University of Kerbala
College of Computer Science & Information Technology
Computer Science Department

# Developing a unified Hierarchical Representation Model for Planning and Acting in Autonomous Robots

A Thesis

Submitted to the Council of the College of Computer Science & Information Technology / University of Kerbala in Partial Fulfillment of the Requirements for the Master Degree in Computer Science

**Written by**

Zainab Abbas Fadhil

**Supervised by**

Asst. Prof. Dr. Ahmed Abdulhadi Ahmed

2023 A.D. 1444 A.H.

بسم الله الرحمن الرحيم

(يَعْلَمُ مَا بَيْنَ أَيْدِيهِمْ وَمَا خَلْفَهُمْ وَلَا يُحِيطُونَ بِهِ عِلْمًا ۝وَعَنَتِ الْوُجُوهُ لِلْحَيِّ الْقَيُّومِ وَقَدْ خَابَ مَنْ حَمَلَ ظُلْمًا ۝وَمَنْ يَعْمَلْ مِنَ الصَّالِحَاتِ وَهُوَ مُؤْمِنٌ فَلَا يَخَافُ ظُلْمًا وَلَا هَضْمًا ۝وَكَذَلِكَ أَنْزَلْنَاهُ قُرْآنًا عَرَبِيًّا وَصَرَّفْنَا فِيهِ مِنَ الْوَعِيدِ لَعَلَّهُمْ يَتَّقُونَ أَوْ يُحْدِثُ لَهُمْ ذِكْرًا ۝ فَتَعَالَى اللَّهُ الْمَلِكُ الْحَقُّ وَلَا تَعْجَلْ بِالْقُرْآنِ مِنْ قَبْلِ أَنْ يُقْضَى إِلَيْكَ وَحْيُهُ وَقُلْ رَبِّ زِدْنِي عِلْمًا)

صدق الله العظيم

## Supervisor Certification

I certify that the thesis entitled **(Developing a unified Hierarchical Representation Model for Planning and Acting in Autonomous Robots)** was prepared under my supervision at the department of Computer Science/College of Computer Science & Information Technology/ University of Kerbala in partial fulfillment of the requirements of the degree of Master in Computer Science.

Signature:

Supervisor Name: Assist. Prof. Dr. Ahmed Abdulhadi Ahmed

Date: 4 / 7 /2023

## The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "**Developing a unified Hierarchical Representation Model for Planning and Acting in Autonomous Robots**" for debate by the examination committee.

Signature:

Assist. Prof. Dr. Muhannad Kamil Abdulhameed

Head of Computer Science Department

Date: 4 / 7 /2023

## Certification of the Examination Committee

We hereby certify that we have studied the dissertation entitled **(Developing a unified Hierarchical Representation Model for Planning And Acting in Autonomous Robots)** presented by the student (Zainab Abbas Fadhil**)** and examined her in its content and what is related to it, and that, in our opinion, it is adequate with (**Very Good**) standing as a thesis for the degree of Master in Computer Science.

Signature:
Name: Dr. Nidhal K. El Abbadi
Title: Professor
Date: 7 / 7 / 2023
**(Chairman)**

Signature:
Name: Dr. Ali Retha Hasoon
Title: Assistant Professor
Date: 4/ 7/ 2023
**(Member)**

Signature:
Name: Dr. Elham Mohammed Thabit A. Alsaadi
Title: Assistant Professor
Date: 4 / 7 / 2023
**(Member)**

Signature:
Name: Dr. Ahmed Abdulhadi Ahmed
Title: Assistant Professor
Date: 4/ 7/ 2023
**(Member and Supervisor)**

Approved by the Dean of the College of Computer Science & Information Technology, University of Kerbala.

Signature:
Name: Assist. Prof. Dr. Ahmed Abdulhadi Ahmed
Title:
Date: 4 / 7 / 2023
**(Dean of Collage of Computer Science & Information Technology)**

# Dedication

This work is dedicated to the family of the Prophet Muhammad "peace be upon him and his family"

This work is also dedicated to

My supervisor, Dr. Ahmed

My Grandfather

My Parents

My brothers and sisters

My second family

My husband

My Children

# Acknowledgement

Know that the success you have achieved is only by God's will and success. I can only thank him and express my deep gratitude to him. God's grace and support have been essential in my research journey, and I owe God all my progress.

I would like to express my thanks to my professors in the Department of Computer Science and Information Technology

Grateful to have had the support and guidance of my supervisor, Dr. Ahmed. He demonstrated a high level of commitment and dedication by closely monitoring my progress and providing valuable feedback and suggestions, his patience and understanding during challenging times were truly appreciated.

I would like to express my heartfelt gratitude to my late father, who has been a constant source of everything.

To my dear mother, thankful for your support and endless inspiration. Your love and encouragement have been invaluable to me.

To my brother (Fadhil and Ali) and my sisters (Tabark and Nabaa)
I would like to express my deep gratitude to you for your continued support and encouragement, and I consider myself fortunate to have you here with me.

Thankful to My father-in-law, for his constant encouragement and guidance. His wisdom and advice have been invaluable in motivating me to strive for success.

My mother-in-law, you have always played a great role in encouraging me and pushing me toward achieving my ambitions. Thank you for your friendship and unconditional love.

To my loving husband, I am grateful for your unwavering support throughout my journey. Your belief in my abilities and your desire to see me succeed has been instrumental in my achievements.

To my beloved children (Muhammad, and Ibrahim), you are the light of my life.

# Abstract

Hierarchical frameworks have traditionally been used by humans to organize their thoughts on complex problems. One application of artificial intelligence involves the creation of intelligent agents that break down challenging problems into layers of abstraction, simplifying the problem-solving process. The Hierarchical Task Network (HTN) structure enables tasks to be effectively performed.

In the context of autonomous robots operating in closed and deterministic environments, such as domestic environments, task planning is crucial for achieving a high level of accuracy. To address this, the HTN planner and descriptive action models were employed to determine the next state in the state transition system during the planning process for the generated plan. A formal representation for operational action models was introduced, and their effective utilization in conjunction with the Refinement Acting Engine (RAE) and the Dijkstra algorithm to determine optimal task approaches was explained for the execution of this plan.

The integration planning and acting algorithm was designed to operate hierarchically, utilizing a transit tree and backtracking to handle the error for re-planning and execution in the event of execution errors. Careful attention was given to method implementation to avoid system failure or suboptimal performance.

The methodology used in the text involves addressing the task planning challenges faced by autonomous robots operating in closed and deterministic environments, specifically domestic environments. To achieve a high level of accuracy in task execution, the Hierarchical Task Network (HTN) planner and descriptive action models were employed with representation for operational action models the Refinement Acting Engine (RAE), and the Dijkstra algorithm to determine optimal task approaches, The hierarchical structure and error handling mechanisms contribute to achieving accurate and efficient task completion while maintaining system reliability depending on The integration

planning and acting algorithm was designed to operate hierarchically, utilizing a transit tree and backtracking algorithm.

Through experiments conducted using the Python language (Spyder), it was demonstrated that the algorithm developed outperforms the commonly used planning and acting approach, providing an optimal path within the system. The results showed a precision value of approximately 94.9%, a recall value of approximately 94.9%, and an F1-score of approximately 94.79%. These findings validate the effectiveness of the approach in improving the planning and acting process.

## Declaration Associated with this Thesis

Some of the works presented in this thesis have been published and accepted as listed below.

- Z. Al-Ghanimi and A. A. Al-Moadhen, "Deliberative Robotics Behavior Systems: Asurvey," 2022 International Conference on Data Science and Intelligent Computing (ICDSIC), Karbala, Iraq, 2022, pp. 147-152, doi: 10.1109/ICDSIC56987.2022.10076116.
- Zainab Al-Ghanimi and Ahmed Abdulhadi Al-Moadhen, "Integrating Planning and Acting in Autonomous Robots through a Unified Hierarchical Representation Model", A Canadian journal of applied mathematics, computer science and statistics, vol. 120, pp. 382–392, Jun. 2023.

# Table of Contents

Dedication ..... i
Acknowledgement ..... ii
Abstract ..... iii
Table of Contents ..... vi
List of Tables ..... viii
List of Figures ..... ix
List of Abbreviations ..... xi

**CHAPTER ONE: INTRODUCTION .....**

1.1 Overview ..... 1
1.2 Motivation ..... 3
1.3 Problem Statement ..... 6
1.4 The Aim of the Thesis ..... 6
1.5 Objectives ..... 7
1.6 Thesis Organization ..... 8

**CHAPTER TWO: THEORETICAL BACKGROUND .....**

2.1 Overview ..... 9
2.2 Robotics ..... 9
2.2.1 Planning ..... 10
2.2.2 Acting ..... 11
2.2.3 Planning versus Acting ..... 11
2.3 Deliberation ..... 14
2.3.1 Deliberation Models ..... 15
2.3.2 Knowledge Representations ..... 16
2.4 Planning Domain and Problem Representation ..... 17
2.5 Techniques for Planning System ..... 18
2.5.1 STRIPS Model ..... 18
2.5.1 Definition of the PDDL Language ..... 18
2.6 Approaches Task Planning ..... 19
2.6.1 Classical planning ..... 19
2.6.2 Hierarchical Task Network ..... 21
2.7 Refinement Acting Engine ..... 27

2.8 Dijkstra Algorithm ..... 28
2.9 Related Work ..... 30
2.9.1 Planning system ..... 30
2.9.2 Planning and Acting System ..... 31
**CHAPTER THREE: PROPOSED METHODOLOGY .....**
3.1 Overview ..... 37
3.2 HTN PLanner ..... 37
3.3 Deterministic environment ..... 41
3.4 Refinement Acting Engine ..... 44
3.5 Integration planning and acting ..... 46
3.6 Plan validation ..... 53
**CHAPTER FOUR: RESULTS AND DISCUSSION .....**
4.1 Overview ..... 54
4.2 Domestic Environment ..... 54
4.3Domain environment ..... 55
4.3.1Tasks Robot ..... 55
4.4 Feature HTN Planning ..... 58
4.4 Evaluation system ..... 59
**CHAPTER FIVE: CONCLUSIONS AND FUTURE WORK .....**
5.1 Overview ..... 64
5.2 Conclusion ..... 64
5.3 Limitations ..... 65
5.4 Future work ..... 66
**REFERENCES .....67**

# List of Tables

***Table 2.1:*** *Summary of previous works.....34*

***Table 4.1:*** *presents a displaying the number of actions for TP, TN, FP, FN, Precision, Recall, and F1-score .....63*

# List of Figures

***Figure 1.1:*** *Deliberative Architectures.....5*

***Figure 2.1:*** *Planning as a process of searching and a succession of phased predictions [2]. .....12*

***Figure 2.2:*** *Continual online planning and acting. .....16*

***Figure 2.3:*** *A planning domain is viewed as a system in which states change. .....22*

***Figure 2.5:*** *example for travel task.....26*

***Figure 3.4:*** *Refinement tree for task using differ methods.....45*

***Figure 3.5:*** *instance method for refinement.....50*

***Figure 4.1:*** *Domestic Environment.....54*

***Figure 4.2:*** *system implementation result.....58*

***Figure 4.3:*** *Displays the result of calculating metrics to evaluate Precision, Recall, and F1-score.....62*

# List of Algorithms

***Algorithm 2.1:*** *Dijkstra algorithm…….. .29*
***Algorithm 3.1:*** *Refinement Algorithm. ……47*
***Algorithm 3.2:*** *Backtracking Algorithm. ……51*

# List of Abbreviations

| Abbreviation | Description |
|---|---|
| APE | Acting Planning Engine |
| APEplan | Acting-and-Planning-Engine Planner |
| BT | Behaviour Trees |
| FN | False Negative |
| FP | False Positive |
| GOAP | Goal-Oriented Action Planning |
| HGN | Hierarchical Goal network |
| HTN | Hierarchical task network |
| PDDL | Planning definition domain language |
| PRS | Procedural Reasoning System |
| RAE | Refinement Acting Engine |
| RAEplan | Refinement-Acting-Engine Planner |
| RRT | Rapidly-exploring random trees |
| STRIPS | Stanford Research Institute Problem Solver |
| TN | True Negative |
| TP | True Positive |
| UCT | Upper Confidence Bound on Trees |
| UPOM | UCT Planner for Operational Models |

# CHAPTER ONE:

# INTRODUCTION

## 1.1 Overview

Over time, one key area of interest in task planning has been advancing planning methods for robotic systems. Intelligent autonomous mobile robots are being extended in numerous applications to handle complicated tasks. These jobs might be local household duties performed within human houses, on remote planets, and can even underwater. To enable these robots to interact with their environment and carry out their assigned tasks, the environment should be intelligently discovered. This discovery will allow a robot to reason about its activities and available resources in a flexible and efficient manner. Planning and environment should also be integrated through the deliberation process because the knowledge base is crucial in expressing the organization of a robot's surroundings, as well as the relationships between entities (each an item and a task) and their properties [1].

Deliberation for acting includes selecting the actions to take and how to carry them out in order to attain a goal. It pertains to a thought process that takes place both prior to and during the execution of an action. It also deals with what will happen if an agent takes an action and which actions should an agent take, and how to perform this action to achieve this intended effect. The reasoning enables the agent to forecast, determine what to do and how to accomplish it, and integrate numerous acts that contribute to the goal. In Artificial Intelligence (AI), planning is studied as a deliberation process and performed computationally in the intellectual aspect of acting. This cognitive process is an explicit, abstract form of reasoning that involves selecting and organizing actions based on predictions of their outcomes[2].

The objective of this deliberation is to accomplish as many predefined goals as possible. The AI planning plan development part is mostly concerned with state transition difficulties. These issues have a starting state, the desired goal, and a collection of feasible actions to modify the state. A plan consists of activities that can be initiated from the starting condition and, when implemented, will change it into a state where the objective has been met. The most basic type of planning issue is the "classical planning problem refers to the general planning of restricted state-transition systems. This type of planning presupposes a deterministic, discrete, and non-temporal world model. The problem is modeled in a language that tells what information about the world can be true at the time. The information that are relevant to the planning process are known as predicates or facts. Predicates provide us with details that may become important at a later stage of planning. These predicates together form the state, which represents the current state of the world at a specific point in time [3].

The Hierarchical Task Network (HTN) is a planning subfield in AI that organizes plans hierarchically. While HTN planning shares similarities with classical planning, as each world state is depicted by a sequence of atoms, there are certain distinct characteristics, particularly in terms of what they plan for and how they prepare for it, it differs from traditional planners. In the context of Hierarchical Task Network (HTN) planning, an environment's state is depicted as a set of atoms, with each action corresponding to a deterministic change in state. The primary aim of an HTN planner is not to accomplish a particular set of goals, but rather to complete a sequence of tasks[3].

## 1.2 Motivation

The hierarchy is one of the most common frameworks used to comprehend and conceptualize the state of the world lies in its effectiveness and widespread applicability. The hierarchical structure allows for the organization and categorization of complex information in a systematic and manageable manner.

Human beings naturally seek ways to make sense of the world around them, especially when faced with intricate and multifaceted phenomena. The hierarchy provides a clear and structured approach to understanding complex systems by breaking them down into smaller, more manageable components.

By utilizing a hierarchical framework, individuals can identify relationships, dependencies, and patterns within a system. This allows for a better grasp of the overall structure and functioning of the subject matter under consideration. The hierarchical arrangement facilitates the classification and organization of information, enabling easier navigation and comprehension.

Furthermore, hierarchy promotes a top-down approach, where overarching concepts or categories are defined first, followed by the subdivision of these concepts into more specific subcategories. This step-by-step progression helps in building a comprehensive understanding of the subject matter and facilitates effective communication and knowledge sharing.

Overall, the motivation behind using hierarchy as a framework for comprehending and conceptualizing the state of the world is rooted in its ability to provide structure, organization, and clarity to complex systems,

There by enhancing our understanding and enabling effective decision-making and problem-solving.. HTN planners' use of an intuitive hierarchical architecture makes it very simple to integrate readily available expert information about a subject to drive the search process. Task networks record practical procedural control knowledge or instructions on how to carry out a task defined in terms of a breakdown into subtasks. There are several domain-independent HTN planners available (SHOP, SHOP2, O-PLAN, and O-PLAN2 [4][5][6][7]). To create a plan in HTN planning, the process involves breaking down tasks into increasingly smaller subtasks until basic, executable tasks with constraints are obtained. Effective methods can assist an HTN planner in achieving desirable outcomes. Hence the inclusion of search control information can accelerate the HTN planning process beyond the speed of classical planning [3]. HTN planning has received a widespread application in robotics mission planning, as well as gaming AI creation [8], [9]. As demonstrated in Figure (1.1), AI planning and action models can be defined in two fundamental ways: through descriptive models and operational models [3]. Descriptive models are more abstract than operational models. Descriptive models abstract away the intricacies of an action and focus on the primary effects; they are appropriate at higher levels of a deliberative hierarchy.

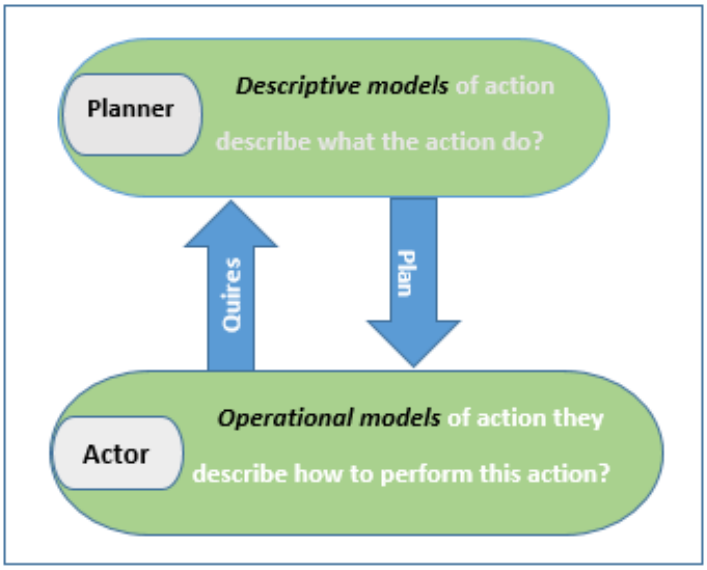


***Figure 1.1:*** *Deliberative Architectures.*

At runtime, the actor typically deliberates on how to carry out the tasks it is currently undertaking. The deliberation continues incomplete until the actor achieves its goal, which may include flexible modifications to its plans and return results [2].

In a state transition system, HTN planners rely on descriptive action models to effectively determine the next states. While this approach works well in a closed, static, and deterministic environment, it falls short in domains that are open, dynamic, and non-deterministic/probabilistic. These are commonly encountered in real-world scenarios. The planning domain rarely provides a completely accurate representation of the actor's environment, and executing plans may result in failure due to various reasons, such as action execution failures, unexpected events, or incomplete/incorrect knowledge during the planning stage [3].

## 1.3 Problem Statement

1- Importance of effectively integrating planning and action in a unified hierarchical model. This integration is necessary to address the process of deliberation between planning and action, ensuring that plans align with the actions taken.

2- Selecting the optimal path or method when executing an action in an actor, particularly in the context of a refinement acting engine within a Hierarchical Task Network (HTN) planning framework.

3- Handling action failures during execution and proposing a solution by combining an HTN planner with an RAE to enable backtracking and re-planning when actions fail or unexpected events.

## 1.4 The Aim of the Thesis

1- Propose a hierarchical representation that unifies the descriptive and operational model, with actions ranging from abstracted levels to more detailed ones, commands.

2- Develop an algorithm that enables the planner and actor to work consistently using the unified representation.

## 1.5 Objectives

1- Search the descriptive action models required for planning and the operational action models required for acting and maintaining consistency between them.

2- Using artificial intelligent Algorithms Including 1- Hierarchal Task Network, 2- Refinement Acting Engine, 3- Dijkstra Algorithm, 4- Run-Lazy-Lookahead, and 5- backtrack algorithm.

3- Develop a hierarchical system with integrated planning and acting algorithms that employ both descriptive and operational models. By maintaining the hierarchy within the planning solution, users are presented with a solution tree that includes the plan and a set of refinement techniques that provide different ways of handling tasks and responding to actions for closed-loop online decision-making., as well as how to select the by using optimal path between methods in the operational models.

4- A proposed development algorithm that integrates an HTN planner with an RAE, allowing for backtracking and re-planning when actions fail to execute as intended.

## 1.6 Thesis Organization

The thesis is organized into five chapters, the remaining chapters of the thesis are outlined as follows:

**Chapter Two**: presents the content and characteristics of robotics-related approaches and the most commonly used methods, and explains their importance and limitations, and described some of the work associated with these techniques.

**Chapter Three**: presents the methodology used, including the structure and related algorithms in detail.

**Chapter Fourth:** presents the performance evaluation and results of the simulation. We describe the domain and tasks we used to conduct our experiments.

**Chapter Five:** draws conclusions related to this research work and suggests future work that will help future researchers to improve the performance of (the hierarchical system).

# CHAPTER TWO:

# THEORETICAL BACKGROUND

## 2.1 Overview

In this chapter, we have explained what robots are, their parts, and the various environments that deal with them, as well as described the deliberation process with its models, and planning system techniques. It also explained the most used approaches to solve planning tasks and described some of the work associated with these techniques.

## 2.2 Robotics

Robotics is a field of study that focuses on designing mechanical devices that are capable of autonomous movement. Its a multidisciplinary field of study that revolves around the design, development, and implementation of mechanical devices, known as robots, with the ability to perform tasks autonomously. It combines various branches of engineering, such as mechanical, electrical, and computer science, along with elements of mathematics and physics [10].

The primary objective of robotics is to create machines that can perceive their environment, make decisions, and manipulate objects or interact with their surroundings without continuous human intervention. These robots are typically equipped with sensors to gather information about their environment, processors to process that information, and actuators to execute physical actions based on the processed data [11].

The study of robotics encompasses various subfields, including robot kinematics and dynamics, control systems, perception and sensing, artificial intelligence, machine learning, and human-robot interaction. Researchers and engineers in robotics strive to develop robots that can perform a wide range of tasks efficiently and effectively, such as industrial automation, medical assistance, exploration of hazardous environments,

and even social interactions[10]. The objective of robotics is to create machines capable of aiding humans in diverse tasks. These machines are typically complex and can perform a variety of actions automatically, depending on computer programming. A robot that is capable of functioning independently without the need for human input or guidance is considered an intelligent machine. Such machines can complete tasks and operate effectively in a given environment. In many cases, robots that possess a high degree of autonomy can carry out tasks that would normally require human labor[12].

### 2.2.1 Planning

The process of creating an action plan to complete a task is known as planning. In order to automate planning, a computer program must represent: 1- the world, 2- represent actions and their effects on the world, 3- reason about the effects of sequences of such actions, 4- reason about the interaction of actions that are happening at the same time, 5- control the search process to find plans with a level of efficiency.

A major challenge for artificial intelligence is the capacity for action-based reasoning. Common sense reasoning is often employed by individuals, whereas the reasoning function is an explicit and abstract process of deliberation that selects and organizes actions by anticipating their consequences[13]. The primary objective of this deliberation is to active the maximum number of predetermined goals possible. Several planning systems are available[11]:

1. Motion and task planning
2. Temporal planning
3. Probabilistic planning

4. Planning in open domains

The task planner was used in this thesis. A high-level plan is created through task planning[14]. The task planner has to be given a description of the manipulable items, the task environment, the robot, and the beginning and intended ultimate states of the environment.

The ultimate result has to be a robot software that can change the original condition into the required final state [12]. Classical planning and hierarchical planning are the most often used methods for addressing planning challenges. They typically rely on heuristic search, Nonetheless, traditional planning approaches are increasingly sophisticated, especially with regard to domain-independent heuristics.[15].

### 2.2.2 Acting

This section discusses how robots must engage in planning tasks in order to operate effectively within their physical environment. As a result, the planner's actions will be decomposed into a sequence of actions that the robot system will execute. Throughout the execution process, the actor directs the designated system elements using commands [2].

### 2.2.3 Planning versus Acting

Creating a deliberative machine that integrates planning and acting tasks is a crucial challenge in design. Planning aims to generate a set of logical actions that can accomplish a specific task.

For example, you may do this by using a lookahead strategy, that combines prediction phases Figure (2.1), while in the state (s), action (a) is anticipated to build a state (s') inside of a search through several sets of actions for a set that leads to the ideal target state. Refers lookahead to a technique used in decision-making processes to anticipate the potential outcomes of future actions. It involves simulating different future scenarios and evaluating their potential consequences before selecting the best course of action. Lookahead is particularly valuable in domains where decisions have long-term consequences and where the decision-maker seeks to optimize outcomes based on a forward-looking perspective. The difficulty of planning is influenced by the types of tasks that require planning, the types of predictive models required, and the types of plans considered suitable. Specific planning techniques for particular types of problems have been developed to tackle different challenges in various domains [2].

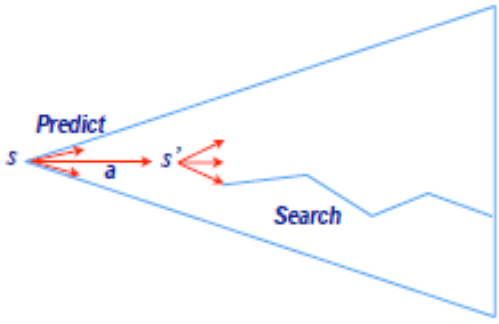


***Figure 2.1:*** *Planning as a process of searching and a succession of phased predictions [2].*

The process of acting include selecting how to execute the actions that have been selected or decided upon while adjusting to the context in which the activity is taking place. This can be done with or without the assistance of a planner.

Each action is regarded as a general task that will undergo refinement into more specific actions or instructions based on the existing circumstances. Planning involves looking for projected states, whereas acting necessitates continuously comparing the current condition to predicted states and making adjustments as necessary. Acting, also entails responding to unanticipated changes and external occurrences. Depending on how straightforward planning is and how rapidly the environment changes, there are several methods to arrange the interaction between acting and planning. One straightforward sequence of steps is the linear progression of "plan then act" which does not accurately describe the relationship between planning and action. instead, it suggests that the relationship is more complex and dynamic. The section likely elaborates on the limitations of the linear progression model by highlighting factors such as feedback loops, ongoing adjustments, and the iterative nature of planning and action.

It may discuss how planning and action interact with each other in a continuous cycle, rather than being distinct and separate stages. For example, during the execution of a plan, unforeseen circumstances or new information may arise, necessitating adjustments to the original plan. These adjustments can then inform subsequent actions, which in turn may require further planning modifications. This iterative process highlights the ongoing interplay between planning and action throughout the course of a task or project.

By explaining the limitations of the linear "plan then act" model, the section emphasizes the importance of recognizing the dynamic nature of planning and action.

It highlights the need for flexible approaches that allow for adaptation, learning, and continuous improvement based on the feedback received during the execution of a plan.

It's not always possible or necessary to seek a clear plan before taking action. When the environment is well-modeled and predictable, such as in a manufacturing production line. This approach is necessary when the cost or risk of acting is high and the actions are irreversible. In such applications, it is often necessary for the designer to minimize the diversity of the environment beyond what can be modeled and predicted [2].

## 2.3 Deliberation

A set of components that are hierarchically arranged can carry out the decision-making process of the robot. A robotic system can be given more extensive, adaptable, and durable functions by using the current and significant study area of deliberation in robotics. The use of prediction models and the acquisition of these models are both necessary for deliberate action. An actor can also need to develop their ability to adjust to different roles and circumstances. Deliberate action cannot be boiled down to a single function; planning is merely the first step. Integrating many functions coherently is crucial. It requires the functions to be consistent with one. Another deliberation may necessitate several different functions. The robot's cognitive architecture involves five main components: Planning, Observing, Acting, Goal reasoning, Monitoring, and Learning. In this research, we focus on the processes of planning, acting, and integrating them. The relationship between planning and acting is fundamentally based on the idea of deliberation. Deliberation for acting entails selecting the actions to take and how to carry them out.

It takes one or more acts that are justified by the hierarchal sequences planned to achieve this goal. Deliberation refers to a process of reasoning that occurs both before and while acting. For instance [2]:

- What will happen if an agent acts?
- Which steps should an agent choose should adhere to in order to have the desired impact, and how should the agent go about doing so?

Autonomy, or the ability to carry out one's intended tasks without being directly commanded by a human, is what drives agents with deliberative capacities. They assume that instructions, a collection of primitives that perform sensory-motor control, are used to carry out actions. The actor's actions are carried out by putting directives into action. It employs models of how these commands function to engage in deliberation. The process of planning comprises choosing and arranging the procedures required to achieve a certain objective. The steps that need to be followed are frequently known [2].

### 2.3.1 Deliberation Models

To choose which acts to do and how to take them, an entity engaged in acting needs anticipatory models of its actions. Descriptive and operational models are used, respectively, to express these two forms of information. One can create both descriptive and operational models of activities by utilizing state variables that represent the state of an actor and its surroundings.

- Descriptive action models define the state or collection of possible states that can occur as a result of completing an action. Plans produced using descriptive models operate effectively under the assumptions of a closed, unchanging, and deterministic reality (this plan).

- Operational models, on the other hand, define how to carry out an action involves determining the appropriate commands to perform in the current environment and organizing them in a way that achieves the desired outcome. A plan's action result in state transformations, which change the set of true facts in accordance with the effect of the action (execution of plan).

Acting needs online planning and deliberation continual. As shown in Figure (2.2), an actor must break down their actions into smaller phases in order to access the necessary operators [16].


***Figure 2.2:*** *Continual online planning and acting.*

### 2.3.2 Knowledge Representations

An agent must carefully plan and execute its actions, which requires models for organizing, monitoring, modifying, and adapting its strategies and plans. The agent also needs methods for breaking down its activities into manageable steps or assigning them to specific capabilities.

Descriptive models, which are essential for reasoning, address issues such as causes, consequences, motives, benefits, and costs of actions. In contrast, operational models, often referred to as task performance standards, deal with the "how" aspects of actions. Certain models might possess specifications for both operational and descriptive aspects [17].

The required knowledge representations for actors must facilitate the uniform and generic specification of both "know-what" and "know-how" action models. It is important to note that the actor's environment must be taken into account in both types of models. The knowledge representations that actors are required to use should make it easier to specify both the "know-what" and "know-how" models of actions. They ought to support effective algorithms for deliberation [18].

## 2.4 Planning Domain and Problem Representation

The terms "planning domain" and "problem representation" denote the approach of encoding all relevant information about the robot's environment for all the tasks that carry out. The elements of the world that describe the problems are formalized in the resulting domain.

Given that each style has a particular level of expressiveness and complexity, this formalization of knowledge representation plays a crucial role in determining the types of problems that can be represented and solved [19]. It goes without saying that a language can represent a wider range of issues the more expressive it is. As a result, complex systems and concepts encountered in the actual world may be represented more easily in expressive languages.

However, the intricacy of the language frequently increases the computational difficulty of the methods employed to solve the problem, which results in a longer processing time [20].

## 2.5 Techniques for Planning System

Since the last years, the study of planning systems has been ongoing activity for study. And numerous methods, algorithms, and systems have been put forth. They are very different from one another in terms of how they model the world or how they approach problem-solving.

### 2.5.1 STRIPS Model

The Stanford Research Institute Problem Solver (STRIPS) planner is frequently cited as the first system to implement a dedicated planning algorithm, and its action modeling approach, which defines prerequisites, positive effects, and negative effects as sets of atomic facts, is still being utilized today[21]. The STRIPS style represents planning problems using the triplet "I, A, G," which includes the initial state (I), a list actions (A), as well as a list of goals (or desired end-states(G)). The states are represented as a collection of predicates based on first-order predicate logic[22].

This type of planning was constantly changing states to achieve the desired state by using the heuristic and already-applied action details.

### 2.5.1 Definition of the PDDL Language

The Planning Domain Definition Language is an industry-standard language for creating STRIPS domains and problem sets (PDDL) (The

most basic subset of PDDL is referred to as "STRIPS")[21]. Most of the code can be written in PDDL using English words so that it can be easily read and understood. Simple AI planning problems can be written using this method fairly easily[23].

The PDDL The definition of a planning problem can be divided into two components: the domain and the problem itself. The domain component specifies: the state variables and actions in a generic model of the relevant environment, while the problem describes a specific instance referring to the problem definition within the given domain, including the starting state and target condition. STRIPS and PDDL can be utilized to solve a variety of problems. If a limited number of actions, prerequisites, and effects is capable of illustrating the world domain, a PDDL domain and problem can be created to solve it [21].

## 2.6 Approaches Task Planning

Two of the most common methods for completing planning tasks are classical planning and hierarchical planning. Solvers typically use heuristic search, while classical planning approaches are more sophisticated, especially with regard to domain-independent heuristics [7].

### 2.6.1 Classical planning

The so-called "classical planning problem" is the most basic type of planning problem often discussed. It makes the assumption that the universe is predictable, discrete, and fundamentally non-temporal [21].

Classical planning involves finding a series of actions that transforms a starting state to a target condition, under the assumptions of a deterministic environment and actions. The primary computational difficulty is developing efficient methods for generating such action sequences, which are commonly referred to as plans [2]. Planning refers to the process of discovering a series of actions or steps in a deterministic environment that transforms a starting state into a desired state. Task is to develop effective methods for obtaining such action sequences, commonly referred to as plans, which presents a computational challenge. A typical planning problem can be presented as a directed graph, where the nodes correspond to states, and the edges indicate the actions that change the state of the edge's source node to the state of its destination node. One way to express the problem is as a challenge of finding a path or sequence of actions. Hence, a plan can be considered as a path from the starting node in the graph to a node that represents one of the target states. [15]. The following is a formal model for a traditional planning issue:

(Classical Planning Model) Definition. A planning model = [S, $s_0$, $S_G$, A, fi] is made up of the following elements [24]:

- The state space is composed of an initial state s0 S, a finite and discrete collection of states' S,
- A group of objective states SG S,
- A sequence of acts
- The appropriate A(s) A actions for each state's S, and (s).

The outcome of an action, a, in a state, s', is f(s, a), often referred to as s[a]. The use of a series of activities to change a state may be characterized as

$$\mathbf{S\,[a_0 \ldots a_n] = (s\,[a_0 \ldots a_{n-1}])\,[a_n]} \tag{2.1}$$

### 2.6.2 Hierarchical Task Network

The AI planning approach, one of type of planning is known as Hierarchical Task Network (HTN) planning, differs from traditional planning. The fundamental concept of this approach involves a specification of the initial states, a task network that serves as a set of achievable objectives, and the approach involves utilizing domain knowledge that encompasses networks of both simple and complex tasks. Each task that can be performed is represented in a hierarchical manner in a task network or decomposed into more detailed subtasks if the task is primitive [25]. The planning process commences by decomposing the primary task network and continues until all composite tasks have been decomposed, which denotes the finding of a solution. This approach consists of a series of fundamental steps that apply to the world's initial condition and carried out with the aid of the planning operators. Creating problem-solving "recipes" that mimic the problem-solving approaches of a human expert in a particular domain. Approach a planning challenge is made simple with HTN approaches. The plan is usually represented as a symbol[11].

Definition from transition state (s; a), Figure (2.3), which depicts what happens when a state undergoes an action, explains the outcome and uses the same concept as traditional planning. To define a planning problem and its solutions, the language on the other hand contains tasks, methodologies, and task networks[2].

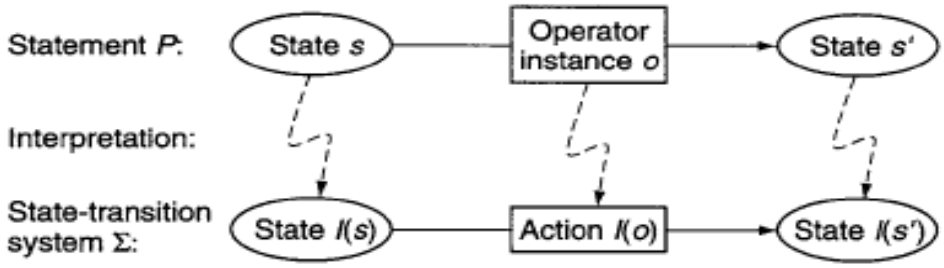


***Figure 2.3:*** *A planning domain is viewed as a system in which states change.*

The words "problem representation" and "planning domain" refer to the methodical coding of all pertinent information about the robot world for all of the tasks the robot does. The aspects of the world that will be utilized to define issues are formalized in the resultant problem. The theoretical framework is made up of tasks, operators, task networks, techniques, a planning issue, and a solution. The planning system's input consists of a number of operators and techniques, each of which describes how to break down a task into a number of smaller tasks (smaller tasks)[26]. Planning progresses by constantly breaking down non-primitive tasks this task decomposed into increasingly smaller subtasks using the methods employed until it approaches Primitive tasks are tasks that can be directly handled by the planning operators [8].

The planning problem is the set of beliefs the actor has about the world, divided to 4-part:

The problem definition $S_0$ issue is broken down into:

$$\mathbf{P= (S_0, W, O, M)} \qquad (2.2)$$

- Initial state by enumerating all the right data in that state, the (init) section defines the starting state used to represent the issue. These details are referred to as facts or predicates.

Before adding any action to convert the robot site to another, for instance, while creating the work environment, the initial condition of the robot site is a basic fact.

- A task network is composed of a set of task nodes, denoted as U, and a set of constraints denoted as C. These constraints define conditions over U that must be satisfied during the planning process and by the resulting HTN solution [27] and planning domain:-

$$\mathbf{D = (M, O)} \tag{2.3}$$

- An acting domain's specification we simulate a 4-tuple for **method** Using:

$$\mathbf{m = (name\ (m), task\ (m), subtasks\ (m), constr\ (m))} \tag{2.4}$$

- Name (m): One of the unique features of the approach is to ensure that no two methods in the planning domain can have the same method symbol. The names of the methods enable us to refer to them without explicitly specifying the preconditions and effects when replacing method instances.
- Task (m): non-primitive job, divides u into smaller tasks (m). A task network consists of (Subtasks (m), Constraint (m)). A task may have numerous ways, each reflecting a distinct approach of refining that work.
- An action is a process that converts the state of the robot. This includes perceiving and learning about its environment. The hierarchy of abstraction comprises multiple levels, where each level represents a different degree of abstraction It is possible to observe an activity. It is primitive at some levels and compound at others.

For example, opening (a door) is an abstract basic action and a compound job that may be developed into a set of actual actions when necessary. At the bottom of the hierarchy, the most fundamental action is a command. The robot platform can execute commands instantly. A three-tuple action is:

$$\mathbf{a = (name\ (a), precond\ (a), effects\ (a))} \tag{2.5}$$

Before the action is performed, the state **S** must fulfill the precondition (**a**), and after the action is completed, the state must satisfy the effects (**a**).

- A name: which may include arguments.
- A precondition list: is a set of facts that must be true in order for an action to be carried out.
- An effect: list of facts made true by executing the action.All these sequences of action are depending on the heuristic system. We can see in Figure (2.4) Task network visualizations [27].

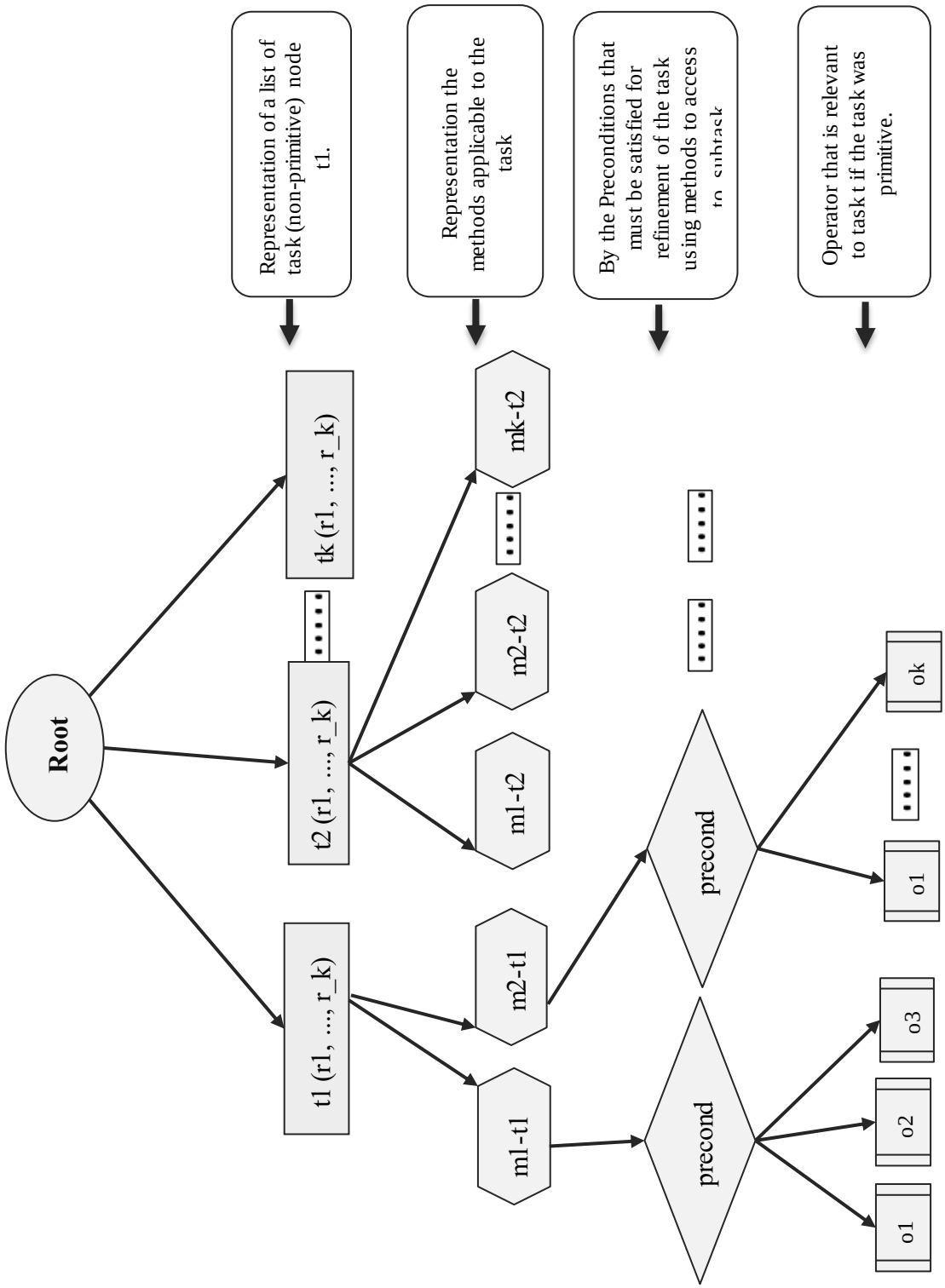


***Figure 2.4:*** *Task network visualizations involve representing a set of tasks and their interdependencies in a graphical format.*

Can describe this chart as a simple example in Figure (2.5). We divide the Travel task from more abstract task access more detailed commands.

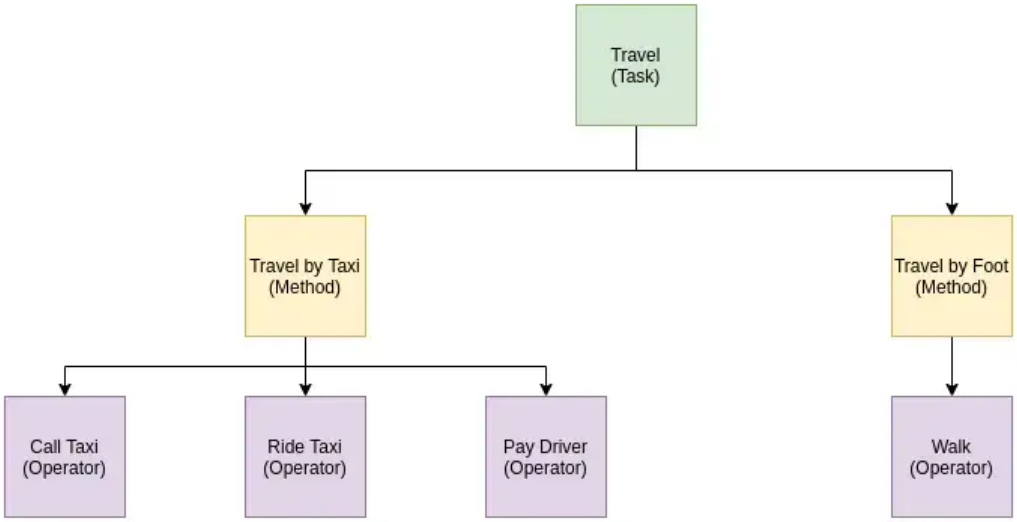


***Figure 2.5:*** *example for travel task*

This example illustrated that there are two methods for completing a travel task: either by taxi or on foot. Based on factors like our location, our financial situation, and other factors like the state of the world, we can select the best method.

A task is considered primitive if one of its symbols is a symbol for an operator; otherwise, it is non-primitive.

The method symbol, which represents a method's name, is the second new symbol. A method exists for non-primitive activities.

There is a way for decomposing an abstract task into less abstract tasks, primitive tasks, or a combination of both for non-primitive activities.

A method has the following components:

1- Name of method for example, travel by taxi.

2- Task for example, travel, a task that it can solve.

3- Preconditions: the world's current condition determines whether the approach is still applicable.

4- task network (or subtasks) a task network used for visualizing more tasks in networks [2].

## 2.7 Refinement Acting Engine

Hierarchical Task Networks (HTNs) are commonly used in robot planning to represent complex tasks and their dependencies. To execute an actor-dependent HTN in planning a robot, you need to define the HTN Structure: This involves breaking down the complex task into smaller sub-tasks and organizing them hierarchically based on their dependencies, Define actor: Actors refer to the agents that are responsible for completing the tasks.

In this case, the robots will be the actors, assign tasks to the actor: Based on the dependencies defined in the HTN structure, you need to assign the tasks to a robot, Monitor progress: As the robots start working on the assigned tasks, you need to monitor their progress and ensure that they are following the correct sequence defined in the HTN. In a summary, executing an actor-dependent HTN in planning a robot requires careful planning, organization, monitoring, and optimization to guarantee the effective accomplishment of the intricate task [28]. We propose an action technique called Refinement Acting Engine RAE incorporates the necessary techniques for performing actions using this representation. Describes an operational model formalism based on refining techniques. This method describes how to carry out a task (an abstract task of some kind) by breaking it down into more concrete tasks. RAE uses a variety of refinement methods. To provide commands for the execution platform, abstract activities are iteratively refined into less abstract ones [29].

The Refinement Action Engine (RAE), which has the capability to test approaches in making judgments, offers the procedures necessary to deal with this representation. To manage the acts that the actor must do and new events to which he must respond, RAE employs a library of methods called M. RAE receives the following inputs:

- A collection factual information representing the world as it is right now.
- Collection task that need accomplished.
- A series events indicating external occurrences the performer might have to respond.

Task-defining sources, such as planners or users are where tasks originate. RAE chooses a pertinent method m and builds a LIFO stack for each task in the input stream to keep track of the refinement's status. If m doesn't work, RAE will try an alternate strategy that is appropriate right now and hasn't been used before.

The last-in, first-out (LIFO) list known as a refinement stack is composed of tuples with the following format: (T; m; I, tried) [30] I is a reference to body of m[30], and it is initially set to 1, tried is a collection of refining method executions for T that have already been attempted but failed. The standard push, pop, and top operations are used to manage a stack [30].

In a stack, progressing means going up one step at a time [30]. If this part is a command, RAE sends it to the execution platforms; if it is a task, RAE chooses an appropriate method and places it at the top of the stack; and if this step is to exit from m, RAE removes m from the stack.

## 2.8 Dijkstra Algorithm

There are many algorithms that deal with calculating distances and choosing the most appropriate path based on system characteristics and employing them for this work, such The Dijkstra algorithm, named after its inventor E.W. Dijkstra, is used to compute the shortest path between a starting point (the source) and a destination in a graph [31].

As the Dijkstra algorithm in 2.1 can compute the shortest paths from a single source to all nodes in a graph simultaneously. This issue is also known as the single-source shortest paths problem. GPS technology employ this algorithm to one can determine the most optimal path from the current location to the destination by finding the shortest route, It has many industrial uses, particularly in fields where modeling networks is necessary

Dijkstra's Algorithm Fundamentals:

- The Dijkstra's Algorithm begins by selecting a starting node (also known as the source node) and then evaluates the graph to identify the shortest possible path from that particular node to every other node present in the network.

- After identifying the shortest path between the source node and the second node, the algorithm marks the second node as "visited" and adds it to the path.

This process continues until all nodes in the graph are included in the path, connecting the source node to every other node through the shortest route. At each iteration, the algorithm updates the distances between adjacent nodes and selects the node with the shortest distance to the source node as the next one to be processed [32].

***Algorithm 3.1: Dijkstra algorithm.***

```
1.  Initialize-single-source(S,G)
2.  S ≠ 0                          //first-method-using-in-actor
3.  Q ← V [G]                      //combine-other-node
4.  While Q ≠ Ø                    //multiple-method-using
5.  Do u ← extract-min (Q)         // a linear search through all of Q's vertices
6.  Return ← u
```

## 2.9 Related Work

This section covers the different aspects of work that are related to the planning system and its integration with the acting system.

### 2.9.1 Planning system

In the past years, there are plentiful published works on the planning primitives, which reduced the acting function. J. Orkin, at el. [33] (GOAP) Goal-Oriented Action Planning is a simple method of action planning. R. E. Fikes and N. J. Nilsson, [22] STRIPS is a planning system designed primarily for controlling the conduct of autonomous characters in video games in real time. Real-time planning grants AI characters the ability to think. Think about the difference between planning and predefining state transitions. Real-time planning enables you to simulate the impact of different variables on logic and modify behavior accordingly. More recent games have included HTN planning after [33] showed that real-time planning is a workable approach for the current generation of games. There are numerous HTN planners available.

Some of the Interactive Planning and Execution System ken Currie and Custin Tate, [4] O-Plan (Open Planning Architecture) is the design and execution of a more flexible system targeted at assisting planning research and development, allowing for planning approaches, and allowing for powerful search controlling heuristics. A. Tate, at el. in A. Tate, B. Drabble, and R. Kirby [5] O-Plan2 A key contribution of this study to it is the comprehensive conceptualization of a planning and control system that is modular and adaptive.

In D. YueCao AmnonLotem Hector Muftoz-Avila, SHOP [6] as well as SHOP2's successor, and D. Nau, et al. [7] SHOP3, R. P. Goldman and U. Kuter [34]designed for video game AI planning. Shivashankar et al.[35] Create a task semantics for planning that was simply corresponding to the goal semantics for classical planning.

### 2.9.2 Planning and Acting System

Researchers sought to integrate acting and planning, so an actor may use a Refinement Acting Engine (RAE). S. Patra, et al. in this context[36] [37] To complete tasks in constantly changing contexts in a dynamic environment an RAE leverages hierarchical operational models in the planning procedure, as well as planning and learning algorithms, The experimental findings suggest that employing two separate measurements, specifically learning techniques considerably increases RAE performance based on its efficiency and success. Creating and advancing a cohesive acting-and-planning system that utilizes identical operational models for both components for the actor and make use of a hierarchical task-oriented language with cutting-edge Control structures designed for making decisions in real-time within a closed-loop system. F.Fklix Ingrand,et al. [38] The PRS system motivates the action [9], and it may seek guidance from the planner.

The RAEplan planning algorithm generates plans by simulating the actions of the actor operational models using Monte Carlo rollout simulations. RAEplan also use proper refining techniques to determine how to alter activities or occurrences.

Experiments demonstrate how much more effective the acting and planning systems are. The operational models used by the actor in the

system appear to be used in both acting and planning [37]. To demonstrate acting-and-planning, the APE algorithm is employed, which leverages hierarchical operational models based by the PRS system. In contrast to the reactive PRS method, APE makes decisions the planner, RAEplan, generates plans by utilizing Monte Carlo sampling to simulate executions of the actor's operational models.

APE-plan executes a subset of the available refinement techniques each time APE is to choose the method for refining a job, subtask, or event. When a refining technique is utilized, a command to the execution platform is issued at each site. The investigation contains a number of exciting realistic domain aspects, For example, factors such as dynamicity, real-time sensing requirements, information gathering, collaboration, and simultaneous operations.in S. Patra, et al [39]. To distinguish between domains with and without dead ends, the researchers utilized three distinct Metrics for evaluating performance include the success ratio, retry ratio, and speed of achieving success.

In D. S. Nau,at el. [40] The discrepancy between the descriptive action models needed for planning was the main emphasis of this work Acting requires operational action models, It should be addressed the difference between the descriptive action models needed for acting and the operational action models needed for planning.

Patra, et al.[16] [25] Using the actor's operational models, APE and UPOM integrate acting with planning. In V. Alcázar, at el. [41] RRT structures and search algorithms were created for use in continuous path planning issues as a search method for automated planning, with RRTs adapted for implicit and discrete search spaces. In D. J. Musliner, at el. [42] integration of planning and acting was performed by having the

planner synthesise a new plan, which can take a long time: the old plan was thus performed continuously in a loop, with the new plan only installed once planning was complete.

A. Menif et al. [43] Domain compilation and procedural task application/decomposition techniques are combined in the SHPE methodology.

It is designed to give video games quick planning capabilities. In Amanda Coles, at el. [44] COLIN is a forward-chaining heuristic search planner that reasons using Continuous Linear numeric change as well as accessing the full temporal semantics of PDDL to utilise classical planning and organise time. Simone Fratini, at el.[45] Organize time by offering knowledge representation the system describes a new controller composed uses the APSI-timeline-based TRF methodology to model and solve planning challenges.

M. Colledanchise [10] applied behaviour trees (BT) were applied for online planning and acting, which allowing the old plan to be run in a loop while the planner generates a new plan hence, the new plan is not installed until the planning is completed. BT can also react to unexpected events, though no refinement techniques (a mechanism to express several alternative refinements of jobs) were offered.

In R. Lallement,at el. [46] demonstrate the use of HTN planning in robots. The authors J. Wolfe, at el. [47]discuss how they used an HTN strategy to integrate task and motion planning. Motion primitives are evaluated for cost and feasibility using a specific solver and sampling.

Ron Alford, at el. [27] makes a contribution by formalizing GTN planning, a hybrid formalism that makes task and goal breakdown easier by using Goal-Task Network (GTN) planning, a method that integrates goal and task planning. Yuan, at el. [48] Use SHOP HTN Task modifiers

concept was introduced and how to improve the planning algorithm so that it can take advantage of the task modifier and give it goal reasoning abilities.

***Table 2.1:*** *Summary of previous works.*

| Authors | Years | Method used | Result |
|---|---|---|---|
| Simone Fratini, et al. [45] | (2011) | based on ESA APSI technology | Explains a brand-new planning module that models and resolves planning issues using a timeline-based methodology. |
| Amanda Coles, et al. [44] | (2012) | Linear program (LP) | Time managing Integrated planning and outcomes representation of knowledge. |
| Shivashankar, et al. [35] | (2012) | HGN | Provide task semantics that easily matched the goal semantics of classical planning and that, when used in the domains of classical planning, offered higher soundness guarantees. |
| Dana S, et al. [16] | (2015) | RAE | The discrepancy between operational action models, which are needed for action, and descriptive action models, which are needed for planning. |
| Ron Alford, et al. [27] | (2016) | GTN | Through the use of goal and task planning |

| | | | |
|---|---|---|---|
| | | | combined in a method called Goal-Task Network (GTN) planning. Contributions included formalizing Both task and goal decomposition are supported by a hybrid formalism known as GTN planning. |
| Xenija Neufeld, et al. [49] | (2017) | HTN | This paper summarizes and analyses the planner implementations used in several of these games in light of the various planner components. |
| Sunandita Patra, et al. [37] | (2018) | RAE & APE-plan | Operational models make it possible to deal with a range of situations and react to unanticipated results and occurrences in a constantly shifting environment, Performance is based on its efficiency and success. |
| Sunandita Patra, et al. [25] | (2020) | APE,UPOM and learning strategies | It was suggested to integrate acting and planning using the actor's operational models. The speed to success metric was used to evaluate the overall time required for planning and executing, including the failure cases. |
| Sunandita Patra, et al. [36] | (2020) | RAE , UPOM and learning strategies | Used hierarchical refinement operational models to provide a novel system for integrating action and planning. |

| | | | |
|---|---|---|---|
| | | | Results reveal that UPOM and the learning techniques greatly raise RAE's performance using the following two metrics: success and effectiveness |
| Weihang Yuan, et at. [48] | (2021) | THN | Task modifiers concept was introduced and How to improve the SHOP HTN planning algorithm so that it can take advantage of the task modifier and give it goal reasoning abilities. A task modifier modifies the task list in response to unexpected observations in the environment. |

# CHAPTER THREE:

# PROPOSED METHODOLOGY

## 3.1 Overview

In this chapter, we define the HTN planning formulation 3.1 after defining some of the fundamental words and representations. In Section 3.2, we define the environment in which this system operates. 3.3 Discuss the actor and their handling of the refinement process during the execution of operations. The refining approach is dealt with in 3.4 by employing an optimal path, along with the actor definition and how to connect with the planner. Finally, in 3.4, how to determine whether a plan is valid.

## 3.2 HTN PLanner

We adhere to the HTN formulation described in [3] which is presented in further detail in chapter two. Terms, literals, operators, actions, and plans used by us from traditional planning. A Hierarchical Task Network (HTN) is an action-planning approach that employs the hierarchical decomposition of network operations to address problems at different levels of abstraction [50]. A Task network is entered into a planning system together with the problem that has to be solved, or the task planning's objective. The task network consists set of tasks that define additional tasks. Planning operates by recursively decomposing non-primitive tasks into smaller sub-tasks until primitive tasks are reached that can be executed directly using planning operators.

We demonstrate how to convert HTN domain descriptions into PDDL so that traditional planners can use them (provided they comply with certain requirements). Our approach may considerably enhance the performance of a classical planner when arranging data in accordance with the hierarchical system's requirements into PDDL [51].

The planning domain and problem have been formalized by using planning domain and definition language (PDDL) which refers to all environmental data relevant to the robot's activities that are done for its surroundings. The PDDL has used the STRIPS model to represent the main components of the structure PDDL has. The concept a planning problem is separated into two components: The beginning state and goal condition are specified by the problem However in accordance with the hierarchical system's structure, the goal takes the shape of a collection of tasks sequential.

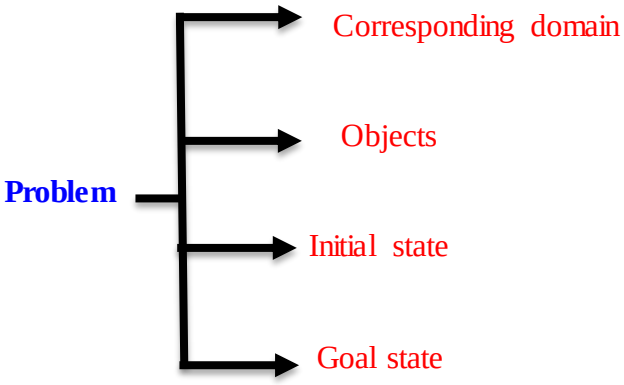


***Figure 3.1**: The problem representation in planning domain definition language.*

Every element of the world's first state that is pertinent to the problem must be explicitly stated in the initial state. Both static and dynamic data can be stored in the initial state. For instance, item f, a refrigerator, is an illustration of static information since it is assumed to remain constant throughout the action planning process. A further example of dynamic information is the condition in which I might propose that r is a robot that begins at location 1, as it is expected that it would move about while action planning is underway[52]. The G objective is to carry out a series of tasks in order to accomplish the ultimate objective.

For instance, the first task's objective is for the robot to conduct navigation, and the second task's objective is for the robot to fetch a particular object. Up until the robot reaches the desired destination, the work is sequential [8].

**Domain planning**: Planning consists of two steps: establishing the domain, which offers a broad representation of a broad representation of the pertinent components of the world, and identifying the issue, which is a particular instance within the domain that identifies the beginning point and the desired outcome. According to the problem definition:

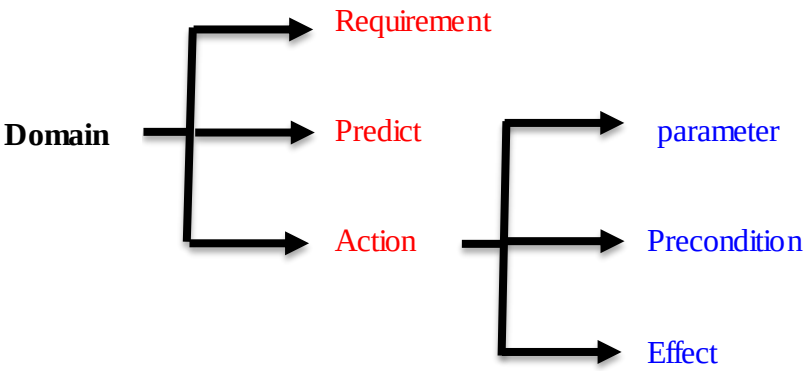


***Figure 3.2:*** *The domain representation in planning domain definition language.*

**Requirement: -** It is made up of every object and predicate-argument that should be displayed with their type in domains where the corresponding constraints are given as (: types).

**Predicates**: One part of the domain definition is the section that includes the list of state variables in the model. These variables are binary and represent fact either true / false.

**Actions**: A knowledge base is a collection of facts and predictions that may either be true or false.

According to PDDL, the parts of the problem and domain can be clarified, as in the simple example:

```
Procedure problem():
   Domain: domestic_env
   Objects: robot
   Init:
      in?(robot, base)
   Goal:
      in?(robot, Location1)

Procedure domain(domestic_env)

Requirement: navigation task

Prediction:
   in?(robot, base)
   not in?(robot, location1)

Precondition:
   not in?(robot, location1)
   in?(robot, base)

Action Search:
   Effect:
      found?(robot, Location1))
```

The problem contains the initial state, the desired goal, and the things to do with their domain, as shown in the example, depending on the name of the domain in the field of "domestic environment".

Requirements to do "task navigation" And the initial facts before the action effect that is the initial location of the robot before moving.

The name of the action that the robot performs according to the precondition and the desired effect as shown, in the beginning. One of the facts of the robot is in location 1(r.loc1 is true), after implementing the required effects to this action (is navigating to another location) the results will be different from the initial fact (r.loc1 is false).

## 3.3 Deterministic environment

To model the architecture for continuous online planning and acting, we employed deterministic environments. This framework consists of the software creating the robot's "plan" by identifying a series of simple actions that allow the robot to move from the beginning state to the end state after receiving descriptions of the starting state and final state. This sequence may correspond to the actions that the robot must do. Mainly, we use an autonomous robot that is employed largely for domestic chores is referred to as a domestic robot.

We assume that the robot performs the navigation and fetching tasks in the order of a deterministic environment.

The following is how the tasks are arranged in the hierarchical system as Figure (3.3):

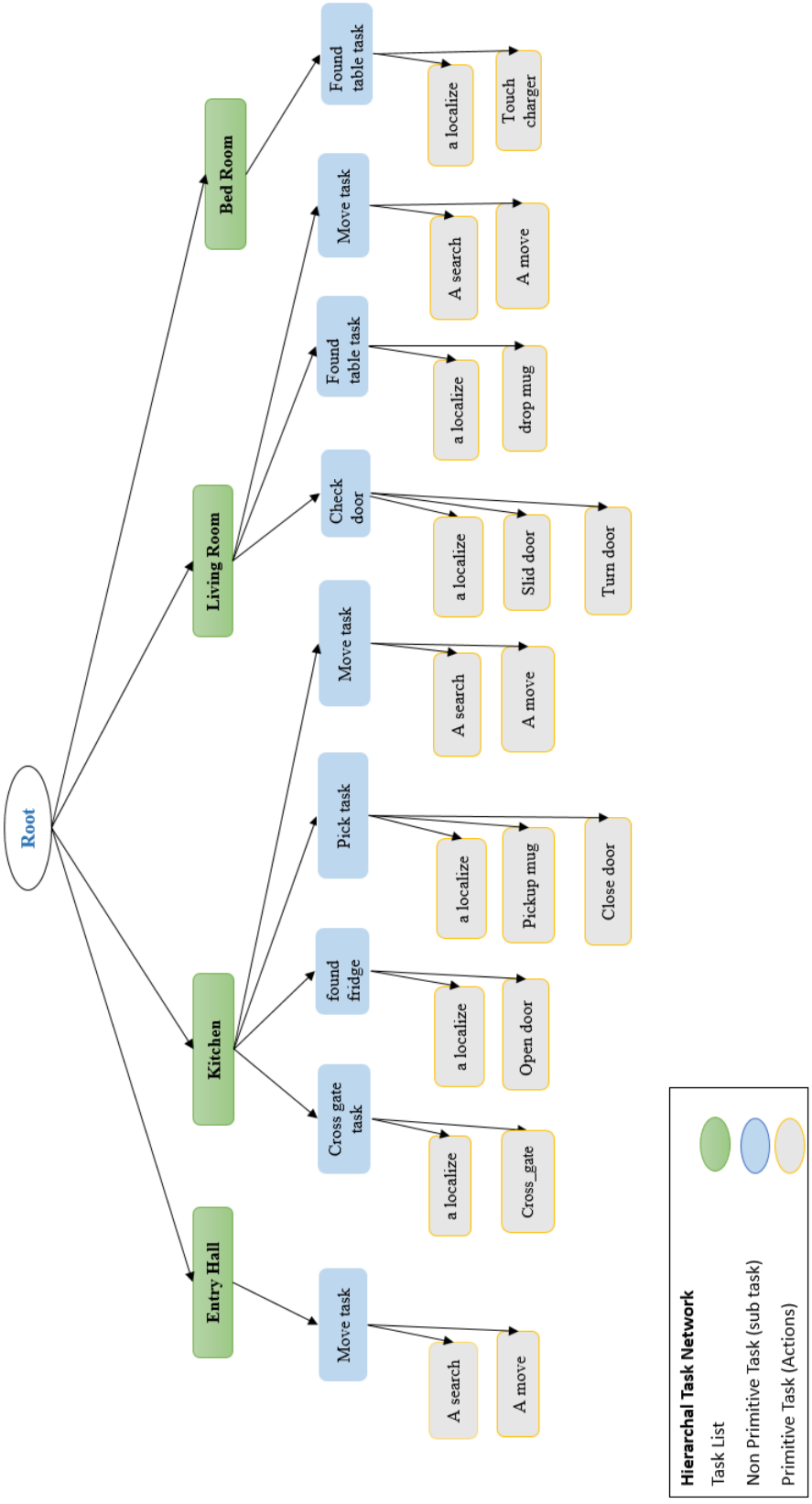


**Figure 3.3:** an illustration of a planning in domestic environment hierarchically

Making a list of activities and objectives for the system that will be carried out in the same order as anticipated is a part of the planning process. If the initial state of the tasks to be scheduled has a full order, when the initial tasks are fully ordered, we may use the W instead of a sequence of tasks using graph notation for the task network w = (U, C), where W = t1, t2,..., tk. The task in the first node of the U graph is represented by t1, the job in the second node by t2, and so on. Even if the jobs are fully ordered, we refine them using a tree generation/transit technique. Instead of being shown as a series of tasks, as initially suggested in the HTN planner formulation, they are shown as a task network in an acyclic digraph form. The task network-handling planner's variable definitions state that U is a collection of task nodes, which are handled as follows:

- Task (u) specifies the task ($t_1$,...$t_k$) associated with u.
- The node that has been refined is represented by refined (u).
- If task t were a simple task (primitive task), operator (u) would represent the operator that would be relevant to the task.
- Whether a node has been visited is indicated by the variable visited (u) (true, false).
- State (u) depicts the node's state at the time it was first visited.
- Methods (u) refers to the techniques employed for refining that are appropriate for task t.

**Planning:** To utilize the Hierarchical System Algorithm for planning, the planning problem is segmented into two components: the domain and the problem definition. The domain definition entails specifying the current variables (facts that can be true or false) and actions. On the other hand, the problem definition focuses on determining the initial, objective state (task) to formulate a plan. Uses a descriptive model to predict what the actions will do. Comparative to operational models, descriptive models are more abstract. Planning can be difficult because building extremely comprehensive prediction models is frequently too complex. Descriptive models are useful at higher levels of a deliberative hierarchy because they abstract away the details and focus on an action's fundamental impacts. Additionally, since these models need information that was not known at the time of planning, abstraction is required. Furthermore, reasoning with intricate models is extremely computationally challenging.

## 3.4 Refinement Acting Engine

Actors refer to the agents that are responsible for completing the tasks. Based on the dependencies defined in the HTN planner structure by the refinement, RAE assumes always methods prioritizing the refinement of tasks using the first method over the others. However, if this method fails, it can be replaced by the following methods, depending on the priorities and their order. It is possible that the other methods are superior to the first one.

As a solution, we proposed incorporating the Dijkstra algorithm along with the refinement engine in the operational model to evaluate the available methods and select the best suited for the given tasks. One distinguishing feature of this Dijkstra algorithm enable to identify the source node (i.e., the first method associated with the task) and compare it to other nodes to determine the best among them. For instance, when moving from one location to another, there may be multiple safe routes to reach a specific point to accomplish the task. However, by incorporating this algorithm, it selects the most efficient path among them. Figure (3.4) below demonstrates that there are multiple methods to access the same procedure and to choose the best of these.

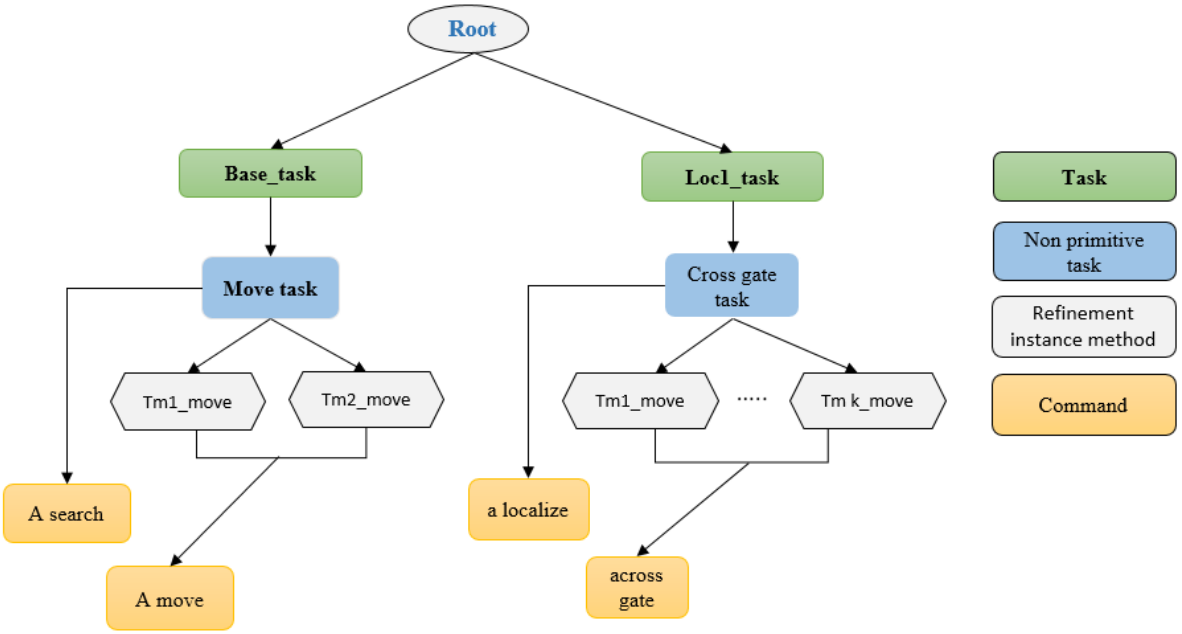


***Figure 3.4:*** *Refinement tree for task using differ methods.*

## 3.5 Integration planning and acting

In order to effectively utilize planning algorithms, actors must cooperate with planners. It is important to recognize that an actor's environment may not be fully captured by a planning domain. Therefore, even if a planning algorithm forecasts that a plan would be successful in achieving a target, there may be unforeseen problems that arise during execution, such as execution failures, unexpected events, incorrect information, or partial information. Actors must therefore have the ability to modify their plans in response to such problems.

To address these issues, an online planning algorithm called Run-lazy-Lookahead can be used to facilitate interaction between planning and acting. In this approach, (Σ, s, g) represents a planning problem, and the algorithm incorporates modifications to enable actors to modify their plans when necessary [53].

The run-lazy-lookahead algorithm involves calling Lookahead and then executing the plan as far as possible. Its Simulate function is responsible for testing whether the remainder of the plan can be executed

correctly. It can either simply compute γ(s, π) or perform a more detailed analysis.

However, this algorithm has some potential issues, such as waiting too long to re-plan if execution fails or missing opportunities to replace the plan with a better one[54].

We have made some modifications to the algorithm. As mentioned earlier, the run-lazy-Lookahead planner's signature is (∑; s; g), whereas the HTN planners' signatures are (S; W; O; M). In the case of hierarchical systems, ∑ represents the planning domain, and goal g is presented in a hierarchical task network as shown in algorithm 3.2, rather than in sequential tasks. Hierarchical action implementation has its advantages, as each task and its associated actions can be defined hierarchically. When errors occur in the implementation phase of a specific procedure, it is possible to track this procedure using the tracking algorithm and deal with the sub-tree associated with it.

***Algorithm 3.1: Refinement Algorithm.***

1. I ← simplified the observed state.
2. N ← refine (∑, I, N) //N is task network
3. If N =fail
4. Return fails
5. π ← accessibility to simple activities (actions) by DFS (N).
6. while π != empty and simulate (∑, I, N) != fail do
7. a ← pop-first-action in π
8. execute(a)
9. I ← simplified the observed state.
10. If π != empty then
11. N ← Un-refine (N, a)
12. N, a ← backtrack (N, a)
13. Else
14. Break

The Refinement algorithm outlines a refinement process for hierarchical task networks (HTNs) in the context of planning and task execution. Here's an explanation of the steps involved:

1: Simplify the observed state (I): This step involves reducing the complexity of the current observed state to a more manageable representation. It helps in focusing on the relevant information for planning and decision-making.

2: Refine (∑, I, N): The algorithm takes as input the refined HTN (N), the simplified observed state (I), and the initial state (∑). The refinement process aims to decompose the high-level tasks in the HTN into more detailed sub-tasks.

3: If N = fail: If the refinement of the HTN fails, indicating that the task cannot be further decomposed or refined, the algorithm returns a failure.

4: Return fails: If the refinement fails, the algorithm terminates and reports failure.

5: Accessibility to simple activities (actions) by DFS (N): This step determines the accessibility of simple activities or actions within the refined HTN. It employs a depth-first search (DFS) to traverse the HTN and identify the available actions that can be executed.

6: While Π is not empty and simulate (∑, I, N) is not fail do: This loop executes as long as there are remaining actions (π) and simulating the execution of the HTN (simulate (∑, I, N)) does not result in failure.

7: Pop the first action in Π (a): The algorithm selects and removes the first action from the list of remaining actions (π).

8: Execute(a): The selected action (a) is executed, potentially involving interactions with the environment or other agents.
9: Simplify the observed state (I): After executing the action, the observed state is simplified to reflect any changes resulting from the action execution.

10: If π is not empty then: If there are remaining actions (π) in the list, indicating more tasks to be executed.

11: Un-refine (N, a): This step involves undoing the refinement of the HTN (N) related to the executed action (a), reverting the task structure to a previous state.

12: Backtrack (N, a): The algorithm backtracks to a previous point in the HTN (N) where the executed action (a) was selected, potentially exploring alternative paths or options.

13: Else: If there are no remaining actions in π, indicating that all tasks have been completed.

14: Break: The algorithm breaks out of the loop, indicating successful completion of the refined HTN and task execution.

Overall, this algorithm iteratively refines and executes an HTN, handling failures, and making use of simplified observed states to guide the

planning and execution process. It employs a depth-first search for accessibility analysis and backtracking to explore different execution paths when necessary.

The re-planning process is limited only to the failed action and method associated with the task, and it is modified instead of re-planning the entire plan. When performing tasks in a sequential manner, as is the case in run-lazy-Lookaheads, all procedures will be sequential and re-planning the plan, when a problem occurs in one of the procedures, the tasks are tracked from actions last to the problem that occurs. This procedure will be costly from the computational point of view. However, depending on the task tree, the actions associated with this task can be determined faster, as shown in Figure (3.5).

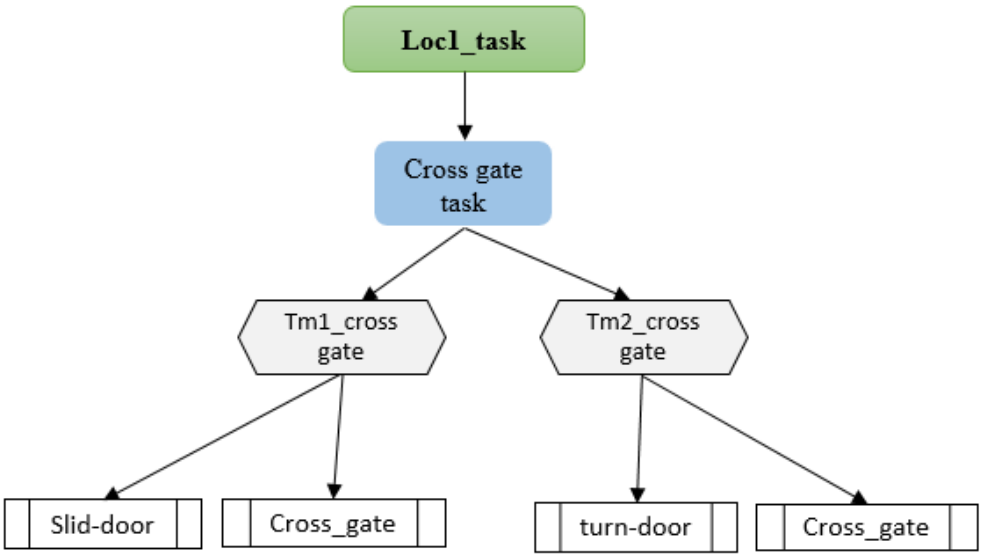


***Figure 3.5:*** *instance method for refinement.*

The refinement algorithm runs each plan as far as possible and only runs refinement again when a plan simulator shows that the plan will not work as intended.

An example in Figure (3.5) shows that opening a door depends on whether the door turns or slides. If the simulation of the plan reveals that it will not work correctly, the simulator should return a failure.

When the plan cannot be carried out as anticipated, the Backtrack algorithm of an HTN planner is used to retrace the steps as utilized in (3.3) [52].

***Algorithm 3.2: Backtracking Algorithm.***

```
1-   Backtracking (N, a)
2-   Mp ← parent of(a)        //method connect actions
3-   Np preorder DFs(Mp)
4-   for each v reverse (Np) do      //access root of sub-tree
5-       Refine (v) ← false
6-       If v is non primitive then
7-           Nv ← Successors(v)
8-           n ← n/N
9-           return n, v
10-  return root(n)
```

The procedure named "backtrack (N, a)" is utilized to update the task network in case the refinement of node u fails. Backtracking in reverse is thoroughly explained in the context of backtracking (preorder DFS) of the task network . Figure (3.6) represents our workflow hierarchy.

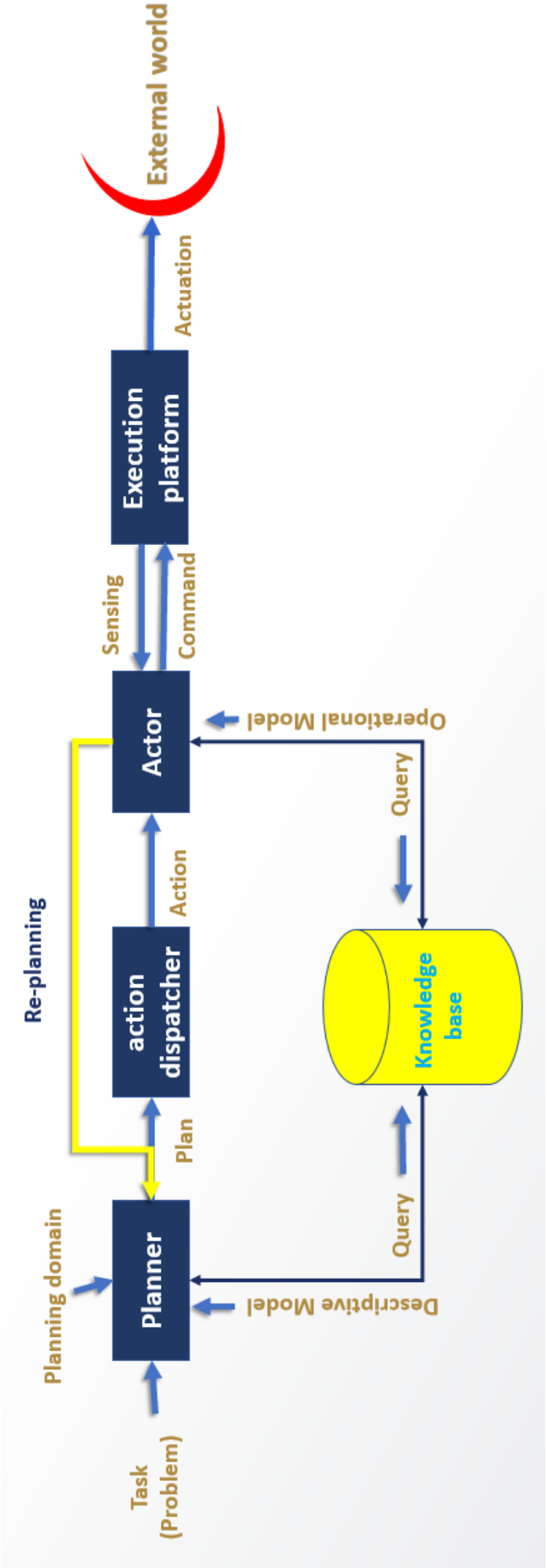


**Figure 3.6:** diagram represents the planning and action and the combination between them

## 3.6 Plan validation

The validation plan is a strategy that should specify what needs to be done. It is written at the beginning of the validation project before being created plan. In the checking stage, the actual state (calculated from the database in system) is compared to the expected state. This comparison aims to see if there is a contradiction between the two states that might point to an unexpected situation. The expected state is written and compared to what is performed by the planner from creating a plan. Expected state = [(a_search_for_loc1), (a_move_from_base_to_loc1), (a_localize_gate_in_loc1)] in this example the expected state is written based on the task tree in the system in Figure (3.3) when compared to the Plan case to see the validity and purpose of the plan: Find out if there is a contradiction between results and expectations.

Is there a lack of information related to the results of the plan or missing information? It is possible that if there is a deficiency, this information resulting from the validation can be used for re-planning [50].

# CHAPTER FOUR:

# RESULTS AND DISCUSSION

## 4.1 Overview

The preceding chapters have provided a comprehensive overview of the hierarchical system, delving into its intricacies and functionalities. In the present chapter, we shift our focus towards the evaluation of the system's performance and the outcomes obtained through simulations. In this chapter, we present the results of our experiments with a system. Sections 4.2 and 4.3 delve into the domain and tasks employed for our experimental setup, while Section 4.4 is dedicated to the thorough evaluation of the system.

## 4.2 Domestic Environment

This thesis deals with an autonomous robot intended for use in a domestic environment, which is a type of robot designed to perform various tasks within a household setting. These robots are capable of navigating through the home, identifying objects, and interacting with them. We have designed a domestic environment that includes a living room, kitchen, bedroom, and sitting room utilization in Figure (4.1).

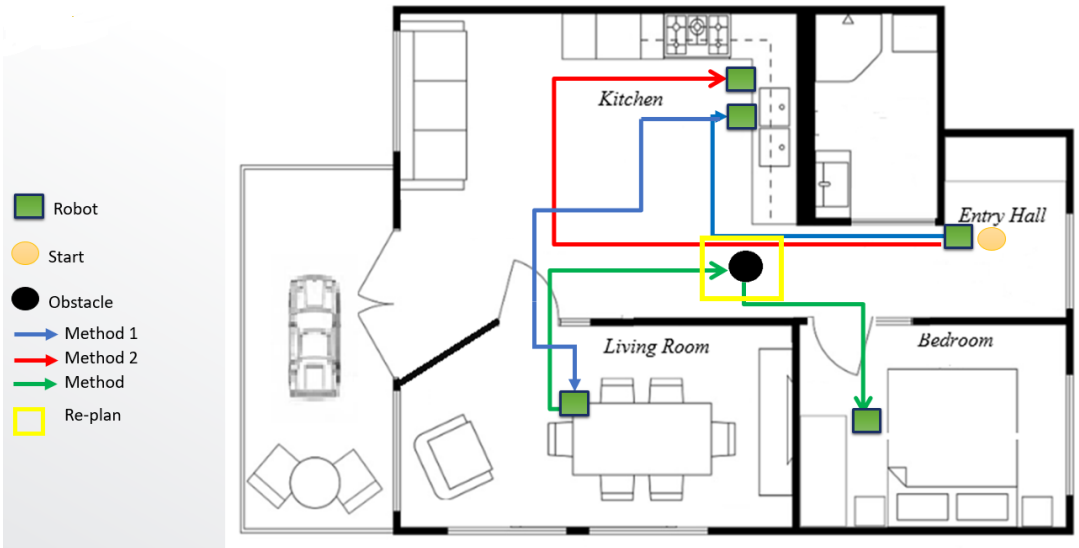


***Figure 4.1:*** *Domestic Environment.*

### 4.3Domain environment

The fundamental goal of the system is to demonstrate its autonomy by interacting with the first task and progressing to the subsequent tasks until all tasks are completed.

### 4.3.1Tasks Robot

- **The Navigation Task**

  Within the field of robotics, the process of moving a robot from one location to another within a predefined area is an essential task. To accomplish this, the robot must possess the ability to determine its own position, utilize a map or have awareness of its surroundings, and make informed decisions regarding the optimal path to the desired destination.

- Localization plays a crucial role in the capabilities of an autonomous robot as it forms the foundation for effective decision-making. It involves accurately determining the robot's position and orientation in relation to its surroundings, utilizing a combination of mapping and knowledge base. Without precise localization, the robot's ability to plan and execute tasks effectively would be compromised.
- The move action, which includes physically moving the robot from one place to another after taking a sequence of steps in the environment, is an essential part of the navigation process. For the robot to get where it's going, the move action must be completed successfully [55].

- A search task in robotics is the process of locating a certain object or position within an environment. The robot may take extra steps to finish the job after examining the surroundings to locate the target. A search task's principal goal is to locate a certain target within the supplied context. Despite the fact that they have different focuses, search and localization activities can both be accomplished by robots. While search tasks are concentrated on locating a specific target within the environment, localization tasks require identifying the robot's location within the environment[56].

➢ **Fetching Task**

- Robotics fetching jobs include a variety of actions in which the robot collects, transports, and delivers objects to a predetermined location. By utilizing its gripper to pick up the things, moving them to their intended place, and setting them down appropriately, the robot must be able to plan, carry out, and interact with its environment [57].
- In robotics, a pickup job comprises a robot using a gripper to grab or capture an item. In order to pick up an object, the robot may need to execute a number of smaller tasks, including object detection, localization, grasp planning, and grip execution.
- Additionally, touching can be a crucial component of the fetching process since it allows the robot to detect the existence or characteristics of an object [58].

The sequential completion of the following tasks is required by the automated system:

**1-** The robot is assigned a set of tasks and starts from the entry hall, with the first one being to navigate to the kitchen. This task is divided

into sub-tasks, starting with searching for the kitchen using the "search" instruction. Once the location of the kitchen is known, the robot uses the "move" instruction to reach it,

**2-** Utilizing "localization" techniques to ascertain the robot's position, the Dijkstra algorithm was employed to calculate the optimal path, specifically the shortest distance route from the entry hall to kitchen. If the distance to the kitchen is 3 meters and the distance to an alternative location is 6 meters, the optimal path" to cross gate "between the two is chosen. The other part of the task is to locate the refrigerator, which is done using "localization" after the robot arrives in the kitchen, A sequence of procedures is then performed, beginning with the "open" instruction to open the refrigerator door, followed by "localization" to identify the location of the mug, and finally "fetch process" through "pick up" actions to take the mug using the robot arm. This completes the first task.

**3-** To navigate to the second location, the robot executes the "*search*" command to determine the location. It then uses the existing knowledge base and house map to navigate to the living room, which it does by executing a series of sub-tasks. Once the robot arrives at the living room by "*move*" instruction, it inspects the door and determines the type of lock it "*slid*" or "*turn*". It then searches for the table food by executing the "*localization*" procedure and places the mug on the table by executing the "*drop*" command, thus completing its second task. The robot uses the Dijkstra algorithm to find the optimal path to access the living room.

**4-** Following the completion of the second task, the robot proceeds to the next task by searching for the bedroom and moving towards it using the "*search*" and "*move*" commands based on a Dijkstra algorithm that determines the optimal path. Upon reaching the bedroom, the robot

This study employed a planning domain that involved 20 primitive task operators and 23 task refining techniques for dealing with 16 non-primitive, which allowed us to obtain an integrated plan in our system Figure (4.2) is used to display the results.

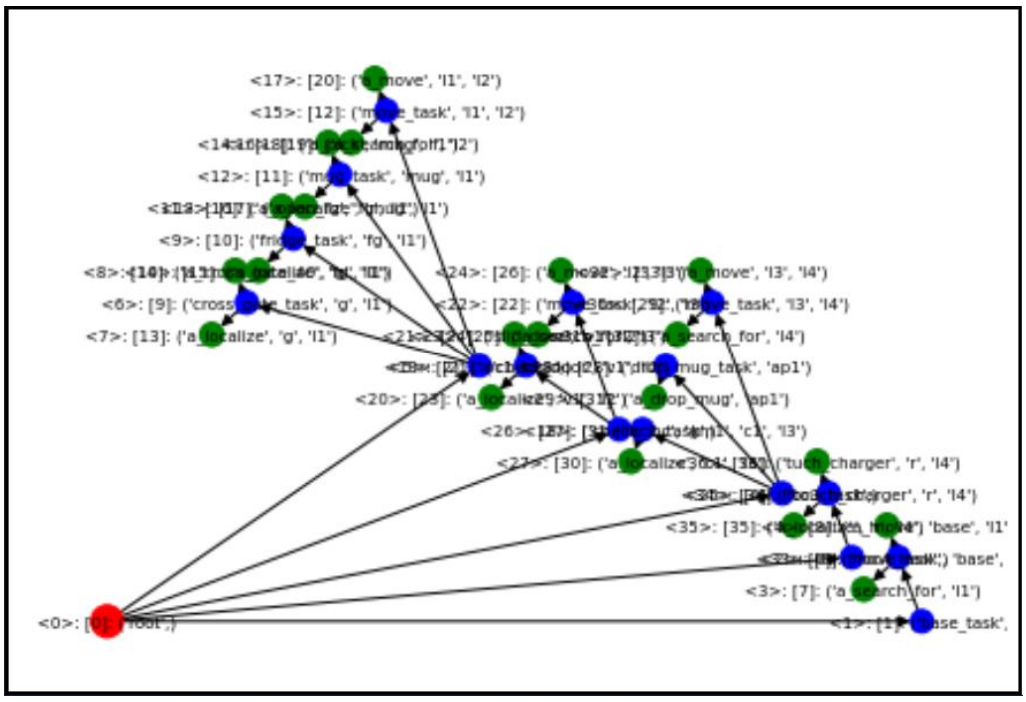


***Figure 4.2:*** *system implementation result.*

## 4.4 Feature HTN Planning

We compared the system with the previous system in order to assess its limitations and advantages:

**GTPyhop** is a Python-based automated planning system that utilizes hierarchical planning techniques to generate action plans for tasks and goals[59]. It extends Pyhop to encompass goal planning in addition to task planning, incorporating aspects of HTN planning from Pyhop and SHOP[6].

The GTPyhop software combines planning and acting algorithms from the Run-Lazy-Lookahead actor, but its use of recursion poses two limitations for successful re-planning:

- Firstly, the use of recursion prevents the code from being re-entered, meaning the only recourse for re-planning in the event of an action failure is to reconstruct the plan by re-running the system, which may lead to inaccurate results.
- Secondly, GTPyhop only returns the plan and not the refinement tree, which is necessary for accurate re-planning.

Unlike HTN planners that use schema-specific languages, this system does not have prior knowledge of the preconditions and subtasks of its methods. It instead invokes the method directly via Python code[59].

Our experiments show that the new algorithm we developed can improve upon the common planning and acting approach by modifying the integration of planning and acting with the Run-Lazy-Lookahead algorithm in two ways:

- The algorithm uses an iterative transfer tree procedure for task refinement, with tree traversal algorithms handling refinement and backtracking. This provides greater control over how the algorithm refines tasks. The algorithm can process a partial task tree and provide a complete solution task network, enabling hierarchical knowledge inclusion in the re-planning process.

## 4.4 Evaluation system

Numerous tests are run in a controlled setting as part of the thesis review to assess the effectiveness of the system is one of the assessment metrics employed. Precision, recall, and F1-score are computed using these numbers, giving information about the system's performance.

**Precision** a performance indicator called precision is used to assess how accurately a model can identify instances of success that is determined[60]. Precision can be expressed mathematically as:

$$\textbf{Precision} = TP / (TP + FP)$$

**Recall,** is a parameter that quantifies the completeness of a model's positive predictions[60][61]. The mathematical formula for the recall is:

$$\textbf{Recall} = TP / (TP + FN)$$

**F1-score** is a statistic that, by combining accuracy and recall, the formula for calculating the F1-score is:

$$\textbf{F1-score} = 2^* (Precision^* Recall) / (Precision+ Recall)$$

calculate precision, recall, and the F1-score, we utilize The values representing true, false positive, and true, false negative as They can be briefly described as:

**(TP)** The number of elements in the intersection between the set of occurrences where a certain action appears in the robot output and the set of instances where the same action appears in the expected plan is measured by the TP metric.

To understand this metric, let's break down its components:

1. Set of occurrences where a certain action appears in the robot's output: This set consists of the instances or occurrences where a specific action is detected or identified by the robot or the system being evaluated. It represents the actions that the robot outputs or recognizes during its operation.

2. Set of instances where the same action appears in the expected plan: This set represents the instances or occurrences where the same action is expected or intended to occur according to the predefined or expected plan. It is a reference set that defines the correct actions that should be performed. (Depending on the initial environment)

The TP metric then measures the number of elements (occurrences) that are common or shared between these two sets. In other words, it counts how many times the robot's detected actions align with the expected actions.

The TP metric is useful for assessing the accuracy or correctness of action recognition or planning algorithms.

By comparing the TP metric with other evaluation metrics like False Positive (FP) and False Negative (FN), we can get a more understanding of the performance of the system in terms of action recognition and planning.

**TP** = | {instances where robot output contains an action} ∩ {instances where action is in expected plan}|

**(FP)** is the cardinality (number of actions) of the intersection between the set of times when an action is present in the robot output and the set of times when it is absent from the expected plan.

**FP** = | {instances where robot output contains an action} ∩ {instances where action is not in expected plan}|

**(TN)** represents the number of actions (or cardinality) that are absent in both the robot output and the expected plan.

It measures the intersection between the set of instances where a specific action is not present in the robot output and the set of instances where the same action is not present in the expected plan.

**TN** = | {instances where robot output does not contain the action} ∩ {instances where the action is not in the expected plan}|

**(FN)** the value of corresponds to the number of actions (or cardinality) that are absent in the robot output but present in the expected plan. It represents the intersection between the set of instances where a specific action is not present in the robot output and the set of instances where the same action is included in the expected plan.

**FN** = | {instances where robot output does not contain the action} ∩ {instances where action is in expected plan}|

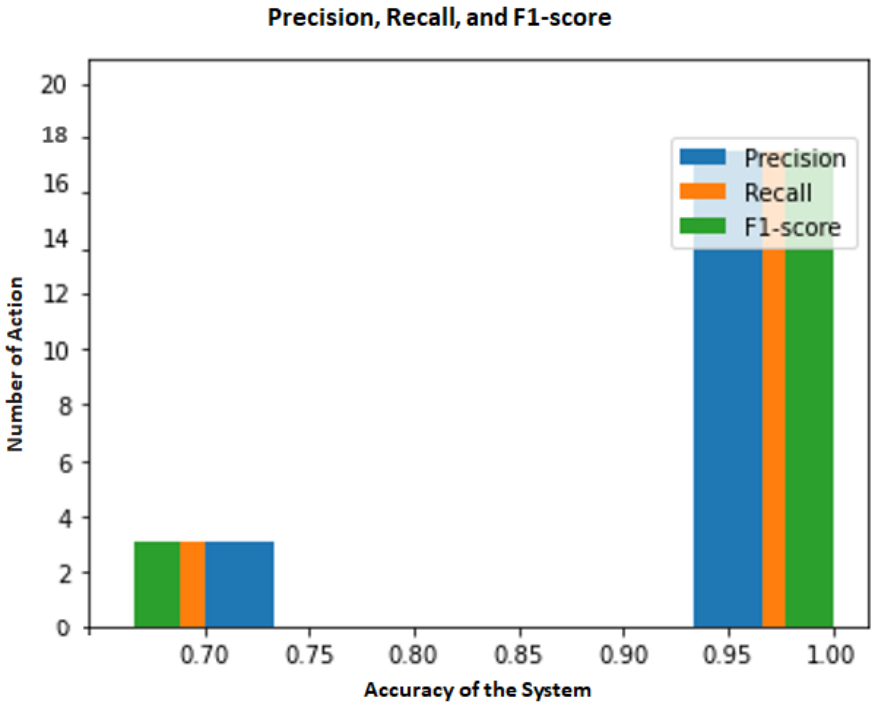


***Figure 4.3:*** *Displays the result of calculating metrics to evaluate Precision, Recall, and F1-score.*

***Table 4.1:*** *presents a displaying the number of actions for TP, TN, FP, FN, Precision, Recall, and F1-score .*

| Num_action | TP | TN | FP | FN | Precision | Recall | F1-score |
|---|---|---|---|---|---|---|---|
| **1** | 2 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **2** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **3** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **4** | 2 | 0 | 1 | 1 | 0.66 | 0.66 | 0.66 |
| **5** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **6** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **7** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **8** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **9** | 2 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **10** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **11** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **12** | 2 | 0 | 1 | 1 | 0.66 | 0.66 | 0.66 |
| **13** | 2 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **14** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **15** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **16** | 2 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **17** | 2 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **18** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **19** | 3 | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| **20** | 2 | 0 | 1 | 1 | 0.66 | 0.66 | 0.66 |

The table indicates that the system achieved a precision value of approximately 94.9%, a recall value of approximately 94.9%, and an F1-score of approximately 94.79%.

# CHAPTER FIVE:

# CONCLUSIONS AND FUTURE WORK

## 5.1 Overview

This chapter provides a conclusion on the research phases and objectives, as well as future work for other researchers

## 5.2 Conclusion

In conclusion, this section highlights the achievements and findings of the presented thesis. The thesis focused on developing algorithms for HTN planning, acting, and integrating planning and acting processes. The key points discussed are as follows:

1. HTN planner and descriptive action models: The thesis proposed an HTN planner that works in conjunction with descriptive action models to calculate the next state during the planning process. This approach demonstrated effective utilization and provided a foundation for further development.

2. Operational action models: The introduction of operational action models, represented formally, showed their successful integration with the Refinement Acting Engine (RAE) and the Dijkstra algorithm. This integration helped in determining optimal task approaches, leading to improved performance.

3. Integration of planning and acting: The planning and acting algorithm was designed hierarchically, utilizing a transit tree and backtracking mechanisms. This design allowed for re-planning in case of execution errors, ensuring system reliability and avoiding suboptimal behavior.

4. Experimental evaluations: The developed algorithms were evaluated using the Python language (Spyder). The result showed that the algorithm outperformed conventional planning and acting approaches by providing optimal paths within the system.

5. Precision, recall, and F1-score: The system demonstrated accurate identification of positive instances while maintaining completeness. The precision value of 94.9%, recall value of 94.9%, and F1-score of 94.79% highlighted the system's ability to achieve high accuracy and completeness in its performance.

Overall, the presented thesis made significant contributions to HTN planning and acting, showcasing the effectiveness of the proposed algorithms in terms of system reliability, optimal performance, and accuracy in task identification.

## 5.3 Limitations

It is essential to be aware of the constraints imposed by the existing planning and action system in addition to the prospective topics for future investigation. These restrictions offer chances for more study and development. The system has the following limitation:

Scalability: When faced with difficult tasks or large-scale settings, the system's performance may suffer. The planning and acting algorithms may require extra improvement as the environment's size and complexity rise in order to ensure efficient and successful functioning.

Adaptability to Unknown environment: When operating in Unknown or Unstructured settings, the system's performance may be jeopardized. It mainly relies on pre-existing maps or information bases, This constraint may be overcome by creating methods that allow the system to dynamically adapt and learn from its surroundings in real-time.

The planning and acting system can overcome present difficulties and grow into a more reliable, adaptive, and intelligent solution for diverse applications in the field of robotics by addressing these constraints via more research and development.

## 5.4 Future work

Two prospective topics of investigation were found to help the research in the future.

The D* algorithm may first be integrated, making use of its effective path-planning skills to dynamically adapt to the environment as it changes in real-time. The system's capacity to choose the best routes and react to changing circumstances would be improved by this improvement.

Second, by introducing learning strategies into the system, the robot would be better equipped to learn new information and make better decisions. The system may learn from previous events, see trends, and make better judgments based on accumulated information by utilizing machine learning techniques.

It is envisaged that the planning and acting system will continue to develop by embracing these future developments, offering ever more effective, adaptive, and intelligent robotic systems for various applications.

# REFERENCES


[1] A. Al-Moadhen, R. Qiu, M. Packianather, Z. Ji, and R. Setchi, "Integrating robot task planner with common-sense knowledge base to improve the efficiency of planning," *Procedia Comput. Sci.*, vol. 22, pp. 211–220, 2013, doi: 10.1016/j.procs.2013.09.097.

[2] Ghallab, M., Nau, D., & Traverso, P. (2016). Automated Planning and Acting. book. Cambridge, UK: Cambridge University Press.

[3] Ghallab, M., Nau, D., & Traverso, P. (2016). Automated Planning: Theory & Practice. book. Cambridge University Press.

[4] Currie, K., & Tate, A. (1991). Oplan - The Open Planning Architecture. Artificial Intelligence, 52, 49-86. doi: 0004-3702/91/$03.50

[5] Tate, A., Drabble, B., & Kirby, R. (year). O-Plan2: An Open Architecture for Command, Planning, and Control. Artificial Intelligence Applications Institute, University of Edinburgh, South Bridge, Edinburgh EH HN, United Kingdom. 1994.

[6] Nau, D., Cao, Y., Lotem, A., & Muñoz-Avila, H. (1999). SHOP: Simple Hierarchical Ordered Planner. In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99) (pp. 968-973).

[7] Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN Planning System. Journal of Artificial Intelligence Research, 20, 379-404.

[8] Georgievski, I., & Aiello, M. (2015). HTN planning: Overview, comparison, and beyond. Artificial Intelligence, 222, 124-156. doi: 10.1016/j.artint.2015.02.002.

[9] Thompson, F., & Galeazzi, R. (2020). Robust mission planning for Autonomous Marine Vehicle fleets. Robotics and Autonomous Systems, 124, 103404. doi: 10.1016/j.robot.2019.103404.

[10] Michele Colledanchise. (2017). Behavior Trees in Robotics. Doctoral Thesis. Stockholm, Sweden: KTH Royal Institute of Technology. TRITA-CSC-A-2017:07. ISSN-1653-5723. ISRN-KTH/CSC/A-17/07-SE. ISBN 978-91-7729-283-8. Robotics Perception and Learning. School of Computer Science and Communication. SE-100 44 Stockholm, Sweden. Copyright © 2017 by Michele Colledanchise except where otherwise stated. Tryck:

Universitetsservice US-AB 2017.

[11] Ingrand, F., & Ghallab, M. (2014). Deliberation for autonomous robots: A survey. Artificial Intelligence, 1(143), 1-35. doi:10.1016/j.artint.2014.11.003

[12] Alia Karim Abdul Hassan . Robotics And Planning. 2009 Journal Nasa . https://www.nasa.gov/audience/forstudents/k-4/stories/nasa-knows/what_is_robotics_k4.html

[13] D. E. Wilkins, "Domain-independent planning Representation and plan generation," *Artif. Intell.*, vol. 22, no. 3, pp. 269–301, 1984, doi: 10.1016/0004-3702(84)90053-5.

[14] M. Fox, G. Gerevini, D. Long, and I. Serina, "Plan stability: Replanning versus plan repair," *ICAPS 2006 - Proceedings, Sixt. Int. Conf. Autom. Plan. Sched.*, vol. 2006, pp. 212–221, 2006.

[15] D. Höller, P. Bercher, G. Behnke, and S. Biundo, "On guiding search in HTN planning with classical planning heuristics," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2019-Augus, pp. 6171–6175, 2019, doi: 10.24963/ijcai.2019/857.

[16] D. S. Nau, M. Ghallab, and P. Traverso, "Blended planning and acting: Preliminary approach, research challenges," *Proc. Natl. Conf. Artif. Intell.*, vol. 6, pp. 4047–4051, 2015.

[17] M. Ghallab, D. Nau, and P. Traverso, "The actor's view of automated planning and acting: A position paper," *Artif. Intell.*, vol. 208, no. 1, pp. 1–17, , doi: 10.1016/j.artint.2013.11.002.

[18] C. da Costa Pereira and A. G. B. Tettamanzi, "Reasoning about actions with imprecise and incomplete state descriptions," *Fuzzy Sets Syst.*, vol. 160, no. 10, pp. 1383–1401, 2009, doi: 10.1016/j.fss.2008.11.019.

[19] A. A. Al Moadhen, A. M. Abdulhussein, and H. G. Kamil, "Planning and acting framework under robot operating system," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 433, no. 1, 2018, doi: 10.1088/1757-899X/433/1/012090.

[20] M. Plan, "HAMILTON GARDENS OPERATIVE," 2014.

[21] P.Haslum, N.Lipovetzky, , D.Magazzeni, and C.Muise. Title: An Introduction to the Planning Domain Definition Language. 2019 DOI 10.2200/S00900ED2V01Y201902AIM042

[22] R. E. Fikes and N. J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving" Journal: Artificial Intelligence. 1971.Publisher: North-Holland Publishing Company Copyright: © 1971 by North-Holland Publishing Company

[23] M. Ghallab, A. Howe, C. Knoblock, I.D. McDermott A. Ram, M. Veloso D. Weld, and D. Wilkins. PDDL | The Planning Domain Defnition Language. journal: Tech Report CVC TR-98-003/DCS TR-1165, 1998.

[24] N. Lipovetzky. " Structure and Inference In Classical Planning" 2014 ISBN 978-1-31-246621-0. Published by AI Access.

[25] S. Patra, J. Mason, M. Ghallab, D. Nau, and P. Traverso, "Deliberative Acting, Online Planning and Learning with Hierarchical Operational Models," Oct. 2020, doi: 10.1016/j.artint.2021.103523.

[26] Alford, R., Shivashankar, V., Kuter, U., & Nau, D. (2012). HTN Problem Spaces: Structure, Algorithms, Termination. In Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12), 1475-1481.

[27] R. Alford, V. Shivashankar, M. Roberts, J. Frank, and D. W. 2016 Aha, "Hierarchical planning: Relating task and goal decomposition with task sharing," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2016-Janua, pp. 3022–3028, 2016.

[28] B. Hayes and B. Scassellati, "Autonomously constructing hierarchical task networks for planning and human-robot collaboration," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2016-June, pp. 5469–5476, 2016, doi: 10.1109/ICRA.2016.7487760.

[29] S. Stock, M. Mansouri, F. Pecora, and J. Hertzberg, "Online task merging with a hierarchical hybrid task planner for mobile service robots," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2015-Decem, pp. 6459–6464, 2015, doi: 10.1109/IROS.2015.7354300.

[30] S. Patra, J. Mason, M. Ghallab, D. Nau, and P. Traverso, "Deliberative acting, planning and learning with hierarchical operational models," *Artif. Intell.*, vol. 299, pp. 1–68, 2021, doi: 10.1016/j.artint.2021.103523.

[31] M. A. Javaid, "Understanding Dijkstra Algorithm," *SSRN Electron. J.*, no. January 2013, 2013, doi: 10.2139/ssrn.2340905.

[32] L. S. Liu *et al.*, "Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach," *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/8881684.

[33] J. Orkin, "Three states and a plan: the AI of FEAR," *Game Dev. Conf.*, vol. 2006, doi=10.1.1.92.8551&rep=rep1&type=pdf

[34] R. P. Goldman and U. Kuter, "Hierarchical Task Network Planning in Common Lisp: the case of SHOP3," *(*2019), doi:10.5281.

[35] V. Shivashankar, U. Kuter, D. Nau, and R. Alford, "A Hierarchical Goal-Based Formalism and Algorithm for Single-Agent Planning," Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

[36] S. Patra, J. Mason, A. Kumar, M. Ghallab, P. Traverso, and D. Nau, "Integrating Acting, Planning, and Learning in Hierarchical Operational Models," 2020. University of Maryland, College Park, MD 20742, USA 2LAAS-CNRS, 31077, Toulouse,.

[37] S. Patra, M. Ghallab, D. Nau, and P. Traverso, "APE: An Acting and Planning Engine," 2018. HAL Id: hal-01959098

[38] F. Fklix Ingrand, R. Chatila, R. ABami, and dQic Robert, "PRS: A High Level Supervision and1 Control Language for Autonomous Mobile Robots," , International Conference on Robotics and Automation Minneapolis. Minnesota - April 1996

[39] S. Patra, M. Ghallab, D. Nau, and P. Traverso, "Using Operational Models to Integrate Acting and Planning," 2018. ICAPS Workshop on Integrated Planning, Acting and Execution, Jun 2018, Delft, Netherlands. hal-01959118

[40] D. S. Nau, M. Ghallab, and P. Traverso, "Blended Planning and Acting: Preliminary Approach, Research Challenges." (2015) [Online]. Available: www.aaai.org

[41] V. Alcázar, M. Veloso, and D. Borrajo, "Adapting a rapidly-exploring random tree for automated planning," *Proc. 4th Annu. Symp. Comb. Search, SoCS 2011*, pp. 2–9, 2011.

[42] D. J. Musliner, M. J. S. Pelican, K. D. Krebsbach, R. P. Goldman, and E. H. Durfee, "The evolution of CIRCA, a theory-based AI architecture with real-time performance guarantees," *AAAI Spring Symp. - Tech. Rep.*, vol. SS-08-02, pp. 43–48, 2008.

[43] A. Menif, É. Jacopin, and T. Cazenave, "SHPE: HTN planning for video games," *Commun. Comput. Inf. Sci.*, vol. 504, no. 2011, pp. 119–132, 2014, doi: 10.1007/978-3-319-14923-3.

[44] A. Coles, A. Coles, M. Fox, and D. Long, "COLIN: Planning with continuous linear numeric change," *J. Artif. Intell. Res.*, vol. 44, pp. 1–96, 2012, doi: 10.1613/jair.3608.

[45] S. Fratini, A. Cesta, R. De Benedictis, A. Orlandini, and R. Rasconi, "APSI-

based deliberation in Goal Oriented Autonomous Controllers," *11th Symp. Adv. Sp. Technol. Robot. Autom.*, 2011.

[46] R. Lallement *et al.*, "HATP : Hierarchical Agent-based Task Planner To cite this version : HAL Id : hal-01944178 HATP : Hierarchical Agent-based Task Planner," 2018.

[47] J. Wolfe, B. Marthi, and S. Russell, "Combined task and motion planning for mobile manipulation," *ICAPS 2010 - Proc. 20th Int. Conf. Autom. Plan. Sched.*, pp. 254–257, 2010, doi: 10.1609/icaps.v20i1.13436.

[48] W. Yuan, H. Munoz-Avila, V. Gogineni, S. Kondrakunta, M. Cox, and L. He, "Task Modifiers for HTN Planning and Acting," *Proc. Ninth Annu. Conf. Adv. Cogn. Syst.*, no. 2016, pp. 1–13, 2021.

[49] X. Neufeld, S. Mostaghim, D. L. Sancho-Pradel, and S. Brand, "Building a planner: A survey of planning systems used in commercial video games," *IEEE Trans. Games*, vol. 11, no. 2, pp. 91–108, 2019, doi: 10.1109/TG.2017.2782846.

[50] K. Erol, J. Hendler, and D. S. Nau, "HTN planning: Complexity and expressivity," *Proc. Natl. Conf. Artif. Intell.*, vol. 2, pp. 1123–1128, 1994.

[51] R. Alford, U. Kuter, and D. Nau, " Translating HTNs to PDDL : A Small Amount of Domain Knowledge Can Go a Long Way "2004. University of Maryland, College Park, Maryland 20742, USA.

[52] S. J. Russell and P. Norvig, *Solving problems by searching*.Book. 2016.

[53] M. Ghallab, D. Nau, P. Traverso, I. LAAS-CNRS, University of Toulouse, France, University of Maryland, USA, FBK ICT IRST, Trento, and Manuscript, "Automated Planning and Acting," *Autom. Plan. Act.*, pp. 1–354, 2016, doi: 10.1017/CBO9781139583923.

[54] M. Ghallab, D. Nau, and P. Traverso, "Deliberation in Planning and Acting Automated Planning and Acting Deliberation in Planning and Acting Part 2: Refinement Models." June 22, 2017.

[55] E. T. H. Library, "A robot using a modular navigation system," 2002.

[56] R. Siegwart and I. Nourbakhsh, "Mobile Robot Localization," *Robot.*, pp. 159–230, 2011.

[57] A. Bouguerra, "Robust Execution of Robot Task-Plans A Knowledge-based Approach." Publisher: Örebro University 2008 issn 1650-8580
isbn 978-91-7668-610-2

[58] A. Bhalerao, K. Chopade, P. Doifode, and J. Gaikwad, "Pick and Place Robotic

ARM using PLC," Publisher by: *(IJERT) ISSN: 2278-0181*
*Vol. 8 Issue 08, August-2019*

[59] D. Nau, Y. Bansod, S. Patra, M. Roberts, and R. Li, "GTPyhop: A Hierarchical Goal+Task Planner Implemented in Python."(2021) Published ICSPS

[60] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," no. May, 2020, doi: 10.9735/2229-3981.

[61] H. Dalianis, "Evaluation Metrics and Evaluation," *Clin. Text Min.*, no. 1967, pp. 45–53, 2018, doi: 10.1007/978-3-319-78503-5_6.

# الخُلاصَة

لطالما استخدم البشر الأطر الهرمية لتنظيم أفكارهم حول المشكلات المعقدة. .
يتضمن أحد تطبيقات الذكاء الاصطناعي إنشاء عوامل ذكية تقسم المشكلات الصعبة إلى طبقات من التجريد ، وتبسيط عملية حل المشكلات. يتيح لنا هيكل شبكة المهام الهرمية (HTN) أداء مثل هذه المهام بفعالية.

في سياق الروبوتات المستقلة التي تعمل في بيئات مغلقة وحتمية ، مثل البيئات المحلية ، يعد تخطيط المهام أمرًا بالغ الأهمية لتحقيق مستوى عالٍ من الدقة. لمعالجة هذا الأمر ، استخدمنا مخطط HTN ونماذج عمل وصفية لتحديد الحالة التالية في نظام انتقال الحالة أثناء عملية التخطيط. لقد قدمنا تمثيلًا رسميًا لنماذج الإجراءات التشغيلية وشرحنا كيف يمكن استخدامها بشكل فعال مع محرك عمل الصقل (RAE) وخوارزمية Dijkstra لتحديد أساليب المهام المثلى.

تم تصميم خوارزمية التخطيط والتنفيذ الخاصة بنا للعمل بشكل هرمي ، باستخدام شجرة عبور والتراجع لإعادة التخطيط في حالة حدوث أخطاء في التنفيذ. تم إيلاء اهتمام دقيق لتطبيق الأسلوب لتجنب فشل النظام أو الأداء دون المستوى الأمثل.

تتضمن المنهجية المستخدمة في النص معالجة تحديات تخطيط المهام التي تواجهها الروبوتات المستقلة التي تعمل في بيئات مغلقة وحتمية ، على وجه التحديد البيئات المحلية. لتحقيق مستوى عالٍ من الدقة في تنفيذ المهام ، تم استخدام مخطط شبكة المهام الهرمية (HTN) ونماذج الإجراءات الوصفية مع تمثيل لنماذج الإجراءات التشغيلية ، ومحرك الصقل الفعال (RAE) ، وخوارزمية Dijkstra لتحديد نهج المهام المثلى ، يساهم الهيكل الهرمي وآليات معالجة الأخطاء في تحقيق إنجاز دقيق وفعال للمهام مع الحفاظ على موثوقية النظام اعتمادًا على تخطيط التكامل وخوارزمية التمثيل التي تم تصميمها للعمل بشكل هرمي ، باستخدام شجرة العبور وخوارزمية التراجع.

من خلال التجارب التي أجريت باستخدام لغة Python (Spyder) ، أوضحنا أن الخوارزمية التي طورناها تتفوق في الأداء على نهج التخطيط والتنفيذ الشائع الاستخدام ، مما يوفر المسار الأمثل داخل النظام. أظهرت النتائج دقة تقارب 94.9%، وقيمة استدعاء تقارب 94.9%، ودرجة F1 تقارب94.79%. تثبت هذه النتائج فعالية نهجنا في تحسين عملية التخطيط والتنفيذ.

جامعة كربلاء
كلية علوم الحاسوب وتكنولوجيا المعلومات
قسم علوم الحاسوب

عنوان الرسالة

# تطوير نموذج تمثيل هرمي موحد للتخطيط والعمل في الروبوتات المستقلة

رسالة ماجستير
مقدمة الى مجلس كلية علوم الحاسوب وتكنولوجيا المعلومات / جامعة كربلاء وهي جزء من متطلبات نيل درجة الماجستير في علوم الحاسوب

كتبت بواسطة

زينب عباس فاضل

بإشـــراف

أ.م. د. أحمد عبد الهادي احمد

1444 هـ

2023 م