Università degli studi di Genova

DIBRIS

Department of Computer Science and Technology,
Bioengineering, Robotics and System Engineering

ARTIFICIAL INTELLIGENCE FOR ROBOTICS II

Assignment 1: AI Planning

Authors:

-Ines Haouala S5483776

-Aicha Manar Abbad S5565902

-Romaissa Benkredda S5434673

-Triki Karim S5528602

Professors:

-Mauro Vallati

-Fulvio Mastrogiovanni

May,9th,2023

● Table of Content

6. Result Analysis :

7. Conclusion:

8. References:

9. Appendices:

9..1. Appendix A:

• Introduction:

The advent of robotics and artificial intelligence has altered the way we go about our daily lives. One area where these technologies are beginning to make an impact is in the food and beverage business, specifically in the form of robotic coffee shops. To maintain optimal efficiency and customer satisfaction, the successful integration of robots in such situations necessitates careful planning and implementation. This is where AI planning and robotic experience come into play. However, performance evaluation can still be expanded to incorporate the utmost complexity level conceivable within the current issue arrangement, emphasizing the importance of ongoing research and development in this field. Overall, this project allows you to demonstrate the possibilities of robotics and AI in the domain of Coffeeshops and FastFood.

1. PDDL (Planning Domain Definition Language):

The Planning Domain Definition Language (PDDL) is known as a "modeling language," with ramifications that extend beyond its intended intent. If the purpose of PDDL is to promote theoretical research, such as comparing novel planning algorithms, then it should be assessed using a limited set of criteria. However, if PDDL is to be a practical language that assists engineers in reliably and effectively encoding an application domain into a planning domain model. [1]

2. LPG Planner:

The LPG technique in PDDL (Planning Domain Definition Language) refers to a specific methodology for discovering plans or solutions to PDDL-expressed planning issues.

LPG is an abbreviation for "LPG planner," a popular planning system that uses this strategy. The LPG technique is based on heuristic search, which means it guides its search for a solution using heuristics or rules of thumb.[2]

The LPG technique begins with the planner producing an initial state and a goal state based on the input PDDL issue description. It then searches across the universe of alternative plans, employing a heuristic to assess the "goodness" of each plan.[3]

The LPG planner employs a heuristic function to calculate the cost of moving from one state to another. This heuristic is intended to direct the planner to the most promising pathways in the search space, reducing total search time.

The LPG planner also employs a technique known as "A* search" to boost its efficiency. A* search is a form of search algorithm that combines the cost of getting to a given state with a heuristic estimate of how much it will cost to get to the destination state. This enables the planner to concentrate on the most viable options while avoiding unnecessary study of less potential paths.

In general, the LPG method is a strong and effective solution for resolving planning issues in PDDL. [4]

2.1. The Positive Points Of Using LPG Planner:

- **Efficiency**: The LPG planner is renowned for being effective at solving planning issues. It can swiftly investigate potential paths while avoiding the examination of unpromising ones thanks to the application of heuristics and A* search.
- **Flexibility**: The LPG planner can be used to represent a variety of planning issues, including those with complicated domains and expansive state spaces. It supports a wide range of PDDL components.
- **Accuracy**: The LPG planner's heuristic function is made to produce precise estimates of the costs involved in moving from one state to another. This aids in pointing the planner in the direction of ideal or nearly ideal options.
- **Open source**: The LPG planner is an open-source program, which means it is freely available for usage and can be updated or enhanced to meet individual needs.
- **Widely used**: The LPG planner is a well-known tool in the field of automated planning, and it has been utilized in numerous research projects
- **Support durative actions**: The LPG Supports durative actions that we need it in our project

2.2. The Negative Side Of Using LPG:

- **Limited expressivity**: While the LPG planner supports a wide range of PDDL constructs, it may not be able to express certain sorts of planning issues that require more advanced modeling elements or non-standard constraints.
- **Tuning parameters**: The performance of the LPG planner might be significantly reliant on the specific issue domain and the parameters of the heuristic function. Finding the best parameters may necessitate some trial and error or experimenting.
- **Complexity**: The LPG planner can be difficult to configure and use, especially for users who are unfamiliar with the PDDL language or automated planning in general.
- **Resource requirements**: The LPG planner may require extensive computer resources to solve big or complicated planning issues, which may limit its utility in some applications.

2.3. Download LPG Planner:

Please visit the following URL to obtain the LPG planner:
http://helios.hud.ac.uk/scommv/storage/lpg++

Open your terminal and browse to the LPG++ directory once it has been downloaded. In the terminal, type "./lpg++ -o waiterrobot.pddl -f waiterrobot_p4.pddl -n 1" to start the planner.

The planner will be launched using the given problem and domain files.

3. Problem Statement:

In a coffee shop, there are two robots: a barista who prepares drinks and a waiter who serves them. Customers place orders via a specific interface at the entry, and all orders are known at the start of the planning issue. The barista needs 3 time units to produce a cold drink and 5 time units to prepare a heated drink. When the drinks are ready, they are placed on the bar for the waiter to collect. The waiter can handle one drink at a time with a gripper or up to three drinks on a tray. When using a tray, the waiter's speed is reduced to 1 meter per time unit.

The tray must be returned to the bar after usage and cannot be left on a table. A table requires two time units per square meter to clean, and the robot cannot clean a table while carrying the tray.

The coffee shop includes a bar counter at the top and four one-meter-apart tables for customers. Tables 1 and 2 are 2 meters distant from the bar. Table 3 is the only one with two square meters, while the others have one.

There are four possible issue scenarios:

- Problem 1: Two clients at table 2 bought two cold beverages, and tables 3 and 4 require cleaning.
- Problem 2: Four people at table 3 requested two cold beverages and two hot drinks, and table 1 needs cleaning.
- Problem 3: Two clients from table 4 bought two warm beverages, while two customers from table 1 ordered two warm drinks. Table 3 needs cleaning.
- Problem 4: Two clients at table 4 and two at table 1 bought cold beverages, while four customers at table 3 wanted warm drinks. Table 4 needs cleaning.

4. Description of Domain:

4.1. The Domain file : Waiter robot:

The domain file under the name of waiter robot is the common domain for all of the 4 problems presented in the assignment. The 4 problems have different tasks to be completed by the robot but share some aspects such as the layout of the coffeeshop, the initial states of the robot waiter and barista, etc as shown in our problem 1 to 4 files refer to Appendix.

The waiter robot domain is a representation of the planning domain that declares and lists the objects, predicates, actions and events used. It also describes how events and actions are triggered through preconditions. This will automate the planning and generate plans for the robot waiter to complete its tasks. In this case, the tasks can vary from taking orders, serving the clients, or cleaning tables. The structure of the domain file coffeeshop is given below, also refer to Appendix where you can find the complete PDDL code explained:

4.1.1. *Types definition:*

The first block defines the types and subtypes where gripper, location and order are subtypes of object, counter and table subtypes of location.

These will be used to describe the objects and will later be used to establish the relationships between the objects.

4.1.2. *Predicates:*

The predicates in the domain file define the state of the objects and establish the relationships between the objects; this will be used later in the functions and actions blocks. Comments are provided on the PDDL code to have a clear understanding of the representations of the predicates, refer to Appendix.

4.1.3. *Functions:*

The function block is used to define and measure numerous aspects of the domain, examples are: the speed of the robot, number of drinks on tray, etc. Refer to the Appendix for further details.

4.1.4. *Durative Action move-robot:*

The "move_robot" is a durative action that makes the robot waiter move from one location to another, within a duration that is based on the distance between the two locations and the speed of the robot. The parameters of this action are: the robot, starting location and end location. In the initial state, the robot must be free and it should be positioned at the end location.

To execute the action we need the robot's initial and final positions and the path in between them. The robot moves to its final location at the end of the action and it is no longer in its initial position.

4.1.5. *Durative Action prepare-order:*

The "prepare-order" durative action makes the barista prepare the order. The parameter of this action is the drink, counter bar, and table. As a condition for the action, the order is not prepared or currently preparing , and the order is known. When the action is triggered, the barista starts preparing the drink. By the end, the preparation process is done, the drink is set on the counter and the number of drinks to be served is increased by one.

4.1.6. *Action take-order*:

The "take-order" durative action makes the robot waiter take the order and carry it to the table of the client. As a precondition for the action, the robot should be positioned at the counter with its grippers free. Additionally, the order should be already prepared by the barista and ready for serving. If the preconditions are satisfied, the robot waiter takes the order and carries it. As a result, its grippers are no longer free, and the drink is now carried by the robot.

4.1.4. *Action serve-order:*

The "serve-order" action makes the robot release the order at the client's table. The preconditions for this action to happen are the tray and the gripper are both free, the robot must be at the right table and carrying the order. The effect of this action is that the drink order is marked as served, and the gripper is set to free.

4.1.5 *Durative Action clean-table*:

The "clean " durative action makes the robot waiter clean the table at the coffee shop. The parameter of this action is the location of the table. For this action to be triggered, the robot waiter should be at the correct place, the gripper must be free, and the table should be marked as needing to be cleaned. The effect of the action is that the table is marked as cleaned.

4.1.6. *Action Take the last order*:

The “take-last-order” is an action that takes the last order from the counter and puts it on the tray. The action has as parameters the drink needed to be served, and the counter bar. The preconditions of the action ensure that the robot is at the counter, the tray is not carrying any order, the gripper is free, and there is at least one drink to be served. The order predicate is additionally used to confirm that the last order is still at the counter and has not yet been fulfilled. The drinks_in_tray predicate is tested one last time to make sure the tray has enough space to hold at least one more drink. The effect updates the state of the robot to indicate that it is carrying the last order.

4.1.7. *Action prepare-tray*:

The “prepare-tray” is an action that prepares a tray of drinks for the robot to carry. The action takes as parameters the order needed to be prepared and the counter. The preconditions of the action have several conditions from which: the robot has to be located in the counter, robot gripper must be free, at least to drink ready to be served, and the tray can carry on three drinks at a time. The effect of this action is the update the state of the planning domain: Mark the order as carried, remove the drink order for the specific drink from the counter, the number of drinks still to be served is decreased by one, increase the number of drinks on the tray by one, and mark the tray as being used.

4.1.8. *Action serve-tray-order*:

The “serve-tray-order” is an action that serves drinks while carrying a tray. The action takes as parameters the drink to be served and the location of the table. The precondition for the action includes the following conditions: The robot's grippers must not be free, there must be at least one drink on the tray, the robot must be carrying the specified drink order, the robot must be at the specified table location, the specified drink order must be associated with the specified table location. The effect of the action is to mark the robot gripper as occupied, place the specified drink order on the specified table, decrease the number of drinks on the tray by one, and mark the specified drink order as no longer being carried by the robot.

4.1.9. *Action take-tray*:

The “take-tray” is an action that prepares the tray to be carried by the robot. The action takes as parameters the location of the counter. The precondition for the action includes as conditions: The robot's grippers must be free, the robot must be at the counter, the tray must be ready and waiting at the counter, and the tray must not already be on the robot. The effect of this action is to: Mark the tray as no longer being ready and waiting at the

counter, mark the tray as no longer being occupied, mark the robot's grippers as no longer being free, mark the tray as now being carried by the robot, assign a speed of the robot, and mark the tray as being in use by the robot.

4.1.10. *Action serve-tray*:

The “server-tray” is an action that serves the order carried by the robot using a tray. The action takes as a parameter the location of the counter. The precondition of this action includes these conditions: The tray must be in use by the robot, the robot's grippers must be busy, the robot must be at the counter, and there must be less than 1 drink on the tray. The effect of this action is the following: Mark the tray as no longer being in use by the robot, assign a speed of 2 to the robot, mark the tray as being ready and waiting at the counter, and mark the robot's grippers as free

5. Description of the Problem files:

5.1. *Problem 1*:

The PDDL problem file that covers problem 1 is saved under the name of "waiterrobot_p1", refer to Appendix. This problem file satisfies the requirements of problem 1 where the robot must deliver 2 drinks and clean 2 tables. The problem file declares the following objects c1 and c2 as subtypes of order, drink1 and drink2 as subtypes of drink, t1, t2, t3, t4 as subtypes of table and counter as subtype of counter. In the initialization section, the layout of the coffeeshop is predefined, listing the distance between the 4 tables. Moreover, the following is also predefined: the time needed to prepare each of the hot or cold drinks is predefined, how fast the robot moves and the number of drinks on tray. These are constant between the 4 problem files. Initially, the robot and the tray should be free. There are 2 drinks that are ordered at table 2, where the 2 drinks are cold. Additionally, table 3 and 4 should be cleaned. In the goal scope/section, the robot should successfully serve the drinks at the respective table and clean the set tables, in the least amount of time.

5.2. *Problem 2*:

The PDDL problem file that covers problem 2 is saved under the name of "waiterrobot_p2", refer to Appendix. This problem file satisfies the requirements of problem 2 where the robot must deliver 4 drinks and clean 1 table. The problem file declares the following objects c1, c2, w1 and w2 as subtypes of order, t1, t2, t3, t4 as subtypes of table and counter as subtype of counter. In the initialization section, the layout of the coffeeshop is predefined, listing the distance between the 4 tables. Moreover, the following is also predefined: the time needed to prepare each of the hot or cold drinks is predefined, how fast the robot moves and the number of drinks on the tray. These are constant between the 4 problem files. Initially, the robot and the tray should be free. There are 2 cold drinks ordered at table 3 and 2 hot drinks that are ordered at the same table. Additionally, table 1 should be cleaned. In the goal scope/section, the robot should successfully serve the drinks at the respective table and clean the set table, in the least amount of time.

5.3. *Problem 3*:

The PDDL problem file that covers problem 3 is saved under the name of "waiterrobot_p3", refer to Appendix. This problem file satisfies the requirements of problem 3 where the robot must deliver 4 drinks and clean 1 table. The problem file declares the following objects w1, w2, w3 and w4 as subtypes of order, t1, t2, t3, t4 as subtypes of table and counter as subtype of

counter. In the initialization section, the layout of the coffeeshop is predefined, listing the distance between the 4 tables. Moreover, the following is also predefined: the time needed to prepare each of the hot or cold drinks is predefined, how fast the robot moves and the number of drinks on tray. These are constant between the 4 problem files. Initially, the robot and the tray should be free. There are 4 hot drinks ordered: 2 hot drinks at table 1 and 2 hot drinks ordered at table 4. Additionally, table 3 should be cleaned. In the goal scope/section, the robot should successfully serve the drinks at the respective table and clean the set table, in the least amount of time.

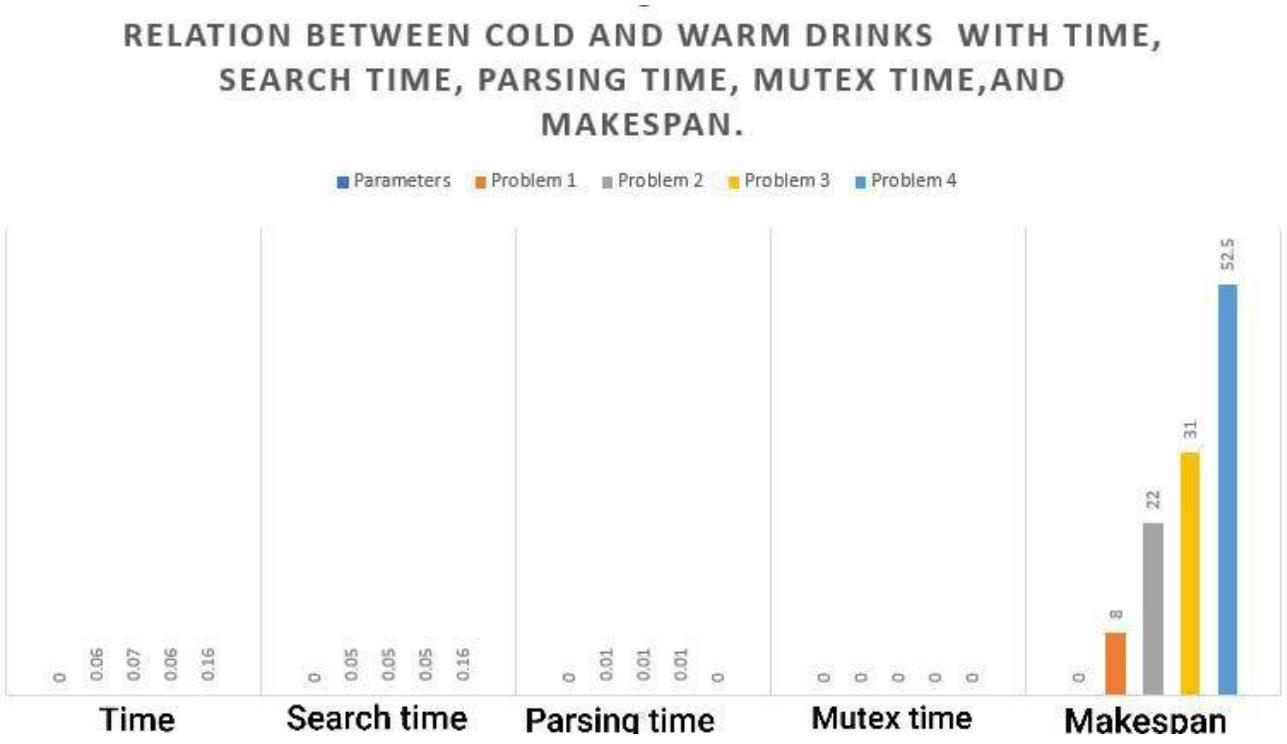
5.4. *Problem 4*:

The PDDL problem file that covers problem 4 is saved under the name of "waiterrobot_p4", refer to Appendix. This problem file satisfies the requirements of problem 4 where the robot must deliver 8 drinks and clean 1 table. The problem file declares the following objects c1, c2, c3, c4, w1, w2, w3 and w4 as subtypes of order, t1, t2, t3, t4 as subtypes of table and counter as subtype of counter. In the initialization section, the layout of the coffeeshop is predefined, listing the distance between the 4 tables. Moreover, the following is also predefined: the time needed to prepare each of the hot or cold drinks is predefined, how fast the robot moves and the number of drinks on tray. These are constant between the 4 problem files. Initially, the robot and the tray should be free. There are 4 hot and 4 cold drinks ordered: 4 hot drinks at table 3, 2 cold drinks at table 4 as well as 2 cold drinks at table 1. Additionally, table 4 should be cleaned. In the goal scope/section, the robot should successfully serve the drinks at the respective table and clean the set table, in the least amount of time

6. Result Analysis :

The table below displays the tabular compilation of the output produced in response to the assigned problems that was exported into the corresponding SOL Files. Plotting the data allowed researchers to examine the relationships between various parameters and spot trends that were in opposition to them. Figures 2, 3, and 4 show the graphs in detail.

Parameters	Problem 1	Problem 2	Problem 3	Problem 4
Time	0.06	0.07	0.06	0.16
Search time	0.05	0.05	0.05	0.16
Parsing time	0.01	0.01	0.01	0.00
Mutex time	0.00	0.00	0.00	0.00
Makespan	8.00	22.00	31.00	52.50



The graphical representation indicates that the problems analyzed, namely problem

1, problem 2, problem 3, and problem 4, exhibit negligible differences in terms of time , parsing time and search time , as well as absence of mutual exclusion time , while the makespan displays a progressive increase from problem 1 to problem 4.

7. Conclusion :

In conclusion, the robotic coffee shop scenario presented in this assignment was successfully modeled with efficient and effective plans. The task involved creating models for a barista and waiter robot, assigning tasks such as preparing drinks, serving customers, and cleaning tables. The problem was solved by taking into account the time required to prepare cold and warm drinks, the capability of the waiter robot to grasp a single drink or use a tray, and the time required to clean tables..

However ,the evaluation of performance can also consider the highest complexity level achievable within the current problem arrangement. For instance, when the coffee shop reaches its maximum capacity, with all tables fully occupied by 10 customers ordering warm drinks. In future studies, a more powerful computing system could be used to assess the performance of such a complex problem.

Overall, the assignment provided an opportunity to showcase AI planning and robotic expertise in developing the best possible model for the coffee shop scenario

8. References:

[1] McCluskey, T.L. (2003) PDDL: A language with a purpose? In: ICAPS-03, 13th International Conference on Automated Planning & Scheduling, June 9-13 2003, Trento, Italy.

[2] "LPG Planner." [Online]. Available: http://www.lpg.tu-bs.de/. [Accessed: April 25th, 2023].

[3] M. Ghallab, D. Nau and P. Traverso, "Planning as heuristic search: a unified approach to robotics, AI, and control," Artificial Intelligence, vol. 33, no. 1, pp. 1-52, 2004.

[4] J. Hoffmann and B. Nebel, "Automated planning and acting," AI Magazine, vol. 38, no. 4, pp. 13-34, 2017.

9. *Appendices* :

9.1. *Appendix A* : Planning in PDDL Domain Files, Output Files, and Problem Files

Link:
https://drive.google.com/drive/folders/1Gior0jjHIH9cTHWyMWu2E4UlFo6Mky0x?usp=sharing

In this link , we provide a folder on the PDDL domain files, output files, and problem files used in the planning process.

A.1 PDDL Domain Files

The PDDL domain file contains the following element named " waiterrobot"

A.2 Output Files

The output of a planning process is a plan that describes the sequence of actions required to achieve the goal state. The plan is typically saved in a file in a format that can be read by the planning software.

The output files are named :

- waiterrobot_p1
- waiterrobot_p2
- waiterrobot_p3
- waiterrobot_p4

A.3 Problem Files

A PDDL problem file specifies the initial state of the planning problem and the goal state that needs to be achieved. It defines the objects used in the problem and assigns them values based on the domain types defined in the domain file.

The problem files :

- plan_waiterrobot_p1.pddl_1
- plan_waiterrobot_p2.pddl_1
- plan_waiterrobot_p3.pddl_1
- plan_waiterrobot_p4.pddl_1