AI for Robotics II

Your task is to model an automated warehouse 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.

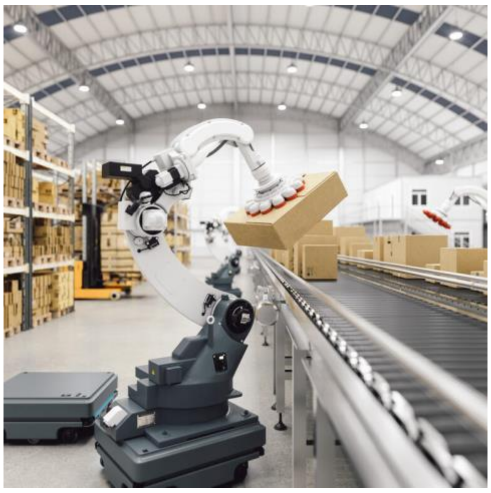

Figure 1 (imaginima/E+ via Getty Images)

A company is interested in demonstrating how an automated warehouse solution would work for their business. They 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 company is looking for the best possible models and design, so they 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

In the considered warehouse, there are two robots that move crates around. The crates are to be loaded on the conveyor belt by an additional robot (as in the picture above). The idea is that when the warehouse receives new orders, the corresponding crates to be moved are communicated to the robots, that take them to a loading bay where a dedicated robot can put them on the conveyor belt. Usually, one robot is enough to move light crates in the warehouse. However, in the presence of heavy (>50kg) crates, two robots are required.

Details

In the warehouse, there are three robots: one *loader* and two "*mover*". The loader is in charge of loading crates on the conveyor belt. It cannot move, as it is basically only a robotic arm. The mover robots are in charge of moving crates from their initial position in the warehouse to the loading bay, so that they can then be loaded on the conveyor belt by the loader. Crates to be loaded on the conveyor belt are selected a-priori via a dedicated company system. You don't need to model that, and you can assume that, at the start of your planning problem, all the crates to be loaded on the conveyor are known.
The weight and initial position of each crate are of course known. The position of a crate is only given in terms of its distance from the loading bay. You can assume that distances are always straight line distance, and that there is no risk of interference between the paths of different crates to the loading bay.
Light crates, which weigh less than 50kg, can be moved around by a single mover robot. The time needed to move a crate around depends on its weight, and can be calculated as: *distance * weight / 100*. So for instance, a crate that is initially positioned 20 units from the loading bay and weights 40kg, can be moved by a mover to the loading bay in 20*45/100 = 9 time units.
Heavy crates, which weigh more than 50kg, need 2 mover robots. The time needed to move them is to be calculated as for light crates.
It should be noted that also light crates can be moved around by 2 mover robots together. In that case, the time needed to move a light crate around is calculated as *distance * weight / 150.* For the crate considered in the example (20 units, 40kg) the required moving time would then be 20*45/150 = 6.
When moving around with no crates, mover robots cover 10 distance units per time unit. At the start of the planning problem, the mover robots are both at the loading bay.

It takes the loader robot 4 time units to load a crate on the conveyor belt, and it can load a single crate at a time. The loading bay must be kept free while the loader is in the process of loading a crate on the belt. In other words, no crates can be put on the loading bay during the 4 time units mentioned above.

Optional extensions

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

1. *Some crates go together*, some of the crates from the warehouse need to be loaded subsequently on the conveyor belt to help the delivery guy. Those that need to go together are characterised by the same letter, and it is indicated in the corresponding problem.

2. *There are 2 loaders*. The company decided to invest in a second loader to help the loading process on the conveyor belt. This second loader uses the same loading bay as the other one, and can be used while the other one is loading. However, the company decided to invest in a cheap robotic arm, that is not capable of loading heavy crates.
3. *Mover robots need recharging*. The mover robots have a limited battery capacity of 20 power units. A robot consumes a power unit for each time unit in which it is actively doing something (moving around or moving crates). To recharge, a robot needs to go to a recharging station that is positioned at the loading bay. The recharging process does not interfere with loading processes. *Please be aware that this extension can lead to problems that are quite challenging to be solved for the planning engine*.
4. *This is fragile!* Some crates are fragile and needs extra care when being moved around. In particular: fragile crates always need 2 movers to be taken to the loading bay, and the loader robot works at reduced speed to avoid any potential damage. This means that loading a fragile crate on the conveyor belt takes 6 time units instead of the usual 4.

You can include one or more of the extensions in your model. You can also consider whether some of the reformulation techniques discussed in class can be applied to your model, and investigate if they are beneficially impacting performance or not.

Problems to be encoded

As mentioned above, the movers always start from the loading bay, and distances for crates are provided as straight line distances. There is no risk of interference on the paths between different crates and the loading bay.

- Problem 0.5 (**needed only for Part II**): There are 2 crates:
  - crate, weight 70kg, 10 distance from loading bay.
  - crate, weight 20kg, 20 distance from loading bay. Crate in group A for extension 1
- Problem 1: There are 3 crates:
  - crate, weight 70kg, 10 distance from loading bay.
  - Fragile crate, weight 20kg, 20 distance from loading bay. Crate in group A for extension 1.
  - crate, weight 20kg, 20 distance from loading bay. Crate in group A for extension 1
- Problem 2: There are 4 crates:
  - crate, weight 70kg, 10 distance from loading bay. Crate in group A for extension 1.

    - Fragile crate, weight 80kg, 20 distance from loading bay. Crate in group A for extension 1.
    - crate, weight 20kg, 20 distance from loading bay. Crate in group B for extension 1
    - crate, weight 30kg, 10 distance from loading bay. Crate in group B for extension 1
- Problem 3: There are 4 crates:
    - crate, weight 70kg, 20 distance from loading bay. Crate in group A for extension 1.
    - Fragile crate, weight 80kg, 20 distance from loading bay. Crate in group A for extension 1.
    - crate, weight 60kg, 30 distance from loading bay. Crate in group A for extension 1
    - crate, weight 30kg, 10 distance from loading bay.
- Problem 4: There are 6 crates:
    - crate, weight 30kg, 20 distance from loading bay. Crate in group A for extension 1.
    - Fragile crate, weight 20kg, 20 distance from loading bay. Crate in group A for extension 1.
    - Fragile crate, weight 30kg, 10 distance from loading bay. Crate in group B for extension 1
    - Fragile crate, weight 20kg, 20 distance from loading bay. Crate in group B for extension 1.
    - Fragile crate, weight 30kg, 30 distance from loading bay. Crate in group B for extension 1
    - crate, weight 20kg, 10 distance from loading bay.

Additional information

This part of the submission tests your understanding of planning models, and of operational use of planning engines. You must produce a single 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, in other words, make sure that all the constraints and distances can be represented appropriately. You can take inspiration from existing PDDL benchmarks and models.
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)

- Check planners available and supported by the Unified Planning Framework https://github.com/aiplan4eu/unified-planning
- Numeric and Temporal planning: try LPG https://lpg.unibs.it/lpg/ (**ask Mauro for a version of LPG if the one you download from the above link does not work properly**)
- Patty for Planning as Satisfiability with Patterns (only for numeric and temporal planning): https://github.com/matteocarde/patty (**ask Matteo for a version of Patty if the one you download from the above link does not work properly**)

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.

Part II: Solving, Heuristics, and Patterns

Once you have demonstrated that the models can be solved using an existing planning engine or system, it is your turn to design approaches that can help in solving problems from the considered domain. Specifically:

- You should provide the pseudocode for an heuristic that you believe could be beneficial in assessing the quality of states and in driving search towards a goal. Focus on problem 0.5 to show how it works. Provide a description on how you would compute the heuristic function h: S -> N, which, given a state, returns a number. Discuss on why the heuristic is useful. For instance by considering cases where given two states S1 and S2, with S2 closer to a goal state, it guarantees that h(S2) < h(S1). Provide an intuition on why the heuristic should/should not be admissible. Please bear in mind that heuristics should be cheap to compute yet informative. **There is no need to actually implement the heuristic**.
- You should provide a definition of a pattern for problem 0.5. Write a simple pattern (i.e. without repetitions) that would cover the plan. Discuss what would be the bound if you used the pattern in a symbolic approach for the handcrafted domain. What about the other larger problems? Would the bound be always the same or would it increase? How would you change the pattern in those cases?

What to submit

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 generated by the planning engine.

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.;
- The designed heuristic, with pseudocode and description;
- The pattern, with description and discussion.