6

Our Result : 40

PROBLEM 3:

	Fragile?	weight	distance	group	durations		
					reach the crate	pick and return	load
c1	no	70	20	A	2	move_heavy=14	4
c2	yes	80	20	A	2	move_heavy=16	6
c3	no	60	30	A	3	move_heavy=18	4
c4	no	30	10	none	1	~~move_1= 3~~ ~~or~~ move_2= 2	4

Optimal result : 62

Our Result : 63

PROBLEM 4:

	Fragile?	weight	distance	group	durations			
					reach the crate	pick and return	tot	load
c1	no	30	20	A	2	move_1= 6 or move_2= 4	6	4
c2	yes	20	20	A	2	move_2=2,66	4,66	6
c3	yes	30	10	B	1	move_2=2	3	6
c4	yes	20	20	B	2	move_2=2,66	4,66	6
c5	yes	30	30	B	3	move_2=6	9	6
c6	no	20	10	none	1	move_1= 2 or move_2= 1,33	2,33	4

Optimal Result : 33.7

Our Result : 60.7

AUTHORS

- Sinatra Gesualdo
- Zhouyang Hong
- Samuele Depalo
- Abdelghani Bakour

GROUP H