Artificial Intelligence II: First Assignment

Students Barış Aker (5430437)
Claudio Tomaiuolo (5630055)
Nicholas Attolino (5656048)
Teodoro Lima (5452613)

Abstract

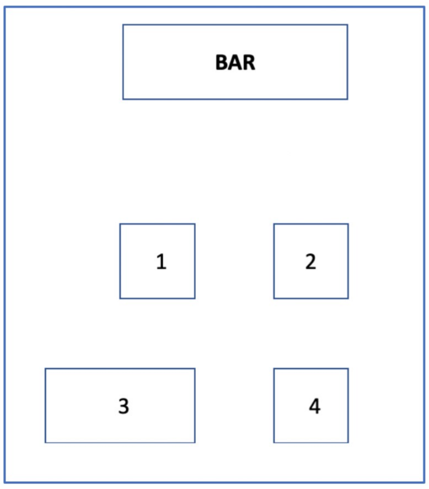
Robotics and Artificial Intelligence are among the current issues that are posing the great challenges for our times and society. The future of health, work and mobility are closely interwoven with the development of robotics and AI [1]. But developments in the field of high-tech are already changing various environments and scenarios, one example being the restaurant sector.
In Japan, a few coffee shops are using robots as waiters and baristas. However, as the number of customers increases, the shops realise that they need to efficiently and effectively plan the way in which robots perform their tasks, to properly run the shop.
In this paper, we analysed the efficiency of our PDDL planning model, which has the task of managing the movements and actions of a waiter robot and a barista robot running a coffee shop. The model is performed with ENHSP Planning System (version: ENHSP-20), using different engines, due to the limits of ENHSP. The environment is defined and the robots have limits and specific rules to comply with in order to perform and satisfy customers' needs.

Introduction

AI and Robotics are changing the future of society in areas such as work, education, industry, farming, and mobility, as well as services like banking.
Figures such as Prague's Golem of the 16th century, from Jewish mysticism, formed from dust or earth to serve as a helper and companion, and later the Frankenstein monster by British writer Mary Shelley in 1816, are evidence of the human desire to create an anthropomorphic servant for himself. The current development of robotics ad artificial intelligence aims to fulfil precisely this yearning for a man-made helper.
The first pioneer in this sense was Leonardo Da Vinci, who, in order to comprehend the medical research of his time (15th-16th century), dissected bodies and acquired information, used it to translate the human locomotive system into mechanisms. The "automa cavaliere" (or "knight") was the first mechanism able to walk, sit and move its arms.
In centuries past, the subject of a 'helper automaton' might have seemed like science fiction and/or frightening, nowadays it is becoming increasingly real. In more technologically developed cities, it is already common to find robot waiters roaming the tables and aisles of restaurants, to help manage orders and facilitate the transport of items between the kitchen, dining room and warehouse.

Environment and constraints

In the coffee shop, there are two robots: one barista and one waiter.
The barista is in charge of preparing drinks. Cold drinks (3 time units) are faster to prepare than warm drinks (5 time units). Once ready, the drinks are put on the bar, where the waiter can pick them up.
The waiter takes drinks to the tables and clean them when customers leave. The robot can grasp a single drink using its grippers (not one drink per each gripper) or can use a tray, then the waiter can carry up to 3 drinks at the same time. The waiter moves at 2 meters per time unit; 1 meter per time unit if it is using the tray. The tray can be taken from the bar, and must be returned there after use. The waiter is not allowed to leave the tray on a table. The waiter takes 2 time units per square meter to clean a table. The robot cannot clean a table while carrying the tray.



The coffee shop layout is shown in the figure above. It has the bar counter on the very top, and 4 tables for costumers. Each table is 1 meter apart from any other (let's assume Euclidean geometry does not apply here, so also tables 1 and 4 are 1 meter from each other). The bar is 2 meters away from tables 1 and 2. Table 3 is the only table of 2 square metres, all the others are of 1 square metre.

Optional constraints

1. Warm drinks cool down, so it needs to be served to the customer before the drink gets too cold. A warm drink takes 4 time units to become cold. A customer will not accept a warm drink delivered more than 4 time units after it was put on the bar by the barista. (Implemented)
2. There are 2 waiters. The coffee shop is doing well, so the owner decided to buy a second waiter. This is of course making things a bit more complicated: only one waiter can be at the bar at a given time; further, the owner does not want that a table is served by both the waiters: only one waiter can deal with the orders of a given table.
3. Finish your drink. After receiving the drink, a customer will finish it in 4 time units, and will leave after that. When all the customers of a table have left, the waiter robot can clean it. The waiter robot is required to clean all the tables to complete the problem.
4. We also serve food. The coffee shop is also serving delicious biscuits. They need no preparation, and can be picked up by the waiter at the bar counter. All the customers with cold drinks will also receive a biscuit, but only after having received their drink. The waiter can not bring at the same time the drink and the biscuit, but must deliver the drink, and then go back to the counter to take a biscuit for the customer. For the sake of moving biscuits around, the same limitations as for drinks apply.

Problems to solve

Problem 1 There are 2 customers at table 2: they ordered 2 cold drinks. Tables 3 and 4 need to be cleaned.

Problem 2 There are 4 customers at table 3: they ordered 2 cold drinks and 2 warm drinks. Table 1 needs to be cleaned.

Problem 3 There are 2 customers at table 4: they ordered 2 warm drinks. There are also 2 customers at table 1: they ordered 2 warm drinks. Table 3 needs to be cleaned.

Problem 4 There are 2 customers at table 4 and 2 customers at table 1: they all ordered cold drinks. There are also 4 customers at table 3: they all ordered warm drinks. Table 4 needs to be cleaned.

ENHSP Planner System

ENHSP, Expressive Numeric Heuristic Search Planner, is a PDDL automated planning system that supports: *Classical and Numeric Planning (PDDL2.1); Optimal Simple Numeric Planning, Satisficing Non-Linear Numeric Planning, Planning with Autonomous Processes and Discrete Events (PDDL+), Global Constraints and Expressive Formulas in Preconditions and Goals.*
Automated planning is what allows intelligent agents to come up with an organisation of actions to achieve some sort of goal state starting from an initial state of things [3].

How it works

The planner reads in input a PDDL domain and problem file and provides a plan (a sequence of actions). The plan is time-stamped: associated to each action, it is possible to find the time at which that instance of the action has to be executed. In dealing with autonomous processes, ENHSP discretises the problem (with a delta=1sec by default); so the plan is guaranteed to be valid only with respect to that discretisation.
PDDL is the standard de facto language to express planning problems. The domain file expresses the signature of your predicates, functions (for the support of autonomous processes) and also all the actions/processes/events available, in a parametrized way. The problem file expresses the particular instance of the planning problem (e.g., what is the initial value of predicate A? What is the goal?).
ENHSP supports PDDL 2.1 in particular, and PDDL+ (for the support of autonomous processes) and also events [3].

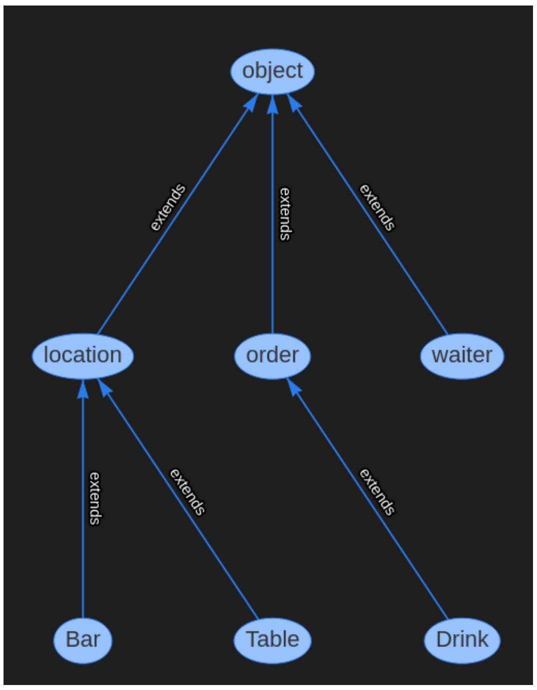
The general command used to run the code is:

```
java -jar enhsp-dist/enhsp.jar -o <domain_file> -f <problem_file>  -planner <configuration>
```

Code

Inside the Domain file there are: `:requirements, :types, :predicates, :functions, :actions`.
`:requirements` used are: `:adl`, which adds `:strips`, `:typing`, `:disjunctive-preconditions`, `:equality`, `:quantified-preconditions`, `:conditional-effects`; `:typing`, allowing the usage of typing for objects; `:fluents`, allowing the inclusion of a `:function` block which represent numeric variables in the domain; `:time`, necessary for including processes and events.

`:types` used are: *location*, *order* and *waiter* as **objects**, *Bar* and *Table* as **location** and *Drink* as **order** (as shown below).



`:predicates` are:
(waiter-at ?w – waiter ?t – location) represents "where is the waiter?"
(start-moving-to ?w - waiter ?to - location) represents "the waiter is moving to where"
(tray-taken) to inform "tray has been taken"
(tray-empty) to inform "tray is empty"
(carrying-3-drinks ?w - waiter) represents "the waiter carries three drinks"
(carrying-2-drinks ?w - waiter) represents "the waiter carries two drinks"
(carrying-1-drink ?w - waiter) represents "the waiter carries one drinks"
(free-grippers ?w - waiter) represents "waiter's grippers are free"
(drink-ordered ?d - Drink ?t - location) represents "drink ordered to which table"
(barista-free) represents "the barista is free" *(waiter-free ?w - waiter)* represents "the waiter is free"
(start-preparing-order ?d - Drink) represents "the barista is starting to prepare the drink"
(cold-drink ?d – Drink) represents "the type of the drink"
(warm-drink ?d – Drink) represents "the type of the drink"
(prepared-order ?d - Drink) represents "the drink was prepared"
(ready-order ?d - order) represents "the order is ready on the bar"
(carried-order ?d - order ?w - waiter) represents "the waiter is carrying the order"
(delivered-cold-order ?d - order ?t - Table) represents "the cold order has been delivered to the right table"
(delivered-warm-order ?d – order ?t – Table) represents "the warm order has been delivered to the right table"
(table-clean ?t - location) represents "which table needs to be cleaned"
(start-table-cleaning ?w - waiter ?t - location) represents "the waiter starting to clean the table"
(free-table ?t - location) represents "which table is free"

Inside the `:functions` part, there are:
(distance ?t1 – location ?t2 – location) represents "distance between tables and bar"
(table-size ?t – location) represents "the size of the tables"
(velocity-with-tray ?w - waiter) represents " the velocity of the waiter with tray"
(time-for-order) represents "the need time to prepare an order"
(time-to-clean-table) represents "the need time to clean a table"
(distance-from-goal) represents "the distance to the goal position"
(count-ready-on-bar) represents "the number of drinks ready on the bar"
(cooldown-warm-drink) represents "the cooldown of a warm drink when is ready on the bar"

Actions, Processes and Events

`:action` **act-starting-to-prepare-cold**, checking that the drink to be prepared is cold, the barista is free, the barista has not started to prepare the order; as results we will have the bartender busy, starting to prepare the order and assigning the value 0 to the function representing the time to be spent.
`:action` **act-starting-to-prepare-warm**, the same as before but with little differences: (warm-drink) instead of (cold-drink) in preconditions and the function "cooldown-warm-drink" where we assign the value 5 (we consider 5 to 1 a the cooldown of 4 seconds, so when it is equal or less than 1 the drink will have cooled down).
`:process` **prs-preparing-order**, it checks that order preparation has begun and, if it turns out to be true, increases the time required for order preparation.
`:event` **evt-order-preparation-terminated**, it verifies that the preparation of the order has started and that the time value is equal to 3 seconds if a cold drink is being prepared or 5 seconds if a warm drink is being prepared, obtaining as a result the prepared drink and ready, the barista free and increases the value of the counter ready drinks on the bar by 1.

`:process` **prs-cooldown-to-cold**
`:event` **evt-cooldown-warm-drink**
In the process it is verified that the system only checks warm and ready drinks on the bar with the effect of decreasing the value of the cooldown by 1.

`:action` **act-pick-up-cold-order-prepared**
`:action` **act-pick-up-warm-order-prepared**
`:action` **act-put-down-cold-order-prepared**
`:action` **act-put-down-warm-order-prepared**

There are two actions to pick up and put down a cold drink and two actions to do the same with a warm drink: for the cold drink, it checks the velocity of the waiter (because it is different if he takes a tray, so in this case he doesn't have a tray and his speed is 2 m/s; it is used the functions equals to 0 if without a tray on his hands, or equals to 1 if with a tray on his hands), the position of the waiter, if the drink is ready, that the grippers of the waiter must be free and if the drink is cold. As results we have the drink carried from the waiter and the counter of ready drinks on the bar decreased by 1. To put down the cold drink it is checked that the drink corresponds to the table it was ordered from, the position of the waiter, if the drink is cold, if the speed of the waiter is 2m/s and if the drink is being brought by the waiter. As a result it is obtained that the drink has been delivered and that the waiter has his free grippers; for the warm drink is essentially the same but with the difference that to pick up a drink it is checked that the drink must be warm and the cooldown equals to 5. As a result, the cooldown must be greater than or equal to 1.

`:action` **act-starting-the-movement**, it foresees the free waiter and the table from which he will start, obtaining as a result that the starting table will be free, that the waiter will have started moving and will be assigned a value to the "distance-from-goal" function equal to the distance between the starting and finishing tables.
`:process` **prs-movement**, it is verified that the move action has started and as a result the distance to the target is decremented based on the speed of the waiter.
`:event` **evt-movement-finished**, it verifies that the arrival table is free and that the distance from the target has become a value equal to or less than 0, obtaining as a result the waiter free and at the arrival table.

`:action` **act-clean-table**, in order to clean a table, it is necessary to have the waiter free with his hands free and to be at the table to be cleaned (considering that the table has not been cleaned previously); obtaining as a result that the waiter will have started cleaning the table and a time value will be assigned for cleaning the table equal to 2 times the size of the table in question.
`:process` **prs-cleaning-table**, it verifies that the cleaning action has started obtaining the decrease over time of the time required for cleaning.
`:event` **evt-end-clean-table**, it verifies that the time required for cleaning is less than or equal to 0, resulting in the table being clean and the waiter free.

`:action` **act-pick-2-drinks-with-tray**
`:action` **act-pick-3-drinks-with-tray**
In order to bring drinks with a tray, tow actions were performed: one for the first two drinks and one for the third drink (It should be noted that it was decided to allow only "cold" drinks to be transported as they reasoned manually and imagined a real case, the team decided that due to the cooldown for warm drinks and the understanding abilities of the Engine used, it is not possible to deliver warm drinks on the tray). In the first action, it is checked if the waiter is at the bar with his free grippers, if the tray is empty and if the drinks are cold and ready on the bar (taking into account that there must be at least 2 drinks on the bar) obtaining as a result that tow drinks will find on the tray in the waiter's grippers and that his speed will become equal to 1m/time unit. In the second action, it is verified that the waiter is already carrying two drinks on the tray and if there is at least one cold and ready drink on the bar, obtaining as a result that the waiter is carrying three drinks on the tray.

`:action` **act-put-down-third-drink**, it checks that the waiter is carrying two drinks on the tray, that he is at the order table, bringing the correct order on the tray. As a result he will no longer be carrying the third drink because it will be delivered and therefore he will only be carrying two drinks.
`:action` **act-put-down-second-drink**, it has the same characteristics as the previous one as it checks for two drinks on the tray, the second correct drink and scales the drinks on the tray by one once the second one is delivered.
`:action` **act-put-down-first-drink**, it is the same as mentioned above with only one difference where the last drink will be delivered and the tray will be empty.

`:action` **act-to-leave-the-tray**, it allows the waiter to leave the tray at the bar and to do this, it checks that he has an empty tray, that his speed is reduced since he has the tray in his grippers and that he is at the bar. As a result we will have that his speed will be normal again, that he will no longer have the tray in his grippers and that his grippers will be free.

Results

The different problems proposed were executed using different engines of the ENHSP Planning System, due to its limitations and different optimisations offered by its planner configurations.

Problem 1

The first problem requested presents two customers seated at table 2, who order two cold drinks. Then tables 3 and 4 need to be cleaned. Objects have been defined as: bar, tables (tab1, tab2, tab3, tab4), drinks (d1, d2) and a waiter w. The problem is executed with the `opt-blind` configuration, optimised for complex problems, but with not too long plans: no more than 20 actions. The problem is successfully solved with an average planner execution time of about 0.5 second, Grounding Time is 25 and Expanded Nodes are 17685. The actual time taken by the robots to perform all required actions is 13sec (as shown below).

```
Found Plan:
    0: (act-starting-the-movement bar tab4 w)
    0: -----waiting---- [2.0]
    2.0: (act-clean-table tab4 w)
    2.0: -----waiting---- [4.0]
    4.0: (act-starting-the-movement tab4 tab3 w)
    4.0: (act-starting-to-prepare-cold d1)
    4.0: -----waiting---- [5.0]
    5.0: (act-clean-table tab3 w)
    5.0: -----waiting---- [7.0]
    7.0: (act-starting-to-prepare-cold d2)
    7.0: -----waiting---- [9.0]
    9.0: (act-starting-the-movement tab3 bar w)
    9.0: -----waiting---- [11.0]
    11.0: (act-pick-2-drinks-with-tray bar w d2 d1)
    11.0: (act-starting-the-movement bar tab2 w)
    11.0: -----waiting---- [13.0]
    13.0: (act-put-down-second-drink tab2 w d2)
    13.0: (act-put-down-first-drink tab2 w d1)
```

Problem 2

The second problem involves 4 customers sitting at table 3, ordering 2 hot and 2 cold drinks. Table 1 then needs to be cleaned. Objects have been defined as: bar, tables (tab1, tab2, tab3, tab4), drinks (d1, d2, d3, d4) and a waiter w.
The problem is executed with the `opt-blind` configuration, with a planner execution time of almost 5 seconds, Grounding Time is 24 and Expanded Nodes are 570401. The execution time of the actions by the robots is 19 seconds.

Problem 3

The third problem presents a scenario where there are 2 customers at table 4, who order 2 hot drinks, and 2 customers at table 1, who order 2 hot drinks. Next, table 3 must be cleaned. Objects have been defined as: bar, tables (tab1, tab2, tab3, tab4), drinks (d1, d2, d3, d4) and a waiter w.
The problem is executed once again with the `opt-blind` configuration, which takes less than 4 seconds to find an optimised plan, Grounding Time is 29 and Expanded Nodes are 441278. The actions execution time is 22 seconds.

Problem 4

The last problem proposed has 2 customers at table 4 and 2 customers at table 1 and they all order a cold drink. There are also 4 customers at table 3 and they all order a hot drink. Table 4 must be cleaned later. Objects have been defined as: bar, tables (tab1, tab2, tab3, tab4), drinks (d1, d2, d3, d4) and a waiter w.

The problem cannot be executed with the previously used configuration (`opt-blind`), due to its action limit being set to 20. This time it was used `sat-hmrph` configuration, which takes 0.5 seconds to find a plan, Grounding Time is 33 and Expanded Nodes are 688. The actions execution time is 54 seconds.

Conclusions

The development of robotics and artificial intelligence has led to a significant transformation in several industries, including the restaurant industry. In this paper, the efficiency of a PDDL planning model used to manage the movements and actions of a robot waiter and a robot barista in a cafeteria was analysed. This work identified specific constraints and challenges, such as the need for efficient task planning and time management. The results of this study provide insights that could help the catering industry to improve its services and efficiency for greater customer satisfaction. With the continuous development and evolution of robotics and AI, these technologies are expected to have an increasingly significant impact on our society, leading to new opportunities and challenges.

References

[1] Henry A Kissinger, Eric Schmidt, Daniel Huttenlocher. (2021) *The Age of AI: And Our Human Future*. John Murray Press.

[2] Braun, J., Archer, M.S., Reichberg, G.M., Sánchez Sorondo, M. (2021). *AI, Robotics, and Humanity: Opportunities, Risks, and Implications for Ethics and Policy.* Springer, Cham.

[3] *The ENHSP Planning System*. Accessed 27 April 2023, <https://sites.google.com/view/enhsp/home>.