COMPLEX INTERACTIONS BETWEEN MULTIPLE GOAL OPERATIONS IN AGENT GOAL MANAGEMENT

A Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

By

SRAVYA KONDRAKUNTA
M.S., Wright State University, 2017
B.Tech., Koneru Lakshmaiah University, 2015

2021
Wright State University

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

January 4, 2022

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Sravya Kondrakunta ENTITLED Complex Interactions between Multiple Goal Operations in Agent Goal Management BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Dr. Michael T. Cox, Ph.D.
Dissertation Co-Director

Dr. Mateen M. Rizki, Ph.D.
Dissertation Co-Director

Dr. Yong Pei, Ph.D.
Computer Science & Engineering PhD Program Director

Barry Milligan, Ph.D.
Vice Provost for Academic Affairs
Dean of the Graduate School

Committee on
Final Examination

Dr. Michael T. Cox

Dr. Mateen M. Rizki

Dr. Michelle A. Cheatham

Dr. Matthew M. Mollineaux

Dr. Michael L. Raymer

ABSTRACT

Kondrakunta, Sravya. P.h.D., Department of Computer Science & Engineering, Wright State University, 2021. *Complex Interactions between Multiple Goal Operations in Agent Goal Management.*

A significant issue in cognitive systems research is to make an agent formulate and manage its own goals. Some cognitive scientists have implemented several goal operations to support this issue, but no one has implemented more than a couple of goal operations within a single agent. One of the reasons for this limitation is the lack of knowledge about how various goals operations interact with one another. This thesis addresses this knowledge gap by implementing multiple-goal operations, including goal formulation, goal change, goal selection, and designing an algorithm to manage any positive or negative interaction between them. These are integrated with a cognitive architecture called MIDCA and applied in five different test domains. We will compare and contrast the architecture's performance with intelligent interaction management with a randomized linearization of goal operations.

Contents

List of Figures

List of Tables

Acknowledgments

I am genuinely honored to take this opportunity to express my profound gratitude towards my advisor, **Dr. Michael T. Cox**. I find his enthusiasm toward seeking new research problems very contagious. In addition, he constantly inspires me through his thoughts, work ethic, and professionalism. I respect him for his guidance throughout the pursuit of my Ph.D. Furthermore, he always provides enough space and time to develop my ideas while encouraging me to ask questions. He profoundly impacted my personality and my research approach.

In my second set of acknowledgments, I would like to extend my thanks to **Dr. Michelle A. Cheatham**. Dr. Cheatham is one of my role models in the Department of Computer Science at Wright State. Her skills, along with her cheerful and friendly attitude, always remind me to remain humble. Furthermore, I want to extend my gratitude to **Dr. Matthew M. Molineaux**. Dr. Molineaux constantly inspires me through his confidence, work ethic, and solution approaches to research problems. In addition, I want to thank **Dr. Mateen M. Rizki** for his support throughout my master's and Ph.D. Finally, I express my thanks to **Dr. Michael L. Raymer** for his ideas and suggestions on my thesis.

I want to show my warmest appreciation to my family for their immense support. First, I thank My parents, **Tulasi Ram Anjaneyulu Kondrakunta** and **Varalakshmi Kondrakunta** for their courage, patience, unwavering support, and encouragement throughout my six long years in graduate school. Second, I thank my sister, **Dr. Bhagya Sree Divya Kondrakunta**, my best friend, cheers me up every time. Finally, I thank My Husband, **Dr. Venkatsampath Raja Gogineni**, for his support in all aspects of my life. I deem myself very lucky to have such a wonderful family.

The National Science Foundation supports this research under grant 1849131, AFOSR Grant Number FA2386-17-1-4063, and the Office of Naval Research under grant N00014-18-1-2009.

I dedicate this thesis to my parents, my sister, and my husband.

Introduction

One of the main goals of artificial intelligence is to build a robust *Intelligent Physical System (IPS)*. The IPS must hold several types of intelligent behaviors to work robustly in the real world. In theory, multiple types of intelligence [39] exist. Some include linguistic intelligence, logical-mathematical intelligence, spatial intelligence. Linguistic intelligence deals with the ability to learn, understand, express using a language. Logic-mathematical intelligence implements one's ability to reason about numbers and quantitative parameters. Spatial intelligence includes the ability to detect patterns, understand and manipulate patterns around the space. Over the years, because of the advent of technology, we are approaching the idea of IPS. A couple of examples include Sophia [54], and Waymo [98]. Although both are awe-inspiring accomplishments, they exploit only linguistic or spatial intelligence. However, robustness demands all kinds of intelligent behaviors. Furthermore, an agent containing all types of intelligence will compare to human intelligence and will possess human-like capabilities.

We claim that there are four main capabilities that the IPS must hold. The capabilities include perception, thought, action, and communication. Perception gathers information about the world around it. Thought processes the collected information, generates goals, and creates plans to achieve them. Action performs behavior in the real world to execute the plans and achieve the goals. Finally, communication can express, justify, and explain its thought process to other humans or agents. We also claim that the agent-containing all four capabilities will be called a IPS with general intelligence. Several research areas attempt to reach general intelligence behaviors in autonomous agents. Some of these include developing the agents through neural nets, reflex agents, utility-based agents, learning agents, and cognitive architectures. This thesis will consider the approach of cognitive architectures, which subsumes many of the approaches mentioned above. We use a goal-based cognitive architecture, where we expect the agent to achieve specific goals. We argue that one of the critical abilities for such a goal-based IPS is to manage its goal set (i.e., the agenda of the agent) by reasoning about selecting, formulating, monitoring, achieving, evaluating, delegating, sharing and changing its goals.

We define a *goal* as a future state that the agent wants to achieve. The state could include the environment and the objects around the agent, including itself. An external source can provide goals to the agent, or the agent can formulate its own goals. In most cases, humans provide an agent with multiple goals, and it must be able to achieve a majority of these. In such cases, the agent should be wise to perform several operations on the goal set to be highly flexible and robust. These operations include:

- Selecting and ordering its goals;
- Formulating new goals to tackle new situations;
- Monitoring if the conditions of the environment are valid to achieve a goal;
- Achieving and performing actions in the real world;
- Evaluating if an agent achieved its goal;
- Delegating a goal to a cooperative agent when it cannot achieve its goals;
- Sharing a goal with a cooperative agent when one agent cannot achieve it;
- Changing its goal set when the environment changes.

Such goal operations make an agent flexible and self-reliant, and they improve common sense and both interpersonal and intrapersonal intelligence. Specifically, this thesis focuses on three goal operations: goal selection, goal change, and goal formulation. But, in addition to examining the mechanism of each goal operation in isolation, *we investigate the interaction between goal operations.* Here We develop a novel goal management algorithm when multiple-goal operations interact. This contribution stands as the central idea of the thesis.

The goal selection operation selects one or more goals from the goal set of the agent. For example, humans perform goal selection almost every day. We prioritize personal goals according to individual preferences and work toward achieving the goals. Similarly, the goal selection is required for the agent when it has a list of goals to pursue, and the agent cannot spend time on all of them simultaneously or when there is a limit to the amount of resources present. In the two scenarios above, the user must provide specific goals to the agent. which is very tedious when working on complex domains. So, goal selection makes the agent self-reliant when multiple goals exist.

The goal change operation helps the agent change its current goal to a similar goal, such that the main objective of the original goal remains the same. This operation arises when the current goal is no longer achievable due to several reasons. For example, some of the reasons are that the situations in the world might have changed and the current goal is no longer valid, the agent is out of resources to achieve the current goal, or some other agent has already achieved the goal. In the situations mentioned earlier, if the agent cannot change its goal, it might get stuck while acting, repeat the completed task, or perform actions that make it seem unintelligent. So, goal change makes the agent flexible and intelligent while protecting itself and its environment.

Finally, the goal formulation operation helps the agent generate its own goals without the help of an external source. For example, consider a situation where the agent has more than a sufficient amount of resources. There could be instances when the agent can use those resources to achieve a goal that might be useful to the user in the future or to handle an emergency like a fire. In the latter case, the agent should formulate a goal to save itself when a person is not around to save the agent. Goal formulation will be crucial in these and similar situations. This, goal formulation allows the agent to be self-reliant, smart, and a reasonable entity.

Apart from working on the above three-goal operations individually, we also focus on the interactions among the operations. It is essential because negative interactions may lead to poor performance if not chosen carefully, and neglected positive interactions represent missed opportunities. The next subsection details the contributions of the thesis.

1.1 Thesis Contributions

The thesis focuses on developing a goal management algorithm for handling multiple-goal operations that co-occur. We study all the goal operations under a relatively new research area called *goal reasoning*. Goal reasoning research that casts thinking in terms of goal operations [23, 58, 61, 45] is relatively recent and underdeveloped. The problem of interaction has never before been examined. The following are the main contributions of the thesis:

- This work implements the goal selection operation using three strategies. Specifically, we implemented a first in first out strategy [58], a cost to benefit strategy (which draws inspiration from the work of [55]) and Hill Climbing [63] strategy (which is a greedy algorithm). The first in first out strategy strategy represents achieving the goals in the order the agent receives. The cost to benefit strategy analyses the costs and benefits of attaining all individual goals in the goal set and chooses one goal with the best cost to benefit ratio. Finally, Hill Climbing implements a greedy algorithm based on domain-specific values for selecting goals. We implement the selection operation in the construction, restaurant, and marine survey domains. In addition, we also provide a formal representation of the goal selection operation (Chapter 3, 6).

- This work implements the goal change operation in several domains. Some domains include the construction domain and the marine survey domain. We implement the goal change operation by changing the predicate or arguments of the goal. We also formally represent the goal change operation (Chapter 4, 6).

- This work implements a novel method to perform the goal formulation operation in a naval mine clearance domain, a labor relations domain, a construction domain, and a marine survey domain. The thesis also formally represents the goal formulation operation (Chapter 5, 6).

- The main contribution of the thesis is the novel goal management method to aid in an agent's decision-making when multiple-goal operations co-occur. In addition, the thesis also implements and tests the method across two different domains to prove its generality. The domains are the marine survey domain and the construction domain (chapter 6).

The idea of individual goal operations is not new. Several research efforts realize the implementation of each goal operation to some extent. Still, no one has attempted to integrate all of the operations in a single agent framework. This thesis addresses the knowledge gap present by developing a novel method for the agent to handle such situations. We evaluate the developed method by generating experimental evidence and showing its statistical significance. The generated evidence also supports the following claims.

- **Claim 1**: Individual goal operations aid the agent in many situations. We show that the individual goal operations are limited in their ability to handle many decision-making situations robustly.

- **Claim 2**: There is a measurable advantage to the agent performance with the inclusion of the goal management. In a dynamic environment where there is a possibility of interaction of multiple goal operations, we need the extra decision-making algorithms for the agent to act robustly.

This work implements goal operations using a cognitive architecture called the Metacognitive Integrated Dual-Cycle Architecture. Before diving deep into it, the following subsection provides a brief background on several alternative cognitive architectures.

1.2 Cognitive Architectures

There are many cognitive architectures in development; some well-known cognitive architectures in the research area are SOAR, ACT-R, CLARION, and ICARUS. We will focus on the primary operations and principles of each of these architectures.

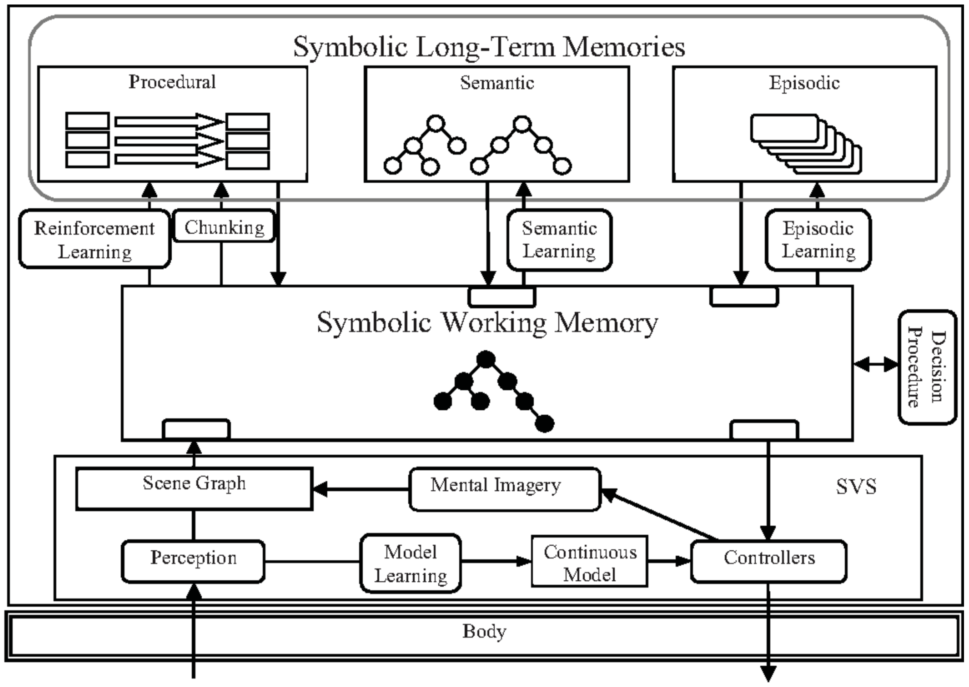

Figure 1.1: Soar Cognitive Architecture [66]. The SOAR interacts with the real world through the perception and action modules. All the data is initially stored in short-term memory and then classified into long-term (procedural, semantic, episodic) after processing.

Soar [66] derives its name from this basic cycle of State, Operator, And Result. Figure 1.1 depicts the general architecture of SOAR. It contains perception and action modules to interact with the world. The SOAR architecture also has different types of memory called short-term, procedural, semantic, and episodic. It also has a decision module that would make decisions according to the algorithms implemented. Soar updates the procedural memory module with the help of reinforcement learning techniques.

The theory behind developing the cognitive architecture is the Problem Space Hypothesis. While attempting to achieve a goal, the Problem Space Hypothesis states that an agent can achieve the goal-oriented behavior by searching through a space of possible states (a problem space). At each step, a single operator is selected and then applied to the agent's current state, leading to internal changes, such as retrieval of knowledge from long-term memory or modifications or external actions in the world. The Soar architecture is the oldest and most famous cognitive architecture. Hence, it works on several game domains like Towers of Hanoi, Tic Tac Toe, and others that might use AI agents such as StarCraft and Minecraft.

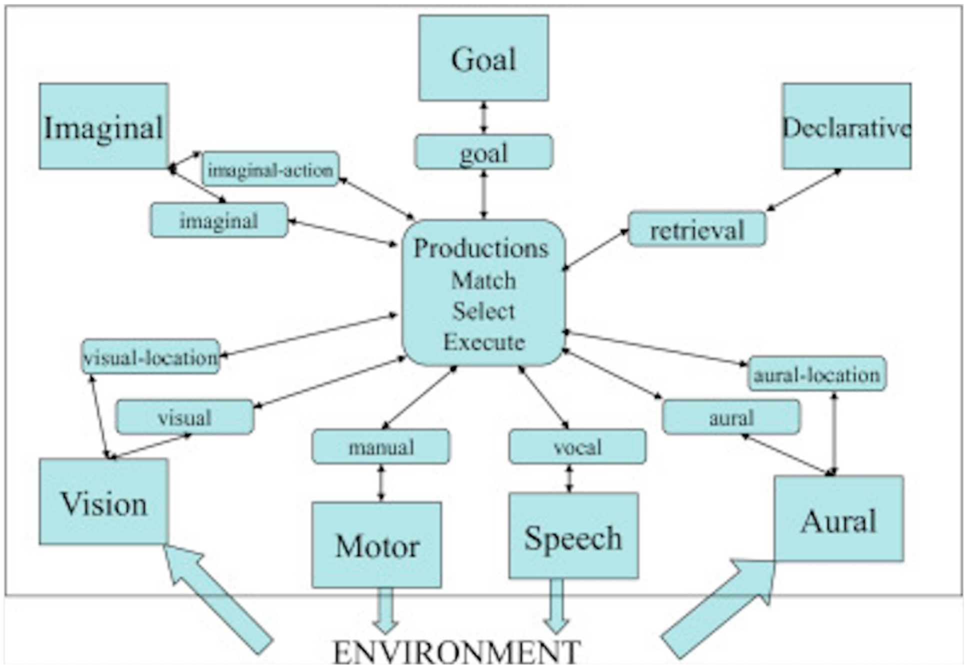

Figure 1.2: Act-R Cognitive Architecture [91]. The Act-R interacts with the real world through the motor, speech, vision, and Aural modules.

Figure 1.2 depicts the general architecture of Adaptive Control of Thought - Rational (ACT-R) [91, 102, 5]. It contains vision, aural, motor, and speech modules to interact with the world. It also has a central decision module that would make decisions called the productions module. This module works similarly to the procedural knowledge rules. Initially written in LISP, the developers created Act-R in 1973.

Unlike Soar, ACT-R focuses on human cognition and human intelligence to understand them closely and attempts to reproduce the observations. Scientists try to understand human behavior by various physiological tests. In general ACT-R looks like a programming language. It is a framework to create multiple models, which adds the user's assumptions to the agent. In addition to using deductions from psychology experiments conducted on humans, ACT-R also uses information about the domain to model cognition. The ACT-R evaluates the results by comparing the accuracy of results, measuring the time taken to perform an operation, and from the neurological data of FMRI. One of the ACT-R implementations, which is very successful, is ”the cognitive tutor for mathematics.” One of the works relevant to this thesis here is an integrated model of cognitive control in task switching [3] which discusses how humans control their everyday tasks and switch between those tasks. The author developed a cognition control model for these abstractly and then linked six basic behavioral effects to the model. They have finally compared the results of task switching with task repetition and depicted them in graphical format. The agent showed an increase in its performance because of the task switching.

Next is CLARION, Connectionist Learning with Adaptive Rule Induction On-line [99, 100]. Figure 1.3 depicts the architecture of CLARION with its four subsystems (Sun, 2007). In the metacognitive subsystem, the architecture does not behave like a single-minded system. Instead, it has the flexibility to choose its behavior by collecting all available information through other subsystems and interacting with them. CLARION has four subsystems that have their roles.

- Action-centered subsystem: This controls all types of actions that the agent performs.

- Nonaction-centered subsystem: This maintains the knowledge of the agent.

- Motivational subsystem: This provides the inspirations for perception, action, and

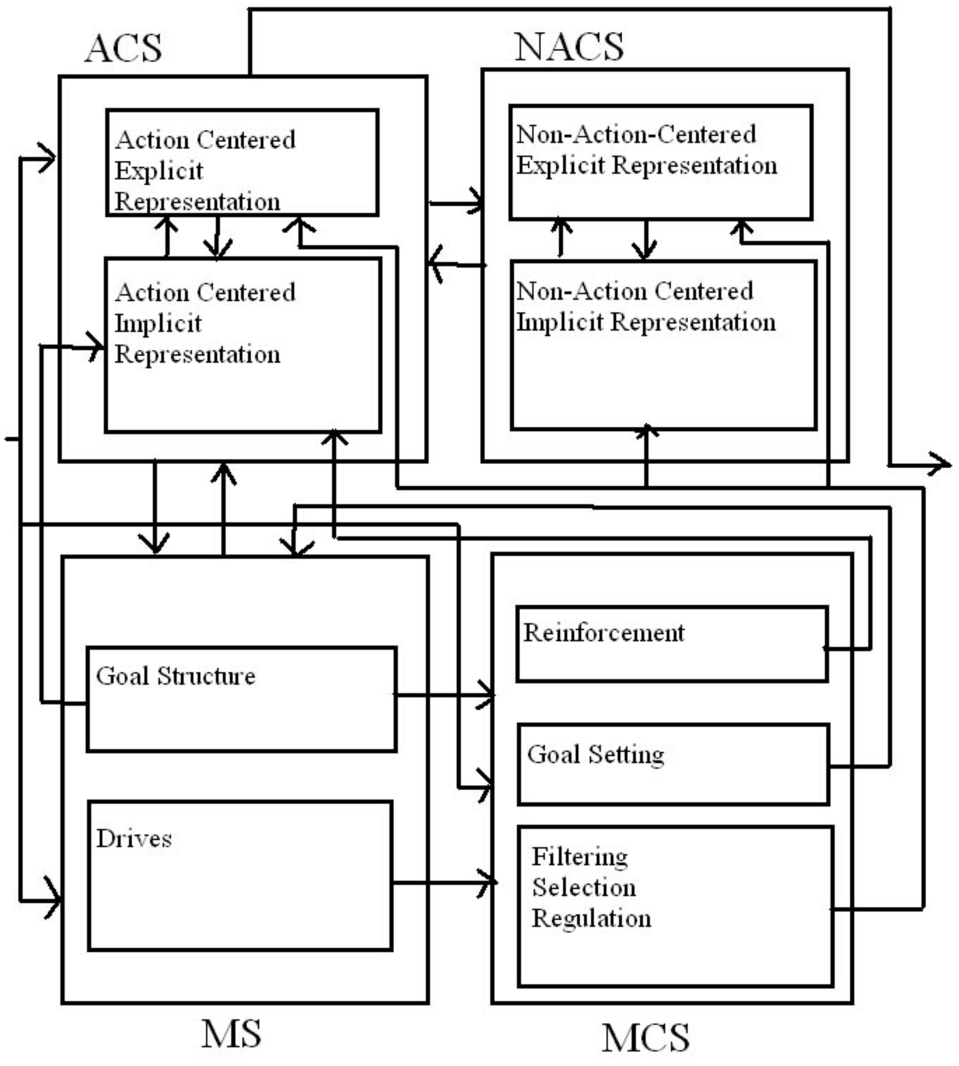
ACS
NACS
Action Centered Explicit Representation
Action Centered Implicit Representation
Non-Action-Centered Explicit Representation
Non-Action Centered Implicit Representation
Goal Structure
Drives
Reinforcement
Goal Setting
Filtering Selection Regulation
MS
MCS

Figure 1.3: Clarion Cognitive Architecture [99]. The image also depicts all four subsystems of Clarion: action-centered subsystem, nonaction centered subsystem, motivational subsystem, and Meta-cognitive subsystem.

cognition.

- Meta-cognitive subsystem: This monitors and modifies all the other subsystems.

The ICARUS architecture [96, 16] took some of its assumptions from the Soar and ACT-R cognitive systems. However, the ICARUS architecture stands alone from other systems by focusing on perception and action to develop a cognitive system.

Figure 1.4 depicts ICARUS, its memories, and processes. The modules of ICARUS

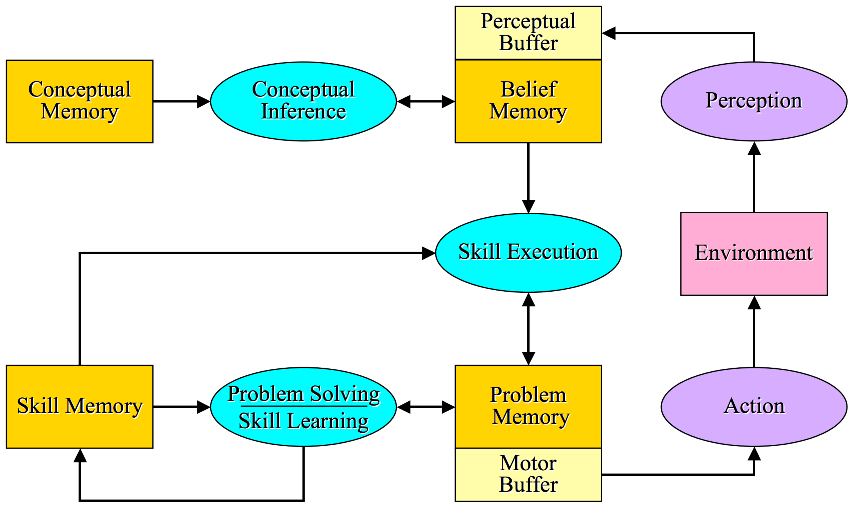

Figure 1.4: Icarus Cognitive Architecture [96]. Icarus interacts with the real world using perception and action modules.

include learning, problem-solving, skill execution, and conceptual interface. All the modules are cascaded, and the lower-level modules help or provide their outputs to the higher-level modules. ICARUS [15, 67] also performs the goal selection operation from a given set of goals by assigning priority values to each goal. The range of the values varies from 0 to 10; the value zero indicates the least priority, and ten indicates the highest priority. However, the author did not put much focus on how the values are assigned.

We introduced four different cognitive architectures in this subsection. In addition, we also discussed the operations and motivations behind each of the cognitive architectures. The next subsection describes the implementation of the goal operations in the four architectures.

1.3 Goal Operations in Cognitive Architectures

The Soar architecture heavily concentrates on selecting, evaluating, and achieving its problem-solving operators as opposed to goals. Therefore, goal reasoning task is not directly accomplished or implemented in the Soar. However, the agent performs goal reasoning to some extent using operators. However, we believe that goal reasoning could be a crucial improvement in some domains of Soar architecture. For example, consider Soar's implementation in one of the game domains mentioned in the earlier sections. Consider the game of tic-tac-toe. Instead of always playing towards the goal to win the game, the agent can lose the game. One example of such a case is when trying to teach a kid. Therefore, we can use reasoning capabilities directly and quickly with the goal reasoning module instead of choosing from many operators in the problem space (or) waiting for one goal from a user.

ACT-R produces a goal-directed behavior by assigning weight values to goals. One version of ACT-R presents a Goal-Restricted Production System (GRAPES) model of ACT-R [4]. In addition, ACT-R also implements the sub-goaling of a given goal. Sub-goaling refers to breaking down a goal into several small goals that achieve the original goal when combined. Although it seems like ACT-R could perform some level of goal reasoning, both Soar and ACT-R are limited to one top-level goal. Explicitly, the focus of both architectures is not on goal reasoning or goal operations. However, implicitly, both architectures perform goal achievement.

Clarion emulates the goal-directed behavior with the help of its four reasoning subsystems. First, Clarion performs goal setting, which is similar to the selection/ formulation operation in the Motivational Subsystem (MS). The Action Centered Subsystem (ACS) focuses on goal achievement. The Meta-Cognitive Subsystems (MCS) reasons about goals and pushes the MS toward goal setting. Although the Clarion does not implement goal reasoning with explicit goal operations, it implicitly performs some goal operations like selection/formulation.

The Icarus architecture implements the goal selection operation by assigning priority values to goals. In addition, it performs goal achievement and goal re-achievement if environmental conditions make the goal invalid. So, the agent must be performing implicit/explicit goal monitoring. Furthermore, Icarus also performs goal formulation operations using rules in the long-term memory.

After examining the goal reasoning capabilities in each architecture, we believe that all the architectures would greatly benefit from including explicit goal reasoning in each of them. Let us now turn our focus toward implementing all the goal reasoning capabilities using one architecture. We present the architecture in the next chapter 2of the thesis. However, before diving into MIDCA, let us look at the organization of the thesis document in the following subsection.

1.4 Outline of the thesis

We organize the next chapters of the thesis in the following format. Chapter 2 presents the background knowledge about MIDCA. Chapter 3, chapter 4, and chapter 5 represent the goal selection, goal change, and goal formulation operations in detail. Chapter 6 represents the ideas on the interactions of the goal operations. It also presents a novel algorithm that addresses the problem. Chapter 8 outlines the literature review and offers the ideas of other researchers in a similar field. Finally, chapter 9 concludes the thesis.

More specifically, chapter 2 introduces the cognitive architecture which we use for this thesis. It describes the cognitive architecture in general and discusses goal management in the architecture with a knowledge structure called the goal graph. This chapter will also talk about various other goal operations and briefly describes each of them. In addition, this chapter will provide the user with the necessary knowledge to understand the context and content of the thesis.

Chapter 3 discusses the goal selection operation in detail and introduces a formal representation of the operation. In addition, we also describe the domains in which we implement the operation. Next, chapter 3 presents the evaluation method used in the domains to test the agent's performance and finally depicts the results in graphical format.

Chapter 4 explains details about the goal change operation. First, it introduces the formal algorithm of the operation and the formal representation of the predicate transformation, which aid in the implementation of the operation. Next, we describe the domain in which we implement the operation. The domain is called the construction domain. Finally, we provide the method of evaluation and show the results.

Chapter 5 provides the details about goal formulation. It also provides a formal representation of the operation. This chapter also describes the rich domains in which the operation is implemented and presents the evaluation method. Finally, we present the results.

After describing these three goal operations, chapter 6 introduces the main problem which the thesis attempts to solve. Then, it explains the developed method to manage the interaction of goal operations. It also evaluates the implementation against several different agents. Finally, the chapter 7 provides the performance comparison among different agents in graph format.

Chapter 8 provides the literature review for this thesis from relevant papers. This chapter includes the work on various goal operations like selection, change, and formulation. In addition, it also looks at several cognitive architectures to provide a contextual comparison with the MIDCA architecture. Furthermore, the chapter also presents work on multi-agent systems to present future extensions of the current work. Finally, we will explore some of the planning agents and psychology papers relevant to the thesis.

Chapter 9 summarizes the thesis work and its contributions and presents future ideas to develop and refine the research further.

MIDCA

The *Metacognitive Integrated Dual-Cycle Architecture (MIDCA)*[22] is the cognitive architecture for our implementations. The concept of cognition was around for a long duration. However, Cox initially put forth the concept of computational metacognition [19, 24] having taken his inspiration from the work of Nelson and Narrens [77] on metamemory in the early 1990s. The current architecture was first proposed in 2011 by Cox, Oates, and Perlis [25] which was influenced by the work of Don Norman [79] and Michael Lewis [68]. MIDCA is capable of both cognitive and metacognitive tasks. In addition, MIDCA uses an approach called *Goal-Driven Autonomy (GDA)*. GDA manages the goals of an agent, which allows the agent to better distinguish between goals and actions. The earliest GDA approach, INTRO, originally described an agent that generated its own goal while trying to explain an anomaly in the world [20].

As the name suggests, MIDCA has two cycles of operation: one a cognitive cycle and the other metacognitive. The cognitive cycle achieves goals related to the physical world to manage the environment. The metacognitive cycle works on meta-level goals to manage the cognitive cycle. Figure 2.1 below depicts the MIDCA architecture. The orange represents the cognitive cycle, and the blue represents the metacognitive cycle. Six phases in each cycle perform distinct operations. The operations of each phase [22] in the cognitive layer are as follows.

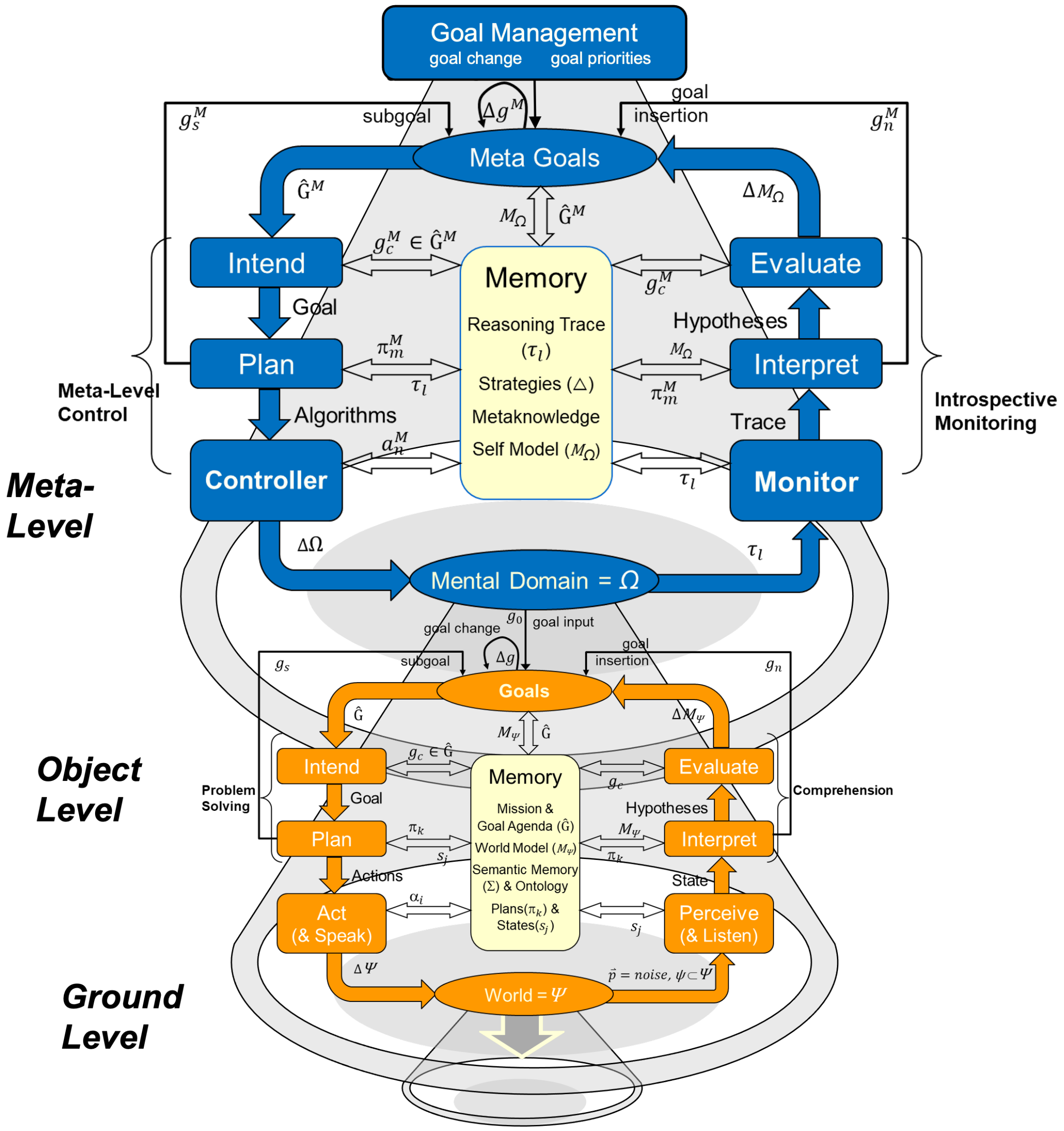

Figure 2.1: MIDCA Cognitive Architecture. MIDCA operates on three levels: ground level, object level, and meta-level. The ground level represents all the real-world objects and simulators. The object level (cognition) processes information from the ground level and interacts with the ground level to achieve goals in the real world. The meta-level (metacognition) works on the object level and modifies the agent's cognition. In addition, the orange color represents the phases in the cognitive cycle, and the blue color represents the phases in the metacognitive cycle.

- **Perceive**: Monitors the environment (Ψ), inputs percepts ($\vec{p}$) and updates the representations of the world in MIDCA as states (s_j). Currently, we represent the world states using a first-order predicate-argument format.

- **Interpret**: Validates all the state information, detects anomalies and reasons about them to formulate any new goals (g_n). This phase also gets all the goals from the user and stores them in the goal agenda ($\hat{\boldsymbol{G}}$). The agenda is represented by a structure called the goal graph (see section 2.1).

- **Evaluate**: Checks to see if a goal state is achieved or not. If it is, then it removes the achieved goal from the agenda.

- **Intend**: Performs the goal selection operation. This phase chooses a current goal set (g_c) from all the goals in the goal agenda by following various strategies.

- **Plan**: Gets all the selected goals from the Intend phase. It checks if a plan (π_k) exists to achieve g_c. If a plan exists, then nothing happens. Otherwise, it generates sequence of actions ($\pi_k = \langle \alpha_1, \alpha_2, ...\alpha_n \rangle$) to constitute the current plan.

- **Act**: Iterates through the plan and performs one action per cycle. This is executed in a simulator or in the real world when MIDCA is connected to a robot or other physical platform.

The phases in the metacognitive cycle perform similar actions to the cognitive cycle. For example, the Monitor phase monitors the cognitive cycle. The meta-level Interpret phase generates the goals. The meta-level Evaluate phase evaluates if an agent achieved its goal. The meta-level Intend phase selects the current goal from the meta-level goal set. The meta-level Plan phase generates a plan to achieve the current goal. Finally, the Control phase performs actions on the cognitive cycle. In MIDCA, there is no strict ordering of the phases; it provides the flexibility to add or remove phases according to the application demand. Earlier, we introduced a structure called the goal graph. It is a knowledge structure that helps MIDCA maintain its current goal set. The following section describes the functionalities and importance of the goal graph.

2.1 The Goal Graph

As introduced earlier, a goal graph helps manage the goal and the plans for the current goal agenda; it does this by providing partial ordering to all the unique goals. A goal graph is a simple data structure representing each goal as nodes, root nodes, or leaf nodes. According to the nature of the goals, there might be one or more roots for the goal graph tree structure. For example, consider the two goals on(A, B) and on(D, C), where letters represent toy blocks stacked on one another. These two goals are similar, so they are considered different root nodes with the same priority. Later, Intend will select one of them based on various factors. MIDCA initializes an empty goal graph whenever MIDCA first starts. Below is how each phase interacts with the goal graph:

- **Perceive**: No interaction
- **Interpret**: Gets the goals from the user and inserts them into the goal graph. Interpret also inserts the formulated goal when an anomaly is detected and places it above the root node to prioritize it.
- **Evaluate**: Checks to see if the agent achieved its current goal and, if so, removes the goal set and the corresponding plan from the goal graph.
- **Intend**: Checks to see if the goal graph is empty. If it is empty, then the agent skips Intend. Otherwise, it checks if the current goal set is empty. If it is, then based on strategies (example: first in first out, hill climbing, or information measures), select a new goal set and assign it to the current goal. If a current goal exists, then Intend is skipped.
- **Plan**: Checks the goal graph for a matching plan. If one exists, it checks validity. If no matching plans exist or are not valid, the Plan phase generates a new plan and inserts it into the goal graph.

- **Act**: Iterates over the plan from the goal graph and implements one action at a time until the agent achieves all actions in the plan.

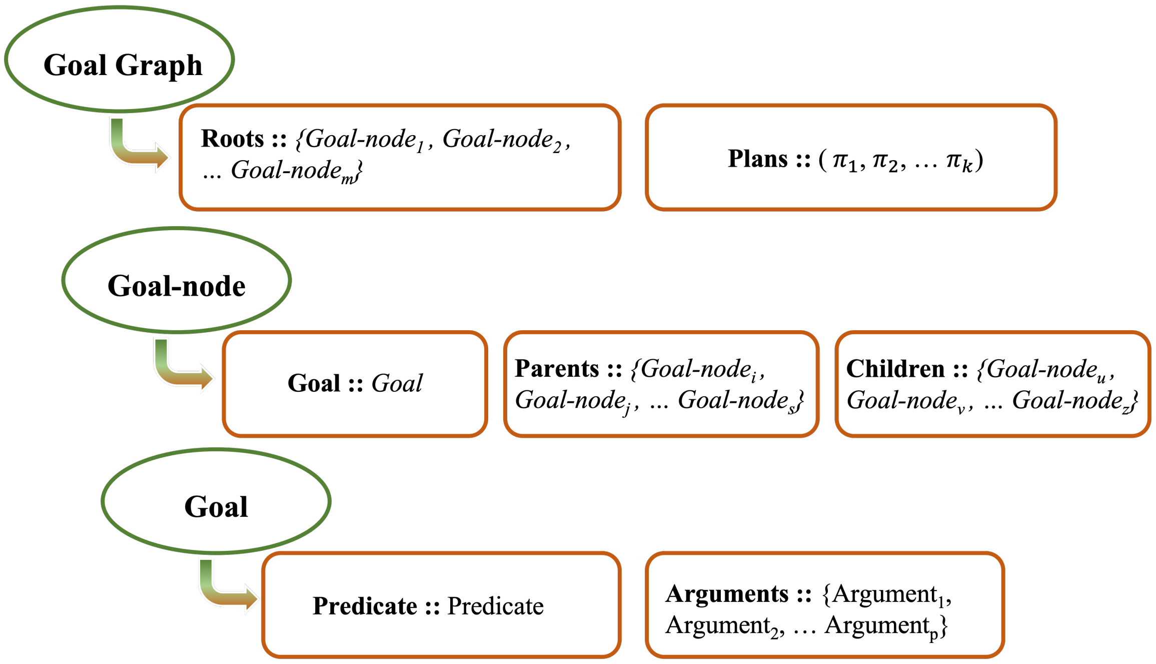
Goal Graph
Roots :: {Goal-node$_1$, Goal-node$_2$, ... Goal-node$_m$}
Plans :: ($\pi_1, \pi_2, \dots \pi_k$)
Goal-node
Goal :: Goal
Parents :: {Goal-node$_i$, Goal-node$_j$, ... Goal-node$_s$}
Children :: {Goal-node$_u$, Goal-node$_v$, ... Goal-node$_z$}
Goal
Predicate :: Predicate
Arguments :: {Argument$_1$, Argument$_2$, … Argument$_p$}

Figure 2.2: The goal graph knowledge structure. The knowledge structure contains three main classes: goal, goal node, and goal graph.

Figure 2.2 presents all the components of the goal graph knowledge structure. There are three classes for the goal graph: goal graph, goal node, and goal. The goal graph class creates the overall goal graph by establishing the set of roots. Each element of the roots is a goal node object. In addition, the goal graph also stores all the plans. Each plan contains a pointer to the goal that it is trying to achieve. The goal node contains the goal class object and has pointers to the parents and children of each goal. Each element of the parents and children is also the goal node's object. The goal class constitutes a goal in the format of a predicate and its arguments.

The goal graph knowledge structure is a directed tree/lattice structure, an instance of the structure is shown in Figure 2.3. A child node has lower precedence than its parent, and all the sibling nodes have equal priority. Here the ¬onfire(A) is the root and has the highest importance. The child nodes on(B, D), on(A, B), and on(D, C) have equal priority, with the preference being less than their parent goal.

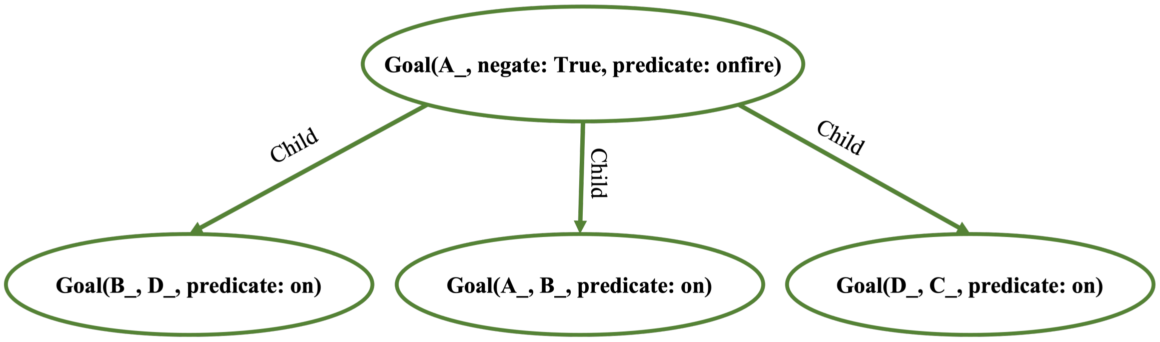
Goal(A_, negate: True, predicate: onfire)
Child
Child
Child
Goal(B_, D_, predicate: on)
Goal(A_, B_, predicate: on)
Goal(D_, C_, predicate: on)

Figure 2.3: The figure depicts one instance of a goal graph knowledge structure. The block, A, not being on fire is the goal with the highest priority. That is the root node in the image. The child nodes are the goals to stack blocks on top of one other.

In conclusion, the goal graph helps MIDCA better manage its goals by inputting all goals, validating them, and organizing them in a tree structure. A goal graph validates a goal by checking the predicates and arguments of a goal and checking if they are present in agents' memory. The validation is essential to easily read, achieve, and remove from the goal graph. Since we learned how MIDCA manages its goals, let us look at the operations MIDCA can perform on its goals.

2.2 Goal Operations

One of the factors in determining an agent's performance is its capability to create, understand, and manage its goals. MIDCA instantiates such capabilities as goal operations. There are several actions which the agent can perform on goals [23, 58], some of which are as follows.

- **Goal Selection**: Selection of one or more goals from the goal agenda ($\hat{G}$) of the agent to become the current goal set (g_c). This operation helps organize the goals of the agent by prioritizing them according to the situation using various strategies [58].

Currently, we use three approaches to perform this operation, which we elaborate on in subsequent chapters.

- **Goal Change**: Changes the current goal set (g_c) to another set because of several reasons. Some of the reasons include sudden environmental changes or the agent being out of resources to achieve g_c. In such situations, checking for alternate goals that could satisfy the goal. Currently, we achieve goal change through predicate transformations in MIDCA [23]. Look at chapter 4 for details.

- **Goal Formulation**: An agent generating its own goals (g_n) without the help of an external user. Goal formulation is a high-level task that is the basis of the GDA approach. Currently, MIDCA formulates its goals when a discrepancy is detected and perceived as a problem [61, 64]. We include details about the operation in chapter 5.

- **Goal Delegation**: Transfer of one or more goals to either a human or another agent. This operation is advantageous in a multi-agent domain [45]. Some triggering reasons for this operation are: the agent has more goals than it can accomplish, or the agent does not know how to plan the goal.

- **Goal Achievement**: Checking the current goal set (g_c) and monitoring it to see whether the agent achieved its goal or track the percentage of the goal achieved. Currently, MIDCA checks to see if the whole goal is achieved or not.

- **Goal Monitoring**: Monitoring the changes in the environment and updating them to see if the reasons for the goal are still valid [33]. Monitoring is an essential goal operation as it continuously informs the agent when planning or plan execution may no longer be necessary.

Among the above goal operations, this thesis focuses on three: goal selection, goal change, and goal formulation—the reasoning behind why the mentioned three-goal operation is not the focus of this thesis. Let us now examine each of them in detail in the following three chapters.

Goal Selection

The goal selection operation allows the agent to choose and prioritize its goals according to its preferences. This powerful operation enables the agent to be independent when multiple goals exist in its agenda. We implement the goal selection operation in the Intend phase of MIDCA. Intend is a good fit for goal selection because the selection is the main functionality of the phase. Currently, three strategies exist to implement the goal selection: First In First Out (FIFO) [60], a cost to benefit ratio from the domains [60], and a greedy hill-climbing selection [63]. We discuss the first two methods in the subsequent paragraphs. However, we describe the greedy approach in the later chapters.

As the name suggests, FIFO selects the goals in the same order as the input. It is a simple implementation method for the selection operation. In this method, an agent accomplishes the goals without considering limitations or available resources. This method is suitable when a relatively unlimited amount of resources exist in the world, and the order of the goals does not matter. However, since that is not the case in many domains, we developed a second method that uses domain-specific performance metrics.

The idea to use the metrics from domains to prioritize or select goals was inspired by the work of Johnson [55]. Johnson uses domain-specific metrics in an airport domain. The airport domain contains one airport and two office buildings, which are partially observable to the agent. The agent's goal is to locate an officer in the environment within a specific time limit. As the agent traverses through the environment, it updates its knowledge and reduces the amount of uncertainty about the world to find the officer. The author uses information metrics such as area traversed and time-lapsed to select the goals. First, the author calculates the ratio of information measure (area_traversed and time) for every goal. After which, the algorithm selects the goal with a minimum ratio because it minimizes the uncertainty of the location. This work is strictly domain-specific, and we cannot implement it in domains other than time-limited search. In this work, we consider performance metrics similar to the information measure in each domain to aid in the goal selection. The metrics used to perform the goal selection will act as a measure of the performance of the agent in that domain. Currently, the user provides the agent with performance measures for the agent in a domain. In general, the performance measure ratio is the ratio of a performance measure over a limiting value related to the specific domain, as represented in the formula below.

$$Ratio = \frac{Expected\_Performance\_measure}{Limiting\_Value} \tag{3.1}$$

.

In equation 3.1, the performance measure refers to a resource gain of the agent after achieving the goal. The limiting value represents the number of resources the agent needs to spend to achieve the goal.

3.1 Formal Representation of Goal Selection

This section represents the goal selection operation using formal notation. This representation allows the user to understand the operation with all the necessary conditions to perform the selection operation.

Table 3.1 tabulates the formalism for goal selection [60]. The $\hat{G}$ is the goal agenda of the agent. All goals in the agenda follow a first-order predicate-argument structure ($p^x(obj_a, obj_b, ..., obj_y)$), where p^x is the predicate and $obj_a, obj_b, ..., obj_y$ are objects. When given $\hat{G}$, the goal selection operation verifies three preconditions ($pre(\delta^{se})$) and results in a goal set g_c for immediate achievement. The first precondition ($pre_1(\delta^{se})$) checks if the agent's current goal set is empty. The second precondition ($pre_2(\delta^{se})$) iterates through all goals in agent's goal agenda, $\hat{G}$, applies the selection function and returns a final goal set g_{ret}. Finally, the third precondition ($pre_3(\delta^{se})$) assigns the returned goal g_{ret} to the current goal set g_c of the agent and verifies the goal validity. To validate the goal, agent checks if the predicates of each goal belong to the class hierarchy tree (CL) [8], which is present in the domain knowledge of the agent. Bergmann [8] presents the notation of the CL, where the are leaf classes that have a root class, and the root class has a superclass. In addition, it also checks if all the objects of the goal $obj_1, obj_2, ...obj_m$ belong to the objects in the domain $Objs$. After satisfying all the conditions in $pre(\delta^{se})$ the agent can now output a selected goal(s) ($res(\delta^{se})$) and work towards achieving it.

Table 3.1: The table presents the formal representation of the goal selection operation. The operation inputs goal agenda ($\hat{G}$) and output the current goal set (g_c) for immediate achievement. We use δ^{se} to represent the goal selection operation. CL represents the class hierarchy of all predicates, we borrow the notation of CL from [8]. $Objs$ represent all the objects in the domain.

$$\boldsymbol{\delta^{se}(\hat{G} : G) : G}$$
$$head(\delta^{se}) = selection$$
$$parameter(\delta^{se}) = \hat{G} = \{g_1, g_2, ..., g_n\}$$
$$pre_1(\delta^{se}) = g_c = \{\}$$
$$pre_2(\delta^{se}) = g_{ret} \leftarrow argmax_{g \in \hat{G}}\{SelectionFunction\}$$
$$pre_3(\delta^{se}) = (g_c \leftarrow g_{ret}) \wedge g_{ret} = p(obj_1, obj_2, ...obj_m) \wedge p \in CL$$
$$\wedge obj_1 \in Objs \wedge obj_2 \in Objs... \wedge obj_m \in Objs$$
$$pre(\delta^{se}) = \{pre_1(\delta^{se}), pre_2(\delta^{se}), pre_3(\delta^{se})\}$$
$$res(\delta^{se}) = g_c = \{g_{ret}\}$$

So far, we have presented the goal selection operation. In addition, we also represented the strategies used by the Selection Function in pre_2. To meet the second precondition, the agent starts implementing the strategies (example: FIFO). Although we do not explicitly represent it in the table 3.1, the agent implements an appropriate strategy to choose the current goal set g_c. The next sections present the implementation of goal selection operation in the construction and restaurant domains.

3.2 Implementation and Empirical Evaluation of the Goal Selection operation

In this section, we implement the goal selection operation in two different domains [58, 60]. The first is the construction domain, which extends the blocks world domain. The second is the restaurant domain, which captures and implements the main functions of a real-world restaurant. We then present the experimental setup and provide the empirical results in each domain mentioned above.

3.2.1 The Construction Domain

As mentioned, the construction domain is an extension to the blocks world domain, and the goals generated are to build towers. For example, there are a specific number of blocks in the domain. Each block has a distinct identifier (A_, B_, Z_). The agent attempts to stack the blocks on top of one other to build towers (goals). We randomly generate the goals (the number of towers to construct with their heights). The height of a tower ranged between one and seven. We have two assumptions for this domain, first no two goals in one goal agenda ($\hat{G}$) share the same height. In addition, they do not share the same block. Second, If we use a block named 'A_' in one tower, we do not use it in a different tower in the same agenda. The two assumptions are essential to reduce the ambiguity and avoid additional goal generation.

Initially, we assume all the blocks are in a warehouse, which we do not show in the simulator. The construction site would be empty at the beginning of each goal set. Whenever we generate a new goal set, the agent selects one current goal (g_c) for immediate achievement. The agent selects g_c by FIFO and the ratio of the performance measures. After selecting g_c, the agents generate a plan (π_c) to achieve the goal, g_c. Agent implements relevant actions using *operators* to achieve g_c. After achieving g_c the agent chooses a new g_c. The agent continuously selects until it achieves all goals in the goal set ($\hat{G}$) or is run out of resources to achieve any other goals. Interpret generates a new goal set with a new set of resources for the agent to achieve for experimental purposes. The agent resets the world for the new goal set by erasing the previous goal.

The operators described in this domain include stack, stack_mortared, unstack, unstack_mortared, pickup, putdown, get_from_warehouse, put_out_fire. Stack places one block on top of another block. Unstack picks up one block from the top of another block. stack_mortared is similar to stack operator, but this is more sturdy with mortar. The agent can only execute pickup when the block is on the construction site. Similarly, the agent executes putdown only to place the block on the construction site. Finally, the agent implements get_from_warehouse to fetch the blocks from the warehouse.

3.2.1.1 Experimental Design: The Construction Domain

As stated earlier, goal selection operation in the construction domain uses two methods: FIFO and performance measures. Whenever we initialize MIDCA, Interpret generates a goal set to construct a certain number of towers. Intend then performs goal selection on the input using FIFO and performance measure criteria. As explained, FIFO selects g_c in the order of goal generation, and the performance measure calculates the performance estimate utilizing the ratio of the measures. A scoring function constantly calculates the ratio to help the agent choose g_c.

A scoring function assigns each tower a performance number in the construction domain. The performance number quantifies the gain obtained after successfully constructing a tower within time constraints. For a successful construction of a tower of height $'h'$, let us assume the score achieved is $'h'$. Similarly, for constructing m number of towers the merit is h_1, h_2, ...h_m. A cumulative score is given by 3.2. The towers which exceed the time limit will receive a score of 0.

$$\hat{p} = \sum_{n=1}^{m} h_n \tag{3.2}$$

The construction of any tower takes time. Hence we consider time as a limiting factor in this domain. The agent must be more cautious when building tall towers instead of short towers. Therefore, the time taken to construct a tower increases exponentially with its height. Thus, the time taken is a nonlinear function. In the current implementation, the user provides estimates of time values manually. Table 3.2 below presents the values of time taken for all the possible towers. The units in the virtual simulation are seconds which is not equivalent to the real world.

Table 3.2: The table presents the time taken by the agent to construct a tower. On the left column, it represents the height of the tower. On the right column, is its corresponding time.

Tower height	**Time taken** (sec)
1	1
2	2.2
3	3.4
4	5.4
5	8.4
6	13.4
7	22.4

The agent calculates the ratio of the performance function over the estimated time for all the goals in the goal set ($\hat{G}$) to choose the current goal (g_c). The agent chooses the tower with the highest ratio first. After the agent achieves the first goal, it selects the second goal with the second-highest ratio. The agent continuously performs selection until it achieves all the goals in $\hat{G}$ or the agent runs out of resources such as blocks.

$$C = \frac{\hat{P}}{\hat{t}} \tag{3.3}$$

.

Equation (3.3) shows where $\hat{P}$ is the estimated performance and $\hat{t}$ is the estimated time taken to complete the goal. We evaluate the agent performance by taking in two cases. One has no time limit, and the other has a specific deadline value. Let D indicate the overall deadline for a particular problem.

3.2.1.2 Empirical Results: The Construction Domain

This section presents the results obtained with the evaluation described earlier. In addition to different scenarios, we also compare the FIFO and the performance measure (smart method).

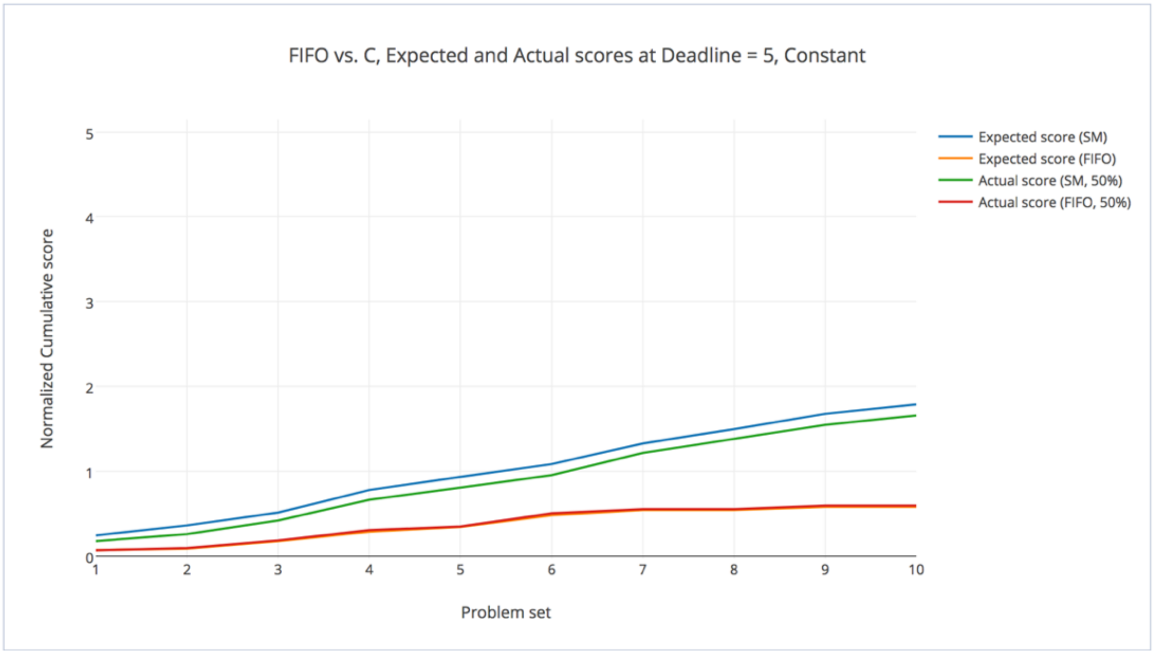

Figure 3.1: The result depicts the comparison between smart and FIFO agents with a deadline of 5 units in the construction domain. The X-Axis represents the problem set, and the Y-Axis shows the cumulative score obtained by the agents. In this graph, the benefit received by the agent varies by 50%. In addition, the time required by the agent to stack blocks varies by 50%.

Figure 3.1 above depicts the results comparing FIFO and the selection method at the constant deadline of 5 seconds. In addition, we introduce a sudden variation in the time taken to construct the tower. We either increase or decrease the time value by 50% of its original value. This variation adds a non-deterministic factor to assess the agent's performance. As the graph depicts, the performance measures' score is higher than using FIFO. In addition, we also depict the expected scores for each of these methods. Expected methods depict the agent performance in a deterministic environment (with no variation in the time taken). The x-axis depicts the problem sets (or) goal agenda sets. Each problem set is averaged across three other sets. The Y-axis presents the agent performance normalized within values 0-5. The maximum percentage of goals achieved by any agent in this experiment is around 35-40%.

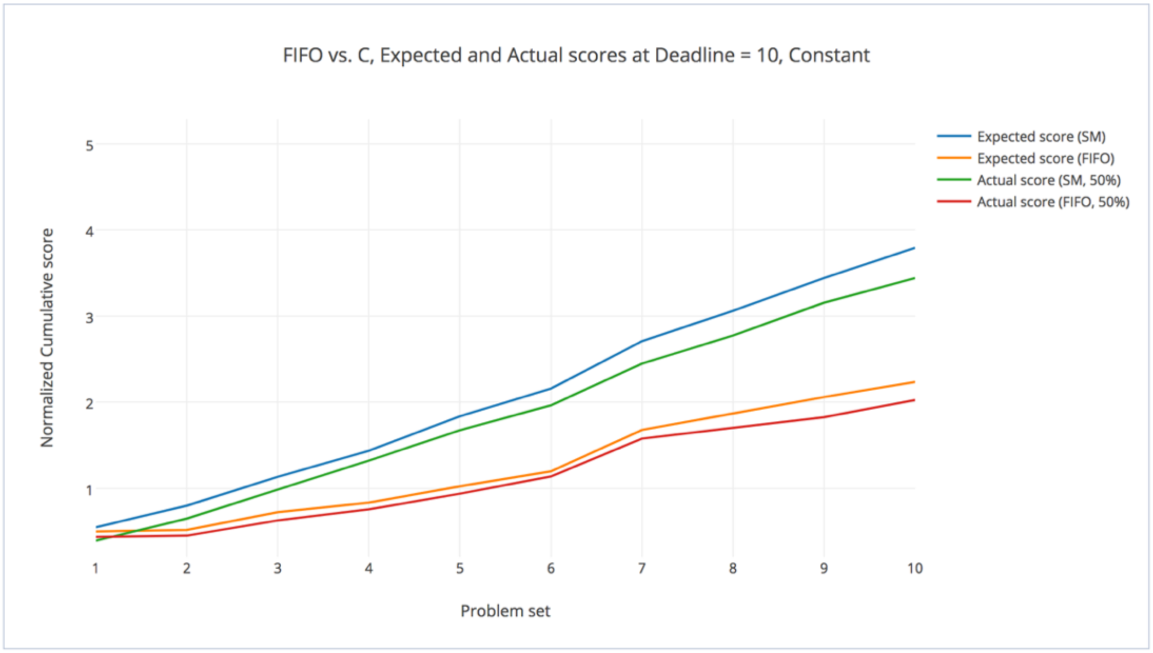

Figure 3.2: The result depicts the comparison between smart and FIFO agents with a deadline of 10 units in the construction domain. The X-Axis represents the problem set, and the Y-Axis shows the cumulative score obtained by the agents. In this graph, the benefit received by the agent varies by 50%. In addition, the time required by the agent to stack blocks varies by 50%.

Similar to the previous results, figure 3.2 above depicts the results comparing FIFO and the selection method. The variation in time taken to construct the tower remains the same as 50%. However, we increase the constant deadline to 10 seconds. This graph follows a similar pattern to the previous results. The graph shows that the performance measures' achieved score is higher than using FIFO. In addition, we also depict the expected scores for each of these methods. The x-axis depicts the problem sets (or) goal agenda sets. Each problem set is averaged across three other sets. The Y-axis presents the agent performance normalized within values 0-5. Any agent's maximum percentage of goals in this experiment is between 75-80$\%$. The increase in agent performance is the increase in its deadline to achieve more goals.

3.2.2 The Restaurant Domain

The restaurant domain is similar to the construction domain. The agent attempts to prepare food orders within a limited menu. It receives orders from a group of customers simultaneously, and each customer might order from one to all the items on the menu. There are fifteen items on the menu varying in cost. Goals in this domain are to serve the customer their order. If a customer orders three different items, the agent receives the order as three goals. In one goal set $\hat{G}$, the agent receives goals from one to eight customers. In this domain, to reduce ambiguity, we assume that the orders of any two customers might contain the same items but not in the same order. There are no customers and no orders in the restaurant domain in the initial state. When MIDCA generates the goal set, the agent prepares orders based on the limited investment. So, the agent can only prepare a certain number of the dishes for the fixed investment. So the goal selection is implemented to choose the goals which yield the best customer satisfaction within the investment amount. We implement goal selection using FIFO and performance metrics similar to the construction domain. Even in this domain, the agent must perform some actions using operators to achieve its current goal g_c.

The operators in this domain include take_order, prepare_order, serve_order. First, Take_order receives the order from the customer and marks the order state as pending. Then the Prepare_order takes the pending orders and prepares the dish. Finally, Serve_order performs the serving action.

3.2.2.1 Experimental Design: The Restaurant Domain

Selection with FIFO is similar to the construction domain in the restaurant domain. However, the performance metrics selection method is different in the restaurant domain. The limiting factor is money in this domain instead of the time in the other one. The calculation of both the functions is as depicted.

A scoring function assigns the score for each customer based on the number of dishes the customer ordered. The agent receives a quantitative score of '1' for each customer's order item or goal. So, if a customer orders m items say i_1,i_2, ...i_m, then the agent will be awarded a score which is a satisfaction (see equation 3.4)

$$\hat{p} = \sum_{n=1}^{m} i_n = m \tag{3.4}$$

. The orders which exceed the investment limit will receive a score of 0. We do not assign scores for partially completed orders.

In addition to score, each item on the menu is also assigned a cost. For example, as mentioned there are fifteen items on the menu with varying costs. So, if a customer orders "n" items with each costing $\hat{m_1}$,$\hat{m_2}$,...$\hat{m_n}$ respectively, then the overall cost is shown in equation 3.5.

$$\hat{m} = \sum_{n=1}^{n} m_n \tag{3.5}$$

.

We depict the equation calculating the performance ratio using both the benefit $\hat{p}$ and cost factors $\hat{m}$ in 3.6. The agent selects the order with the maximum ratio to ensure maximum customer satisfaction. The agent repeats selection until it achieves all the goals in $\hat{G}$ or until it is out of resources.

$$C = \frac{\hat{P}}{\hat{m}} \tag{3.6}$$

.

The next subsection presents the empirical results obtained in the restaurant domain.

3.2.2.2 Empirical Results: The Restaurant Domain

Figure 3.3 depicts the results comparing FIFO and the selection method. The investment limit is \$20. We vary the investment limit by 50%. In this domain, the graph clearly shows that the score achieved using the performance measures is higher than that achieved by using FIFO. In addition to varying the investment by 50%, we also depict the expected scores with no variation. The x-axis depicts the problem sets (or) goal agenda sets. Each problem set is averaged across three other sets. The Y-axis presents the agent performance normalized within values 0-5. The maximum performance achieved is by using performance measures achieved 30%, and FIFO achieved around 25%.

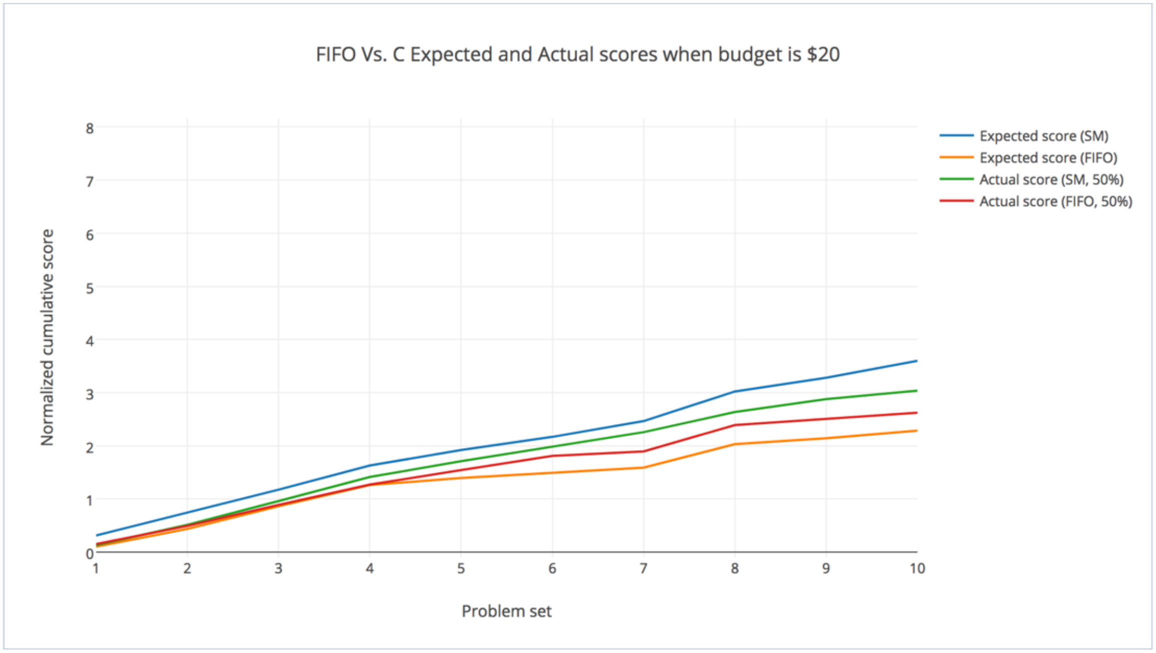

Figure 3.3: The result depicts the comparison between smart and FIFO agents with a budget of 20 dollars in the restaurant domain. The X-Axis represents the problem set, and the Y-Axis shows the cumulative score obtained by the agents. In this graph, the benefit received by the agent varies by 50%. In addition, the money required by the agent to prepare dishes varies by 50%.

Figure 3.4 below depicts the results comparing FIFO and the selection method. The investment limit is \$50. We vary the investment limit by 50%. In this domain, the graph clearly shows that the score achieved using the performance measures (smart method) is higher than that achieved by using FIFO. In addition to varying the investment by 50%, we also depict the expected scores with no variation. The X-axis depicts the problem sets (or) goal agenda sets. Each problem set is averaged across three other sets. The Y-axis presents the agent performance normalized within values 0-5. The maximum performance achieved is with the cost-benefit method, the is greater value is greater than 75%, and FIFO achieved around 70%.

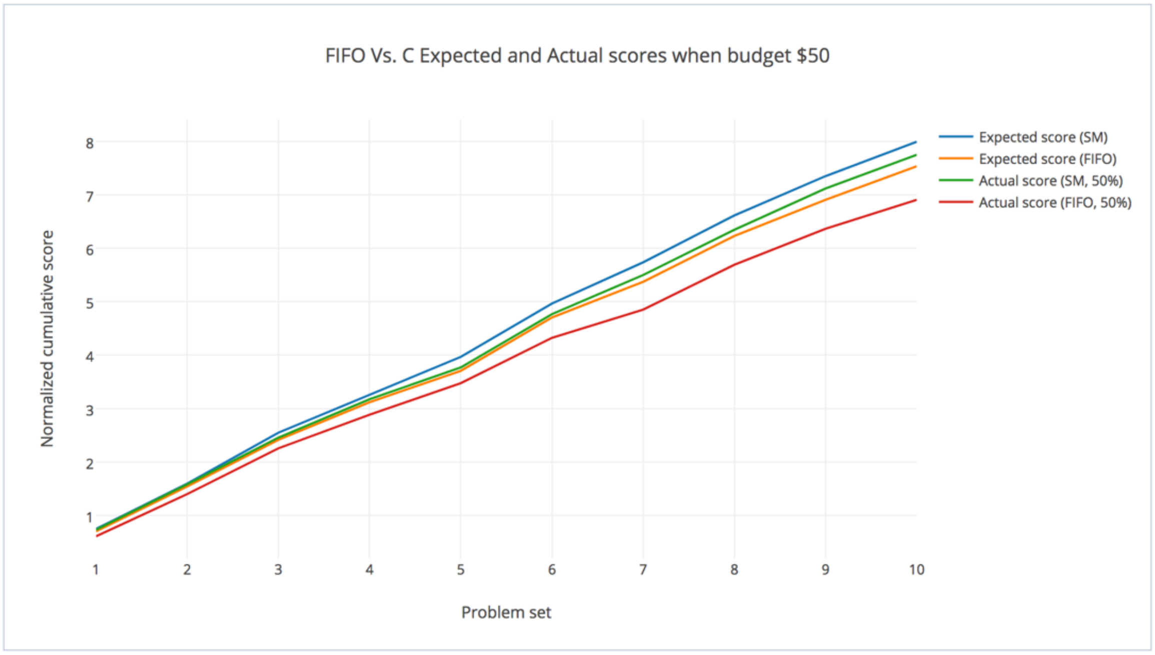

Figure 3.4: The result depicts the comparison between smart and FIFO agents with a budget of 50 dollars in the restaurant domain. The X-Axis represents the problem set, and the Y-Axis shows the cumulative score obtained by the agents. In this graph, the benefit received by the agent varies by 50%. In addition, the money required by the agent to prepare dishes varies by 50%.

3.3 Summary of Goal Selection

Goal selection allows the agent to make independent decisions when encountering multiple goals. We presented the goal selection operation with simple FIFO and cost-benefit ratio methods in two domains. The cost-benefit method provides a noticeable improvement in the agent's performance in both the construction and the restaurant domains.

Some ideas to further improve the selection operations are to improve the current cost-benefit method to fit into other domains. We can extend this method to include other qualitative factors such as the quality of the building or food. Currently, we use one factor to measure the cost or benefit, but we can extend the equations 3.3, 3.6 to include multiple parameters. We can implement the goal selection operation in different cognitive architectures and compare their performances. Furthermore, the work of Schank [93] classifies the goals into several goal types based on the nature of the goal. Each goal type is different and may have its advantages to the agent. For example, achievement goals are the professional goals a human user sets, and crisis goals are the goals the agent generates in a crisis event. This type of classification might be useful for the agent when assigning a priority to the goals. Finally, we also implemented the selection operation using a greedy hill-climbing approach. We depict the method in Chapter 6

Goal Change

The goal change operation is the process of changing the current goal to a similar goal. This operation is significant because it aids the agent in situations where the agent is out of resources or in an undesirable state in the environment.

As mentioned in earlier chapters, metacognition reasons about the agent's cognition. Therefore, metacognition allows the agent to assess its own goals and decisions. Since the goal change operation requires the agent to reason about its own goals and resources, we claim goal change to be a meta-level operation. Similar to the process of metacognition, the meta-layer in MIDCA controls the cognitive layer. Hence, we implement the goal change operation in the meta-level Control phase of MIDCA. Currently, we implement goal change using three predicate transformations[27]: generalization, specialization, and identity.

Generalization changes the predicate of the goal to a more abstract predicate in the conceptual hierarchy of predicates. Specialization changes the predicate of the goal to a more specific predicate in the hierarchy. Identity makes no change and is an essential decision not to change the goal. Cox proposed the method to perform the goal change operation using predicate transformations [27]. This paper suggests the importance of goal change and its implementation in an Air Camping Planning (ACP) domain. In the ACP domain, the agent's main goal is to make the river impassable for the enemy troops. The commander estimates and allows one resource for every bridge to destroy it, thus making the river impassable. However, when working in the environment, if the agent discovers an extra bridge, it should still be able to make the river impassable. The agent can achieve the goal by either getting more resources or changing its goal of ”making the river impassable” to ”restricting the movement across the river.” In both scenarios, there implements a goal change action. The author refers to the first change as goal intrusion and the next as goal erosion transformation by the author.

There are other predicate transformations besides the ones mentioned above: abstraction, concretion, and escalation. Each of these predicate transformations performs a distinct type of change. An agent can choose any predicate transformations and order them according to its situation to reach the desired goal. For this purpose, there is a choose functionality in the goal change operation. Let us now look at the formal representation of the operation.

4.1 Formal Representation of Goal Change

This section depicts the formalism [23] for the predicate transformations: identity (δ^I), generalization(δ^{ge}), and specialization (δ^{sp}). It also represents the algorithm for the goal change operation. As stated earlier, generalization transformation chooses a generic predicate; specialization transform changes the predicate to a specific predicate, and identity transform represents no change. Table 4.1 shows the preconditions and results required and obtained for each transformation. The generalization transform (δ^{ge}) has three preconditions $(pre(\delta^{ge}))$: the first precondition $(pre_1(\delta^{ge}))$ states that the current predicate should belong to the class hierarchy [8] tree and all the objects related should belong to the object class; the second precondition $(pre_2(\delta^{ge}))$ specifies that there should be another predicate that belongs to the superclass of the current predicate; and the third precondition $(pre_3(\delta^{ge}))$ checks if the number of resources present is sufficient to achieve the current goal. This transformation results in a new goal with a more generalized predicate. Specialization (δ^{sp}) also has three preconditions $(pre(\delta^{sp}))$: the first $(pre_1(\delta^{sp}))$ is the same as generalization transform; the second $(pre_1(\delta^{sp}))$ specifies that the current goal predicate should not be the leaf node of the class hierarchy. The third $(pre_1(\delta^{sp}))$ checks if there are extra resources to achieve the

Table 4.1: The table presents the formal representation of the predicate transformations for the goal change operation. Specifically, we present the generalization, specialization, and identity transformations. We use δ^{Δ} to represent the goal change operation and δ^{ge}, δ^{sp}, and δ^{I} to represent generalization, specialization, and identity transformations. Each transformation inputs the current goal set (g_c) and outputs the changed goal for immediate achievement. CL represents the class hierarchy of all predicates; we borrow the notation of CL from [8]. Finally, $Objs$ represents all the objects in the domain.

$\boldsymbol{\delta^{ge}(g_c : G) : G}$
$head(\delta^{ge}) = generalization$
$parameter(\delta^{ge}) = g_c = \ p(obj1, obj2)$
$pre_1(\delta^{ge}) = p \in \boldsymbol{CL} \wedge obj1 \in Objs \wedge obj2 \in Objs$
$pre_2(\delta^{ge}) = \exists p, p' \mid p \in \boldsymbol{CL} \wedge p' \in \boldsymbol{CL} \wedge p_{superclass} = p' \wedge p = ((p_{name}, p', (p.A_1, p.A_2, ...p.A_m)) \wedge p' \neq \top$
$pre_3(\delta^{ge}) = limitedResourcesForGoals(s, g_c)$
$pre(\delta^{ge}) = \{pre_1(\delta^{ge}), pre_2(\delta^{ge}), pre_3(\delta^{ge})\}$
$res(\delta^{ge}) = p'(obj1, obj2)$

$\boldsymbol{\delta^{sp}(g_c : G) : G}$
$head(\delta^{sp}) = specialization$
$parameter(\delta^{sp}) = g_c = \ p(obj1, obj2)$
$pre_1(\delta^{sp}) = p \in \boldsymbol{CL} \wedge obj1 \in Objs \wedge obj2 \in Objs$
$pre_2(\delta^{sp}) = \exists p', p \mid p' \in \boldsymbol{CL} \wedge p \in \boldsymbol{CL} \wedge p'_{superclass} = p \wedge p' = ((p'_{name}, p, (p'.A_1, p'.A_2, ...p'.A_m)) \wedge p' \notin L_c$
$pre_3(\delta^{sp}) = surplusResourcesForGoals(s, g_c)$
$pre(\delta^{sp}) = \{pre_1(\delta^{sp}), pre_2(\delta^{sp}), pre_3(\delta^{sp})\}$
$res(\delta^{sp}) = p'(obj1, obj2)$

$\boldsymbol{\delta^{I}(g_c : G) : G}$
$head(\delta^{I}) = identity$
$parameter(\delta^{I}) = g_c$
$pre(\delta^{I}) = \top$
$res(\delta^{I}) = g_c$

---

transformed goal. This transformation results in a new goal with a more specific predicate. Finally, identity transformation (δ^I) has one precondition ($pre_1(\delta^I)$) that always remains true and yields the same goal.

Table 4.2 below represents the algorithm β for goal change. As mentioned earlier, goal change requires a choose functionality $choose()$ to select the number and order of the transformations to reach a specific goal. The inputs to the algorithm β are the agent's state s, current goal g_c, and the list of all transformations possible Δ where the order of the operations is reversed for implementation purposes and the agent stores it as $\hat{\Delta}$. If a specific transform δ^* is in the list of transformations, then the agent applies it to the current goal g_c in the goal agenda $\hat{G}$ or if there are multiple transformations δ^1, δ^2,... δ^m then they are applied to the current goal g_c. For example, δ^* is an insertion transformation. We will talk more about it in the next chapter. Finally, the $choose()$ includes all possible transformations that satisfy the preconditions in the order of their existence in the Δ function.

Table 4.2: The table presents the method to choose one predicate transformation δ^* out of all possible transformations Δ for the goal change operation. The function takes the current state (s), the current goal set (g_c), and the list of all possible transformations (Δ) as input. Finally, the function outputs the modified goal agenda, $\hat{G}$, as the output.

```
$\boldsymbol{\beta(s : S; g_c : G) : G}$
$\hat{\Delta} \leftarrow reverse(choose(s, g_c, \Delta)$
$if \delta^* in \hat{\Delta} then$
  $if \hat{\Delta} = \langle\delta^*\rangle then$            $//insertion \quad only$
    $\hat{G} \leftarrow \{g_1, g_2, ...g_c, ...g_n\} \cup \delta^*()$
    $\beta \leftarrow g_c \wedge \delta^*$
  $else \langle\delta_1, \delta_2, ...\delta_m\rangle = \hat{\Delta} \leftarrow \hat{\Delta} - \delta^*()$       $//insertion \quad plus \quad others$
    $\hat{G} \leftarrow \{g_1, g_2, ...\delta_m(...\delta_2(\delta_1(g_c))), ...g_n\} \cup \delta^*()$
    $\beta \leftarrow \delta_m(...\delta_2(\delta_1(g_c))) \wedge \delta^*()$
$else \hat{G} \leftarrow \{g_1, g_2, ...\delta_m(...\delta_2(\delta_1(g_c))), ...g_n\}$       $//no \quad insertion$
  $\beta \leftarrow \delta_m(...\delta_2(\delta_1(g_c)))$


$choose(s : S; g_c : G, \Delta = \{\delta_1 |, \delta_2, ...\} : poset) : sequence$
$if \Delta = \{\} then\ choose \leftarrow \langle\rangle$
$elseif \forall x \mid x \in pre(\delta_1) \wedge satisfied(x) then$
    $choose \leftarrow \delta_1 \mid choose(\Delta - \{\delta_1\})$
  $elsechoose(\Delta - \{\delta_1\})$
```

4.2 Implementation and Empirical Evaluation of the Goal Change operation

4.2.1 The Construction Domain

We implement the goal change operation in the construction domain. The construction domain is the same as the one mentioned in the goal selection operation, but the goals are different. Here the goal is to build 'stable towers' rather than just building towers. The number of towers and resources present are random. When the agent is out of resources, it chooses to either abandon the goal and wait for more resources or to change its goal to a more generic goal of 'building towers' and continue to work on the generic goal. The goal change operation is decided based on the preferences of the user. If the agent is fine with wobbly building towers rather than sturdy ones, the agent can change its goal. If not, the choice is to abandon the goal until it finds resources. Then, the agent can change its current goal to a generic goal and achieve the maximum possible success in the current version. The goal change also uses two different trees called predicate and object trees. These two are specified while defining the domain. So, according to the transformation requirement, the respective tree is parsed to obtain the result.

The operators in this domain are also the same as before, but our focus will be on stack and stack_mortared. The operator stack_mortared builds a sturdy tower using a mortar block in the middle of regular blocks, and the stack builds a less sturdy tower by placing the blocks on top of each other. The goals in this domain are to build a stable tower always. Let us now look into the experimental setup in the domain.

4.2.2 Experimental Design

In this domain, if the agent is out of resources, then it tries to complete the goal predicate by replacing the goal of ”stable-on” with ”on” (see fig 4.1). Next, we evaluate the performance of MIDCA with and without goal change. First, we assign a score of 2 for ”stack_mortared” each operation and 1 for ”stack” each operation. We then record the scores by varying the number of resources for the same 30 problem sets by counting the ”stack_mortared” and ”stack” operators.

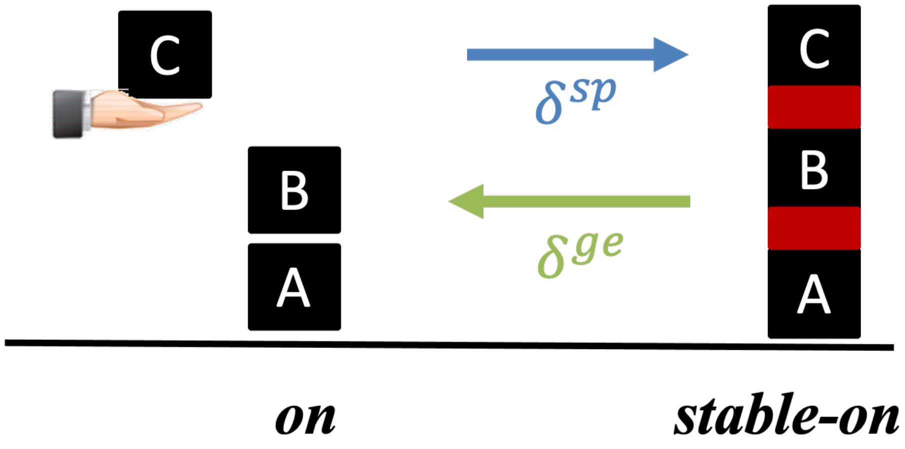

Figure 4.1: Goal Change operation from ”stable-on” to ”on” and vice versa. The image illustrates the generalization and specialization transformations. The black squares with alphabets are blocks and the red rectangle is mortar.

For example, consider the problem set of two towers T1 and T2, with five and six respective heights. The number of mortars present is seven. All the goals here are to construct a stable tower using mortar blocks. To construct the stable tower T1, we need four mortar blocks. After the completion of T1, the agent has three mortar blocks. Now T2 will be constructed, and after stacking four stable blocks, the agent will be out of resources, and the goal is transformed from ”stable-on” to ”on,” and the agent places the remaining two blocks. In this scenario, the score achieved will be that the score for T1 is 4*2=8, the score for T2 is (3*2)+1+1= 8. Therefore, the overall score is score(T1)+score(T2) = 8+8 = 16.

For the same scenario but without goal change, the agent will stop the construction when the agent is out of resources, so the score achieved for T2 will be 0 as it does not complete, and the final score achieved for T1 will be 4*2 =8.

4.2.3 Experimental Results

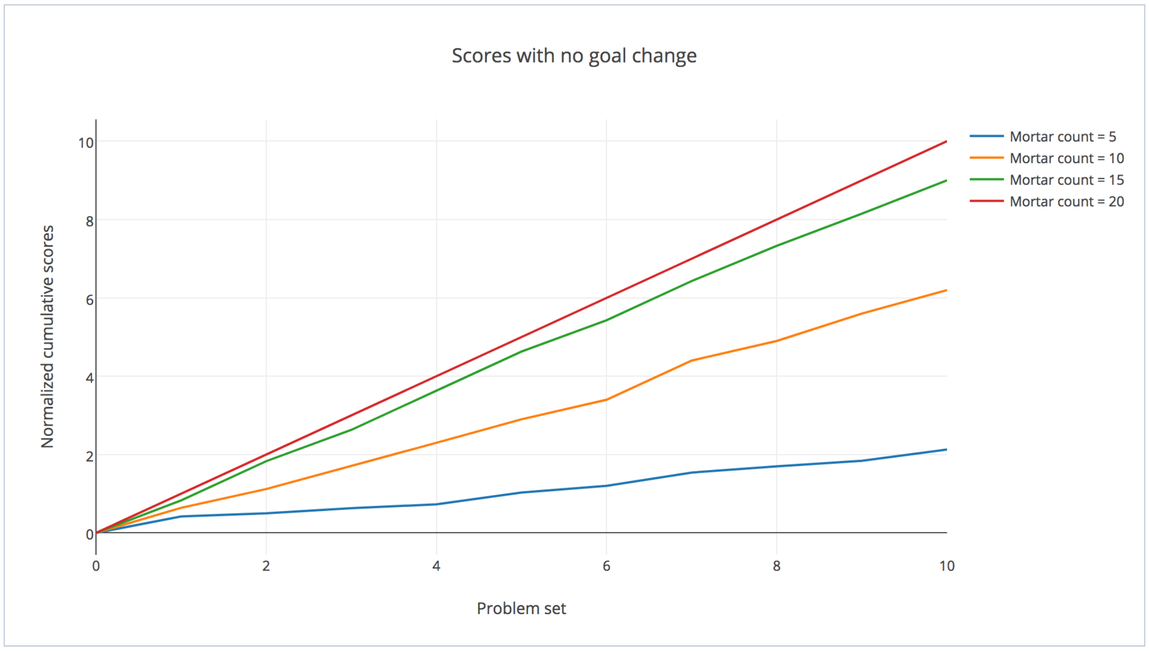

Figure 4.2: The result depicts the agent performance for a different amount of resources without a goal change operation. The X-Axis represents the problem set, and the Y-Axis shows the cumulative score obtained by the agents. In this graph, we vary the number of mortars in increments of five. The initial number of mortars is five, and the final is 20.

Figure 4.2 depicts the scores achieved by MIDCA with no goal change for varying numbers of mortars. Each goal set (problem set) is an average of three trials. The minimum score achieved here is when the available mortar is five and the agent's efficiency is 20%. The maximum score achieved is at the number of mortars being 20. The efficiency achieved is 100% as 20 mortars are sufficient to make all the towers stable.

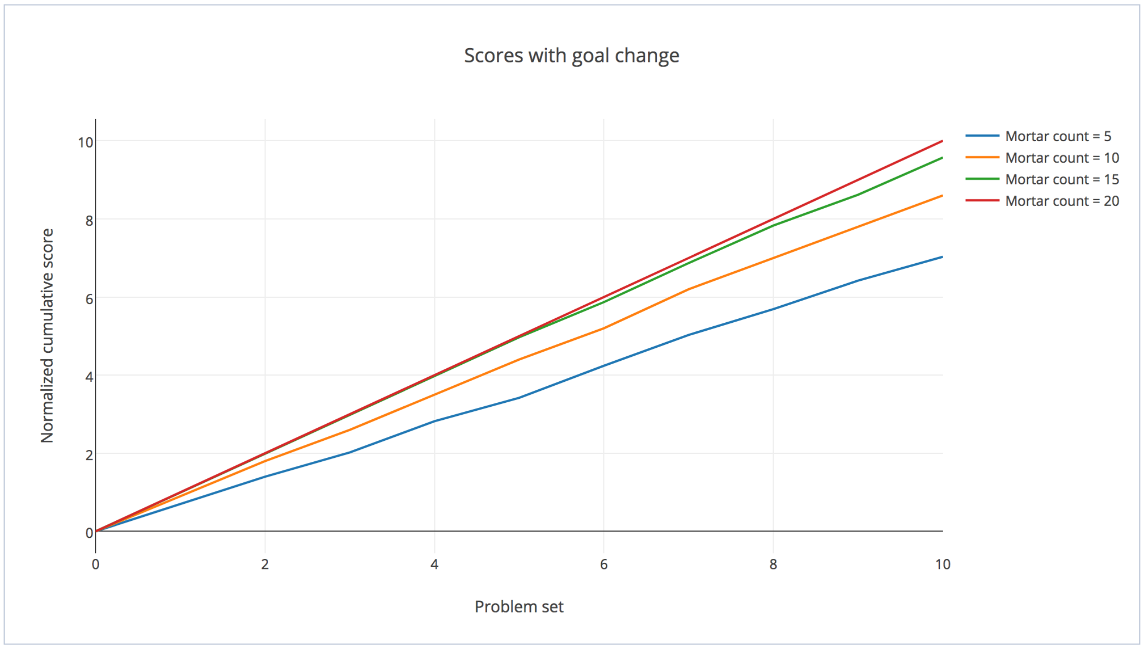

Figure 4.3: The result depicts the agent performance for a different amount of resources with the goal change operation. The X-Axis represents the problem set, and the Y-Axis shows the cumulative score obtained by the agents. In this graph, we vary the number of mortars in increments of five. The initial number of mortars is five, and the final is 20.

Figure 4.3 above depicts the results achieved by MIDCA with goal change for a varying number of mortars. Each problem set is an average of three trials. The minimum score achieved here is when the available mortar is five, and the agent's efficiency was around 70%. Here we can observe that the agent's efficiency with goal change has significantly improved with the same number of resources for the same problem sets. As with both figures 4.2, 4.3 the maximum score achieved is at the number of mortars being twenty, the efficiency achieved at this point is 100% as twenty mortars are sufficient to build all the towers stable.

4.3 Summary of Goal Change

Goal change allows the agent to change its goal. This operation allows the agent to take independent decisions whenever it identifies itself in a position of not achieving the goal. We implement the change operation in the construction domain. The results clearly show an increase in agent's performance with this operation.

In future, we want to implement other predicate transformations apart from generalization, specialization, and identity. In addition, we also want to implement the object transformations to further improve the goal change operation. Apart from the predicate and object transformations, we want to explore other ways to implement the goal change operation. In addition to the construction domain, we implement the goal change in other domains. We present the work in chapter 6.

Goal Formulation

A goal formulation operation allows the agent to generate its own goals. This operation is crucial when the agent cannot achieve its current goal for several reasons. For example, a dynamic environment could render the agent's current goal invalid; an agent with surplus resources and an empty goal set should generate goals and work toward their achievement. Currently, we perform goal formulation when the agent encounters an anomaly (unexpected event). For example, an agent formulates a new goal for anomalies that require a quick response; emergencies like fire generates a goal to put off the fire. In MIDCA, Interpret phase attempts to explain the reasons for an anomaly occurrence. Hence, we implement goal formulation in the Interpret phase of the cognitive layer in MIDCA.

Whenever MIDCA encounters an anomaly, it tries to explain it and formulates a goal if necessary. The current method, which generates the agent's goals, explains every anomaly with an explanation generator called ”Meta-AQUA”[26]. In addition to explaining every anomaly, the explanation generator distinguishes the anomalies that the agent needs to address (problems) from the ones that need not. An agent tackles the problem anomalies by generating a goal with the help of an explanation generator. However, it ignores the regular anomalies. The idea of an agent generating its own goals draws its inspiration from the work done by Cox [23] and in work earlier done by Cox and Ghallab [20][41]. Here, the user's idea of goals provided to the agent is relaxed, and the authors introduce goal formulation as a separate operation. This thesis further elaborates on the goal formulation operation presented in the above works.

5.1 Formal Representation of Goal Formulation

This section presents the formal representation of the goal formulation operation. Table 5.1 depicts the preconditions and results of goal formulation. The first precondition, pre_1, says an explanation exists that provides a cause ω for the anomaly where the expected state does not match the currently observed state s_c.5.2 The second precondition checks if the agent has sufficient resources to generate a goal, which will make the first precondition false. When the agent satisfies all the preconditions, the result will be a new goal to solve the agent's problem.

Table 5.1: The table presents the formal representation of the goal formulation operation. The formulation operation inputs the goal agenda ($\hat{G}$) and output a new goal set (g_n). We use δ^* to represent the goal formulation operation. χ represents an explanation for the anomaly ω. g_n represents the response goal for χ such that the agent is no longer in the anomaly state $\neg\omega$.

$\delta^*(\hat{\boldsymbol{G}} : \boldsymbol{G}) : \boldsymbol{G}$
$head(\delta^*) = formulation$
$parameter(\delta^*) = \hat{\boldsymbol{G}}$
$pre_1(\delta^*) = \exists\, \chi : \omega \rightarrow (s_e \neq s_c)$
$pre_2(\delta^*) = ResourcesFor(g_n)$
$pre(\delta^*) = \{pre_1(\delta^*), pre_2(\delta^*))\}$
$res(\delta^*) = g_n \leftarrow \neg\omega$

Table 5.2 details the first precondition of the table above and represents it formally. It takes the current state (s_c); Background knowledge (Bk); expected state (s_e), Current History (H_c), Remaining plan (π_r) and explanation (χ) and reasons about the inputs by searching through the case-base [45][64][61] and outputs, a Boolean of anomaly, are true or false. Then the explanation backtracks to the cause of the problem and formulates a new goal to eliminate the cause. Cox [21] represents the initial problem in the GDA context and presents a distinction between a classical planning problem and a GDA problem. The paper also puts forth a compelling argument about why a new problem formalism is necessary for the context of GDA. This thesis further modifies the work and builds upon it to refine the

Table 5.2: The table presents the formal representation of the problem in the GDA context. The formulation operation inputs the current state s_c, the background knowledge Bk, the expected state s_e, the current history H_c, the remaining plan π_r, and the explanation χ. The function outputs if the current anomaly is a problem.

$P_{gda} : (s_c, Bk, s_e, H_c, \pi_r, \chi)$
$s_c = \gamma(\pi_c, s_0)$
$Bk = (\Sigma, \Delta)$
$s_e \in S$
$H_c = (\pi_c, g_c)$
$\pi_c \circ \pi_r = \pi_{g_c}$
$\chi = SearchAndRetrieveSimilarCaseFromCaseBase$
$res(P_{gda}) = \top \vert \bot$

formal definition.

5.2 Implementation and Empirical Evaluation of the Goal Formulation Operation

We implement the goal formulation operation in two domains: the naval mine clearance domain and the labor relations domain.

5.2.1 The Naval Mine Clearance Domain

Figure 5.1 depicts the naval mine clearance domain [45, 43, 46, 47, 61, 64]. We build the domain using the Mission Oriented Operating Suit-Interval Programming (MOOS-IvP) framework, a middleware used to simulate the behavior of various underwater platforms. The agent's goal in this domain is to prepare a harbor for use during maritime operations. To achieve this goal, an agent needs to conduct mine clearance activities to ensure the safe passage of ships as they transit between the sea and the port in the harbor. Usually, we establish a network of safe shipping lanes to reduce the size of the search area within the harbor. The area used by the ships to traverse the sea is called a Q-route [43, 47, 64]. The

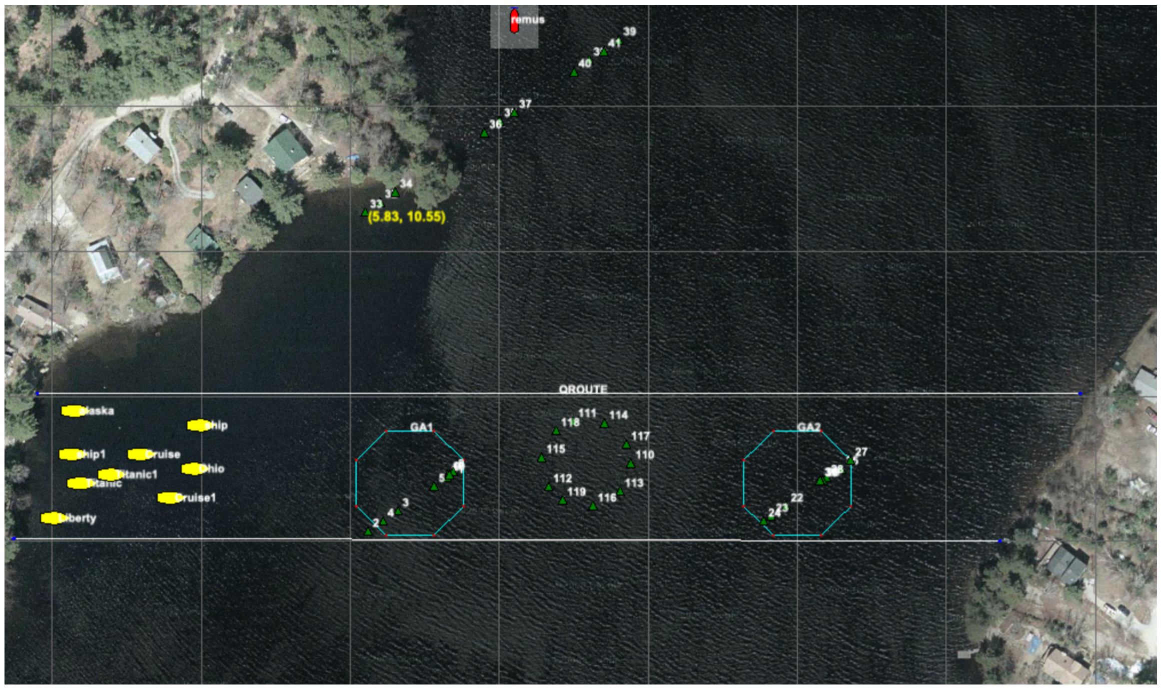

Figure 5.1: The figure depicts the naval mine clearance domain. The agent is the red cylindrical object at the top center of the image. The yellow objects on the left are the freight ships. The area between the two parallel lines is the Q-route. The two octagonal shapes are the green areas.

Q-route in the figure 5.1is the area between the parallel lines. The two octagonal areas are called Green-Area1(GA1) and Green-Area2(GA2), where the agent expects the mines to be present. So, the agent assumes that mines are present only in the two areas. In turn, this leads to the assumption that all mines outside of the green areas are anomalies. In this domain, the agent is an Autonomous Underwater Vehicle (AUV) named Remus-100. It is the red cylindrical object on the upper left side of the image. The shaded area around it is a sonar sensor; Remus detects mines only if they fall under that sensor range. The yellow objects are ships, and the green triangles are mines. The goals in this domain are to clear the mines in GA1 and GA2.

The operators in this domain include fast-survey; slow-survey; clear-mines. The fast survey allows the agent to move quickly in the areas where it does not anticipate mines. The slow survey allows the agent to move carefully in the regions where it expects mines. Finally, when detected, the clear mines operator removes mines from the location.

5.2.1.1 Experimental Design: The Naval Mine Clearance Domain

To evaluate both the domains, we introduce two new agents along with our GDA agent. These are eager agents and baseline agents. They each respond differently to anomalies. Anomaly detection in the GDA agent works on select anomalies which the agent perceives as problems. In contrast, the eager agent addresses all anomalies that it encounters. The baseline agent plans only for its original goals and ignores all anomalies. We assess the three agents' performance, present the results in different environments, and average the results for ten different runs for each mine scenario.

In the naval mine clearance domain, we calculate the performance of the agents based on the number of ships that reach the other side of the harbor safely. Various mine density scenarios exist with six ships and three mine patterns: low, medium, and high. We also include deadlines ranging from 0 to 2 seconds in the domain with increments of 0.5 seconds. These deadlines specify the time gap between the agent starting from home to clear the mines and the ships starting their journey from one side of the shore to the other. Note that the seconds indicate the simulation time, not real-world time. So, the maximum score that an agent can achieve in this domain is 6.

5.2.1.2 Empirical Results: The Naval Mine Clearance Domain

Figure 5.2 shows the scores achieved by the three different agents in all mine density (average of low, medium, and high density) scenarios for the varied deadlines. The X-axis depicts the delay with which the ships start, and the Y-axis indicates the number of ships that safely traverse the Q-route. Here, when we start looking at the values from the left side of the graph, at the delay of 0, very few ships could traverse the Q-route successfully for all three agents. Those that were able to reach the other side could cross the Q-route in the low mine density scenarios, while very few or no ships made it across in the medium and high mine density scenarios.

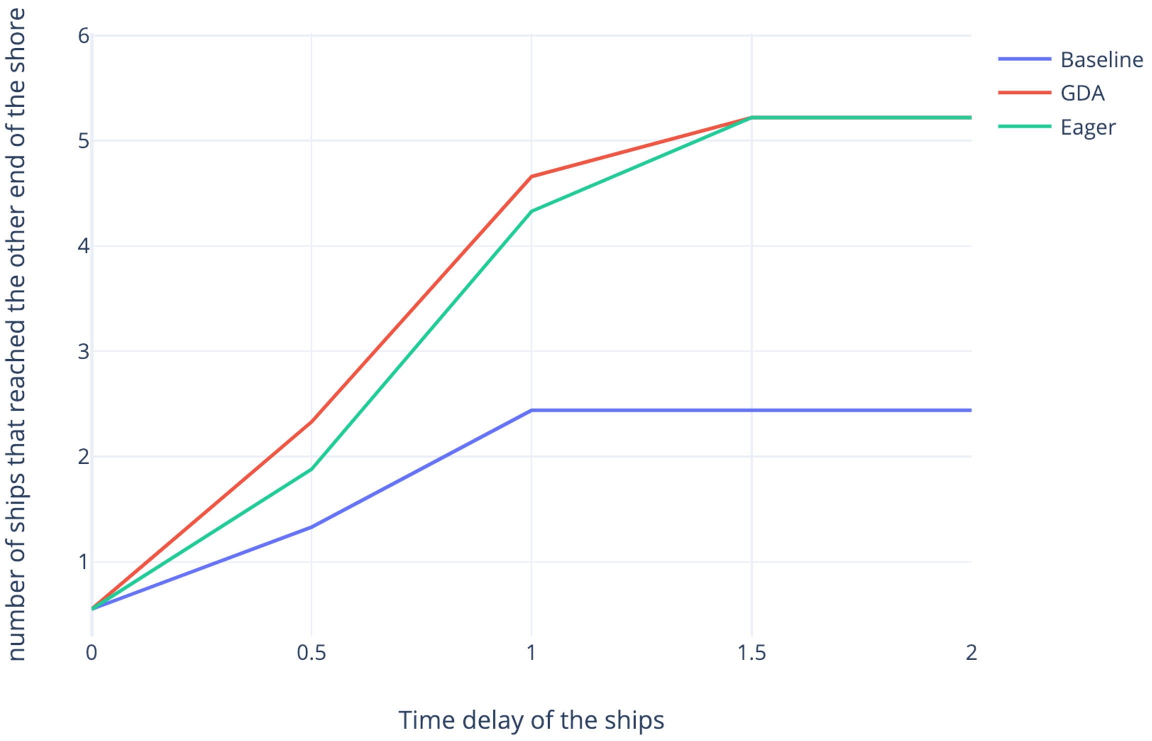

Figure 5.2: The figure depicts the results for the goal formulation operation in the naval mine clearance domain. We compare the performance of the GDA agent with the baseline and the eager agents. The X-Axis represents the time delay of the ships, and the Y-Axis shows the number of ships that reach the other end of the Q-route.

To understand what a delay of 0.5 seconds means, consider what each agent can accomplish within that time frame in a characteristic scenario. For example, after 0.5 seconds, the baseline agent clears mines in GA1 and is on its path to clear the mines in GA2. At the same point in time, the GDA agent cleared all mines in GA1 and some mines within the Q-route; the eager agent cleared mines outside of the Q-route and in GA1.

After 1 second, the baseline agent cleared the mines in green areas and headed towards home. The GDA agent cleared some mines within the path from GA1 to GA2 and some mines within GA2. In comparison, the eager agent works on the mines within the Q-route after clearing the ones in GA1.

In these conditions (delay of 0.5 and 1 seconds), the difference between the performance of the GDA and eager agents does not seem very large. Because the average of the various mine density scenarios also contains low mine density fields where the two agents perform almost identically since the eager agent only has a few mines to clear outside of the Q-route. However, the difference in performance for medium and high mine density scenarios is two ships.

All agents have performed all clearance tasks intended at a delay of 1.5 seconds or greater; thus, performance does not change for delays greater than 1.5 seconds.

5.2.2 The Labor Relations Domain

Second is the labor relations domain; employee strikes at various institutions motivate this domain. Analogous to real institutions, this domain describes a virtual institution where the agent acts as institute head. Several employees and customers exist in this domain. The institution starts with an initial reputation value and a budget represented by numbers. The agent's goal is to enact policies, where implementing a policy requires a known fixed amount of budget and increases the institutes' reputation value by a fixed amount. Disagreements about enacted policies will arise between the head of the institute and the employees with certain intensity values. High-intensity disagreements may lead to a strike. Intensity values vary for each disagreement and are unpredictable. Disagreements can be resolved by negotiating with employees. Negotiating to solve a disagreement also costs money, so the budget varies with the intensity of the disagreement. Moreover, negotiations also decrease the reputation value as a function of intensity. So, the goals are to pass policies, and the anomalies are disagreements. Anomalies with higher intensity values are considered a problem. This domain is not related to trading and marketing agents in artificial intelligence, and we do not simulate it using third-party software.

The operators in this domain include make-policy, inform-policy, resolve-disagreement, ignore-disagreement, and others. Similar to other domains, each operator performs distinct action. For example, make policy designs a policy for the agent to implement, inform policy shares the policy with the employees, resolve disagreement tries to negotiate with employees and tries to resolve disagreement and ignore disagreement ignores the disagreement and does nothing about it.

5.2.2.1 Experimental Design: The Labor Relations Domain

Similar to the previous domain, we introduce two new agents along with our GDA agent. These are eager agents and baseline agents. They each respond differently to anomalies. Anomaly detection in the GDA agent works on select anomalies which the agent perceives as problems. In contrast, the eager agent addresses all anomalies that it encounters. The baseline agent plans only for its original goals and ignores all anomalies. We assess the three agents' performance, present the results in different environments, and average the results for ten different runs for each mine scenario.

To assess the performance of the three agents in the labor relations domain, we compare the reputation values of all agents after they implement a certain number of policies. There are some numerical values in this domain. The initial reputation is 500, and the total budget is \$4000. Implementing any policy reduces the budget by \$25. The intensity value of a disagreement is a random number between (1, 100). When a disagreement arises, the employees can demand a budget amount which is a random number between (1, 25). Providing any amount within 40% to 60% of the amount demanded by employees solves the disagreement.

If there is no disagreement when an agent implements a policy, then the institution's reputation is increased by five. However, if the agent encounters a disagreement, it has two options: solve the disagreement or ignore it and strictly adhere to the initial policy. In the first option that addresses the disagreement, the reputation is neither decreased nor increased, i.e., the reputation change is zero. If the agent does not address the disagreement, then the reputation value is decreased as a function of intensity value. So, if the intensity is $<=34$ then Rep = -Int/100 and, if the intensity is $>=35$ then Rep = -[(n+2)*Int]/100 where n = integer((I-35)/5). The integer() acts as a rounding function. There is an interesting value-added to the budget after implementing every 50 policies with a rate of 2.5%. Reputation values can become negative if the agent does not address disagreements.

5.2.2.2 Empirical Results: The Labor Relations Domain

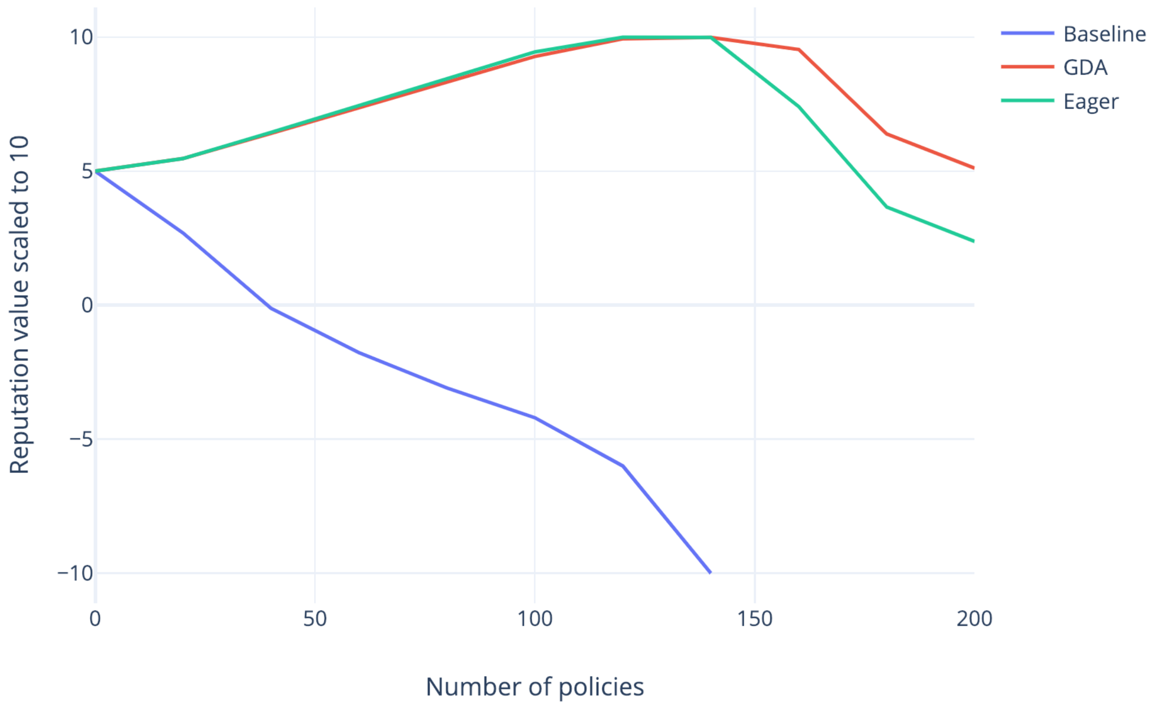

Figure 5.3: The figure depicts the results for the goal formulation operation in the labor relations domain. We compare the performance of the GDA agent with the baseline and the eager agents. The X-Axis represents the number of policies, and the Y-Axis shows the reputation value for the institution.

Figure 5.3 shows the reputation achieved by the three different agents over 200 policies. The X-axis depicts the number of policies implemented, and the Y-axis indicates the reputation value scaled to 10. All the agents have an initial reputation of five. Each point on the lines contains an average of 20 policies. The reputation value is cumulative and can reach a maximum value of 10. In this experiment, if any agent is out of budget, it starts to behave as a baseline agent. The reason is the debt should be as minimum as possible when the agent implements all policies. Starting with the baseline agent, after completing 50 policies, the baseline agent already has a negative reputation because it does not address any of the disagreements, so the reputation value drops and keeps on decreasing monotonically.

The GDA agent gets a little behind the eager agent from 50 to 120 policies because the GDA agent will not address the anomalies with lower intensity values. In contrast, the eager agent addresses all the anomalies and spends its budget on every anomaly. However, this means the eager agent is out of its budget much sooner than the GDA agent.

The eager agent is out of budget at 140 policies and starts behaving as a baseline agent, thus dropping its reputation. The GDA agent preserves its budget and continues to increase its reputation value by around 150. The baseline agent's reputation value is still sinking to much lower values, but we adjust the scale to make the negative values only visible up to -10. These results indicate that the GDA agent should perform better over time and maintain a higher reputation than the eager agent by a significant amount. The GDA agent under-performed by a negligible amount for some time compared to an eager agent because of the higher amount of resources. A smart agent is not needed if the amount of resources present is infinite, but this is not realistic. As long as resources have a limit, there will be a need for the GDA agent to use them sustainably.

5.3 Summary of Goal Formulation

Goal formulation allows the agent to formulate its own goals without the help of an external user. However, it is one of the hardest goal operations to implement because even we humans are unsure about the origination of goals. In this work, we generate goals with the help of case based explanations[61, 43] when unexpected events occur. We implement the goal formulation in the mine clearance domain and the labor relations domain.

We can look into other methods or scenarios of doing this operation. For example, one of the methods includes the agent having plenty of resources and does not have any goals to achieve. Instead, it can develop goals that the agent might need soon.

Interaction of Goal Selection, Goal Change and Goal Formulation

From chapters 3, 4, and 5, it is evident that goal operations increase the agent's performance in dynamic environments. We implement all of the goal operations individually in MIDCA. But, more often, in dynamic environments, multiple goal operations co-occur. For example, consider an agent working toward selecting its current goal g_c when a fire erupts in an area nearby. In this situation, the agent must decide between selecting its current goal g_c or abandoning selection to create a new goal g_n to tackle the fire. Therefore, the next step is to integrate all of the goal operations. This step requires addressing two of the most important issues. First is the knowledge about how goal operations interact with one another, and the second is a rich, complex domain in which all operations co-occur.

The focus of this thesis is to address the first issue and make use of the complex domains to test the claim. To learn how the three-goal operations interact with each other and why the knowledge about their interaction is important, let us take a look at the example scenario in the next section.

6.1 Example Scenario Depicting the Interaction of Goal Operations

Let us consider a scenario in the naval mine clearance domain [43, 47, 61] and look at it in detail to understand the interaction of the three-goal operations.

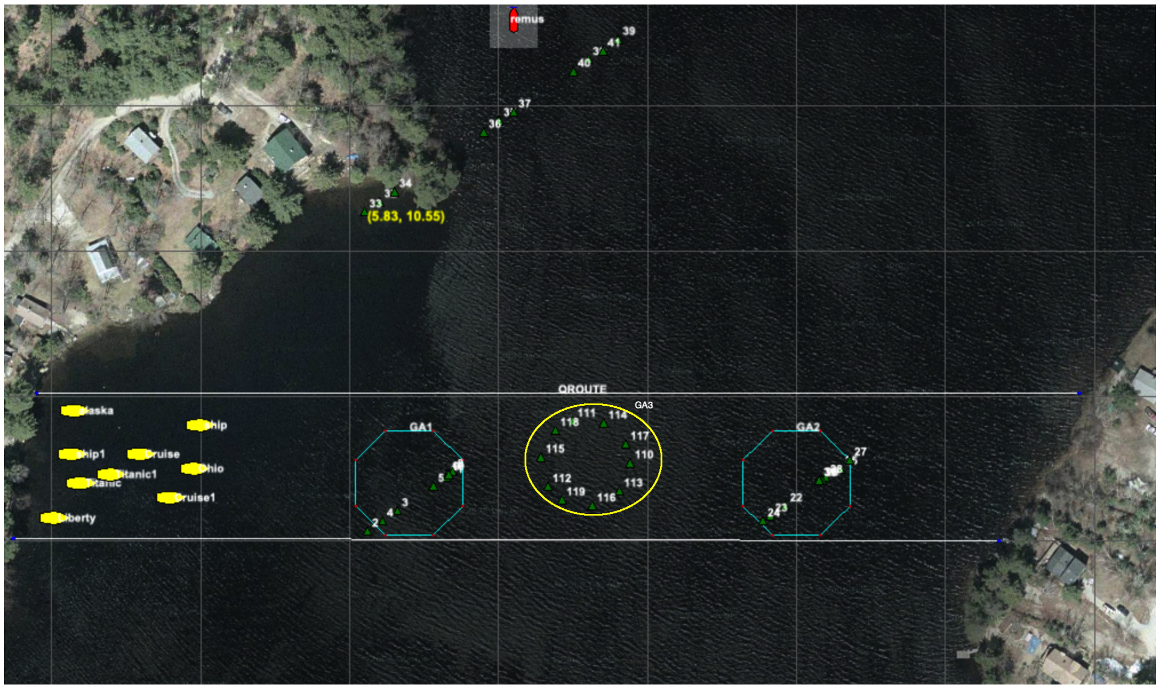

Figure 6.1: The figure depicts an example scenario in the naval mine clearance domain. The agent is the red cylindrical object at the top center of the image. The yellow objects on the left are the freight ships. The area between the two parallel lines is the Q-route. The two octagonal shapes are the green areas. In addition, the yellow circular region is the new green area with mines.

Figure 6.1 depicts a problem in this domain. The initial goal set of the agent is $\hat{\boldsymbol{G}} = \{g_1, g_2, g_3\}$, where $g_1 = \mathit{cleared\_mines}(GA1)$, $g_2 = \mathit{cleared\_mines}(GA2)$ and $g_3 = \mathit{at\_home}(\mathit{Remus})$. The first goal of the agent is to clear mines in GA1, the second is to clear mines in GA2, and the third is to be back home. Since our agent is smart enough to ignore the anomalies that are not problems, it ignores the mines it encounters while traveling from its initial position to GA1 as they do not threaten the shipping. Let us assume that it accomplished g_1 and is headed toward GA2 to achieve g_2. The agent now encounters new mines on its path to GA2. The agent cannot ignore these mines because the ships use the Q-route for transportation purposes, and hence the agent should now generate a goal to handle the situation. Currently, MIDCA just generates a single goal of clearing the encountered mines in Q-route $\delta^* = g_4 = cleared\_mines(m118)$. Apart from this one goal, there is a possibility to formulate several other goals with further reasoning. They are as follows.

- $\delta^* = g_5 = apprehend\_enemy(mission)$. This agent formulates this goal when it tries to reason about the root cause of the anomaly. Let us assume that based on the pattern of the mines that the agent encounters, it identifies that a ship might have laid the mines. It can now formulate a new goal of apprehending the enemy ship to reduce the occurrence of anomalies in the future.

- $\delta^* = g_6 = cleared\_mines(GA3)$. This agent formulates this goal when it tries to reason about the number of anomalies present in the Q-route. After looking at the pattern of mines that it encountered, it can develop a new area GA3 where the agent expects mines to be present.

If we look at the goals g_4 and g_6, they are not two different goals. Goal g_6 can subsume g_4, which means that the agent can effectively change g_4 to g_6 based on its resources. So, only one goal between those two should go into the goal agenda. So, the new goal agenda now looks like $\hat{\boldsymbol{G}}' = \{g_4, g_5, g_2, g_3\}$ if the agent performs the goal formulation operation first and it has a sufficient amount of time to reason about the world.

Let us elaborate on the ordering of the goal operations to understand the last statement in the above paragraph and realize the importance of interaction between the goal operations. The goal agenda after achieving g_1 looks like $\hat{\boldsymbol{G}} = \{g_2, g_3\}$. It is the goal agenda we consider for the scenarios presented below. There is a possibility of two-goal formulations (i.e., g_4, g_5), one goal change (i.e., g_4 to g_6) and goal selection whenever there are more than one goal. Several assumptions are required before we proceed: the goal selection follows FIFO strategy; formulation will only generate one goal (g_4), and goal change will only change the current goal. Consider the following scenarios with different ordering of goals:

- Scenario1. Let us consider the initial situation where the agent formulates a goal and then performs selection then change. (i.e., $\delta^{\Delta}(\delta^{se}(\delta^{*}(\hat{\boldsymbol{G}})))$)
  $\delta^{*}(\hat{\boldsymbol{G}}) = \{g_4, g_2, g_3\} \; //formulates\ the\ goal\ g_4$
  $\delta^{se}(\delta^{*}(\hat{\boldsymbol{G}})) = \{g_4, suspended(g_2)\}//suspends\ the\ current\ goal\ g_2\ and\ selects\ g_4$
  $\delta^{\Delta}(\delta^{se}(\delta^{*}(\hat{\boldsymbol{G}}))) = \{g_6, suspended(g_2)\}//g_4\ is\ changed\ to\ g_6$
  $\implies$ the agent achieves g_6 and later resumes to work on the suspended goal g_2.

  Reason: The goal selection is performed on the goal agenda after the new goal g_4 is formulated. The selection then selects g_4 and then the goal change is performed on the current goal g_4, which will transform g_4 to g_6.

- Scenario2. Let us consider the situation where the agent formulates a goal and then performs change and later selects the goals.(i.e., $\delta^{se}(\delta^{\Delta}(\delta^{*}(\hat{\boldsymbol{G}})))$)
  $\delta^{*}(\hat{\boldsymbol{G}}) = \{g_4, g_2, g_3\}//formulates\ the\ goal\ g_4$
  $\delta^{\Delta}(\delta^{*}(\hat{\boldsymbol{G}})) = \{g_4, g_2, g_3\}//current\ goal\ g_2\ is\ changed\ to\ g_2$
  $\delta^{se}(\delta^{\Delta}(\delta^{*}(\hat{\boldsymbol{G}}))) = \{g_4, suspended(g_2)\}//selection\ suspends\ g_2\ and\ selects\ g_4$
  $\implies$ the agent achieves g_4 and later resumes to work on the suspended goal g_2.

  Reason: The goal change operation is performed on the current goal. In this situation, the current goal will still be g_2 because g_4 is not selected yet. So, the goal change operation just performs an identity transform on g_2 and then goal selection selects g_4.

- Scenario3. Let us consider the situation where the agent performs a change and then performs goal formulation and finally selects the goal.(i.e., $\delta^{se}(\delta^{*}(\delta^{\Delta}(\hat{\boldsymbol{G}})))$)
  $\delta^{\Delta}(\hat{\boldsymbol{G}}) = \{g_2, g_3\}//does\ nothing\ because\ no\ goal\ is\ selected$
  $\delta^{*}(\delta^{\Delta}(\hat{\boldsymbol{G}})) = \{g_4, g_2, g_3\}//inserts\ formulated\ goal\ g_4$

$\delta^{se}(\delta^{*}(\delta^{\Delta}(\hat{\boldsymbol{G}}))) = \{g_4, suspended(g_2)\} // suspends\, the\, current\, goal\, g_2\, and\, selects\, g_4$

$\implies$ the agent achieves g_4 and later resumes to work on the suspended goal g_2.

Reason: The agent changes its current goal g_2 and then formulates a goal a g_4 and finally selects the g_4 suspending g_2. The agent will first work on g_4 and after achieving it moves onto the suspended g_2.

- Scenario4. Let us consider the situation where the agent selects a goal and then performs goal formulation and change operations. (i.e., $\delta^{\Delta}(\delta^{*}(\delta^{se}(\hat{\boldsymbol{G}}))))$

  $\delta^{se}(\hat{\boldsymbol{G}}) = \{g_2\} // selects\ g_2$

  $\delta^{*}(\delta^{se}(\hat{\boldsymbol{G}})) = \{g_4, g_2\} // inserts\ formulated\ goal\ g_4$

  $\delta^{\Delta}(\delta^{*}(\delta^{se}(\hat{\boldsymbol{G}}))) = \{g_4, g_2\} // changes\ g_2\ to\ g_2$

  $\implies$ the agent achieves g_2 and later moves to the next goal g_4 in the agenda.

  Reason: The agent selects g_2 and then the agent will formulate g_4 but will still work on g_2 because there is no selection operation to select g_4. Goal change changes g_2 to g_2. The agent will work on g_2 and move to the next goal g_4.

- Scenario5. Let us consider the situation where the agent selects a goal and then performs goal change and finally formulates a goal.(i.e., $\delta^{*}(\delta^{\Delta}(\delta^{se}(\hat{\boldsymbol{G}}))))$

  $\delta^{se}(\hat{\boldsymbol{G}}) = \{g_2\} // selects\ g_2$

  $\delta^{\Delta}(\delta^{se}(\hat{\boldsymbol{G}})) = \{g_2\} // changes\ g_2\ to\ g_2$

  $\delta^{*}(\delta^{\Delta}(\delta^{se}(\hat{\boldsymbol{G}}))) = \{g_4, g_2\} // inserts\ formulated\ goal\ g_4$

  $\implies$ the agent achieves g_2 and later moves to the next goal g_4 in the agenda.

  Reason: The agent selects the goal g_2 and then the agent performs goal change on g_2 which results in g_2. After achieving it, agent formulates a new goal g_4.

- Scenario6. Let us now consider the situation where performs a goal change operation and then tries to select a goal and then formulates a goal.(i.e., $\delta^{*}(\delta^{se}(\delta^{\Delta}(\hat{\boldsymbol{G}}))))$

  $\delta^{\Delta}(\hat{\boldsymbol{G}}) = \{g_2, g_3\} // does\ nothing\ because\ no\ goal\ is\ selected$

  $\delta^{se}(\delta^{\Delta}(\hat{\boldsymbol{G}})) = \{g_2\} // selects\ g_2$

$\delta^{*}(\delta^{se}(\delta^{\Delta}(\hat{\boldsymbol{G}}))) = \{g_4, g_2\} // inserts\ formulated\ goal\ g_4$

$\implies$ the agent achieves g_2 and later moves to the next goal g_4 in the agenda.

Reason: The agent does not do a goal change operation because there is no goal selected by the agent, so it outputs a null and then selects g_2 and formulates g_4 after it achieves g_2.

In all of the six scenarios above, with only one fixed goal to generate, only one goal change operation, and a fixed selection strategy, there are three distinct outcomes. This clarifies that ordering of the goal operations matter and is worth spending time on to follow the best possible choice. Among the six scenarios presented above, scenario1 is the best for the context above, as it clears most mines by using fewer resources. Scenario2 and scenario3 yield the same result, and they do not clear all the mines in the Q-route but clear the mines m118 the agent encounters within its path and move onto the next goals. The final set of scenarios is scenario4, scenario5, and scenario6, which is the worst because the agent clears the mines in GA2 and then comes back to clear the mine m118. This set uses more resources than the amount required. This raises the question, **How can a system manage interactions among multiple-goal operations?**. This question stands as a central idea to this thesis.

6.2 Goal Management when Multiple Goal Operation Co-occur

Goal operations can co-occur in multiple scenarios. However, the thesis focuses on scenarios when the agent discovers an anomaly, which affects the agent negatively. Also, the algorithm presented focuses mainly on goal selection, goal change, and goal formulation. But, the algorithm is generic enough to include other goal operations, to demonstrate which we provide an example with goal delegation. Similar to other goal operations, we implement the algorithm in MIDCA [63, 62].

We argue that the following three classes of factors are very useful for enabling the agent to pick one goal operation among multiple operations.

- **Anomaly effects:** When the agent is in an anomalous situation, it must reason about the anomaly. As mentioned, the agent only considers the negative effects of the anomaly.
- **Goal priority:** The agent must have a general idea of the importance of every goal in its goal agenda $\hat{G}$.
- **Resource availability:** The agent must also consider the number of resources required for all the goals while deciding on goal operations. Therefore, it must have a consumption estimation of the resources. In addition, it must also continuously update its expectation according to actual observation.

The agent uses the above three-factor classes and decides on a goal operation to pursue. The following few subsections dive deep into each of these factors.

6.2.1 Effects of an Anomaly on the Agent

An unexpected event in a dynamic world can affect the agent positively or negatively. As mentioned previously, the thesis focuses only on the negative anomalies. To study the negative effects of the anomaly, we further classify the effects into three categories: negative effects of the anomaly on agent's goals, negative effects of the anomaly on agent's health, and the number of times the anomaly repeats (which is not a direct negative effect, but it can estimate the cumulative negative effect of the anomaly in future). So far, we have identified three factors to study the negative effects of an anomaly on the agent. To estimate the negative effects the agent must have a model to approximate the values. After building models for each factor, we translate the outputs to qualitative factors. Currently, the qualitative factors we use are high and low. We present the importance of such qualitative classification in later sections.

Figure 6.2 presents the three factors. Specifically, the agent looks at the negative effects of an anomaly on the agent's goals, the negative effects of an anomaly on the agent's anomalies, and the anomaly repetition factor. The next paragraphs subsections the modeling of each of the factors.

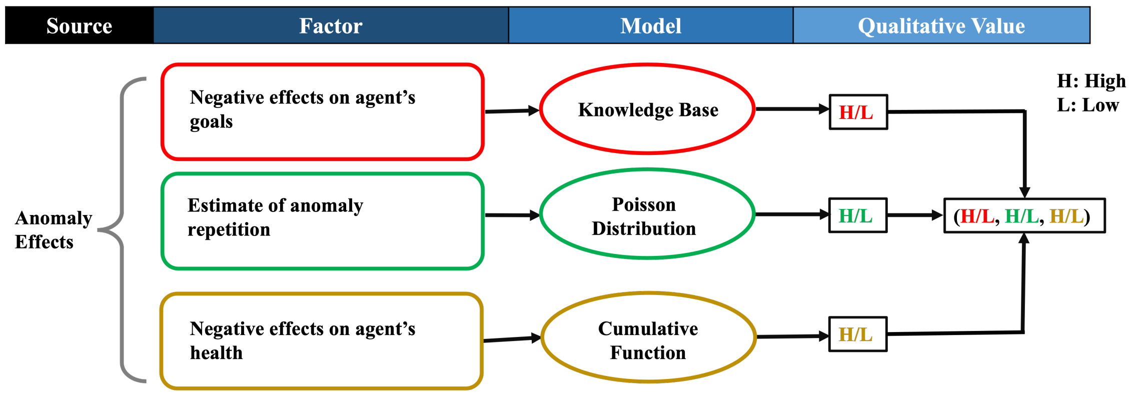

Figure 6.2: The figure represents the negative effects of the anomaly on the agent. We further divide this class into three sub-classes: the negative effects on the agent's health, the negative effects on the agent's goals, and the number of times an anomaly repeats. We model each sub-classes using various approaches (example: using knowledge base). Finally, we classify the output as quantitative values high or low.

6.2.1.1 Negative Effects on Agent's Goals

To model the negative effects of the anomaly on the agent's goals, we use a knowledge base. If the anomaly occurred previously, the agent finds a case relating to the negative effects in the knowledge base and retrieves the case. After retrieval, the agent then classifies the negative effects as high or low. If the agent encounters the anomaly for the first time, the agent considers the anomaly to be a higher risk anomaly. Later, based on its encounter, it creates a new case for the new anomaly and stores it in the knowledge base. The figure 6.3 presents the negative effects on the agent's goals.

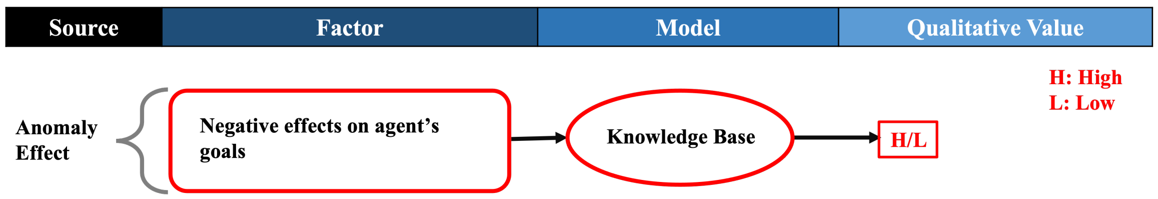

Figure 6.3: The figure represents the negative effects of the anomaly on the agent's goals. We model this using a knowledge base. Finally, we classify the output as quantitative values high or low.

6.2.1.2 Negative Effects on Agent's Health

To model the negative effects of the anomaly on an agent's health, we created a cumulative numeric function 6.1 to track it. We assume that the agent's health starts at a maximum value of 100. The cumulative function keeps track of the agent's health by considering many sub-factors that affect health. For example, consider that n is the total number of sub-factors affecting the agent's health, and h_i is the numeric value of health affected for each factor. Then, we define the overall cumulative function as:

$$H = \Sigma_{i=1}^{n}(h_i) \tag{6.1}$$

Every sub-factor affecting the agent's health is different. Hence, we can define each h_i using one of the basic functions: simple linear, quadratic, exponential, or logarithmic function. For example, the actions an agent performs to achieve its goal linearly reduce its health (over time, the battery of the agent reduces). Therefore, for one goal achieved, we define the factor by which the health decreases as $h_{goal\_number\_1} = x_1 * g_1$; where x is a value between 0 to 1, and g_1 is also a domain-specific value set by a human expert for each goal type. Of course, the exact value of x for each goal differs. Currently, a human expert provides the x value. So, for 'm' goals, the sub-factor (performing actions) would be a simple summation given in (6.2).

$$h_1 = h_{actions} = \Sigma_{y=1}^{m}(h_{goal\_number\_y}) = \Sigma_{y=1}^{m}(x_y * g_y) \tag{6.2}$$

Consider a second sub-factor fire. Fire affects the agent's health exponentially. Hence, similar to a linear function, we should use an exponential function [1] to model the negative effects of fire. Let us indicate the second sub-factor in the equation 6.3where t is the time.

$$h_2 = h_{fire} = e^t \tag{6.3}$$

For example, let us consider the two sub-factors ($h_{actions}$, h_{fire}) defined and substitute them in the cumulative function 6.1. The sequence of steps below presents the calculation for the agent's health.

$$H = \Sigma_{i=1}^{2}(h_i) = h_1 + h_2$$
$$H = h_{actions} + h_{fire}$$
$$H = \Sigma_{y=1}^{m}(h_{goal\_number\_y}) + e^t$$
$$H = \Sigma_{y=1}^{m}(x * g_y) + e^t$$

We perform similar calculations for sub-factor to include them in the cumulative function. We calculate the agent's health value before and after the anomaly. If the difference looks higher than an expert set threshold, we classify the health affected as high; otherwise, it remains low. The figure 6.4 presents the negative effects on the agent's health. For the specific values used for implementation, look at the appendix A and B.

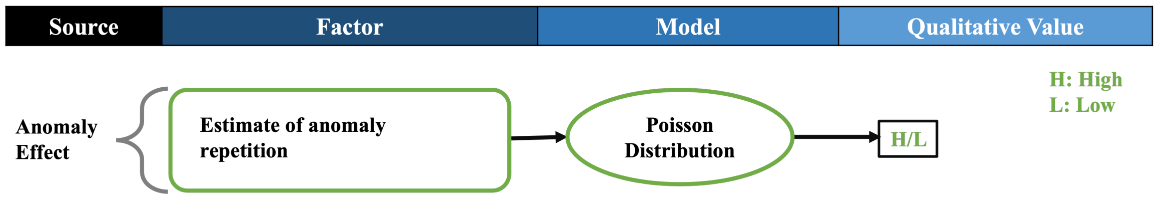

Figure 6.4: The figure represents the negative effects of the anomaly on the agent's health. We model this using a cumulative function. Finally, we classify the output as quantitative values high or low.

[1] e is the Euler's number

6.2.1.3 Anomaly Repetition

Finally, one of the last categories that we need to consider is to check the frequency of the anomaly repetition. Although the anomaly might not affect the agent's goal or health, if it occurs very frequently, the agent should possess knowledge about the anomaly because it might be of interest to the agent in the future. Since the occurrence of an anomaly is an independent event, we use Poisson distribution to predict the frequency of occurrence of the anomaly by storing its past occurrences.

$$P\left(x\right) = \frac{e^{-\lambda}\lambda^{x}}{x!} \tag{6.4}$$

Equation (6.4) depicts the Poisson distribution function. Where λ is the mean of the anomaly occurrences over time, and x is the number of occurrences. The plot of the Poisson values obtained for several x values looks like a bell curve. We then determine the peak value of the bell curve to be the frequency of anomaly repetition. If the determined frequency value obtained exceeds a certain threshold set by an expert, the agent classifies the outcome as high else low. Figure 6.5 presents as the anomaly repetition factor. Look at the appendix A and B.

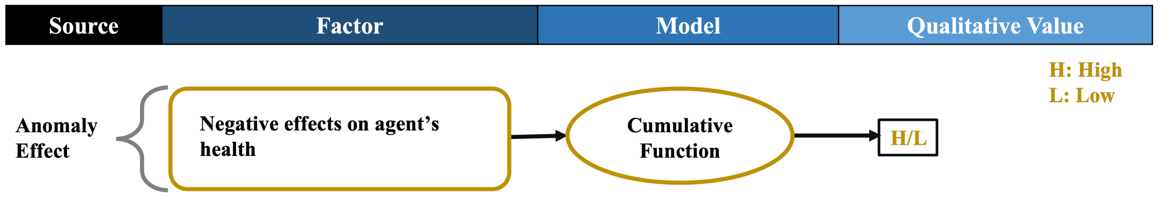

Figure 6.5: The figure represents the number of times an anomaly repeats. We model this using Poisson distribution. Finally, we classify the output as quantitative values high or low.

6.2.1.4 Summary of Negative Effects of an Anomaly on the Agent

This section summarizes the negative effects of the anomaly on the agent. It presents the three anomaly effects we considered: negative effects on agent's goals, health, and anomaly repetition estimate. It also depicts the qualitative classification of the output, high or low.

6.2.2 Importance of the Agent's Goals

The second major factor we used in the algorithm to select goal operations is determining the priority of each agent's goal. Every agent needs to understand the importance of each goal it achieves or tries to achieve. To capture such a factor, we modeled the goal types of [93]. Here Schank broadly categorizes goals into several types and prioritizes one type over another. We used three goal types that best fit our decision algorithm out of all the goal types. Specifically, we consider *crisis:* goals generated in response to a crisis event; *preservation:* goals that help in the agent's self-preservation or resource preservation; and *achievement:* goals provided to the agent to achieve a task or reach a goal state. In general, we prioritize crisis goals over preservation and preservation over achievement goal types. We name these three factors qualitatively as C, P, A. C refers to a crisis, P refers to preservation, and A refers to achievement. The figure 6.6 presents the goal types in pictorial format.

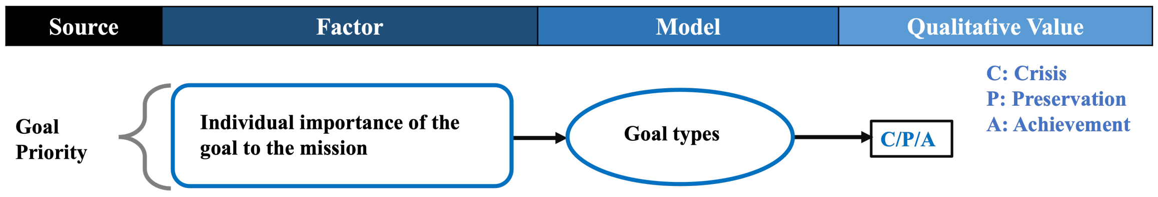

Figure 6.6: The figure represents the priority for each goal using its goal type. There are three goal types[93]: crisis, preservation, and achievement. Crisis goals are the goals generated in response to a problem, preservation goals are the goals to preserve an agent's health or resources, and achievement goals are the regular goals the agent achieves. Therefore, we prioritize crisis over preservation over achievement.

6.2.3 Estimation of Agent's Resources:

The third major factor we use in the algorithm to choose goal operations is to continuously check the resource availability for goal achievement. To model this factor, we use a knapsack algorithm. Such an estimation algorithm is crucial because it can provide a rough estimate of the goals an agent could achieve with a lower cost compared to the planning cost for all the goals. A knapsack algorithm uses a resource value to pick the maximum set of goals an agent can achieve. If the agent predicts that the agent can achieve greater than 85% of its goals with the resources available, then it sends a qualitative value ”Yes”; otherwise, it sends a qualitative value ”No.” The figure 6.7 depicts such in a pictorial format.

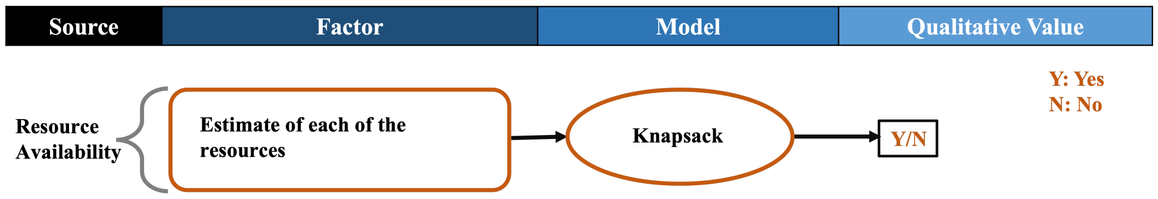

Figure 6.7: The figure represents the class to estimate the number of resources for the agent. We model this using a simple knapsack algorithm. For example, if an agent can achieve 85% of its goals, then we classify the resources as sufficient(yes), else we classify them as insufficient(no).

6.2.4 Goal Management when Multiple Goal Operation Co-occur

Now we have access to all the qualitative factors required to make an informed decision for selecting a goal operation. We use all three factors and formulate generic rules. The generic rules are instrumental across several domains. The figure 6.8 depicts all the components of the algorithm together.

After generating all the qualitative factors, the agent then compares the results to the generic rules in MIDCA. Each rule outputs a specific goal operation. If the factors match any of the rules, the agent chooses a specific goal operation. Specifically, some of the rules we used to decide are as follows. In addition, the agent analyzes the rules in the given order.

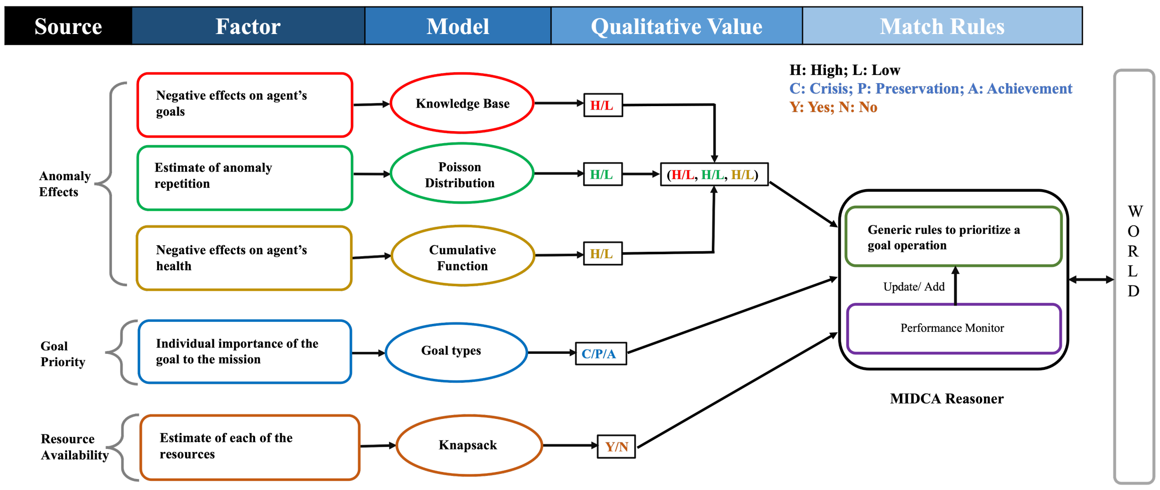

Figure 6.8: The figure represents the entire method to select goal operations. First, we consider three classes of factors: anomaly effects, goal priority, and resource availability. Next, we model each factor and classify the model's output into qualitative factors. Finally, we compare the outputs to the general rules in MIDCA and pick a goal operation appropriate to the agent's situation.

- If any of the anomaly effects is *"high"* and the resources available are sufficient, *"yes,"* then prioritize goal formulation.

- If all the anomaly effects are *"low"* and resources are sufficient, *"yes,"* for both selection and change and selection and change generate different goal types, then prioritize one goal operation based on the goal type, *"Crisis > Preservation > Achievement.*

- If all the anomaly effects are *"low"* and resources are sufficient, *"yes,"* for both selection and change and selection and change generate same goal types, then prioritize one goal operation that uses fewer resources.

The rules mentioned above are generally sufficient to make an informed decision when formulation, change, and selection operations co-occur. However, the algorithm can also easily include other goal operations. For example, we could easily include a delegation rule as follows:

- If resources available are not sufficient, *"no,"* then choose goal delegation over all the remaining operations.

In addition, we also realize that these generic might need adaptation based on the agent's situation in the real world. Therefore, we plan to update the rules or add new rules in the future through the reinforcement obtained from the real world. We implement the goal management method in two different domains: the marine survey domain and the construction domain.

Implementation and Empirical Evaluation of the Goal Management Strategy

As mentioned in the earlier chapter, we implement and evaluate the developed goal management method in the marine survey and construction domains. The marine survey domain attempts to survey water bodies to gather information on the water body and the fish patterns in the water. The construction domain is similar to the one defined in chapters 3, 4, but includes extra anomalies.

7.1 The Marine Survey Domain

In the marine survey domain, we attempt to solve the problem of time-limited surveys of marine environments with autonomous underwater vehicles (AUVs) [63, 62]. Typical goals in this domain include measuring salinity, temperature, and pressure throughout the water column. They can also incorporate acoustic receivers to investigate key aspects of marine life. An important goal within a marine ecosystem is identifying the presence of *hot spots* or regions of high fish density. These areas and the aquatic pathways between them that fish transit represent areas of ecological sensitivity. Thus, discovering the location of major hot spots, especially for endangered species, is an important application. However, many anomalies exist in such environments that hinder agents' performance. For example, sea creatures may attach themselves to platforms and slow progress. In addition, tides and currents exist that also impede progress, and obstacles may appear requiring course change. Finally, conditions may change, limiting the detection range of acoustic receivers [36, 72].

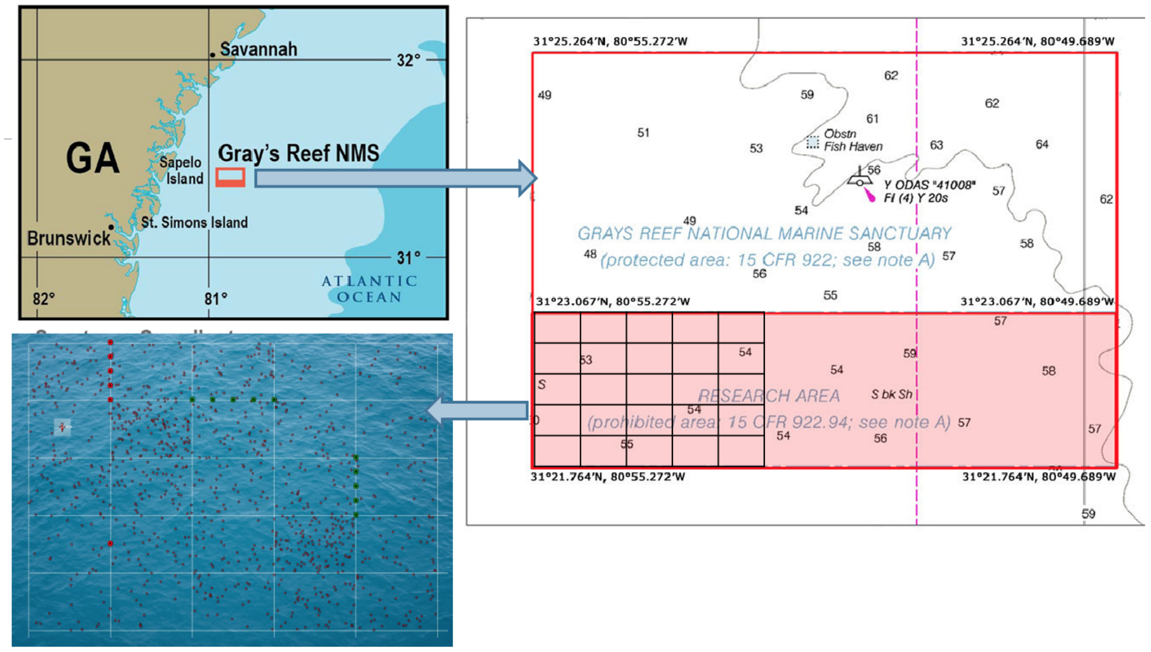

Figure 7.1: Gray's Reef National Marine Sanctuary is located off the coast of Georgia and contains a research area shown in the insert shaded in pink. Within this, we represent a 5x5 subsection. This grid contains fish hot spots that are of interest to marine scientists, here within the cell at locations (1,3) and (3,1). The agent (indicated by the red streak) is in (0,3) cell. The highlighted square around the agent indicates the sensor range for detecting acoustic fish tags (the small red dots).

Our research team regularly deploys AUVs such as Slocum gliders and custom robotic fish for science-driven experiments, testing, and evaluation of new platforms. During missions, the platforms surface to communicate on regular schedules or in response to forced interrupts. AUV surveys make a valuable contribution to management efforts in Gray's Reef National Marine Sanctuary, located on the inner shelf of the South Atlantic Bight off the coast of Savannah, GA (see Figure 7.1). Gray's Reef contains fish tagged with transmitters that send an acoustic signal or 'ping' at a pre-determined frequency (5 minutes for short experiments, 30-180 minutes for long-term tracking) containing identifiers unique to that instrument, allowing researchers to classify detection's by source.

We implemented the domain using an open-source simulator to test search techniques prior to deployment. The simulator is called *Mission Oriented Operating Suite (MOOS)*[7]. It provides modeling for underwater platforms. The lower left of the figure 7.1 shows the portion of the research area modeled by the MOOS simulator, and we split it into 25 cells. We present one scenario of fish distribution in the lower left of the figure. The red dots depict 1000 fish (currently assumed to be static) that emit a ping every 17-time steps. In this scenario, hot spots are near the coordinates (1,3) and (3,1). The red streak represents an AUV controlled by MIDCA, and the highlighted square area represents the receiver detection radius. At Gray's Reef, the detection radius varies with environmental conditions, but currently, the simulator assumes it to be constant. As mentioned, an agent can identify hot spots based on the number of pings.

An example scenario from the marine survey domain can help us understand the importance of the agent using the proposed algorithm to prioritize goal operations. Therefore, we refer to the agent using the proposed algorithm as a smart agent.

The figure 7.2 represents an example scenario in the Marine Survey Domain. As mentioned, there are twenty-five survey goals for the smart agent. We name each cell by its X, Y coordinate values, with both X and Y values ranging from 0-4. The cell's initial location is on the lower left (its coordinate is (0,0)). In addition, several types of anomalies exist in the domain. First, Remora attacks hinder the agent's movement; second, obstacles (represented in red and green lines) hinder the agent's movement from one location to the blocked location. The green obstacles allow movement of the agent by either diving up or down. The agent can only learn about such an obstacle if it stops and inspects the obstacle to gain more information. The red obstacles do not allow the movement of the agent.

Consider a scenario where the smart agent performs goal selection and surveys the

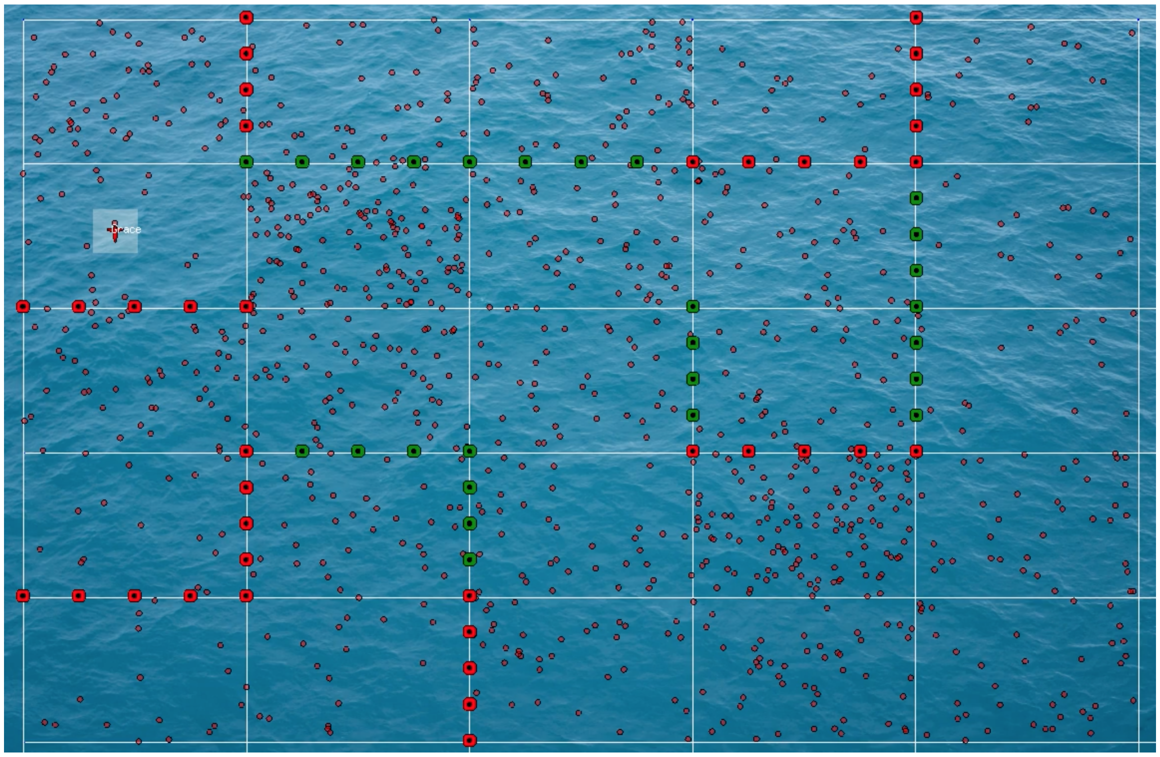
Figure 7.2: An example in the Marine Survey Domain. We divide the region into 25 equal-sized cells. The anomalies in this domain are Remora attacks (cannot be depicted in the image) and obstacles (shown by the red and green lines along cell edges). The agent (indicated by the red cylinder) is at the location (0,3). The highlighted area around the agent indicates the sensor range for detecting acoustic fish tags (the red dots). The hot spots are at (1,3) and (3,1).

location (0,3). It then encounters a Remora attack. The agent has two choices; select a new survey location (Goal Selection), or formulate a goal to remove the organism by gliding backward (Goal Formulation). The agent now reasons about the anomaly effects. Since the Remora attack hinders the agent's movement and the Remora also chips the paint off of the agent, the agent considers the anomaly to be a "high" threat to its health. According to the goal management method from 6.8, if any anomaly effects are high, the agent must choose the goal formulation. So, the smart agent prioritizes goal formulation and glides backward to free itself from the Remora attack. After this, it completes the survey in (0, 3). Since the agent does not have a current goal, it performs a goal selection operation and pursues a new goal. Let us assume that the agent selects the goal of surveying (0, 2). So, the agent must move from its current location (0, 3) to the destination location (0, 2).

The smart agent now encounters an obstacle anomaly that hinders movement from (0, 3) to (0, 2). Thus, it now has two choices; select a new survey location between (1, 3) and (0, 4), or formulate a new goal to inspect the entrance to understand the cause of the obstacle. In this scenario, the obstacle does not affect the agent's health or the agent's goals. In addition, since this is the first occurrence of obstacle anomaly, the anomaly repetition is also low. Hence, the agent prioritizes goal selection and surveys the location given by the selection operation. Similarly, the smart agent prioritizes other goal operations as well. Such prioritization helps the agent address its goals promptly in the dynamic world.

7.1.1 Experimental Design: The Marine Survey Domain

To demonstrate the importance of the goal management procedure developed, we compare the performance of four different agents. The first is a baseline agent that performs only planning and no goal operations. The second is an ideal agent, i.e., an agent achieving its goals in a perfect world (without any anomalies but with goal operations). The third is a random agent that chooses a goal operation in random when multiple co-occur. Finally, the fourth is a smart agent. It implements the developed strategy to choose a goal operation when they co-occur.

In the marine survey domain, the agent searches all cells for hotspots until a maximum deadline of 600-time units. In addition, we create multiple hotspot patterns in the 5x5 survey region. The first scenario has one hotspot at the cell location of (1,4). The second has four hotspots at four corners cells of the 5x5 grid. One trial requires an agent to traverse all the 25 cells of a particular scenario. The agent starts at a given initial location and works towards finding all hotspots before the deadline. For each scenario, we vary trials with 100 different starting locations. We also repeat the experiment by randomizing the anomaly occurrences (with two other seed values). Therefore, we have 300 trials in total.

To set up the experiment in this domain, first, we need to implement the three-goal operations goal selection, goal change, and goal formulation. Then we implement the method to manage goal operations when they co-occur. We discuss the individual implementation of the goal operations in subsequent sections.

7.1.1.1 Goal Selection in The Marine Survey Domain:

We modify the stochastic hill-climbing algorithm [80] to perform a greedy structured search. However, some limitations apply to stochastic hill climbing. For example, it cannot function well if it encounters a local maximum. Therefore, the agent backtracks and continues its search to overcome the inability and continue the search in a structured search strategy. The table 7.1 presents the procedure for structured search.

The agent starts in one specific location of all the 25 cells (line 1). It then adds the cell to the visited stack (line 3). Next, the agent searches the current cell for unique pings (line 4) and records all unique pings. In addition, it also records the location of the pings and classifies them into four boundary edges in a cell. The agent then selects a neighboring cell with the highest unique ping density (line 6). The agent then continuously visits cells that increase in unique pings (lines 9-11). However, when the unique pings in its current cell are less than that of the previously visited cell (line 12), it backtracks and visits the unexplored best neighbor of the last visited cell (line 14). The agent performs this procedure until it visits all cells (lines 7-8) or meets the mission deadline.

CSearch(searchCell, visited) is a support function that accepts inputs from a searched cell and the visited cell stacks and returns the searched cell's best neighbor to visit (lines 18-20). However, if all the input cell neighbors are visited (line 21), it recursively checks for the best non-visited neighbor from a previously visited cell and returns it (lines 25-26). In addition to structured search, we also implement an ergodic search [63] and structured search combined with an ergodic search [63]. However, we do not present the details of the other selection strategies as they are not the focus of this thesis.

Table 7.1: The table depicts the method for structured search (SS). First, the agent surveys the current cell and compares the pings it hears in all four cell directions. Finally, the agent chooses the adjacent cell with the highest pings. In addition, the agent also backtracks in case of local maximums to complete the search of all 25 regions.

```
1. $currentCell \leftarrow startCell$   // Initial location of the agent
2. $visited \leftarrow initStack()$   // Keep track of the surveyed cells for backtracking
3. $visited.push(currentCell)$
4. $currentCellPings \leftarrow$ UniquePings$(currentCell)$  // Survey current cell and get the no:of fish tags
5. loop do
6.     $currentCell, visited \leftarrow$ CSearch$(currentCell, visited)$   // Best neighbor of the current cell
7.     $if\ visited.empty()$
8.         $break$   // Stop the search as all the cells have been surveyed
9.     $currentCellPings \leftarrow$ UniquePings$(currentCell)$
10.    $previousCell \leftarrow visited.top()$   // Get previously surveyed cell
11.    $visited.push(currentCell)$   // Add the current cell to the visited list
12.    $if\ currentCellPings <$ UniquePings$(previousCell)$
13.        $visited \leftarrow SortByMaxUniqueTags(visited)$
14.        $currentCell \leftarrow previousCell$   // Agent is at the top of the hill, perform backtracking

CSearch(searchCell, visited)
15. $N, S, E, W =$ GetExpectedPings$(searchCell)$   // get north, south, east, west expected fish tags
16. $SearchCell =$ UnvstedMaxCell$(N, S, E, W, visited))$  // Unvisited cell with max projected fish tags
17. $if\ searchCell\ is\ None$   // When the neighbors of the current cell are visited
18.    $if\ visited.empty()$
19.        $return\ searchCell, visited$   // Agent has surveyed all the cells
20.    $else$
21.        $newSearchCell \leftarrow visited.pop()$  // Backtrack to the neighbors of the previously visited cells
22.        $return$ CSearch$(newSearchCell, visited)$   // Recursively find the best neighbor
23. $else$
24.    $return\ searchCell, visited$   // Best neighboring cell with max projected fish tags
```

7.1.1.2 Goal Change in The Marine Survey Domain.

Similar to the discussion in chapter 4. We implement the goal change operation using predicate transformations from a brief survey of a cell to a deep survey of a cell. For example, in the marine survey domain, when the agent observes a potential to find a hot spot, it could change its goal to perform a deeper search and collect more values.

7.1.1.3 Goal Formulation in The Marine Survey Domain.

Similar to the discussion in the chapter 5, we implement the goal formulation operation using anomaly detection and explanations. The agent generates goals to tackle the Remora attacks, unexpected flow, and obstacles in this domain. When remora latch onto the AUV and hinder the agent's movement, it needs to glide backward to get rid of it. In the event of an unexpected flow, the agent must dive deeper to reduce the effects of currents. Finally, in the event of an obstacle, the agent must generate a knowledge goal to gain more knowledge about the obstacle. Equipped with this new knowledge, the agent can generate a goal to pass through some obstacles either by diving up or down while the remaining obstacle remains impassable.

7.1.2 Empirical Results : The Marine Survey Domain

This section depicts the empirical results obtained from running an experiment with the setup mentioned above, as our main aim is to maximize the number of goals achieved. We plot the results as a function of time, a function of anomalies.

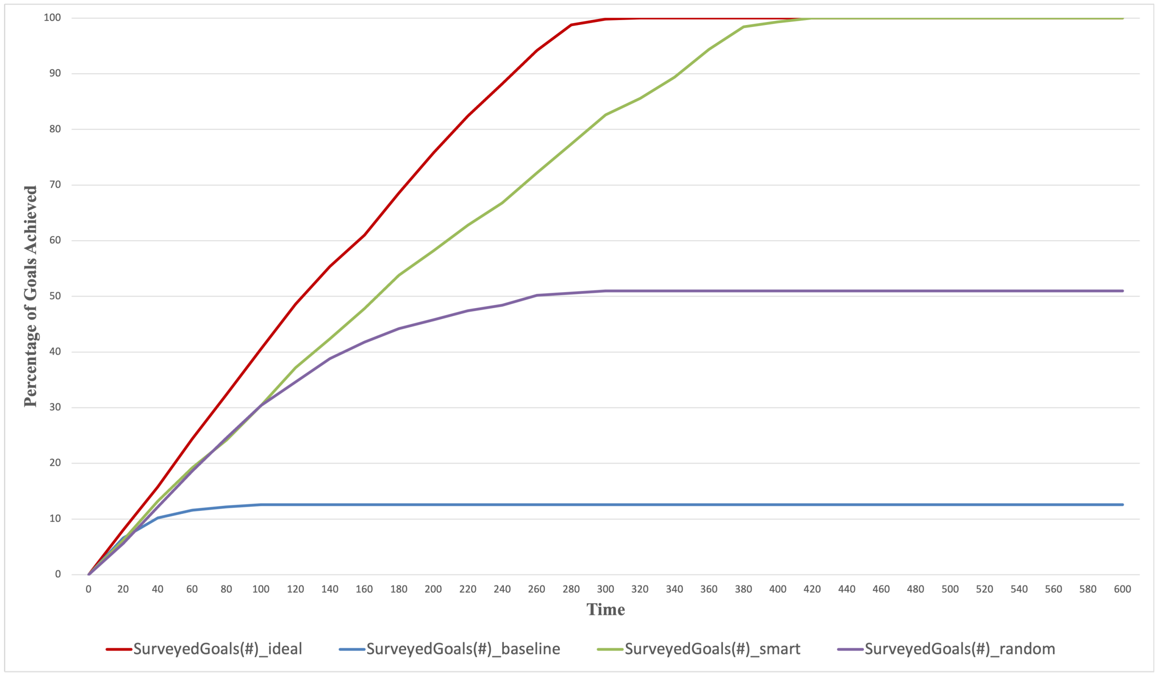

Figure 7.3: Results obtained in the marine survey domain with the one hotspot scenario. The X-axis denotes the time, and Y-axis represents the cumulative percentage of goals achieved. We compare the performance among four different agents: Ideal (agent working in a perfect world), Smart (agent using the goal management algorithm), Random (agent selecting goal operations randomly; and Baseline (the agent ignoring anomalies).

Figure 7.3 presents results for the scenarios with and without the goal management strategy. The X-axis depicts time, and the Y-axis represents the percentage of goals achieved. In addition, the results are obtained with a single hotspot scenario, having the hotspot at location (1,4).

The baseline agent ignores anomalies, so eventually, when one remora attacks, the agent does not respond. It prompts a multi-remora attack, and eventually, the agent comes to a halt. The ideal agent, which works in a perfect world, achieves the highest performance possible because no anomalies occur. The agent choosing goal operations in random performs better than the baseline agent. However, eventually, it runs out of resources due to its poor choices. The smart agent uses the goal management strategy to make an informed decision. It gradually achieves its goals until it matches the performance of the ideal agent.

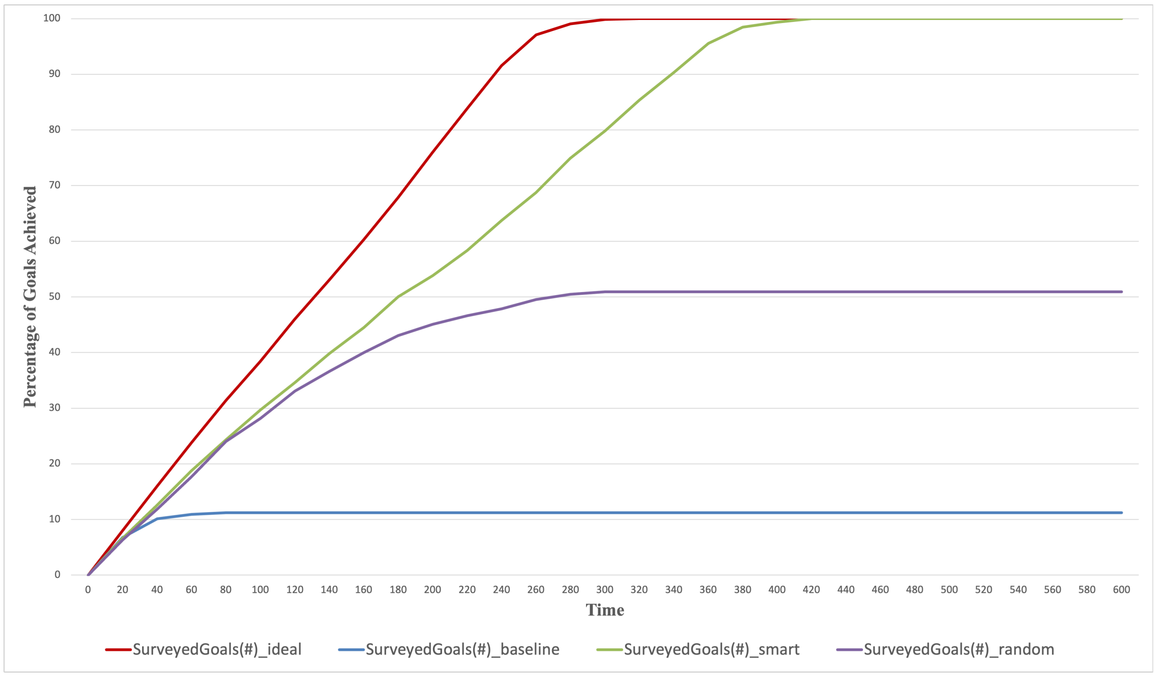

Figure 7.4: Results obtained in marine survey domain with the four hotspot scenario. The X-axis denotes the time, and Y-axis represents the cumulative percentage of goals achieved. We compare the performance among four different agents: Ideal (agent working in a perfect world), Smart (agent using the goal management algorithm), Random (agent selecting goal operations randomly; and Baseline (the agent ignoring anomalies).

Figure 7.4 presents results for the four hotspot scenarios with and without the goal management strategy. The X-axis depicts time, and the Y-axis represents the percentage of goals achieved. In addition, we obtain the results with a multiple hotspot scenario, with hotspots at locations (0,0), (0,4), (4,0), (4,4).

Similar to the previous graph, the pattern remains constant. The baseline agent ignores anomalies. So, it eventually halts, thus making its performance a flat-line. Once again, the ideal agent achieves the highest performance. The agent choosing goal operations in random perform better than the baseline agent but not better than the ideal or smart agents. The smart agent uses the goal management strategy to make an informed decision. As in the figure 7.4, it gradually achieves its goals until it matches the performance of the ideal agent.

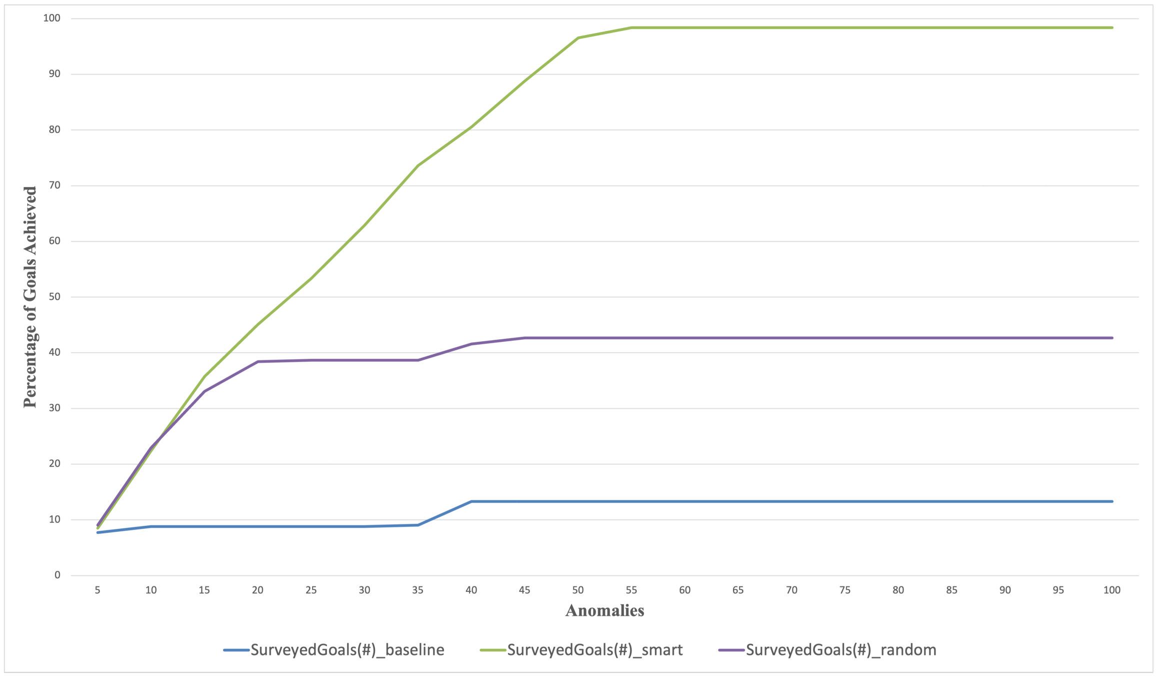

Figure 7.5: The figure depicts the results obtained as a function of anomalies for the single hot-spot scenario. The X-axis denotes the anomalies, and Y-axis represents the percentage of goals achieved. We compare the performance among four different agents: Ideal (agent working in a perfect world), Smart (agent using the proposed algorithm), Random (agent selecting goal operations in random; and Baseline (the agent that ignores anomalies). The results are for a single hotspot scenario.

Figure 7.5 presents results for the scenarios with and without the goal management strategy. The X-axis depicts anomalies, and the Y-axis represents the percentage of goals achieved. In addition, we obtain the results with a single hotspot scenario, with a hotspot at location (1,4).

Since the baseline agent ignores anomalies and halts, it performs very poorly. The ideal agent, which works in a perfect world, does not have any anomalies to plot against. Hence we depict it as a line on X-axis. The agent choosing goal operations in random performs better than the baseline agent. However, eventually, as the anomalies increase. It runs out of resources and flat lines. The smart agent uses the goal management strategy to make an informed decision. So, as the number of anomalies increases, the percentage of goals achieved also increases.

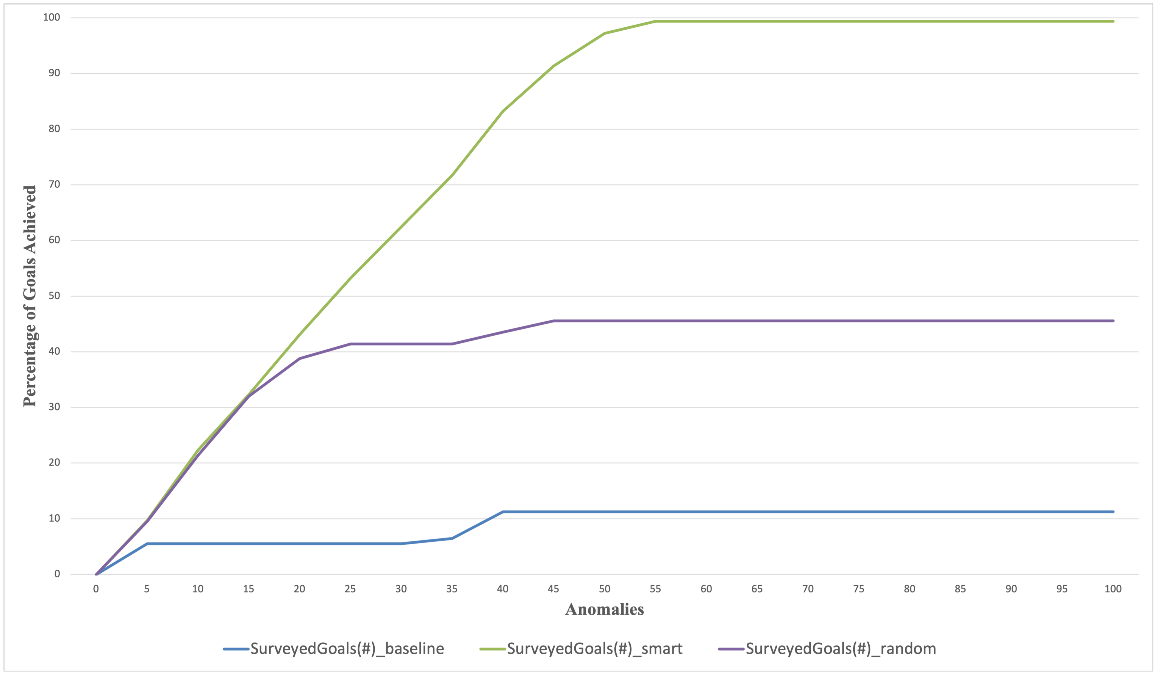

Figure 7.6: The figure depicts the results obtained as a function of anomalies for the four hot-spot scenario. The X-axis denotes the anomalies, and Y-axis represents the percentage of goals achieved. We compare the performance among four different agents: Ideal (agent working in a perfect world), Smart (agent using the proposed algorithm), Random (agent selecting goal operations in random; and Baseline (the agent that ignores anomalies). The results are for a single hotspot scenario.

Figure 7.6 presents results for the scenarios with and without the goal management strategy. The X-axis depicts anomalies, and the Y-axis represents the percentage of goals achieved. In addition, we obtain the results with a multiple hotspot scenario, with hotspots at locations (0,0), (0,4), (4,0), (4,4).

Once again, the pattern remains similar. The baseline agent ignores anomalies and halts it performs very poorly. The ideal agent, which works in a perfect world, does not have any anomalies to plot against. Hence we depict it as a line on X-axis. The agent choosing goal operations in random performs better than the baseline agent. However, eventually, as the anomalies increase. It runs out of resources and flat lines. The smart agent uses the goal management strategy to make an informed decision. So, as the number of anomalies increases, the percentage of goals achieved also increases.

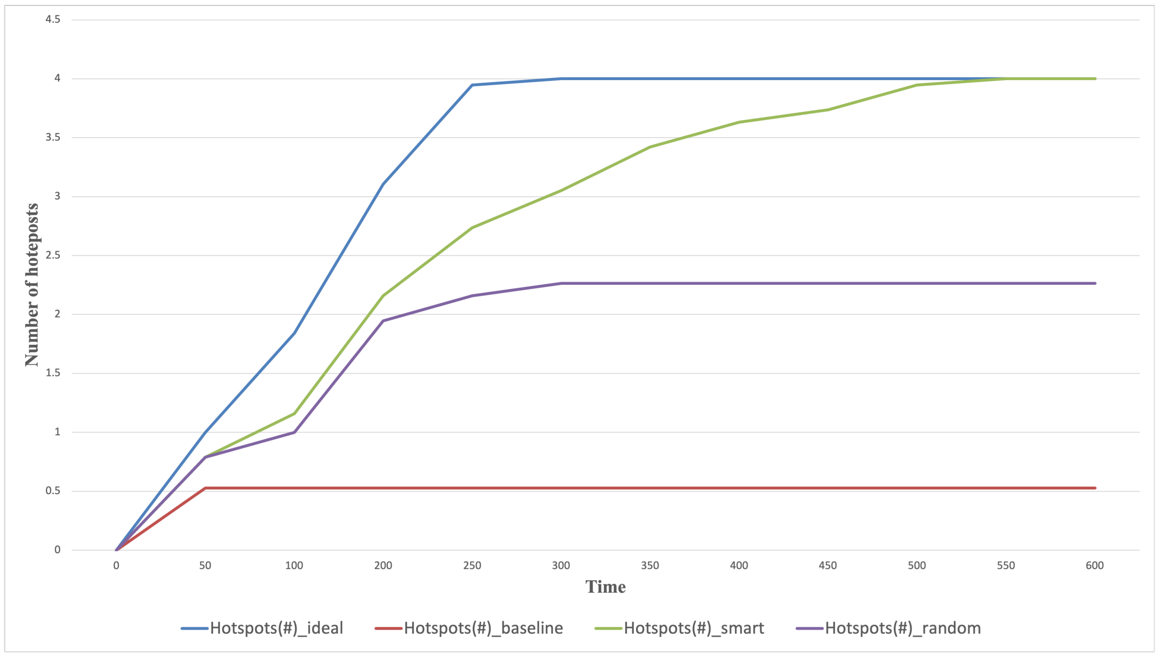

Figure 7.7: The figure depicts the results obtained in the Marine Survey Domain. The X-axis denotes the time, and Y-axis represents the number of hotspots. We compare the performance among four different agents: Ideal (agent working in a perfect world), Smart (agent using the proposed algorithm), Random (agent selecting goal operations in random; and Baseline (the agent that ignores anomalies). The results are for a multi hotspot scenario.

Figure 7.7 presents results for the scenarios with and without the goal management strategy. The X-axis depicts anomalies, and the Y-axis represents the number of anomalies.

Once again, the pattern the agent presents in this graph is as expected. The baseline agent ignores anomalies and halts. So, it performs very poorly. The ideal agent, which works in a perfect world, identifies all hot spots. The agent choosing goal operations in random performs better than the baseline agent but worse than the smart and the ideal agent. However, it manages to identify around half of the total hot spots. The smart agent eventually identifies all the hot spots.

The next section details the second implementation domain, the construction domain.

7.2 The Construction Domain

Similar to the domain in the chapter 4, the construction domain is an extension of the blocks world domain. It is also an extended version of the constriction domain presented in the chapter 3. The construction domain contains several blocks, and each block has a unique name for identification. The domain contains two types of blocks: regular blocks and mortar blocks. The agent attempts to stack one block over the other to build high-rise towers. If the agent stacks the blocks without mortar, the tower is wobbly. Whereas, if it uses mortar blocks, the tower is sturdy. In general, sturdy towers are more desirable compared to wobbly towers. The agent gains reward for construction based on the type and height of the tower. There are two other actors in the domain due to which unexpected events in this domain arise. First, an arsonist destroys the constructed towers by lighting them on fire. The second one is a thief who steals the construction blocks. Both the arsonist and thief act randomly.

The fig 7.8 shows an instance of the goals achieved by the agent in the construction domain. The agent constructs towers of different heights by placing the blocks on top of one another. Each goal provided to the agent is to construct one tower of a particular height. As mentioned, an agent gains a benefit after the successful achievement of the goal and also incurs a cost to achieve a goal. The benefit obtained for each tower is proportional to the tower's height and its type. Similarly, the cost for constructing each tower is proportional to the number of blocks used for the tower. As mentioned, an arsonist lights the constructed towers on fire. In addition, there is also a thief who steals the construction materials (or) blocks at random.

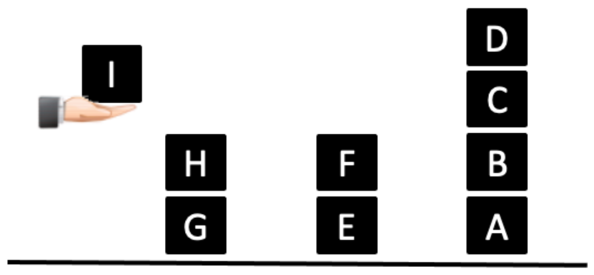

Figure 7.8: The figure depicts an instance of the achieved goals in the construction domain. The agent achieved three goals in the above illustration. The first tower has a height of four, the second tower has a height of two, and the third tower will have a height of three.

The next section demonstrates the importance of choosing the goal operations by implementing and comparing the agent's performance in the construction domain. Finally, it describes the experimental setup and empirical results obtained in both domains.

7.2.1 Experimental Design: The Construction Domain

Similar to the marine survey domain, we use the same four agents in the marine survey domain for comparison, the baseline, ideal, random, and smart agents.

The agent starts with 100 initial blocks and mortar in the construction domain. In addition, the agent builds towers until reaching a deadline of 100-time units. The anomalies such as fire and theft occur randomly. The agent must either generate a goal to apprehend the culprit or continue working on its goals. The agents work on 1000 goal sets (problem sets). Each problem consists of towers varying in height from one to seven. We repeat the problem set with two different seeds. Thus, the results report values averages across 2000 trials.

To set up the experiment in this domain, first, we need to implement the three-goal operations goal selection, goal change, and goal formulation individually. Then we implement the method to manage goal operations when they co-occur. The next subsections elaborate on the individual implementation of goal operations.

7.2.1.1 Goal Selection in The Construction Domain:

We perform goal selection using cost-benefit analysis in this domain. The details of the selection are the same as in the chapter3. We modify the benefit according to the tower type (sturdy vs. wobbly). The sturdy tower gains more benefit. For details on the exact values, see appendix A B.

7.2.1.2 Goal Change in The Construction Domain:

The goal change operation is similar follows a similar strategy as mentioned in the chapter 4. We implement both generalization and specialization goal transformation for goal change. According to the availability of resources, the agent could either change its goal from ”stable-on” to ”on,” or from ”on” to ”stable-on.” Previously, it could only change from ”stable-on” to ”on.” Currently, with the thief, if the agent apprehends the thief, it can successfully restock the stolen resources.

7.2.1.3 Goal Formulation in The Construction Domain:

Similar to the other domains in the chapter 5, the goal formulation operation occurs in case of anomalous events. As mentioned, in this domain, the anomalies are the arsonist and thief. The arsonist burns constructed towers randomly, and the thief steals blocks and mortar at random. The goal generated here is either to ”extinguished fire,” ”apprehended arsonist,” or ”apprehend the thief.” The agent generates goals according to its situation. For example, it will not apprehend the thief if he steals only one block, as it is an unnecessary resource expenditure to try and apprehend a thief (there is a good chance not to find the thief). Hence, the agent is smart in differentiating anomalies with actual problems while goal formulation [61].

7.2.2 Empirical Results: The Construction Domain

This section represents the obtained empirical results in the construction domain with the above experimental setup.

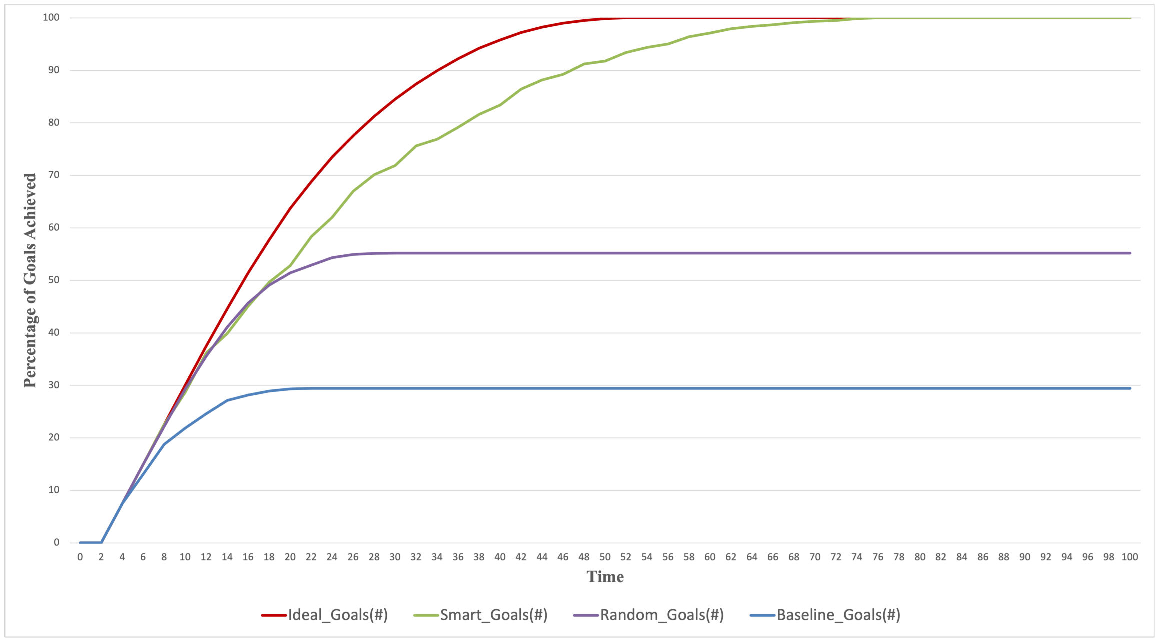

Figure 7.9: The figure depicts the results obtained as a function of time in the construction domain. The X-axis denotes the time, and Y-axis represents the percentage of goals achieved. We compare the performance among four different agents: Ideal (agent working in a perfect world), Smart (agent using the proposed algorithm), Random (agent selecting goal operations in random; and Baseline (the agent that ignores anomalies). The graphs present values that vary across 2000 trials.

Figure 7.9 presents results for the scenarios with and without the goal management strategy. The X-axis depicts time, and the Y-axis represents the percentage of goals achieved. In addition, we obtain the results with values averaged across 2000 runs.

Even in this domain, the baseline agent ignores anomalies, so eventually, either the thief steals all the blocks, or the arsonist burns everything. Since the baseline agent does not respond, it performs very poorly. The ideal agent, which works in a perfect world, achieves the highest performance. The agent choosing goal operations in random performs better than the baseline agent. However, eventually, it runs out of resources due to its poor choices.

The smart agent uses the goal management strategy to make an informed decision. Like the marine domain, it gradually achieves all goals until it matches the performance of an ideal agent. Although the graphs in these two domains asymptote at different values, they follow the same trends.

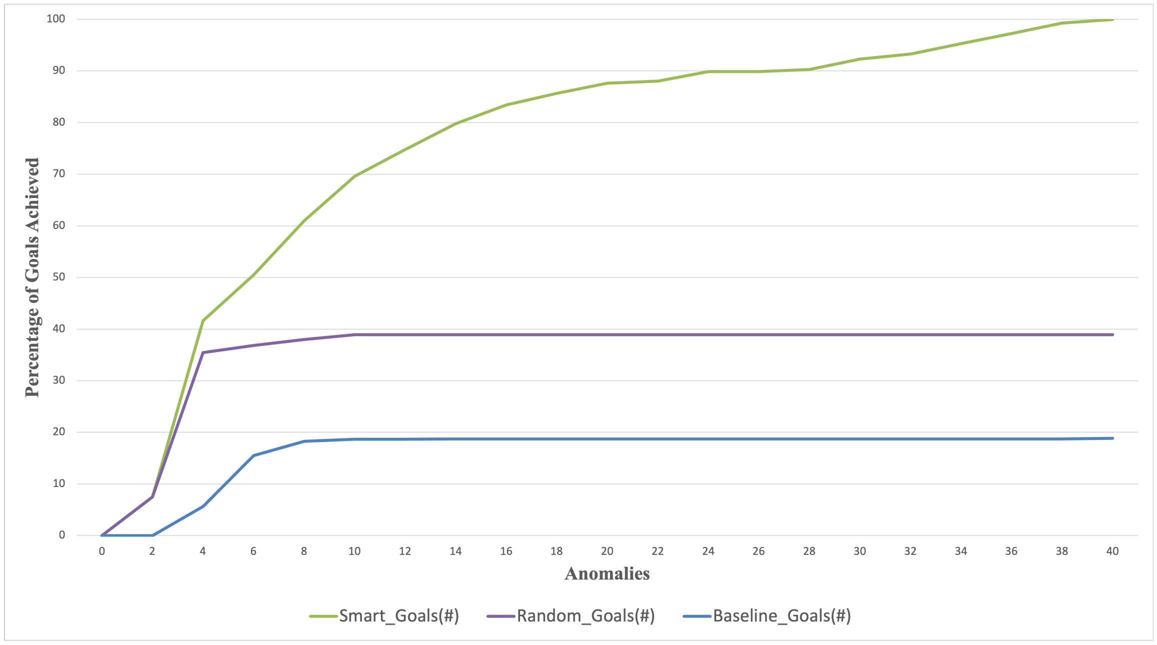

Figure 7.10: The figure depicts the results obtained as a function of anomalies in the construction domain. The X-axis denotes the anomalies, and Y-axis represents the percentage of goals achieved. We compare the performance among four different agents: Ideal (agent working in a perfect world), Smart (agent using the proposed algorithm), Random (agent selecting goal operations in random; and Baseline (the agent that ignores anomalies). the values are averages across 2000 trials.

Figure 7.10 presents results for the scenarios with and without the goal management strategy. The X-axis depicts anomalies, and the Y-axis represents the percentage of goals achieved. In addition, we obtain the results with values averaged across 2000 runs.

Since the baseline agent ignores anomalies and halts, it receives a very low score. The ideal agent, which works in a perfect world, does not have any anomalies to plot against. Hence we depict it as a line on X-axis. The agent choosing goal operations in random performs better than the baseline agent. However, eventually, as the anomalies increase. It runs out of resources and flat lines. The smart agent uses the goal management strategy to make an informed decision. So, as the number of anomalies increases, the percentage of goals achieved also increases.

7.3 Summary of Interaction among Goal Operations

In conclusion, it is essential to address the knowledge gap present about the interaction of various goal operations because the output of various interactions is different and unique in complex situations. Therefore, the agent must possess the knowledge to make a smart choice when it finds itself in similar situations.

We develop a generic goal management strategy to aid the agent take such decisions. The strategy uses three main factors: anomaly effects, goal types, and resource availability. We model each of the methods using various methods. Then classify the output of these values into specific qualitative values. We then use the qualitative values to determine the outcome. All three factors play an important role in the goal management process. We implement the strategy in two very different domains: a real-world marine survey domain and a simulated construction domain. In addition, we also present the empirical evaluation for the results obtained. Since the work is first in its series and the research area does not advertise any official benchmark problem sets. We define four agents that operate with very wide performance variations. This allows us to compare the performance of the developed agent with the lower bound (baseline) and the upper bound (ideal) agents.

In the future, we want to further the research by including more qualitative factors such as medium. We also want to include other goal operations such as goal delegation, sharing, monitoring. One of the above subsections presents an idea about the inclusion of goal delegation within the current goal management strategy. In addition, we can also extend the work by including all the anomaly factors instead of just negative effects.

Literature review

Cognitive systems research is progressing by following many different paths and strategies. Some well-known areas and concepts include Goal-Driven Autonomy(GDA), explanations, expectations, meta-cognition, Belief-Desire-Intention(BDI) agents, and execution monitoring. In this literature review, we will be looking at several papers that belong to the above categories. The literature review would aid in understanding some of the active research areas in the cognitive systems.

INTRO [20] introduced the concept of goal-driven autonomy. The main motivation behind developing such autonomy is to investigate the reasons for the origin of goals. Therefore, GDA focuses on building agents capable of generating and managing their own goals. INTRO implements its research in Wumpus World, where the agent needs to avoid pits and Wumpus while reaching a certain destination. This demonstrates the importance of developing such an agent. Further work on GDA includes implementing the frameworks developed with a cognitive architecture [81]. The cognitive architecture used is called MIDCA. Moreover, since the motivation behind GDA is to investigate the origin of goals, the work also presents some goal generation mechanisms called K-track, D-track, and dual-track. K-track uses TF-trees to generate a goal, D-track uses an explanation system called Meta-AQUA to generate a goal, and the dual-track is the combination of both K-track and D-track. Later, the authors modify the work and extend it to a new domain [82] called the arsonist domain. In this domain, random fire events occur because of the arsonist, and the agent needs to generate goals to investigate the reasons for the cause of the fire. Our current research builds upon the more recent version of the existing MIDCA architecture and builds additional frameworks to facilitate a more robust GDA agent.

In addition to MIDCA, [57] implements GDA in an agent called *Autonomous Response To Unexpected Events(ARTUE)*. The authors implement discrepancy detection and study the performance variation of the ARTUE system in terms of benefits and limitations. The work on ARTUE is also further developed into several later versions called Trainable-ARTUE (T-ARTUE) [86], and M-ARTUE [107]. T-ARTUE performs interactive learning to obtain knowledge to select goals, whereas M-ARTUE uses motivators to formulate goals. The work on implementing GDA in ARTUE and MIDCA demonstrates the generalizability of the GDA frameworks across several platforms. This thesis also performs goal operations such as selection and formulation, as in some of the works mentioned. In addition, however, this thesis also implements other goals operations such as goal change. Moreover, our work also developed a framework to study the interactions between different operations instead of just implementing the individual operations.

GDA also allows the agent to perform mental operations on agent goals called goal reasoning. Goal reasoning facilitates the agent to perform goal operations. Some goal operations include goal selection, goal change, goal formulation, goal delegation, goal monitoring. For example, Cox et al., [23] presents a list of goal operations and elaborates on two particular goal operations: goal change and goal formulation. However, the operations are not only explicit to the GDA framework. Several research works that do not focus on the GDA also develop mentioned goal operations. Let us now look at relevant works for each of the operations.

Goal selection operation allows the agent to prioritize its goals for goal achievement. [55] performs goal selection using Goal Reasoning with Informative Measures(GRIM). This work is domain-specific and implements the selection operation to locate an officer by traversing uncertain locations. GRIM uses information metrics like distance traversed and time to perform the goal selection operation. The extension to this work includes its implementation in multiple domains. [59, 58] implements selection in the construction domain as well as restaurant domain. In addition, the work demonstrates the generality of the work in limited-resource domains. Similarly, T-ARTUE [86] performs interactive learning to learn the knowledge of goal selection from a user. T-ARTUE not only learns through the expert, but it also accepts criticism for a wrongly selected goal and corrects the selection not to repeat it in the future. In addition, more works on selection include [105, 106] which present a goal manager. This goal manager chooses goals based on a priority value. The authors implement their work on autonomous underwater platforms. Further works on selection include Dora the explorer, Hawes et al., [50] explores all the world to fill its gaps in spatial knowledge. Here the priorities for the goals are set by the user manually to select one goal. The goal selection operation is essential in many other applications. For example, one application includes space [13, 14] where the agent triggers new goals based on the outcomes of the previous goals. The work mentioned inspired the work of Rabideau and colleagues, [87] to develop an algorithm for goal selection with oversubscribed resources. In their algorithm, the constraints and priorities define which goal to select among the set of all goals. [103] presents the performance improvement in *Arcsecond Space Telescope Enabling Research in Astrophysics (ASTERIA)* CubeSat. ASTERIA was deployed into space to demonstrate precision photometry in 2017. Although the above work does not explicitly include goal selection, it implements it inherently.

Goal change operation changes the agent's goal to a similar goal due to the evolving environments. Initial work on goal change includes [27] implementation of the operation in the Air Campaign (ACP) domain, where the agent should manage its goals along with changing its goal. Furthermore, it presents a taxonomy of goal transformations, and the goal change occurs due to the following reasons:

- When the planning system anticipates a change in the environment that dictates an adjustment during planning or execution.

- When there are limited or high resources.

. The authors develop this work further in Cox et al., [23], where the authors use predicate transformations to implement goal change operation. Specifically, the authors implement generalization transformation and present the results in a construction domain. [58, 63] later develops this work to include other predicate transformations such as specialization, identity and implements goal change in different domains.

Goal formulation is an essential operation that allows the agent to generate its own goals. Several works focus on developing goal formulation; for example, our previous work generates goals [64, 61] when an agent encounters a problem. We define problems as anomalies that affect the agent or its goals negatively. [107, 105] implements goal formulation using domain-independent heuristics called motivators (opportunity, exploration, and social). The authors calculate the urgency and fitness for each motivator using specific formulas. They test their work in the Mars world domain using the M-ARTUE agent. The works also present an approach to integrating high-level intelligent behavior with motion autonomy while extending the work in multi-agent scenarios. The authors develop this work further in [106], where the agent generates goals and assigns them a priority value for goal management. [18] presents another system, which tries to formulate and manage its own goals. The work includes the motivations of the agent to generate goals. The authors develop the work in the Casper architecture and the NASA MER rover domains. Finally, Hawes [49] develops work on both goal formulation and goal selection using a motivational management framework. This work is a survey paper that also points to several other works on formalization and selection.

This thesis implements mentioned three-goal operations individually, similar to many of the works mentioned. However, it also studies their interactions. The contribution on interactions stands as a novel contribution. None of the works mentioned above examine or implement this contribution.

Some additional works on goals and goal reasoning include the life cycle of a goal from Roberts et al., [92]. The work presents goal lifecycle in the form of goal mode transitions.

Another work on similar lines uses a hierarchy of goals for better representation. This work allows easy management of the goals and goal operations using this hierarchy. Another work also builds the type-level goal hierarchy using agent learned knowledge [51]. The algorithm gradually reduces the cost of the construction of the hierarchy over time. The authors implement their work in the Freeciv domain. Freeciv is a game to build civilizations and conquer others. Although the current work does not implement such goal-level hierarchy or life cycle, it is a very interesting avenue to pursue in the future.

The Wyatt and colleagues [108] also performs some goal management when multiple goals are present for the agent to perform. There is also work on goal delegation [44]. In this multi-agent work, the agent works on multi-agent goal management by delegating goals among multiple agents. Also, [76] uses goal reasoning to improve a sonar sensor's performance. These works on goal management are in line with our solution attempts. However, our thesis's current focus is on single agent's goal management using goal operations which is the main contrast to the above works.

Like all works, the GDA and goal reasoning also has certain limitations; [97] defines and addresses the limitations of GDA. For example, one problem includes the scalability of plan generation algorithms. The solution approach to tackling such problems uses a higher-level goal decomposing into several sub-goals using a Goal Decomposition Planner (GDP). This work is called Hierarchical Goal Networks (HGN). Although our current work does not implement HGN's, it is an interesting future direction to pursue. In addition to all the work we discussed, several other concepts helpful in goal reasoning are expectations and explanations.

Expectations help agents detect abnormal events that occur in the world. This aids the agent in changing its behavior according to its situation. Expectations are very helpful in the dynamic and partially observable worlds. The paper by Dannenhauer and his colleagues[29] is one of the few papers which tries to address the expectations of the agents about their mental cognition. Contributions of the paper are as follows:

- General formalism of the cognitive expectations
- Implementation of meta-cognitive expectations in an agent embodied within a cognitive architecture
- Ablation experimental results indicating that an agent equipped with metacognitive expectations outperforms agents operating without such capabilities

Authors perform experiments in the NBecons and arsonist domains. Some of the prior work on expectations include work done by Dannenhauer[31] where the assumption of applying expectation is applied to partially observable domains to collect data from the world. The authors term these types of expectations as informed expectations. The work also makes calculated planning and sensing costs. In addition, the work presented by Dannenhauer[30] focuses on generating expectations with the help of hierarchical planning principles. The Hierarchical Task Networks (HTN) [69], which are similar to HGN mentioned above are used in shop planners. Types of expectations presented in this work include:

- Immediate: consists of preconditions and effects
- State: state and plan; the agent expects the result of applying the plan to state
- Informed: resulting state after applying tasks to the current state should be defined first

The authors implement their work in Mars world and Arsonist domains. Other works on expectations include the work by Kurup et al., 2012[65], where the agent performs expectation-driven cognition. The agent generates expectations with the help of ACT-R's declarative memory, partial matching, and blending mechanisms. ACT-R is a cognitive architecture that has been around since the 1970s. The author chose a real-world domain for their implementation, pedestrian tracking, and behavior classification. The authors compare the idea and implementation of this work with the KNN classifier. One other work on expectations is by Ranasinghe and colleagues [88] is not entirely focused on expectations.

Still, it identifies surprises or unexpected actions in the world based on condition, action, and precondition. The agent should identify the precondition, and if the precondition is different or very far from the observation, it considers a surprise. They have developed an algorithm for surprise-based learning with rule splitting and some first-order predicate logic. Although, we do not use expectations explicitly. We use the work on expectations to identify anomalies for goal formulation operations. Whenever an anomaly occurs, the agent must explain the reasons behind the anomaly. Hence, we will now focus on some related work on explanations.

Explanations are instrumental to an agent because they allow them to understand and learn more about the world. [45, 46] presents the work on the use of case-based reasoning to explain an anomaly and to aid the agent to come up with a new goal if necessary. The authors develop this work further to aid in goal operations such as selection [47] and goal formulation [43]. Another work that implements explanations using a case-based system is called Meta-AQUA [26]. This work presents a taxonomy of failure cases and inputs them to the system as bias. After which, the agent tries to learn the success cases using the bias provided. Other works on an autonomous system that automatically detects and explains the discrepancies during execution include Williams[104], and Klenk[57]. Both Titan [104], and Kirk [56] choose their actions by tracking the system state using a declarative specification of the system behavior. In addition to the works mentioned, there is work on trying to understand the plan by explaining actions using abduction principles [73]. One more similar work includes learning the world through the surprises the agent encounters, this works presents a variant of the FOIL learning module, and it is implemented in ARTUE system [74]. [94] presents work in which the agent uses explanations to learn about the world. In this work, the author presents a case-based approach to allow the agent to learn and adapt to its environment better. Other works on the agent learning about the world based on the surprises it encounters use prediction rules to predict the results of the actions it does in an unknown world[88]. If the agent encounters a surprise, it tries to reason about it and learn about the world. In this thesis, we make use of implicit explanations when anomalies occur. In addition, in the context of smart agents, almost all of the work presented in expectation, explanation has monitoring inherently or explicitly. Therefore, let us now look at some related works in this regard.

Monitoring is also a very important action that helps the agent modify the model of the world and allows it to reason about the world when some unexpected event occurs. Pettersson [85] presents a survey on execution monitoring in the field of industrial control. A paper on reverse execution monitoring [12] assumes that the effects of the actions are already known to the agent and tries to reason about the action if the outcome is wrong. [101] presents the process of continuous diagnosis. [1] present work on monitoring the plan during plan execution and implementation. The authors extend this work to monitor the reasons to achieve the goals in [33]. Finally, work by Chernova et al., [12] tries to update the understanding of the agent using reverse plan monitoring while making the agent by extracting new features from the world. This thesis only implements the environmental monitoring implicitly. Apart from that, it does not implement any other kinds of monitors. However, it is an exciting venue to pursue in the future.

Another research area is the planning and execution of the work in either simulation or real-world settings. Almost many of the works discussed implement these operations. However, let us add two specific real-world implementations of such work. Research work that incorporates both control and cognitive level intelligence includes the *Control Architecture for Robotic Agent Command and Sensing (CARACaS)*, developed for autonomy in underwater platforms [53]. CARACaS uses intelligent decision-making to make the unmanned underwater agent robust. CARACaS uses three different components to make such decisions. The first is a behavioral engine which is the core of the system. The agent uses it in real-time for several generic behaviors of the agent. The second is a perception engine that produces maps. Agents use these maps for navigation purposes. Finally, a dynamic planner called Continuous Activity Scheduling Planning Execution and

Re-planning is used to handle goal-based planning and re-planning operations on-board. In this work, the burden is on the planner for goal prioritization and goal achievement. In addition, [78] uses a rule-based approach to integrate execution and planning.

Apart from all the topics discussed so far, we also argue that the agent must have a model of itself and the world to work robustly in a real-world setting. One of the early works on metacognition presented by the authors in the paper titled 'Perpetual self-aware cognitive agents' by Cox[20]. Some others working in a similar area include Wyatt[108], where research on self-understanding and self-extension was the focus. The authors develop an architecture for object detection and understanding using CAS architecture. They describe CAS and have presented some good experiments for their architecture. Some other works in the meta-reasoning include [52], [22], where the authors classify meta reasoning into several categories and introduce each category. One another work in self-understanding and how it will improve the agent is outlined in work by Murdock[75], which represents the drawbacks of case-based reasoning, proposes a new self-model, and tries to come up with a new plan whenever necessary. This work uses Task Method Knowledge (TMKL) agents for implementation purposes as they can adapt themselves to a set of transformations whenever necessary. The work presented by Flavall[38] is a theory paper that tries to classify the meta knowledge and organize it. [28] uses expectations to help the agent to build a model of itself. In our thesis, we implement goal change operation at the meta-level because the agent must reason about its own goals to change them.

While trying to understand the world, it is also important for the agent to develop a better representation. However, it should not ignore the costs of planning and sensing. We will now examine works that represent the world better while keeping sensing costs in mind. This would facilitate effective implementation of the goal operations. Golden et al., [48] presents the representation of the sensing actions using UWL (strips-based representation language) and ADL (classical representation framework). This work represents the knowledge goals. Other works that represent world models attempt to include GDA to sense actions have [32].

This work implements the sensing costs while operating in a partially observable world. [42] discusses the hierarchical representation of the capabilities of the agent to understand the world, acquire operations, and plan. This aids the agent in improving its reasoning. One other work which aids in planning when there is an incomplete representation about the actions sensed is [84]. The solution proposes to use the knowledge represented in first-order modal logic and the actions based on how they modify the knowledge states. In addition to the works presented, there is also work on representing the knowledge about the world using some philosophical ideas [71]. STRIPS[37], provides a representation of the world using a simple predicate-argument format. It also searches the goal space and employs a resolution theorem prover to answer some questions that the agent encounters. Baral[6] presents a new representation of states using three values. This is a different version of representation than using the classic Kripke structures. In, [2], the authors represent actions such that they can use them in both the natural language and problems solving fields of artificial intelligence. This paper also presents formalism about representation. Agents can also learn to construct observations, actions, and knowledge by using the radial interactionism [40]. This is useful in understanding the world; the agent obtains input data for the algorithm from sensorimeter readings of the system. There is also work on the reasoning that tries to reason about knowledge, action, and time in certain domains, which is presented in work by Patkos[83]. This is a formal work that makes use of event calculus. Finally, Shanahan[95] presents the use of abductive reasoning methods to construct the world models through the sensor inputs. Our thesis uses a descendant of STRIPS called PDDL in MIDCA for world representation.

Apart from the goal-based agents mentioned until now, one other type of agent is Belief-Desire-Intention(BDI) agents. Initially, these rationale agents are theorized and are presented in practice [89]. Later, the belief change operation is performed using a generalized update [9][10]. There is also work on belief extrapolation by Florence de Saint-Cyr and Jerome Lang[34] which tries to complete the incomplete belief sets from the data obtained through observations. The work provides many extrapolation operators. We think that the work on such agents will better integrate the GDA with real-world control architectures such as CARACaS mentioned above.

The basis for developing cognitive architectures is to understand the psychology of thinking in general. Hence the literature also focuses on some relevant psychology papers. One work tries to focus on the improvement of the knowledge of the agent through constructing the reality by narrating it [11]. The authors define the narrative in ten things: Narrative diachronicity, Particularity, Intentional state entailment, Hermeneutic composability, Canonicity and Breach, Referentiality, Genericness, Normativeness, Context sensitivity and negotiability, and narrative accrual. All these ten at once describe the narrative, and the constructed reality teaches us about the nature of reality built by the human mind. Furthermore, there is also work which investigates how humans construct reality using narration [11]. Michael Mateas and Phoebe Sengers[70] also present various systems that use narrative intelligence. Both represent the importance of narrative intelligence and present its applications. Work by Dennis E. Clayson[17] attempts to address the reason behind students overestimating their exams scores. However, this paper could not determine the exact reason for the overconfidence. Therefore, it lays out various possibilities for overestimation. Dunlosky et al., [35] present a work trying to improve the grades of the students by introducing an effective learning mechanism. There is also work that attempts to investigate the initial feeling of knowing in humans [90]. The long-term goal of this thesis is to build an agent capable of achieving goals and understanding the world similar to or better than an average human.

Conclusion and Future work

The central idea of the thesis is to address the knowledge gap present about the interaction of various goal operations in the research area. We developed a goal management strategy to bridge this gap. The goal management strategy considered three main factors to address multiple co-occurrences of goal operations. The factors include the negative effects of anomalies, goal types, and resource availability. We elaborate on each factor and create models to estimate all of them. We then translate the outputs of the models to a set of qualitative outputs. Finally, we verify if the outputs match any specific rule in MIDCA to output a decision. We implement the goal management strategy in the marine survey domain and the construction domain. We then test the performance of the strategy and provide an empirical evaluation. The results obtained depict the importance of such a strategy to the robust goal achievement of the agent.

In addition to the central idea, we also implement three individual goal operations: goal selection, goal change, and goal formulation. First, We implement the goal selection operation using FIFO, a cost-benefit analysis, and a greedy hill-climbing method. We implement the goal selection operation in the construction, restaurant, and marine survey domains. We also evaluate the individual importance of selection to the agent by obtaining results across all the mentioned domains above. The results show the individual importance of the goal selection operation. Next, we implement the goal change operation using predicate transformations. Specifically, we implement the generalization and specialization transformations. We implement the goal change operation in the marine survey and construction domains. We also evaluate the operation individually in the construction domain to study its importance. Finally, we implement the goal formulation operation using anomaly detection and explanations. We implement this operation in the mine clearance domain, the labor relations domain, the marine survey domain, and the construction domain. We evaluate the operation individually in the first two domains. In the future, we want to further this research with the following ideas.

- **Implement more qualitative factors:** We can extend our work to include other qualitative factors such as medium in addition to high and low. We could also include other goal types such as enjoyment goals and delta goals and study the resulting goal management strategy variations.

- **Implement other goal operations:** We can extend our work to include other goal operations such as goal delegation, goal monitoring, goal evaluation, and goal sharing. We present an approach to include goal delegation to the current goal management strategies in subsection 6.2, but extending this to others promises ever-increasing performance.

- **Include positive anomaly factors:** So far, the thesis has focused on the negative anomaly effects. In the future, we would like to study how the positive anomalies change the decisions and performance of an agent.

- **Include the work of multi-agent systems:** We can extend our work to multi-agent goal management through such high-level decisions. We implement work on multi-agent goal delegation to some extent [45].

- **Improve individual goal operations:** We can also work towards improving each goal operation. For example, the agent can implement goal change using argument transformations in addition to predicate transformations.

- **Embody the work:** We can work toward further embodying our research onto an intelligent physical system.

Bibliography

[1] Zohreh Alavi and Michael T Cox. Rational-based visual planning monitors. In *IJCAI*, pages 3968–3969, 2016.

[2] James F Allen. Towards a general theory of action and time. *Artificial intelligence*, 23(2):123–154, 1984.

[3] Erik M Altmann and Wayne D Gray. An integrated model of cognitive control in task switching. *Psychological review*, 115(3):602–639, 2008.

[4] John R Anderson, Michael Matessa, and Christian Lebiere. Act-r: A theory of higher level cognition and its relation to visual attention. *Human–Computer Interaction*, 12(4):439–462, 1997.

[5] John R Anderson and C Schunn. Implications of the act-r learning theory: No magic bullets. *Advances in instructional psychology, Educational design and cognitive science*, pages 1–33, 2000.

[6] Chitta Baral and Tran Cao Son. Approximate reasoning about actions in presence of sensing and incomplete information. In *ILPS*, volume 97, pages 387–401, 1997.

[7] Michael R Benjamin, Henrik Schmidt, Paul M Newman, and John J Leonard. Nested autonomy for unmanned marine vehicles with moos-ivp. *Journal of Field Robotics*, 27(6)(6):834–875, 2010.

[8] Ralph Bergmann. *Experience management: foundations, development methodology, and internet-based applications*, volume 2432. Springer, 2003.

[9] Craig Boutilier. Generalized update: Belief change in dynamic settings. In *IJCAI*, pages 1550–1556, 1995.

[10] Craig Boutilier. A unified model of qualitative belief change: A dynamical systems perspective. *Artificial Intelligence*, 98(1-2):281–316, 1998.

[11] Jerome Bruner. The narrative construction of reality. *Critical inquiry*, 18(1):1–21, 1991.

[12] Sonia Chernova, Elisabeth Crawford, and Manuela Veloso. Acquiring observation models through reverse plan monitoring. In *Portuguese Conference on Artificial Intelligence*, pages 410–421. Springer, 2005.

[13] Steve Chien, Benjamin Cichy, Ashley Davies, Daniel Tran, Gregg Rabideau, Rebecca Castano, Rob Sherwood, Dan Mandl, Stuart Frye, Seth Shulman, et al. An autonomous earth-observing sensorweb. *IEEE Intelligent Systems*, 20(3):16–24, 2005.

[14] Steve Chien, Rob Sherwood, Daniel Tran, Benjamin Cichy, Gregg Rabideau, Rebecca Castano, Ashley Davis, Dan Mandl, Bruce Trout, Seth Shulman, et al. Using autonomy flight software to improve science return on earth observing one. *Journal of Aerospace Computing, Information, and Communication*, 2(4):196–216, 2005.

[15] Dongkyu Choi. Reactive goal management in a cognitive architecture. *Cognitive Systems Research*, 12(3-4):293–308, 2011.

[16] Dongkyu Choi and Pat Langley. Evolution of the icarus cognitive architecture. *Cognitive Systems Research*, 48:25–38, 2018.

[17] Dennis E Clayson. Performance overconfidence: metacognitive effects or misplaced student expectations? *Journal of Marketing Education*, 27(2):122–129, 2005.

[18] Alex Coddington, Maria Fox, Jonathan Gough, Derek Long, and Ivan Serina. Madbot: A motivated and goal directed robot. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20(4), pages 1680–1681. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[19] Michael T Cox. Field review: Metacognition in computation: A selected research review. *Artificial intelligence*, 169(2):104–141, 2005.

[20] Michael T Cox. Perpetual self-aware cognitive agents. *AI magazine*, 28(1):32–32, 2007.

[21] Michael T Cox. Question-based problem recognition and goal-driven autonomy. In *Goal Reasoning: Papers from the ACS workshop*, pages 10–25, 2013.

[22] Michael T Cox, Zohreh Alavi, Dustin Dannenhauer, Vahid Eyorokon, Hector Munoz-Avila, and Don Perlis. Midca: A metacognitive, integrated dual-cycle architecture for self-regulated autonomy. In *Thirtieth AAAI Conference on Artificial Intelligence*, volume 5(1), pages 3712–3718, 2016.

[23] Michael T Cox, Dustin Dannenhauer, and Sravya Kondrakunta. Goal operations for cognitive systems. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 4385–4391, 2017.

[24] Michael T. Cox, Zahid Mohammad, Sravya Kondrakunta, Venkatsampath Raja Gogineni, Dustin Dannenhauer, and Othalia Larue. Computational metacognition. In *Proceedings of the Ninth Annual Conference on Advances in Cognitive Systems*, pages 1–20. Cognitive Systems Foundation, 2021.

[25] Michael T Cox, Tim Oates, and Don Perlis. Toward an integrated metacognitive architecture. In *2011 AAAI Fall Symposium Series*, pages 74–81, 2011.

[26] Michael T Cox and Ashwin Ram. Failure-driven learning as input bias. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pages 231–236. Erlbaum Hillsdale, NJ, 1994.

[27] Michael T Cox and Manuela M Veloso. Goal transformations in continuous planning. In *Proceedings of the 1998 AAAI fall symposium on distributed continual planning*, pages 23–30, 1998.

[28] Dustin Dannenhauer. *Self monitoring goal driven autonomy agents*. PhD thesis, University of Lehigh, 2017.

[29] Dustin Dannenhauer, Michael T Cox, and Hector Munoz Avila. Declarative metacognitive expectations for high-level cognition. In *Advances in Cognitive Systems*, volume 6, pages 231–250. Springer, 2018.

[30] Dustin Dannenhauer and Hector Munoz-Avila. Raising expectations in gda agents acting in dynamic environments. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 2241–2247. IJCAI Press, 2015.

[31] Dustin Dannenhauer, Hector Munoz-Avila, and Michael T Cox. Informed expectations to guide gda agents in partially observable environments. In *Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2493–2499. IJCAI Press, 2016.

[32] Dustin Dannenhauer, Hector Munoz-Avila, and Sravya Kondrakunta. Goal-driven autonomy agents with sensing costs. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[33] Zohreh Dannenhauer, Matthew Molineaux, and Michael T Cox. Explanation-based goal monitors for autonomous agents. In *Goal reasoning workshop*, 2019.

[34] Florence Dupin de Saint-Cyr and Jérôme Lang. Belief extrapolation (or how to reason about observations and unpredicted change). In *Proceedings of the Eights International Conference on Principles of Knowledge Representation and Reasoning*, pages 497–508. Morgan Kaufmann Publishers Inc., 2002.

[35] John Dunlosky, Katherine A Rawson, Elizabeth J Marsh, Mitchell J Nathan, and Daniel T Willingham. Improving students' learning with effective learning techniques: Promising directions from cognitive and educational psychology. *Psychological Science in the Public Interest*, 14(1):4–58, 2013.

[36] Catherine Edwards, Sungjin Cho, Fumin Zhang, and Sarah Fangman. Field and numerical studies to assess performance of acoustic telemetry collected by autonomous mobile platforms. Technical report, National Marine Sanctuaries Conservation Series, Silver Spring, MD, 2020.

[37] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[38] John H Flavell. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American psychologist*, 34(10):906–911, 1979.

[39] Howard Gardner. *Multiple intelligences: The theory in practice.* Basic books, 1983.

[40] Joseph Garnier, Olivier L Georgeon, and Amélie Cordier. Inferring actions and observations from interactions. In *ACS*, 2013.

[41] Malik Ghallab, Dana Nau, and Paolo Traverso. The actors view of automated planning and acting: A position paper. *Artificial Intelligence*, 208:1–17, 2014.

[42] Yolanda Gil and Jim Blythe. How can a structured representation of capabilities help in planning. In *Proceedings of the AAAI–Workshop on Representational Issues for Realworld Planning Systems*, 2000.

[43] Venkatsampath Raja Gogineni, Sravya Kondrakunta, Danielle Brown, Matthew Molineaux, and Michael T Cox. Probabilistic selection of case-based explanations in an underwater mine clearance domain. In *International Conference on Case-Based Reasoning*, pages 110–124. Springer, 2019.

[44] Venkatsampath Raja Gogineni, Sravya Kondrakunta, and Michael T. Cox. Multi-agent goal delegation. In *Proceedings of the 9th Goal Reasoning Workshop*, pages 1–13, 2021.

[45] Venkatsampath Raja Gogineni, Sravya Kondrakunta, Matthew Molineaux, and Michael T Cox. Application of case-based explanations to formulate goals in an unpredictable mine clearance domain. *International Conference on Case-Based Reasoning*, pages 42–51, 2018.

[46] Venkatsampath Raja Gogineni, Sravya Kondrakunta, Matthew Molineaux, and Michael T Cox. Case-based explanations and goal specific resource estimations. In *The Thirty-Third International Flairs Conference*, pages 407–412. AAAI Press, 2020.

[47] Venkatsampath Raja Gogineni, Sravya Kondrakunta, Matthew Molineaux, and Michael T Cox. Case-based explanations and goal specific resource estimations. In *Proceedings of Thirty-Third International Flairs Conference*, pages 407–412. AAAI, 2020.

[48] Keith Golden and Daniel Weld. Representing sensing actions: The middle ground revisited. *KR*, 96:174–185, 1996.

[49] Nick Hawes. A survey of motivation frameworks for intelligent systems. *Artificial Intelligence*, 175(5-6):1020–1036, 2011.

[50] Nick Hawes, Marc Hanheide, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Moritz Göbelbecker, Michael Brenner, Hendrik Zender, Pierre Lison, Ivana Kruijff-Korbayová, et al. Dora the explorer: A motivated robot. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1617–1618. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[51] Thomas R Hinrichs and Kenneth D Forbus. Beyond the rational player: Amortizing type-level goal hierarchies. In *Goal reasoning: Papers from the ACS workshop*, 2013.

[52] Eric Horvitz. *Metareasoning: Thinking about thinking.* MIT Press, 2011.

[53] Terry Huntsberger and Gail Woodward. Intelligent autonomy for unmanned surface and underwater vehicles. In *Proceedings of the OCEANS'11 MTS/IEEE KONA*, pages 1–10. IEEE, 2011.

[54] Matthew Iklé, Ben Goertzel, Misgana Bayetta, George Sellman, Comfort Cover, Jennifer Allgeier, Robert Smith, Morris Sowards, Dylan Schuldberg, Man Hin Leung, et al. Using tononi phi to measure consciousness of a cognitive system while reading and conversing. *AAAI spring symposium: towards conscious AI systems*, 2019.

[55] Benjamin Johnson, Mark Roberts, Thomas Apker, and David W Aha. Goal reasoning with informative expectations. In *Planning and Robotics: Papers from the ICAPS Workshop. London, UK: AAAI*, 2016.

[56] Phil Kim, Brian C Williams, and Mark Abramson. Executing reactive, model-based programs through graph-based temporal planning. In *IJCAI*, pages 487–493, 2001.

[57] Matthew Klenk, Matt Molineaux, and David W Aha. Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, 29(2):187–206, 2013.

[58] Sravya Kondrakunta. Implementation and evaluation of goal selection in a cognitive architecture. Master's thesis, Wright State University, 2017.

[59] Sravya Kondrakunta and Michael T Cox. Autonomous goal selection operations for agent-based architectures. In *Working Notes of the 2017 IJCAI Goal Reasoning Workshop. IJCAI*, 2017.

[60] Sravya Kondrakunta and Michael T Cox. Autonomous goal selection operations for agent-based architectures. In *Proceedings from Advances in Artificial Intelligence and Applied Cognitive Computing*, page in press, Las Vegas, NV, 2021. Springer.

[61] Sravya Kondrakunta, Venkatsampath Raja Gogineni, Danielle Brown, Matt Molineaux, and Michael T Cox. Problem recognition, explanation and goal formulation. *Advances in Cognititve Systems*, 2019.

[62] Sravya Kondrakunta, Venkatsampath Raja Gogineni, and Michael T. Cox. Agent goal management using goal operations. In *Proceedings of the 9th Goal Reasoning Workshop*, pages 1–15, 2021.

[63] Sravya Kondrakunta, Venkatsampath Raja Gogineni, Michael T. Cox, Demetris Coleman, Xiaobo Tan, Tony Lin, Mengxue Hou, Fumin Zhang, Frank McQuarrie, and Catherine R. Edwards. The rational selection of goal operations and the integration of search strategies with goal-driven autonomy. In *Proceedings of the Ninth Annual Conference on Advances in Cognitive Systems*, pages 1–20. Cognitive Systems Foundation, 2021.

[64] Sravya Kondrakunta, Venkatsampath Raja Gogineni, Matt Molineaux, Hector Munoz-Avila, Martin Oxenham, and Michael T Cox. Toward problem recognition, explanation and goal formulation. *Sixth Goal Reasoning Workshop*, 2018.

[65] Unmesh Kurup, Christian Lebiere, Anthony Stentz, and Martial Hebert. Using expectations to drive cognitive behavior. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, volume 26, pages 221–227, 2012.

[66] John E Laird. *The Soar cognitive architecture*. MIT press, 2019.

[67] Pat Langley and Dongkyu Choi. A unified cognitive architecture for physical agents. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21(2), page 1469. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[68] Michael Lewis. Designing for human-agent interaction. *AI Magazine*, 19(2):66–79, 1998.

[69] Amnon Lotem and Dana S Nau. New advances in graphhtn: Identifying independent subproblems in large htn domains. In *AIPS*, pages 206–215, 2000.

[70] Michael Mateas and Phoebe Sengers. *Narrative intelligence*. John Benjamins Publishing, 2003.

[71] John McCarthy and Patrick J Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Readings in artificial intelligence*, pages 431–450. Elsevier, 1981.

[72] Frank McQuarrie, C. Brock Woodson, and Catherine Edwards. Modeling acoustic telemetry detection ranges in a shallow coastal environment. In *Proceedings of the 14th ACM International Conference on Underwater Networks and Systems (WUWNet'21)*, Guangdong, China, in press. ACM.

[73] Ben Leon Meadows, Pat Langley, and Miranda Jane Emery. Seeing beyond shadows: Incremental abductive reasoning for plan understanding. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[74] Matthew Molineaux and David W Aha. Learning unknown event models. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, volume 28, pages 395–401. AAAI Press, 2014.

[75] J William Murdock and Ashok K Goel. Meta-case-based reasoning: self-improvement through self-understanding. *Journal of Experimental & Theoretical Artificial Intelligence*, 20(1):1–36, 2008.

[76] Jill K Nelson and Steven Schoenecker. Goal driven autonomy as a model for reasoning in cognitive active sonar. In *Proceedings of the 2018 IEEE 10th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 1–5, Sheffield, United Kingdom, 2018. IEEE.

[77] Thomas O Nelson. Metamemory: A theoretical framework and new findings. In *Psychology of learning and motivation*, volume 26, pages 125–173. Elsevier, 1990.

[78] Tim Niemueller, Till Hofmann, and Gerhard Lakemeyer. Goal reasoning in the clips executive for integrated planning and execution. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 754–763, Berkeley, CA, 2019. AAAI Press.

[79] Donald A Norman and Stephen W Draper. *User centered system design: New perspectives on human-computer interaction*. CRC Press, 1986.

[80] Una-May O'Reilly and Franz Oppacher. Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing. In *International Conference on Parallel Problem Solving from Nature*, pages 397–406, Jerusalem, Israel, 1994. Springer.

[81] Matt Paisner, Michael Maynord, Michael T Cox, and Don Perlis. Goal-driven autonomy in dynamic environments. In *Goal Reasoning: Papers from the ACS Workshop*, volume 79, 2013.

[82] Matthew Paisner, Michael Cox, Michael Maynord, and Don Perlis. Goal-driven autonomy for cognitive systems. In *Proceedings of the 36th Annual Meeting of the Cognitive Science Society*, volume 36, page 2085–2090, 2014.

[83] Theodore Patkos and Dimitris Plexousakis. Reasoning with knowledge, action and time in dynamic and uncertain domains. In *Twenty-First International Joint Conference on Artificial Intelligence*, pages 885–890, 2009.

[84] Ronald PA Petrick and Fahiem Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *AIPS*, volume 2, pages 212–222, 2002.

[85] Ola Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53(2):73–88, 2005.

[86] Jay Powell, Matthew Molineaux, and David William Aha. Active and interactive discovery of goal selection knowledge. In *Twenty-Fourth International FLAIRS Conference*, pages 413–418, 2011.

[87] Gregg Rabideau, Steve Chien, and David McLaren. Tractable goal selection with oversubscribed resources. *Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space*, 2009.

[88] Nadeesha Ranasinghe and Wei-Min Shen. Surprise-based learning for developmental robotics. In *2008 ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems (LAB-RS)*, pages 65–70. IEEE, 2008.

[89] Anand S Rao, Michael P Georgeff, et al. Bdi agents: from theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995.

[90] Lynne M Reder and Frank E Ritter. What determines initial feeling of knowing? familiarity with question terms, not with the answer. *Journal of Experimental Psychology: Learning, memory, and cognition*, 18(3):435–451, 1992.

[91] Frank E Ritter, Michael Schoelles, Laura Cousino Klein, and Sue E Kase. Modeling the range of performance on the serial subtraction task. In *Proceedings of the 8th International Conference on Cognitive Modeling*, pages 299–304, 2007.

[92] Mark Roberts, Swaroop Vattam, Ronald Alford, Bryan Auslander, Tom Apker, Benjamin Johnson, and David W Aha. Goal reasoning to coordinate robotic teams for disaster relief. In *Proceedings of ICAPS-15 PlanRob Workshop*, pages 127–138. Citeseer, 2015.

[93] Roger C Schank and Robert P Abelson. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press, 2013.

[94] Roger C Schank and David B Leake. Creativity and learning in a case-based explainer. *Artificial intelligence*, 40(1-3):353–385, 1989.

[95] Murray Shanahan. Robotics and the common sense informatic situation'. In *ECAI*, pages 684–688. PITMAN, 1996.

[96] Daniel Shapiro, Pat Langley, David J Stracuzzi, Dana Nau, Alan Fern, Ray Mooney, Peter Stone, and Pedro Domingos. Transfer learning in integrated cognitive systems. Technical report, Institute for The Study of Learning and Expertise, Palo Alto, CA, 2010.

[97] Vikas Shivashankar, Ron Alford UMD EDU, Ugur Kuter, and Dana Nau. Hierarchical goal networks and goal-driven autonomy: Going where ai planning meets goal reasoning. In *Goal Reasoning: Papers from the ACS Workshop*, 2013.

[98] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[99] Ron Sun. The importance of cognitive architectures: An analysis based on clarion. *Journal of Experimental & Theoretical Artificial Intelligence*, 19(2):159–193, 2007.

[100] Ron Sun. The clarion cognitive architecture: Toward a comprehensive theory of. *The Oxford handbook of cognitive science*, pages 117–133, 2016.

[101] Michael Thielscher. A theory of dynamic diagnosis. *Electronic Transactions on Artificial Intelligence*, 1(4):73–104, 1997.

[102] J Gregory Trafton, Laura M Hiatt, Anthony M Harrison, Franklin P Tamborello, Sangeet S Khemlani, and Alan C Schultz. Act-r/e: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction*, 2(1):30–55, 2013.

[103] Martina Troesch, Faiz Mirza, Kyle Hughes, Ansel Rothstein-Dowden, Robert Bocchino, Amanda Donner, Martin Feather, Benjamin Smith, Lorraine Fesq, Brian Barker, and Brian Campuzano. Mexec: An onboard integrated planning and execution approach for spacecraft commanding. In *Workshop on Integrated Execution (IntEx)/Goal Reasoning (GR), International Conference on Automated Planning and Scheduling (ICAPS IntEx/GR 2020)*, 2020.

[104] Brian C Williams, Michel D Ingham, Seung H Chung, and Paul H Elliott. Model-based programming of intelligent embedded systems and robotic space explorers. *Proceedings of the IEEE*, 91(1):212–237, 2003.

[105] Mark Wilson, Bryan Auslander, Benjamin Johnson, Thomas Apker, James McMahon, and David W Aha. Towards applying goal autonomy for vehicle control. Technical report, Naval Research Lab Washington Dc Navy Center For Applied Research, 2013.

[106] Mark A Wilson, James McMahon, Artur Wolek, David W Aha, and Brian H Houston. Goal reasoning for autonomous underwater vehicles: Responding to unexpected agents. *AI Communications*, 31(2)(2):151–166, 2018.

[107] Mark A Wilson, Matthew Molineaux, and David W Aha. Domain-independent heuristics for goal formulation. In *The Twenty-Sixth International FLAIRS Conference*, pages 160 – 165, 2013.

[108] Jeremy L Wyatt, Alper Aydemir, Michael Brenner, Marc Hanheide, Nick Hawes, Patric Jensfelt, Matej Kristan, Geert-Jan M Kruijff, Pierre Lison, Andrzej Pronobis, et al. Self-understanding and self-extension: A systems and representational approach. *IEEE Transactions on Autonomous Mental Development*, 2(4):282–303, 2010.

Appendix A

The Marine Survey Domain: Details on Numeric Values

This appendix describes all the numeric values used in this thesis for the marine survey domain. Specifically, we present the values for the goal management strategy. The goal management strategy (see chapter 6) uses three classes of factors: the anomaly effects, goal priority, and the resource availability. Out of the three classes of factors the anomaly effects relies heavily on numeric calculations. Therefore, we focus on the anomaly effects in this chapter.

The anomaly effects class has three sub factors: the negative effects on agent's goals; the negative effects on the agent's health; and the anomaly repetition. Among the three sub factors, the agent uses a knowledge base for the first sub factor, hence it does not have numeric values. However, the second and third sub factors use numeric values that are elaborated below:

We measure the health in terms of battery life of the agent. The specific factors we consider to calculate the negative effects on the battery life in this domain are: number of goals achieved, remora attacks, blockades, and flow. We calculate the over all cumulative function using equation (A.1). Here H is the total value, n is four and h_i is the individual sub factor (goals, remora, blockade, and flow).

$$H = \Sigma_{i=1}^{n}(h_i) \tag{A.1}$$

First, the number of total goals in this domain are 25. In addition, there are two types of goal, brief survey, and deep survey. Brief survey uses ten units of battery and deep survey uses 20 units of battery. We assume that type goals linearly effect A.2 the agent's health. Furthermore, we give a weight of 0.1 to both types of goals.

$$h_1 = h_{goals} = \Sigma_{y=1}^{25}(h_{goal\_number\_y}) = \Sigma_{y=1}^{25}(x_y * g_y) \tag{A.2}$$

For example, if the agent performs only a brief search on all 25 goals. The amount of the health reduced is given by, $h_1 = h_{goals} = \Sigma_{y=1}^{25}(0.1 * 10) = 25$.

Second, the we consider the remora attack as a quadratic function. We provide the function in the equation A.3; Where x is the battery affected by the remora attack and where n is the number of remora attacks.

$$h_2 = h_{remora} = \Sigma_{n=1}^{m} x^n \tag{A.3}$$

For now, we consider the value of x to be 5. Therefore, for one remora ($n = 1$) attack the health affected would be, $h_2 = h_{remora} = \Sigma_{n=1}^{1} 5^1 = 5$. The threshold value to classify the negative effect as high is 4. Therefore, the smart agent reacts to every remora attack.

We do not consider that the obstacles or the flow to directly effects the health of the agent. Hence, $h_3 = h_4 = 0$.

Finally the overall cumulative function in this domain is given by the following equation. The equations, depicts a scenario with all 25 goals briefly surveyed and with one remora attack. At any given point, the health of the agent must not change more than 4 threshold points in this domain.

$$H = \Sigma_{i=1}^{4}(h_i) = h_1 + h_2 + h_3 + h_4$$

$$H = h_{goals} + h_{remora} + h_{blockade} + h_{flow}$$

$$H = \Sigma_{y=1}^{25}(h_{goal\_number\_y}) + \Sigma_{n=1}^{m}x^n + 0 + 0$$

$$H = \Sigma_{y=1}^{25}(x_y * g_y) + \Sigma_{n=1}^{1}x^n$$

$$H = \Sigma_{y=1}^{25}(0.1 * 10) + \Sigma_{n=1}^{1}5^1$$

$$H = 25 + 5 = 30$$

The above set of equations conclude the calculation on negative effects on agent's health. Next, we consider the anomaly repetition sub factor A.4, where e is the Euler's constant, λ is the mean for specific time interval (10 units), and x is the expected number of anomalies. We, find the P values for several x values to plot the bell distribution of the poisson function. We then consider the peak of the bell curve, find the number of anomaly occurrences, and compare it to an expert set threshold of 3.

$$P\left(x\right) = \frac{e^{-\lambda}\lambda^{x}}{x!} \tag{A.4}$$

Appendix B

The Construction Domain: Details on Numeric Values

This appendix describes all the numeric values used in this thesis for the construction domain. Specifically, we present the values for the goal management strategy. The goal management strategy (see chapter 6) uses three classes of factors: the anomaly effects, goal priority, and the resource availability. Out of the three classes of factors the anomaly effects relies heavily on numeric calculations. Therefore, we focus on the anomaly effects in this chapter. The anomaly effects class has three sub factors: the negative effects on agent's goals; the negative effects on the agent's health; and the anomaly repetition. Among the three sub factors, the agent uses a knowledge base for the first sub factor, hence it does not have numeric values. However, the second and third sub factors use numeric values that are elaborated below:

We measure the health in terms of time (amount of time agents works without heating its engine) for the agent. The specific factors we consider to calculate the negative effects on its total time in this domain are: number of goals achieved, arsonist, thief. We calculate the over all cumulative function using equation (B.1). Here H is the total value, n is three and h_i is the individual sub factor (goals, arsonist, and thief).

$$H = \Sigma_{i=1}^{n}(h_i) \tag{B.1}$$

First, the number of total goals in this domain vary according to the goal set an agent generates. In addition, there are two types of goal, stable-tower, and wobbly-tower. stable-tower uses two units of time and wobbly-tower uses one unit of time. We assume that type goals linearly effect B.2 the agent's health. Furthermore, we give a weight of the goal corresponding to its height value. Example, a tower of height 1 will have a weight of 0.1, tower or height 2 will have a weight of 0.2. The maximum height of a tower is the agent generates is seven.

$$h_1 = h_{goals} = \Sigma_{y=1}^{n}(h_{goal\_number\_y}) = \Sigma_{y=1}^{n}(x_y * g_y) \tag{B.2}$$

For example, if the agent constructs only five stable-towers with height of two each. The amount of the health reduced is given by, $h_1 = h_{goals} = \Sigma_{y=1}^{5}(0.2 * 2) = 2$.

Second, the we consider the arsonist attack as an exponential function. We provide the function in the equation B.3; Where e is Euler's constant and t is the time at which agent discovers fire (starts at 1).

$$h_2 = h_{fire} = e^t \tag{B.3}$$

For one time unit ($t = 1$) the health affected by the fire would be, $h_2 = h_{fire} = e^1 = 2.71$. The threshold value to classify the negative effect as high is 2. Therefore, the smart agent reacts to every fire event. We do not consider the thief to effect the agent's health. Therefore, $h_3 = 0$

Finally, the cumulative function in this domain for a set of five sturdy goal with height of 2 each, and one fire is given by the set of equations below:

$$H = \Sigma_{i=1}^{3}(h_i) = h_1 + h_2 + h_3$$

$$H = h_{goals} + h_{fire} + h_{thief}$$

$$H = \Sigma_{y=1}^{n}(h_{goal\_number\_y}) + e^t + 0$$

$$H = \Sigma_{y=1}^{5}(x_y * g_y) + e^1$$

$$H = \Sigma_{y=1}^{25}(0.2 * 2) + 2.7$$

$$H = 2 + 2.7 = 4.7$$

The above set of equations conclude the calculation on negative effects on agent's health. Next, we consider the anomaly repetition sub factor B.4, where e is the Euler's constant, λ is the mean for specific time interval (3 units), and x is the expected number of anomalies. We, find the P values for several x values to plot the bell distribution of the poisson function. We then consider the peak of the bell curve, find the number of anomaly occurrences, and compare it to an expert set threshold of 3.

$$P(x) = \frac{e^{-\lambda}\lambda^x}{x!} \tag{B.4}$$

In addition, to the factors. We perform goal selection using a cost-benefit method. In this method, the benefit for stacking each block in sturdy tower is 2; and for each block in wobbly tower is 1. The cost is just the amount of time given in chapter 4.