Assignment for the course
Automated Planning Theory and Practice
Academic Year 2021-2022

Marco Roveri
marco.roveri@unitn.it

Contents

1 Introduction **1**

2 The scenario **1**

3 The Problems **2**
3.1 Problem 1 . . . 2
3.1.1 Initial condition . . . 2
3.1.2 Goal . . . 2
3.2 Problem 2 . . . 2
3.2.1 Initial condition . . . 2
3.2.2 Goal . . . 3
3.3 Problem 3 . . . 3
3.4 Problem 4 . . . 3

4 Deliverables **3**

1 Introduction

The aim of this assignment is twofold. First to model planning problems in PDDL to then invoke a state of the art planner as those provided by `planutils` [3]. Second, to see how the model could be integrated within a robotic setting leveraging the `PlanSys2` [2] infrastructure discussed in the lectures and available at https://github.com/IntelligentRoboticsLabs/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 locations have been injured. The objective of the planning systems is to orchestrate the activities of a set of robotic agents to deliver crates containing emergency supplies to each person.

Let's consider these assumptions:

- Each injured person is at a specific location.
- Each crate is at a specific location and has a specific content such as food or medicine. Crate contents shall be modeled in a generic way, so that new contents can easily be introduced in the problem instance.
- Each person either has or does not have a crate with a specific content. That is, you must keep track of whether they have food or not, whether they have medicine 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 crate is in order to know which people have been given crates!
- Each robotic agent can:
  - pick up a single crate and load it on the robotic agent, if it is at the same location as the crate;

  - move to another location moving the loaded crate;
  - deliver a crate 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).
- Since we want to be able to expand this domain for multiple robotic agents in the future, we 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 4 sub problems, each building on the previous one.

3.1 Problem 1

3.1.1 Initial condition

- Initially all crates are located at a single location that we may call the depot.
- There are no injured people at the depot.
- A single robotic agent is located at the depot to deliver crates.

3.1.2 Goal

The goal should be that:

- certain people have crates with certain contents;
- some people might not need crates
- some people might need both food and medicine, and so on.

This means that the robotic agent has to deliver to needing people some or all of the crates at the depot.

Remarks people don’t care exactly which crate they get, so the goal should not be (for example) that Alice has crate5, merely that Alice has a crate of food.

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 crates.

- the robotic agent can load up to four crates 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 crates from the carrier to a location where it is;
- the robotic agent may continue moving the carrier to another location, unloading additional crates, and so on, and does not have to return to the depot until after all crates on the carrier have been delivered;
- though a single carrier is needed for the single helicopter, there should still be a separate type for carriers;
- for each robotic agent we need to count and keep track of not only which crates are on each carrier but also how many crates 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 crates are located at a single location that we may call the depot.
- There are no injured people at the depot.
- A single robotic agent is located at the depot to deliver crates.
- 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 crates with certain contents;
- some people might not need crates
- some people might need both food and medicine, and so on.

3.3 Problem 3

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 crates at the same time, or pick up a crate 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.

3.4 Problem 4

The final problem consists in implementing within the `PlanSys2` the last problem resulting as outcome of Section 3.3 using fake actions as discussed in the tutorials of `PlanSys2` available at [1].

4 Deliverables

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

- The PDDL files (domain and problems) for each of the problems discussed in Sections 3.1, 3.2, and 3.3. All the PDDL files shall be valid ones, and shall be parsable correctly by at least one planner (it shall be specified which one).
- A folder containing all the code to execute the `PlanSys2` problem as discussed in Section 3.4.
- A (professional) report in PDF describing the approach followed, possible additional assumptions, and evidence of the attempts to solve the PDDL problems formulated, and running the final version within `PlanSys2`. The report shall also discuss the content of the archive and how it has been organized.

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

Submission Form

`https://docs.google.com/forms/d/1IDyKtsVFON6GCrbos2HXXfCOZnijfroRgTLD442rYvw`

References

[1] Francisco Martín and colleagues. PlanSys2 Tutorials, 2021. `https://intelligentroboticslab.gsyc.urjc.es/ros2_planning_system.github.io/tutorials/index.html`. (page 3)

[2] 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)

[3] 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)