Assignment for the course
Artificial Intelligence

Prof. Francesco Scarcello
Ing. Antonio Bono

Academic year 2022/2023

Introduction

The aims of this assignment are threefold:

Task 1: use the PDDL language to model a classical planning problem (7 points);

Task 2: implement your own search algorithm and heuristics for solving the provided instances of the problem (16 points).

Task 3: show how a temporal model could be integrated in a real-world robotic software architecture (7 points).

For the second task, students can implement the algorithm by whatever means they find most appropriate (an example is the `PDDL4J` library [3]). For the third task, the student will refer to `planutils` [2] to solve the problem and the `PlanSys2` [1] package presented during the lectures and available at https://github.com/PlanSys2/ros2_planning_system. All the mentioned software and tools are already installed in the VM provided in the Teams course page.

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

1 Modeling

Let us consider a scenario inspired by an emergency services logistics problem, where a number of people at known locations (and not moving) have been injured. The objective of the planning systems is to orchestrate the activities of one or more robotic agents to deliver boxes containing emergency supplies to each person. Let's consider these assumptions

1. Each injured person is at a specific location, and does not move.
2. Each box is initially at a specific location and can be filled with a specific content, such as food or medicine or tools, if it is 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).
3. Each person either has or does not have 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.

4. There can be more than one person at any given location, and therefore it isn't sufficient to keep track of where a content is in order to know which people have been given content!
5. Each robotic agent can:
   (a) 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;
   (b) empty a box by leaving the content to the current location to deliver it to a person;
   (c) the robotic agent can load up to n boxes onto a carrier, which all must be at the same location;
   (d) the capacity of the carriers (same for all carriers) should be problem specific. Thus, it should be defined in the problem file.
   (e) pick up a single box and load it on an auxiliary vehicle (carrier), if it is at the same location as the box;
   (f) move to another location moving the loaded vehicle, if the boxes have been loaded, otherwise it simply moves;
   (g) the robotic agent can move the carrier to a location where there are people needing supplies;
   (h) the robotic agent can unload one or more box from the carrier to a location where it is;
   (i) 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.
6. 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.

Model such a scenario using the PDDL 1.2 language.

2 Classical Planning

Use *your own planner* to solve the following problem instances. Your solution must report the amount of time used (max 10 minutes to be considered), the numbers of states evaluated and optionally the amount of used memory.

2.1 Instance 1

2.1.1 Initial Condition

- Initially all the five 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. Such contents are always sufficient to satisfy injured people needs.
- There three people: p1, p2 and p3; p1 and p2 are at the same location; p1 needs food and medicine, p2 needs medicine, p3 needs food.
- A single robotic agent and carrier are located at the depot.
- Each carrier can load up to 4 boxes.

2.1.2 Goal

The goal is that all injured people have the items they need.

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

2.2 Instance 2

Use your own planner to solve the same problem of Instance 1 with only the following differences

2.2.1 Initial Conditions

- 2 robots and 2 carriers.
- 3 boxes.
- Carrier capacity is 2.
- 6 people: p1 needs food or tools; p2 needs medicine ; p3 needs medicine; p4 need medicine and food; p5 and p6 needs everything.
- p1 and p2 are at the same location; the other ones are at different ones.

2.2.2 Goal

The goal is that all injured people have the items they need.

2.3 Instance 3 (bonus)

Use your own planner to solve the same problem of Instance 2 with only the following differences

2.3.1 Initial Conditions

- 4 boxes.
- 8 people: p1 needs food or tools; p2 needs medicine ; p3 needs medicine; p4 need medicine and food; p5, p6, p7 and p8 need everything.
- p1 and p2 are at the same location; the other ones are at different ones.

2.3.2 Goal

The goal is that all injured people have the items they need.

3 Temporal Planning & Robotics

3.1 Temporal Planning

We leverage the problem instance reported in paragraph 2.1 with the following extensions.

- Convert the domain defined in paragraph 2.1 to use durative actions. Choose arbitrary but reasonable durations for the different actions.
- Each item has a weight: drug 1; food 2; tools 3. The duration of actions that model movements must take into account these different weights by multiplying the time you chose for the movement action by the total weight of the carrier.

To solve this problem *you should use one of the temporal planners provided by* `planutils` (popf, tfd, etc.).

3.2 Robotics Planning

The final problem consists in implementing within the `PlanSys2` the last problem resulting as outcome of Section 3.1 using fake actions as discussed in the tutorials of `PlanSys2` available at [4].

Remark: To generate the plan do not use the inner temporal solvers of `PlanSys2` but rely on the plan you generated in the previous step. In such a way you do not have to face the current limitations of `PlanSys2` as the the inability to use hierarchical types in the domain definition. In order to use the generated plan in `PlanSys2` it must have the same sintax of a plan generated by popf, i.e.,

```
<time>:  (<action>) [<duration>]
```

Once you edit correctly your plan, it is sufficient to create the problem instance with the `PlanSys2` terminal commands (possibily sourcing a file with them listed) and then execute

```
run plan-file <your_plan.txt>
```

If everything is well setted, you will see the actions progress on the terminal.

4 Deliverables

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

1. The PDDL files (domain and problems) for each of the problems discussed in Sections 2.1, 2.3 and 3.1.
2. The *commented* code of your planner with the instructions to run in on the proposed problem instances.
3. A folder containing all the code to execute the `PlanSys2` problem as discussed in Section 3.2.
4. A report in PDF describing the approach followed, justifying and document in all the design choices of both the modeling and the solver, 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/uDtzecyMZaiUMzA86

References

[1] Francisco Martın et al. "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.

[2] Cristian Muise and other contributors. *PLANUTILS*. General library for setting up linux-based environments for developing, running, and evaluating planners. 2021. URL: https://github.com/AI-Planning/planutils.

[3] D. Pellier and H. Fiorino. "PDDL4J: a planning domain description library for java". In: *Journal of Experimental & Theoretical Artificial Intelligence* 30.1 (2018), pp. 143–176. URL: http://pddl4j.imag.fr/.

[4] Francisco Martin Rico and colleagues. *Plansys2 tutorials*. 2022. URL: https://plansys2.github.io/tutorials/index.html.