



Università degli studi di Genova

DIBRIS

AI for Robotics II

---

Second Assignment

Integrated Task and Motion Planning:  
Modeling a Coffee Shop Scenario

---

***Authors:***

Parinaz Ramezanpour Renani

(5214640)

Danial Sabzevari

(5217544)

***Professors:***

Fulvio Mastrogiovanni

Mauro Vallati

December 17, 2023

## Contents

<b>1</b>	<b>Abstract</b>	<b>2</b>
<b>2</b>	<b>Introduction</b>	<b>2</b>
<b>3</b>	<b>PDDL-Based Setup</b>	<b>2</b>
<b>4</b>	<b>Problem Definition</b>	<b>3</b>
<b>5</b>	<b>Methodology</b>	<b>3</b>
5.1	How It Works . . . . .	3
<b>6</b>	<b>Testing and Validation</b>	<b>5</b>
<b>7</b>	<b>Conclusion</b>	<b>5</b>

## 1 Abstract

In this report, we showcase an innovative framework for integrating motion planning and task planning for robotic operations, modeling a coffee shop scenario with a waiter robot. We use a Planning Domain Definition Language (PDDL) based system to ensure efficiency and accuracy in the execution of robotic tasks.

## 2 Introduction

In the rapidly evolving era of Artificial Intelligence and robotics, the focus is shifting towards creating smarter, more versatile, and more efficient systems. We delve into this issue and address it using an example of a coffee shop scenario, addressing how integrating task and motion planning could revolutionize service-based establishments.

This solution, applied to a **coffee shop** scenario, optimizes the way a waiter robot serves orders, taking into account factors like the distance traveled by the robot. Doing so, it ensures faster service and reduces excessive movement, providing a lean and efficient solution to the task management conundrum in service robotics.

## 3 PDDL-Based Setup

The PDDL-based setup used to address this task involves preparing a PDDL domain file that models the coffee shop scenario and tracks the waiter robot's actions and movements.

## 4 Problem Definition

The problem analyzed in this assignment is the optimization of the path taken by the robot to visit different table regions. By minimizing the distance traveled, operational efficiency is maximized, and service delivery time is minimized.

## 5 Methodology

In our approach, we used predicates to identify different regions in the coffee shop and track the robot's current location. We also defined functions to guide and adjust the robot's movements, thereby ensuring efficient task execution.

### 5.1 How It Works

In this scenario, our robot navigates a space segmented into distinct areas, or 'regions.' Each region hosts one or more waypoints providing essential details about the robot's needed position and orientation for order delivery. There are five regions, and each one features just a single waypoint.

#### Key Tasks of the Order-Delivery Robot

- The robot initiates its task from a predefined spot in the area.
- It's assigned to navigate through all five designated regions.
- The robot has the flexibility to deliver orders from any position around the table within these regions.

To address this challenge, we utilized the popf-tif planner.

We adjusted this planner to meet our problem requirements, altering the external module programmed in C++ to calculate the path cost based on Euclidean distance.

Our primary aim was to adjust the variable that represents the overall cost of the plan. This adjustment was done by considering the distance between the waypoints' positions.

**We received two text files to assist in this process:**

- Region Poses: This file links each region to its corresponding waypoints. Given the structure of our problem, where each region is associated with only one waypoint, this file effectively states that region  $r[i]$  links to waypoint  $w[i]$ .
- Waypoint.txt: This file provides information on the specific posture of the  $i$ -th waypoint within the region. The posture is represented by a position  $(x, y)$  and an orientation  $\theta$ , formatted as  $[x,y,\theta]$ .

We developed a function to compute the Euclidean distance between the starting and ending waypoints of each path step. Our utilized Euclidean distance disregards the robot's posture reflecting the fact that the robot can be placed anywhere around the waypoint. This gave us the following formula:

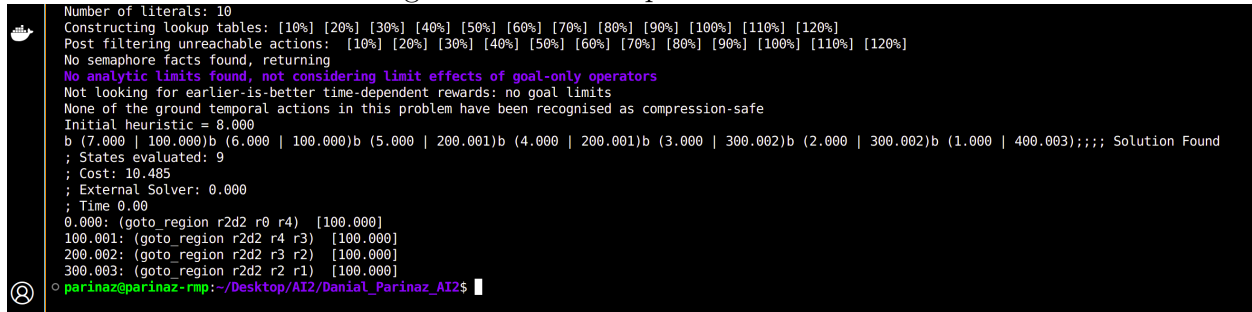
$$D = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$$

By implementing this method, we were able to update the total path cost based on the selected waypoints.

## 6 Testing and Validation

Considering that the plan's total cost is merely the sum of individual path steps, we can easily demonstrate that the implementation works as intended. The observed cost of the

Figure 1: Result of planner execution

A terminal window with a black background and white text. The output shows the planner's internal state and the final solution. The text includes: 'Number of literals: 10', 'Constructing lookup tables: [10%] [20%] [30%] [40%] [50%] [60%] [70%] [80%] [90%] [100%] [110%] [120%]', 'Post filtering unreachable actions: [10%] [20%] [30%] [40%] [50%] [60%] [70%] [80%] [90%] [100%] [110%] [120%]', 'No semaphore facts found, returning', 'No analytic limits found, not considering limit effects of goal-only operators', 'Not looking for earlier-is-better time-dependent rewards: no goal limits', 'None of the ground temporal actions in this problem have been recognised as compression-safe', 'Initial heuristic = 8.000', 'b (7.000 | 100.000)b (6.000 | 100.000)b (5.000 | 200.001)b (4.000 | 200.001)b (3.000 | 300.002)b (2.000 | 300.002)b (1.000 | 400.003);;; Solution Found', '; States evaluated: 9', '; Cost: 10.485', '; External Solver: 0.000', '; Time 0.00', '0.000: (goto\_region r2d2 r0 r4) [100.000]', '100.001: (goto\_region r2d2 r4 r3) [100.000]', '200.002: (goto\_region r2d2 r3 r2) [100.000]', '300.003: (goto\_region r2d2 r2 r1) [100.000]'. The prompt 'parinaz@parinaz-rmp:~/Desktop/AI2/Danial\_Parinaz\_AI2\$' is visible at the bottom.

```
Number of literals: 10
Constructing lookup tables: [10%] [20%] [30%] [40%] [50%] [60%] [70%] [80%] [90%] [100%] [110%] [120%]
Post filtering unreachable actions: [10%] [20%] [30%] [40%] [50%] [60%] [70%] [80%] [90%] [100%] [110%] [120%]
No semaphore facts found, returning
No analytic limits found, not considering limit effects of goal-only operators
Not looking for earlier-is-better time-dependent rewards: no goal limits
None of the ground temporal actions in this problem have been recognised as compression-safe
Initial heuristic = 8.000
b (7.000 | 100.000)b (6.000 | 100.000)b (5.000 | 200.001)b (4.000 | 200.001)b (3.000 | 300.002)b (2.000 | 300.002)b (1.000 | 400.003);;; Solution Found
; States evaluated: 9
; Cost: 10.485
; External Solver: 0.000
; Time 0.00
0.000: (goto_region r2d2 r0 r4) [100.000]
100.001: (goto_region r2d2 r4 r3) [100.000]
200.002: (goto_region r2d2 r3 r2) [100.000]
300.003: (goto_region r2d2 r2 r1) [100.000]
parinaz@parinaz-rmp:~/Desktop/AI2/Danial_Parinaz_AI2$
```

plan is **10.485**, which is the same as calculated as follows:

$$\text{cost} = D(\text{wp0}, \text{wp4}) + D(\text{wp4}, \text{wp3}) + D(\text{wp3}, \text{wp2}) + D(\text{wp2}, \text{wp1}) \approx 2 + 2.828 + 2.828 + 2.828 \approx 10.485$$

where the way-points are:

$$\text{wp0} = [0, 0, 0], \text{wp1} = [2, 0, 0], \text{wp2} = [0, 2, 1.57], \text{wp3} = [-2, 0, 3.14], \text{wp4} = [0, -2, -1.57]$$

This computation aligns precisely with the planner's determined value.

## 7 Conclusion

Through this report, we aimed to showcase the effectiveness of PDDL-based systems in managing task execution in a dynamic environment, such as a coffee shop. The key tasks of the order-delivery robot involve starting from a predefined spot, navigating through all five designated regions, and having the flexibility to deliver orders from various positions around tables within these regions, in which the popf-tif planner was utilized.