Assignment for the course Automated Planning Theory and Practice

Davide Lusuardi
Department of information engineering and computer science
University of Trento
Trento, Italy
davide.lusuardi@studenti.unitn.it

***Abstract*—This report is intended to present and analyse the assignment [1] for the course *Automated Planning Theory and Practice* and discuss some design choices regarding its modelling and implementation.**

The assignment is inspired by an emergency services logistics scenario where a set of robotic agents have the task to deliver crates containing emergency supplies to some injured people.

The assignment is divided into four subproblems, each building on the previous one with increasing complexity. In the first problem, the robot can pick up and move just one crate at time. In the second problem, the complexity increases a bit given that the robot exploits a carrier that can load up to four crates. In the third problem, it is required to use durative actions in order to assign reasonable duration to different actions and model which actions can be executed in parallel. Lastly, the third problem has to be implemented within the `PlanSys2` framework, executing the plan in a simulated environment.

In this report, we show how these problems can be formalized using the PDDL planning language and solved by state-of-the-art planners.

I. Introduction

Planning is a branch of Artificial Intelligence that seeks to automate reasoning about formulating a plan to achieve a given goal in a given situation. Planning is a model-based approach: a planning system takes as input a model of the initial state, the actions available to change it and the goal conditions, producing as output a plan that will reach the goal from the initial situation.

The Planning Domain Definition Language (PDDL) is a formal knowledge representation language designed to express planning models. Developed by the planning research community, it has become a de-facto standard language of many planning systems.

The purpose of the assignment is two-fold: first, to model the given scenario using the PDDL language and find a plan using a state-of-the-art planner; second, leveraging the `PlanSys2` [2] infrastructure, to integrate the model within a robotic setting.

ROS2 Planning System, `PlanSys2` in short, is a framework for symbolic planning that incorporates novel approaches for execution on robots working in demanding environments [2].

The proposed scenario is inspired by an emergency services logistic problem: several injured people are situated at known locations and one or more robotic agents have the task to deliver crates containing emergency supplies to each person. The planning system should formulate a plan for the robotic agents in order to deliver all the crates needed by the injured people.

The document is structured as follows: in Section II, we explain in details our understanding of the proposed scenario and the four subproblems; in Section III, we describe the design choices and the proposed solutions to solve the problem, making some assumptions in order to constrain the problem; in Section IV, we discuss and analyse the achieved results and briefly describe the content of the submitted archive; in Section V, we make some conclusions and future works are proposed.

II. Understanding of the problem

The assignment proposes four subproblems within the same scenario, each building on the previous one.

The proposed scenario consists of an emergency situation in which there are a number of injured people who need emergency supplies. Each injured person is located in a specific place and may need some kinds of emergency supply (e.g., food, medicine, beverage). There may be more people at the same location and some locations may be empty. Each crate is at a specific location and contains just one specific kind of emergency supply. Some people may already have some crates and some others may not need any crate. People don't care which crate exactly they get, only the content type is important. There can be one or more robotic agents that cooperate in order to deliver crates to people. Each location is connected to every other location, so the robotic agents can move directly between them. Initially, the robotic agents and the crates are situated at the depot, where there are not injured people.

The goal consists in delivering all the required crates to injured people.

The above considerations and goal are common for all the four subproblems that we are going to discuss.

A. Problem 1

This is the easiest problem. Just a single robotic agent is present at the depot and it can perform the following actions: pick up and load just one crate that is at the same location; move to another location, moving also the loaded crate if

present; deliver the loaded crate to a person that is at the same location of the robot.

B. Problem 2

In this subproblem, the robotic agent can move crates in a different way. We introduce the concept of carrier that can be loaded with up to a specific number of crates, in this case up to four. This number is problem specific and is modelled into the problem file. At a specific location, the robot can load the carrier with crates situated in that location and can deliver loaded crates to people. The robot can move to another location bringing a carrier with it or without moving any carrier.

Correctly modelling the carrier and its variable capacity has a central role in this problem.

C. Problem 3

In this subproblem, it's required to make use of durative actions. Reasonable durations should be assigned to actions coherently with their real time spans and the possibility to execute actions in parallel should be analysed taking into account real constraints.

D. Problem 4

In this subproblem, it's required to integrate the *Problem 3* within a robotic setting, leveraging the `PlanSys2` infrastructure.

`PlanSys2` is composed of four nodes: the *domain expert* that contains the PDDL model information; the *problem expert* that contains the problem information like current instances, predicates and goals; the *planner* that generate the plan; the *executor* that executes the plan by activating the action nodes. Regarding this problem, it's required to define the PDDL domain file, the terminal commands that specify the problem and the goal, an action node for each specified action and the launcher.

III. Design Choices

In this section, we discuss some design choices and some assumptions that have been made analyzing the scenario and the different subproblems. These assumptions are required to better delineate the problem.

We can start saying that each crate can have just one specific content, each person can need none or more emergency supplies and can initially have some crates. This implies that each person could need more crates. We assume that a single crate with specific content is enough to satisfy the person need of that emergency supply, i.e., if a person needs some food, it is sufficient to deliver it just one crate containing food, no more. For these reasons, the problem file should be defined in a way that does not introduce any form of inconsistency: for example, we should not have the situation in which a person initially has a crate containing food and at the same time the person needs food.

Initially, we need to consider the main object types used in the proposed scenario. We can simply spot the following types: `robot`, `person`, `crate`, `location`, `content`, and, from the *Problem 2* on, the assignment introduces the concept of `carrier`.

We can now start describing the modelled PDDL predicates, actions, and functions. In order to model the location of robotic agents, people and crates, appropriate predicates should be defined. We could simply use just one predicate to express object's location, but we preferred to define a predicate for each type of object to improve clarity. In this way, we have defined the predicates `(robot_at ?r - robot ?l - location)`, `(person_at ?p - person ?l - location)` and `(crate_at ?c - crate ?l - location)`.

As we have said, people need specific emergency supplies. To model this fact, the predicate `(need ?p - person ?co - content)` has been defined. The predicate `(have ?p - person ?co - content)` is used to define that a person has a crate with that content. In order to know if a crate has already been delivered, the predicate `(available ?c - crate)` is used: we assume that when a crate is delivered is no more available to be picked up or moved. This assumption is not strictly required, even if it sounds reasonable, and could be omitted in the modelling.

To model the fact that a robot can be empty or can hold a crate, the predicates `(empty ?r - robot)` and `(hold ?r - robot ?c - crate)` are respectively used. We note that the predicate `empty` is not strictly required but it is useful to simplify the modelling and to increase the efficiency of the planner.

Three different actions have been modelled: `pick_up`, `move` and `deliver`. The action `pick_up`, as the name suggests, models the robot that picks up a crate: the robot and the crate should be at the same location, the robot should be empty and the crate available. The action `move` indicates the movement of the robot from one location to another. The action `deliver` indicates the delivering of a crate to a person: the person should need emergency supplies of the same type of the content of the crate, the robot should hold the crate and should be at the same location of the person.

From *Problem 2* on, the introduction of the carrier, poses the need to change a bit the model introducing the `carrier` type and the `(carrier_at ?ca - carrier ?l - location)` predicate. Instead, the predicates `hold` and `empty` have been removed in favour of the predicate `(load ?ca - carrier ?c - crate)` which indicates that a crate has been loaded into the carrier. In order to model the variable capacity of the carriers, the function `(capacity ?ca - carrier) - number` has been defined: the function is used to check the remaining capacity of a carrier before loading a crate. The action `move_carrier` has also been added and indicates the robot that moves a specific carrier from one location to another. Instead, the action `move` indicates now the movement of the robot without bringing a carrier with it. This action could be useful, for example if we want to model the fact that the carrier is not initially at the same location of the robot.

In *Problem 3*, the actions have been transformed into durative actions, specifying appropriate durations. A constant duration time has been assigned to each action: we decided that `pick_up` takes 2 time units to execute, `deliver` takes just 1 time unit, `move` takes 3 time units and `move_carrier` takes 4 time units since moving a carrier requires more effort. A new predicate `(free ?r - robot)`, that indicates when a robot is not executing any action, has been added in order to help modelling which actions can be executed in parallel: we decided to model the fact that a robot cannot perform more actions in parallel because it sounds more reasonable. This does not imply that actions cannot be executed in parallel since more robots can perform actions at the same time.

In *Problem 4*, we have implemented the *Problem 3* within the `PlanSys2` framework. To do this, we have initially created the project workspace and we have copied within it the PDDL domain file of *Problem 3*. Then we have implemented the four action nodes: they can be defined as *fake actions* in the sense that they do not really make any useful work but they only show to the user the execution progress of the required action. This feedback is given by the function `do_work()` that, in general, should define the code necessary to perform the action. Finally, we have implemented the launcher, a `Python` script that is responsible for selecting the domain and running the action nodes.

The default planner used by `PlanSys2` is `POPF` and it is also the one used in this project.

IV. Results

In this section, we present and analyse the obtained results regarding the execution of the planners on the PDDL files.

The submitted archive contains the solutions to the presented problems and it is organized as follows. We have created four folders, one for each problem, that contain the PDDL domain files and one or more PDDL problem files. For each problem file, the plan found by the planner is reported in a separate file. For *Problem 4*, there is a folder that contains all the code required to execute the `PlanSys2` problem.

A. Problem 1

For this problem, we have modelled two PDDL problem files with increasing complexity. The planner used to found the solution is `Downward` using the option `--alias lama-first`.

In the first problem modelled, there are 4 people, 5 crates, 4 locations and the planner successfully found an optimal solution that requires the execution of 15 actions.

The second problem modelled is a bit more complex: the number of objects modelled increase to 8 people, 8 crates and 5 locations. Also in this case, the planner successfully found an optimal solution that requires the execution of 27 actions.

B. Problem 2

For this problem, the planner used is `Enhsp-2020` that is available in `Planutils`.

The modelled PDDL problem contains 6 people, 6 crates, 4 locations and just one carrier with capacity 4. Executing the planner with the option `-anytime`, it managed to find an optimal solution that requires performing 16 actions.

```
> get plan
plan:
0       (pick_up robot1 carrier1 c1 depot)      2
2.001   (pick_up robot1 carrier1 c2 depot)      2
4.002   (pick_up robot1 carrier1 c3 depot)      2
6.003   (move_carrier robot1 carrier1 depot l1) 4
10.004  (move_carrier robot1 carrier1 l1 l3)    4
14.005  (deliver robot1 carrier1 c3 p6 l3 beverage)     1
15.006  (move_carrier robot1 carrier1 l3 depot) 4
19.007  (pick_up robot1 carrier1 c6 depot)      2
21.008  (move_carrier robot1 carrier1 depot l1) 4
25.009  (deliver robot1 carrier1 c6 p1 l1 beverage)     1
26.01   (deliver robot1 carrier1 c2 p1 l1 food) 1
27.011  (deliver robot1 carrier1 c1 p3 l1 medicine)     1
28.012  (move_carrier robot1 carrier1 l1 depot) 4
32.013  (pick_up robot1 carrier1 c4 depot)      2
34.014  (pick_up robot1 carrier1 c5 depot)      2
36.015  (move_carrier robot1 carrier1 depot l2) 4
40.016  (deliver robot1 carrier1 c5 p4 l2 food) 1
41.017  (deliver robot1 carrier1 c4 p5 l2 medicine)     1
```

Figure 1. Plan found by `POPF` planner.

C. Problem 3

For this problem, the planner used is `Optic` that is available in `Planutils`.

It has been decided to model two PDDL problem files. In the second one, there are two robots and two carriers while in the first one only one of both. In this way, as previously explained, only in the second problem the actions can be performed in parallel.

The planner managed to find a plan of duration 34 seconds for the first problem and a plan of duration 24 seconds for the second one. As can be seen in the second plan, some actions are performed simultaneously as desired.

D. Problem 4

For this problem, the PDDL domain and problem are the same of *Problem 3*. The problem is specified using a list of terminal commands accepted by `PlanSys2`. These commands are reported in a file located in the directory of *Problem 4*.

Executing the `POPF` planner on this problem, it managed to find the plan shown in Figure 1 of duration 42 seconds. This plan is then executed by `PlanSys2` within the `ROS2` framework. Figure 2 shows part of the execution of the plan and the messages sent by the action nodes.

V. Conclusions

We have shown how a typical planning problem like the emergency services logistics scenario proposed in this report can be solved using the PDDL language and classical planners.

Problems with increasing complexity have been discussed and solved, showing how modern planners can efficiently found complex plans.

Lastly, we have shown how to integrate the planning problem within the `PlanSys2` infrastructure in order to be able to execute the plan in a simulated environment.

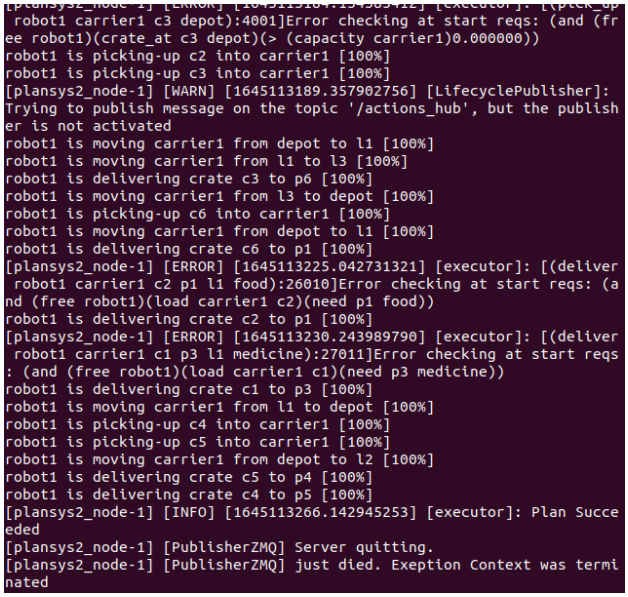

Figure 2. `PlanSys2` execution of the plan.

In future works, using real robotic agents, it could be interesting to implement real action nodes and to study the problems that arise in a real environment.

REFERENCES

[1] Marco Roveri. *Assignment for the course Automated Planning Theory and Practice Academic Year 2021-2022*. Dec. 2021.

[2] Francisco Martín 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.