Assignment for the course
Automated Planning Theory and Practice
Academic Year 2022-2023

Marco Roveri
marco.roveri@unitn.it

Contents

1 Introduction

The aim of this assignment is twofold. First to model planning problems in PDDL/HDDL to then invoke a state of the art planner as those provided by `planutils` [5] or manually compiled (e.g., `fast downward` [2], or `optic` [1]). Second, to see how a temporal model could be integrated within a robotic setting leveraging the `PlanSys2` [4] infrastructure discussed in the lectures and available at https://github.com/PlanSys2/ros2_planning_system.

This assignment can be performed alone or in group of *at most two* students.

2 The scenario

Let us consider a scenario inspired by an emergency services logistics problem, where a number of persons at known a priori locations (and not moving) have been injured. The objective of the planning systems is to orchestrate the activities of a set of different robotic agents to deliver boxes containing emergency supplies to each person.

Let's consider these assumptions:

- Each injured person is at a specific location, and does not move.
- Each box is initially at a specific location and can be filled with a specific content such as *food* or *medicine* or *tools*, if empty. Box contents shall be modeled in a generic way, so that new contents can easily be introduced in the problem instance (i.e., we shall be able to reason on the content of each box).

- Each person either has or does not have a box with a specific content. That is, you must keep track of whether they have food or not, whether they have medicine or not, whether they have tools or not, and so on.
- There can be more than one person at any given location, and therefore it isn't sufficient to keep track of where a box is in order to know which people have been given boxes!
- Each robotic agent can:
  - fill a box with a content, if the box is empty and the content to add in the box, the box and the agent are at the same location;
  - empty a box by leaving the content to the current location, and causing the people at the same location to have the content;
  - pick up a single box and load it on the robotic agent, if it is at the same location as the box;
  - move to another location moving the loaded box (if the box has been loaded, otherwise it simply moves);
  - deliver a box to a specific person who is at the same location.
- The robotic agents can move directly between arbitrary locations (there is no ”roadmap” with specific connections that must be taken), so the graph is fully connected.
- Since we want to be able to expand this domain for multiple robotic agents in the future, we expect to use a *separate type* for robotic agents, which currently happens to have a single member in all problem instances.

3 The Problems

The assignment is structured in 5 sub problems, the second build on the first, both the third and the fourth build on the second, the fifth builds on the fourth.

3.1 Problem 1

3.1.1 Initial condition

- Initially all boxes are located at a single location that we may call the depot.
- All the contents to load in the boxes are initially located at the depot.
- There are no injured people at the depot.
- A single robotic agent is located at the depot to deliver boxes.

There are no particular restrictions on the number of boxes available, and constraints on reusing boxes! These are design modeling choices left unspecified and that each student shall consider and specify in her/his solution.

3.1.2 Goal

The goal should be that:

- certain people have certain contents (e.g., medicine, food, tool);
- some people might not need food, medicine, or tool;
- some people might need both food and medicine, or food and tool, or the three of them, and so on.

This means that the robotic agent has to deliver to needing people some or all of the boxes and content initially located at the depot, and leave the content by removing the content from the box (removing a content from the box causes the people at the same location to have the content).

Remarks people don't care exactly which conten they get, so the goal should not be (for example) that Alice has banana2, merely that Alice has a banana.

3.2 Problem 2

We leverage the scenario in Section 3.1 with the following extensions, where we will have to consider an alternative way of moving boxes (e.g., trucks or helicopter or drones).

- the robotic agent can load up to four boxes onto a carrier, which all must be at the same location;
- the robotic agent can move the carrier to a location where there are people needing supplies;
- the robotic agent can unload one or more box from the carrier to a location where it is;

- the robotic agent may continue moving the carrier to another location, unloading additional boxes, and so on, and does not have to return to the depot until after all boxes on the carrier have been delivered;
- though a single carrier is needed for the single robotic agent, there should still be a separate type for carriers;
- for each robotic agent we need to count and keep track of i) which boxes are on each carrier; ii) how many boxes there are in total on each carrier, so that carriers cannot be overloaded.
- the capacity of the carriers (same for all carriers) should be problem specific. Thus, it should be defined in the problem file.

It might be the case additional actions, or modifications of the previously defined actions are needed to model loading/unloading/moving of the carrier (one can either keep previous actions and add new ones that operates on carriers, or modify the previous actions to operate on carriers).

The initial condition and the goal are the same as the problem discussed in Section 3.1.

3.2.1 Initial condition

- Initially all boxes are located at a single location that we may call the depot.
- All the contents to load in the boxes are initially located at the depot.
- There are no injured people at the depot.
- A single robotic agent is located at the depot to deliver boxes.
- The robotic agent is initially empty.
- Fix the capacity of the robotic agent to be 4.

3.2.2 Goal

The goal should be that:

- certain people have certain contents (e.g., medicine, food, tool);
- some people might not need food, medicine, or tool;
- some people might need both food and medicine, or food and tool, or the three of them, and so on.

3.3 Problem 3

We leverage the scenario in Section 3.2, but in this case we need to address the problem with hierarchical task networks (HTN). To this extent, it is suggested to keep the same actions as per the solution poposed for Problem 3.2, the same initial condition and the same goal. However, here the candidate shall introduce `:tasks` and `:methods` following approaches similar to those discussed in the Lab session on HTN.

3.4 Problem 4

We leverage the scenario in Section 3.2 with the following extensions.

- Convert the domain defined in Section 3.2 to use durative actions. Choose arbitrary (but reasonable durations) for the different actions.
- Consider the possibility to have actions to be executed in parallel when this would be possible in reality. For example, a robotic agent cannot pick up several boxes at the same time, or pick up a box and fly to a destination at the same time.

The initial condition and the goal are the same as the problem discussed in Section 3.2.

3.5 Problem 5

The final problem consists in implementing within the `PlanSys2` the last problem resulting as outcome of Section 3.4 using fake actions as discussed in the tutorials of `PlanSys2` available at [3].

4 Deliverables

The deliverable shall consist of a unique archive containing at least the following contents:

- The PDDL/HDDL files (domain and problems) for each of the problems discussed in Sections 3.1, 3.2, 3.3, and 3.4. All the PDDL/HDDL files shall be valid ones, and shall be parsable correctly by at least one planner (it shall be specified which one). The options used to run the chosen planner.
- A folder containing all the code to execute the `PlanSys2` problem as discussed in Section 3.5.
- A professional report in PDF describing the approach followed, justifying and documentin all the design choices (e.g., predicates, actions, ...), possible additional assumptions (for the part left underspecifed), and in case of deviation from the specification in this document a clear justification. Finally, the document shall include evidence of the attempts to solve the PDDL problems formulated, and running the final version within `PlanSys2` (in form of screenshots and/or output generated by the planner).
  - The report shall also discuss the content of the archive and how it has been organized.
  - The report shall also contain a critical discussion of the results obtained.

The submission shall be performed only through the web form available at the following URL:

Submission Form

`https://forms.gle/c2RhbTreJ62ZYwKj7`

References

[1] J. Benton, Amanda Jane Coles, and Andrew Coles. Temporal planning with preferences and time-dependent continuous costs. In Lee McCluskey, Brian Charles Williams, José Reinaldo Silva, and Blai Bonet, editors, *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI, 2012. URL `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4699`. (page 1)

[2] Malte Helmert. The fast downward planning system. *J. Artif. Intell. Res.*, 26:191–246, 2006. (page 1)

[3] Francisco Martín and colleagues. PlanSys2 Tutorials, 2022. `https://plansys2.github.io/tutorials/index.html`. (page 3)

[4] Francisco Martín, Jonatan Ginés, Francisco J. Rodríguez, and Vicente Matellán. PlanSys2: A Planning System Framework for ROS2. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - October 1, 2021*. IEEE, 2021. (page 1)

[5] Cristian Muise and other contributors. PLANUTILS, 2021. General library for setting up linux-based environments for developing, running, and evaluating planners. `https://github.com/AI-Planning/planutils`. (page 1)