AI for Robotics II

Assignment 1: AI Planning

Deadline: April 30, 2021

Your task is to model a robotic coffee shop scenario from the description that follows. There are many ways to model this scenario. You should strive to be as efficient as you can with your models.

In Japan, a few coffee shops are using robots as waiters and baristas. However, as the number of customers increases, the shops realise that they need to efficiently and effectively plan the way in which robots perform their tasks, to properly run the shop. One shop decided to approach you, as a group of AI planning and robotic experts, to get some help on this matter. You will work for free, but you may be allowed to pass part of the AI for Robotics II exam in exchange.
The owner is looking for the best possible models and design, so he will analyse all the submitted models to identify the best one. The group that provided the best model will be awarded some extra marks.

In a Nutshell

To run a coffee shop, the barista robot needs to prepare the drinks ordered by the costumers. Cold drinks are faster to prepare than warm drinks. The waiter robot is then in charge of serving customers, and to clean tables where customers have already left the shop. The waiter robot can decide to grasp a drink with its gripper, or to use a tray to carry more drinks at once. When using the tray, the robot is moving slower to improve balance.

Details

In the coffee shop, there are two robots: one *barista* and one *waiter*. The barista is in charge of preparing drinks. Orders are placed by customers by using a dedicated interface at the entrance of the shop. You don't need to model that, and you can assume that, at the start of your planning problem, all the orders are known. The customer that put the order is of course known, as it is the place where she sits.
It takes the barista robot 3 time units to prepare a cold drink, and 5 time units to prepare a warm drink.

Once ready, the drinks are put on the bar, where the waiter can pick them up. The waiter robot can grasp a single drink using one of its grippers, and bring it to the table where the corresponding customer is seated. The waiter is not able to grasp one drink per each gripper, but can only bring a drink at a time if it is not using a tray. If it decides to use a tray, then the waiter can carry up to 3 drinks at the same time, but its moving speed is reduced – for ensuring everything is balanced. When using a tray, the waiter cannot carry any additional drink, beside the 3 on the tray. The tray can be taken from the bar, and must be returned there after use. The waiter is not allowed to leave the tray on a table.
The waiter moves at 2 meters per time unit; 1 meter per time unit if it is using the tray.

Finally, the robot has to clean tables: it takes 2 time units per square meter to clean a table. The robot cannot clean a table while carrying the tray.

Optional extensions

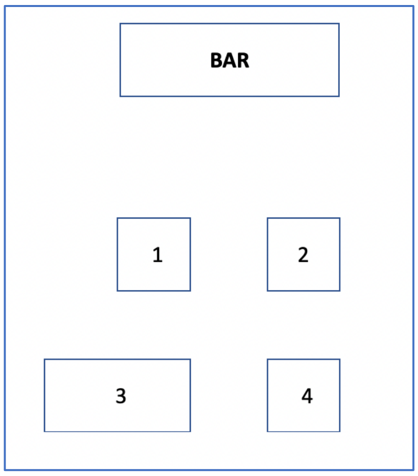
There are a few optional ways in which the model can be extended, to improve your overall final evaluation:

1. *Warm drinks cool down*, so you need to serve the customer before the drink gets too cold. A warm drink takes 4 time units to become cold. A customer will not accept a warm drink delivered more than 4 time units after it was put on the bar by the barista.
2. *There are 2 waiters.* The coffee shop is doing well, so the owner decided to buy a second waiter. This is of course making things a bit more complicated: only one waiter can be at the bar at a given time; further, the owner does not want that a table is served by both the waiters: only one waiter can deal with the orders of a given table.
3. *Finish your drink*. After receiving the drink, a customer will finish it in 4 time units, and will leave after that. When all the customers of a table have left, the waiter robot can clean it. The waiter robot is required to clean all the tables to complete the problem.
4. *We also serve food*. The coffee shop is also serving delicious biscuits. They need no preparation, and can be picked up by the waiter at the bar counter. All the customers with cold drinks will also receive a biscuit, but only after having received their drink.

For the sake of moving biscuits around, the same limitations as for drinks apply (see above).

You can include one or more of the extensions in your model.

Problems to be encoded



The coffee shop layout is shown in the figure above. It has the bar counter on the very top, and 4 tables for costumers. Each table is 1-meter apart from any other (let's assume Euclidean geometry does not apply here, so also tables 1 and 4 are 1 meter from each other). The bar is 2 meters away from tables 1 and 2. Table 3 is the only table of 2 square metres, all the others are of 1 square metre.

- Problem 1: There are 2 customers at table 2: they ordered 2 cold drinks. Tables 3 and 4 need to be cleaned.
- Problem 2: There are 4 customers at table 3: they ordered 2 cold drinks and 2 warm drinks. Table 1 needs to be cleaned.
- Problem 3: There are 2 customers at table 4: they ordered 2 warm drinks. There are also 2 customers at table 1: they ordered 2 warm drinks. Table 3 needs to be cleaned.
- Problem 4: There are 2 customers at table 4 and 2 customers at table 1: they all ordered cold drinks. There are also 4 customers at table 3: they all ordered warm drinks. Table 4 needs to be cleaned.

Additional information

This submission tests your understanding of planning models, and of operational use of planning engines. You must produce a PDDL domain file and the corresponding planning problem files. You are free to choose the version of PDDL that seems more appropriate to you. Make sure that the selected version of PDDL can represent all the constraints and

numeric / temporal aspects. Also, the version of PDDL will have an impact on how you can represent the layout of the coffee shop: it is up to you to decide the best way to model the layout (or to avoid modelling it explicitly). Just make sure that all the constraints and distances can be represented.
The problems must be solvable by a state-of-the-art planning engine. Suggestions about the planning engines to use:

- PDDL+/ Numeric planning: try ENHSP https://sites.google.com/view/enhsp/ (bear in mind it does not support durative actions)
- Numeric and Temporal planning: try MetricFF https://fai.cs.uni-saarland.de/hoffmann/metric-ff.html (it can be pretty picky with regards to the way in which models are encoded, so be patient) or LPG https://lpg.unibs.it/lpg/
- Classical planning: Madagascar https://research.ics.aalto.fi/software/sat/madagascar/ can work well

Feel free to use any configuration of the planning engine that you believe works well, you are not required to stick to the default configuration.

You should submit a zip file including the following:

- A pdf copy of your report.
- The domain model and the problem models, in PDDL.
- The output of the planning engine used.

For the report, you are encouraged to use a concise writing style. The layout of the report is up to you; however, it must provide the following details in a logical order.

- A list of the names and/or URLs of existing PDDL domain models (if any) that you have used to help you create your PDDL models;
- A short description of the meaning of each component of the model, even if some were imported from some other domain model;
- A short analysis of the performance of the planning engine you have used to generate solutions. Try to look into aspects such as the nodes generated, the ground size, etc.