University of Strathclyde Glasgow

The development of a planning system for a swarm of

RoboWasps

A thesis submitted in partial fulfilment for a degree in
Master of Science in Advanced Mechanical Engineering with Aerospace

Supervisor: Prof Massimiliano Vasile

April 2020

Dushyant Khatri
201894465

Dushyant Khatri

Dushyant Khatri

Glasgow, Scotland, April 2020

ABSTRACT

Nature inspires us in many ways. Many scientific theories and engineering innovations are recognised from what nature reveals. Along with this, Technology redefines the world we live in, right from depths of oceans to outer space. Combining the principles from Nature and Technology, this project contributes towards the development of an automated planning system for a swarm of RoboWasps. Inspired from the design of RoboBees and SubCULTron, and in alignment to projects such as the development of social drones by WASP-HS, the project begins by recognising significant attributes and applications of these projects and explores a unique approach of developing a programming-based planning system in implementing towards one of their applications. To begin the implementation, the classical planning model is selected over the five planning models studied during literature study, and the PDDL framework is chosen as an implementation tool for performing analyses. Once the literature study is completed and a logical justification is provided towards the methodology, the research analysis begins by considering a single-agent problem within a 4×4 matrix, gradually progressing to a multiagent problem of four RoboWasps within a 7×7 state space. The simulations are extended further by introducing an organisational structure, and then, by introducing the interference of external agents. With the results obtained, these are evaluated alongside the principles of Swarm Robotics and mathematical criterions of cooperation and collaboration of multiagent systems, with justifications provided on the idea of RoboWasps, the methodology and the analysis performed. Further recommendations are made of including dynamic and nondeterministic factors into the environment and improving the analyses by adding agents' attributes such as beliefs and learning from experience. The project, further, indicates other applications of RoboWasps, such as for smart indoor monitoring, in construction sector, in healthcare, and in hazardous and restrictive spaces. The project concludes with opportunities and challenges that would be faced by RoboWasps, should such an idea be progressed towards a pragmatic development.

ACKNOWLEDGEMENTS

People say that the journey is sometimes more pleasing than the destination itself. This holds true in completing this project; It has been an immense pleasure working under the supervision of one of the most amazing minds in the field of study, Prof Massimiliano Vasile. Not just he has been supportive in developing my technical skills but has also been a great inspiration in enhancing myself on my research skills and solving complex problems with logical reasoning. I would like to sincerely thank him for all his support and believing in my potential to deliver this project with no prior knowledge to the subject.

I am immensely grateful to the entire team of USEV and StrathLoop as this has enabled me in complementing my theoretical understanding of this project to hands-on experience, which resulted in my increased motivation in exploring the electronics field of study.

Last but not least, I would like to thank my parents for trusting in my ambitions and for always being there in those dark points in my life as a light of motivation. Would they have not been financially and emotionally supportive for my second degree, none of this would have been possible.

TABLE OF CONTENTS

L I S T   O F   F I G U R E S

LIST OF TABLES

NOMENCLATURE

Classical Planning

S	Finite and discrete state space
s_0	Initial state
S_G	Set of goal states, $S_G \subseteq S$
a	Action, within a set of actions $a \in A(s)$
$A(s)$	Set of actions
$f(a, s)$	Deterministic state transition function
s'	Resulting state for applying action a
$c(a, s)$	Action cost
δ	Deterministic state transition function
γ	Resulting state function from applying action a

Hierarchical Planning

S	Finite and discrete state space
s_0	Initial state
C	Set of compound task names
A	Set of actions
M	Set of methods
m	Element within the set of methods $m \in M$
tn	Task network
T	Set of unique identifiers
$\prec$	Strict partial order on the identifiers, $\prec \subseteq T \times T$
α	Function that maps identifiers to task names, $\alpha: T \rightarrow A \cup C$
δ	Deterministic state transition function

Contingent Planning

F	Fluent symbols
O	Set of actions

I	Set of clauses over F defining the initial situation
G	Goal situation
L	Literal
x	Set of Boolean variables, $x = \langle x_1, x_2, x_3, \ldots \rangle$
ρ	Environment program
ρ'	Remaining environment program
δ	Agent program
δ'	Remaining agent program

Conformant Planning

$\mathcal{X}$	Finite set of state variables
$\mathcal{V}$	Range function over $\mathcal{X}$
$\mathcal{S}$	Set of states
$\mathcal{A}$	Set of actions
$\mathcal{R}$	Transition relation, $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$
$\mathcal{D}$	Planning domain
$\mathcal{I}$	Initial state
$\mathcal{G}$	Goal state
π	Plan, that is the solution to the contingent planning problem

Epistemic Planning

$Prop$	Set of propositional variables
Agt	Set of agents
OBS	Set of visibility operator
OBS^*	Set of all sequences of visibility operator
S_i	Any sequence of visibility operator
ATM	Propositional variable, called an Atom
σ	Visibility operator, $\sigma \in OBS$
p	Propositional variable, $p \in Prop$
α	Sequence of visibility operator, $\alpha \in ATM$
φ	Agent has a knowledge of φ
$K_{i\varphi}$	Implies that agent i has the knowledge of φ
$\mathcal{L}_k$	Language of multiagent epistemic logic

$\mathcal{L}_{kC}$	Common knowledge shared by every agent $i \in Agt$
M	Mathematical representation of an epistemic model
W	Domain or a finite set of worlds called states
R	Accessibility relation
V	Atomic variation

Coordination and Collaboration of RoboWasps

α	Agent
β	Agent
G_α	Goals corresponding to agent α
G_β	Goals corresponding to agent β
I_α	Initial states for agent α
I_β	Initial states for agent β
D_α	Set of actions for agent α
D_β	Set of actions for agent β
P_α	Plans corresponding to agent α
P_β	Plans corresponding to agent β
P_α^α	Plans corresponding to agent α solely
P_β^β	Plans corresponding to agent β solely
P_α^β	Plans corresponding to agent α involving agent β
P_β^α	Plans corresponding to agent β involving agent α

Durative Actions

A_i	Set of actions
F	Set of facts used to describe the world
$d_{min}(a)$	Minimum duration
$d_{max}(a)$	Maximum duration
$cond_s(a)$	Start condition
$cond_i(a)$	Invariant condition
$cond_e(a)$	End condition
$eff_s(a)$	Start effect
$eff_e(a)$	End effect
$add_s(a)$	Add effect at start

$add_e(a)$	Add effect at end
$del_s(a)$	Delete effect at start
$del_e(a)$	Delete effect at end

Minimal Social Law Algorithm

P_i	Predefined plan
s_k^i	k^{th} state to be visited during execution
k	Variable representation of a specific state
s_o^i	Initial state of agent i
$\langle a_1^i, a_2^i, \ldots, a_{ti}^i \rangle$	Sequences of action executed by the agent i
$A_i'(s)$	Set of forbidden action at s
B_k^j	Set of reachable states at k
C_0	Single initial configuration
τ	Tuple to represent multiple agents in their respective states
L_1	Literal

STRIPS planning

$\mathcal{P}$	Set of ground atomic formulae, also known as conditions
$\mathcal{O}$	Set of operators
pr_+	Positive pre-conditions
pr_-	Negative pre-conditions
po_+	Positive post-conditions
po_-	Negative post-conditions
$\mathcal{I}$	Initial state
$\mathcal{G}$	Goal state
$\mathcal{G}_+$	Positive goals
$\mathcal{G}_-$	Negative goals

ABBREVIATIONS

Abbreviation	Expansion	Definition
ADL	**Action Description Language**	See section 2.6.2, page 27
AI	**Artificial Intelligence**	Ability of a computer-controlled robot to perform tasks commonly associated with intelligent beings [1].
a.k.a	**Also known as**	
ANNs	**Artificial Neural Networks**	A structure to simulate the network of neurons that make up a human brain so that the computer will be able to learn things and make decisions in a human-like manner [2].
DL	**Deep Learning**	A subset of machine learning where that uses many layers of artificial neurons stacked one on top of the other for identifying complex features within the input data and solving complex real-world problems [3].
DNNs	**Deep Neural Networks**	Same as Deep Learning [4].
GA	**Genetic Algorithm**	GAs form a powerful metaheuristic that mimics processes from the Theory of Evolution to establish search algorithms by defining algorithmic analogues of biological concepts such as reproduction, crossover and mutation [5].
IDE	**Integrated Development Environment**	An IDE contains of a source code editor, build automation tools and a debugger.
iff	**If and only if**	
MAV	**Micro Aerial Vehicle**	(here) Air-borne MEMS.

MAS	**Multi-Agent System**	A Multi-Agent System is defined as an organisation of possibly heterogeneous and self-motivated agents that interact with each other [6]. Agents in an MAS could coordinate with each other to accomplish a common mission [6].
MEMS	**Micro-Electro-Mechanical Systems**	Electro-Mechanical Systems that are designed and constructed in micro-scale.
ML	**Machine Learning**	Discipline concerned with implementation of computer software that can learn autonomously. Common approaches within ML are ANNs and GAs [7].
NI	**Natural Intelligence**	A mental attribute that enables humans and/or natural beings to learn from experience, adapt to changes, understand and handle abstract concepts, and use knowledge to manipulate one's environment [8].
PDDL	**Planning Domain Definition Language**	See section 2.6.3, page 28
RL	**Reinforcement Learning**	A type of machine learning in which a computer learns to perform a task through repeated interactions with a dynamic environment. This trial-and-error learning approach enables the computer to make a series of decisions without human intervention and without being explicitly programmed to perform the task [9].
SAR	**Search and Rescue**	The search for and provision of aid to people who are in distress or imminent danger.
SI	**Swarm Intelligence**	See section 1.2.4.2, page 9
SR	**Swarm Robots**	
STRIPS	**Stanford Research Institute Problem Solver**	See section 2.6.1, page 25

CHAPTER ONE

INTRODUCTION TO THE PROJECT

1.1. Motivation

Nature inspires us in many ways. Many scientific discoveries and technological innovations have arisen from being inspired by what the nature reveals – the falling of an apple from a tree led to the discovery of gravitation [10], the bath-tub experiment by Archimedes led to the discovery of buoyancy [11], which is the main principle in marine vehicles, and studying the flight of an eagle led to further improvements for drag reduction in aircrafts [12, 13]. A common root in all these examples was being inspired by natural processes and applying them to a real-world complex problem.

On the other hand, Technology has transformed our lives, right from waking up through to going back to bed and beyond, at every level where humans can interact, from the depths of oceans to outer space. One of the prominent perks of technology has been is that it offers a helping hand in accomplishing tasks swiftly, conveniently and accurately. The consistent growth of technological solutions has redefined Engineering as we know it.

Combining Nature and Technology to form the core foundations, this project addresses advanced technological design factors of wasp-sized drones together with the elements of the nature, such as the swarm of wasps, and recognises a crucial subsystem in exploring, researching and analysing towards the development of a planning system of a swarm of wasp-sized drones. These wasp-sized drones are described as *RoboWasps* throughout the project.

The 'RoboWasp' design is developed due to its anticipated prominent applications. Along with its promising potential, it faces 'real world' problems in its design, for instance, with a significant reduction in size from conventional drones to wasp-sized drones, there are various apparent challenges for their development, for instance, the propulsion system and its efficiency. Such 'real world' problems are solved with introducing 'real world' design solutions such as bio-inspiration. If a biological wasp was to be compared on this attribute, wasps propel by flapping their wings in an oscillatory motion rather than a rotatory motion observed in the conventional drones [14]. Another aspect could be the proper functioning and achieving tasks by RoboWasps, which puts an efficient micro-control system at the core of the development, by considering the biological coordination of a wasp, for example the escape mechanism by a wasp when an imminent danger is recognised and its reaction mechanism of stinging to defend itself. It would not be inappropriate to say that the micro-control system is the heart of the entire design as it would ensure a robust functioning of a RoboWasp. With options of an individual RoboWasp design to be scrutinised, the potential expands by introducing the option of multiple RoboWasps working as a team to deliver common or conflicting tasks. Different subsystems were recognised by linking the anticipated potential of such wasp-sized drones, in an individual and in a swarm capacity. The subsystems of main interest were primarily classified into:

- Aerodynamics and Propulsion Systems
- Structural Analysis and use of advanced materials
- Communication System
- Micro-control system design and architecture

As all of these subsystems would contribute towards a holistic design of RoboWasps, the initial project proposal incorporated all of these subsystems with a preliminary project schedule provided. However, during project planning and further analysis on other project management aspects such as mitigating unforeseen risk factors that might cause significant delays, it was recognised that the time limitation would be a major disadvantage and could potentially result in a compromise on the quality of thesis and anticipated learning. Therefore, one of the four subsystems had to be selected. As the communication subsystem was an advanced topic and seemed achievable in the scheduled timeframe with aspects involving research and industrial practices anticipated to be progressing, this was selected to be the core subsystem of this project. Within the communication subsystem, different domains were explored and analysed - such as robust data transfer amongst the swarm entities, decentralised decision-making and planning system, intelligent task delegation amongst the swarm entities, and image recognition and processing combining different perspectives recorded by the swarm entities. Critically evaluating these aspects with respect to four criterions – the timeline, the current intensity of research, the pre-requisite understanding and the contribution towards the industry, the objectives of the project are defined accordingly in section 1.3.

1.2. A brief overview

With the nature of the project scrutinised in the proposal stage, the project started with recognising the current trends, technologies, research and anticipated projects within the engineering sector. During initial research and feasibility analysis on whether such a project is valid in the present time, many of the projects were explored in understanding the working principle of Swarm Robotics (SR) and their potential in contributing towards the engineering sectors. These acted as sources of inspiration to develop the concept of a swarm of RoboWasps towards Artificial Intelligence (AI). Their features that can be related to the design of RoboWasps are discussed briefly.

1.2.1. Sources of inspiration

Among the different projects that were explored in relating them towards the development of RoboWasps, different possibilities of improving their current design were explored. With many potential abilities demonstrated by these systems within communication, their contribution individually and as a team was researched and related to the design of RoboWasps. Below are two such projects that acted as sources of inspiration in designing the swarm of RoboWasps.

1.2.1.1. RoboBees

"Real-life transformers are being inspired by insects, not cars."
- Williams, A. [15]

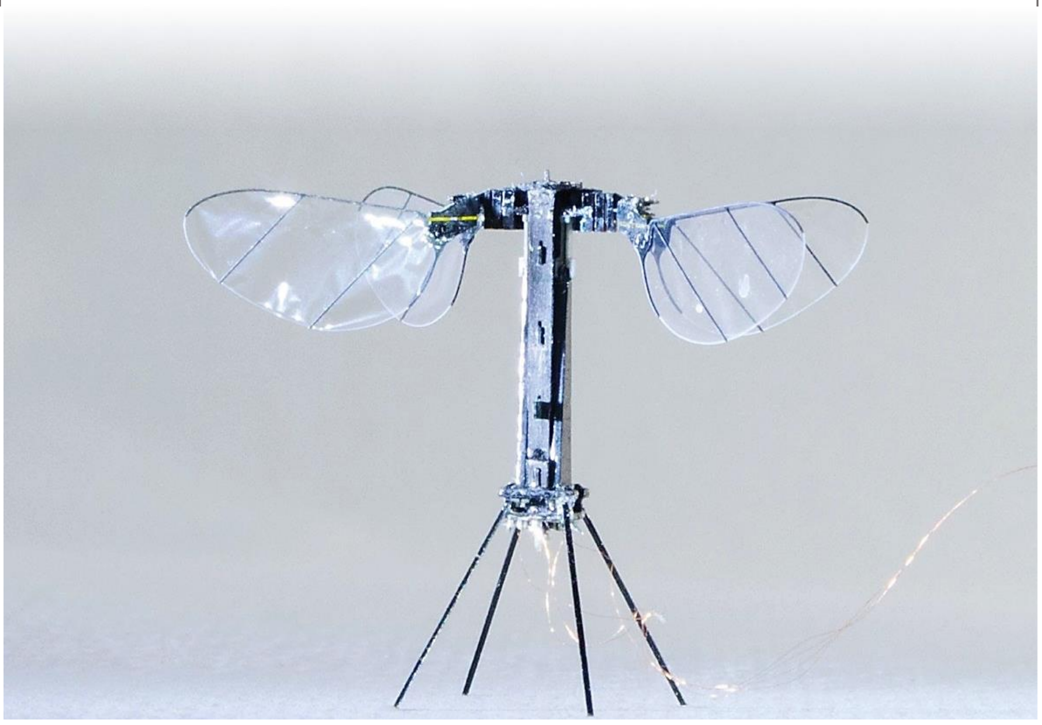


Figure 1 - RoboBee

Inspired by the biology of a bee, researchers at the Wyss Institute at Harvard University are developing RoboBees, a man-made Micro-Aerial Vehicle (MAV) that could potentially be used in crop pollination, search and rescue (SAR) missions, surveillance, as well as high-resolution weather, climate and environmental monitoring **[16]**. A RoboBee would measure about half the size of a paper clip, weighs less than one-tenth of a gram, and flies using artificial muscles compromised of materials that contract when a voltage is applied **[17]**. Its impressive characteristics and its inspiration from nature has attracted further research towards it. A PhD thesis by Landgraf **[18]** explores another interesting characteristic of bees, i.e. their communication among each other by waggle dancing, to be applied to RoboBees. Not just their communication, their cooperative task delegation is very much a burning matter to research upon in the present time. The swarm of RoboBees was originally modelled for performing artificial pollination, where they are capable in communicating with each other and sensing their surroundings, however, their collaborative interaction can further be extended to performing SAR operations [19].

All of these attributes of RoboBees sound very impressive and one could imagine how these would redefine the world we live in by many ways. However, all of these attributes can be viewed as challenges to be tackled in developing the design. Their incredibly light weight, small size and unconventional flight mechanism would require a vigorous resolution. The material used to build the robot must be strong, yet lightweight; Human-engineered actuators and batteries must replicate the power and energy density of a biological tissue; And most importantly, the mind-bogglingly complex algorithms for control and sensing must replicate that used by insects in maintaining steady flight and in manoeuvring [20].

With an almost identical physical design to RoboBees, RoboWasps are likely to face similar challenges in their design. However, when these challenged are overcome, we obtain an engineering micro-marvel to tackle real-world problems with ease.

1.2.1.2. SubCULTron and CoCoRo

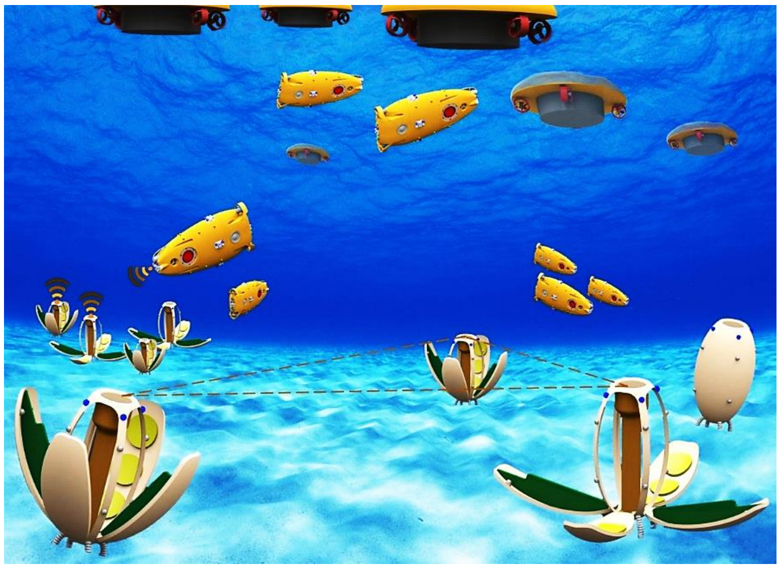


Figure 2 - An Artist's impression of subCULTron [21]

Extending the design of a single RoboWasp to a swarm of RoboWasps, such a possibility and its useful applications were explored by being inspired from agents intended in the exploration of the underwater world. With one of the applications in ecological data acquisition such as temperature, oxygen concentration, salinity, alkalinity, turbulence/currents, particle-density and turbidity, and cloudiness [22], A Horizon 2020-funded project was developed within the Artificial Intelligence Lab of Karl-Franzens

University Graz called subCULTron [23]. The project head, Dr. Mag. Thomas Schmikl [24], indicates that these underwater autonomous robots would not just carry out operations for which they are programmed or trained, but would also develop social skills such as learning how to communicate and interact with other robots as well as with the surrounding environment. A similar project, CoCoRo, shows the flocking behaviour as observed in school of fishes in the underwater world.

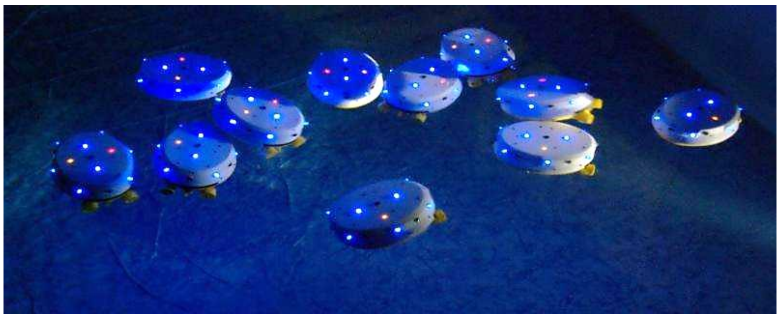


Figure 3 - The CoCoRo Project [25]

Combining the advantages of RoboBees to be small in size with their ability to fly and their ability to be used within restricted spaces, along with the advantages of subCULTron and their ability to interact with one another to complete tasks together, form the sources of inspiration in developing a swarm of RoboWasps.

1.2.2. Artificial Intelligence

With a consistent growth of Artificial Intelligence (AI), tasks performed by humans are now taken over by machines that exhibit intelligent characteristics. AI lends a helping hand in the present day in many forms – ask Siri to play your favourite tunes and she would follow [26], tailored product recommendations miraculously occurring for you to choose from on an e-commerce website [27], or even Google Maps helping you to navigate through the fastest or the shortest route to your destination – are very few examples of AI in everyday life [28]. However, one would wonder on how a machine would think and reason in performing tasks that humans used to perform a few years ago. As a matter of fact, the foundation of AI is strongly derived by how humans think and reason in performing these tasks with their Natural Intelligence (NI) ability [29].

1.2.3. Principles of AI

To recognise the potential of AI towards the development of RoboWasps, the principles of AI are explored. Considering the same examples of AI – Siri, the e-commerce website and Google Maps, all of these follow different approaches to reach the desired solution. For instance, Siri* recognises your voice, converts it into data and searches through for the music in the available database. The key principles observed here are voice recognition, voice conversion to machine language and database searching [30]. The e-commerce website tracks and records the statistical data of your recent searches and websites visited to recognise your interests, and then displays the relevant products tailored to you. The key principles observed here are statistical data acquisition and clustering untagged data [31]. Google Maps takes input of your current location and the destination, analyses on the factors - such as traffic lights, or accidents/traffic lags recorded - that would affect the result, i.e., the least time or the least distance, and works out the optimum path to your destination. The key principle here is manipulating huge data and statistically predict the likelihood of delays from past data recorded whilst new data being recorded at every instant [32]. Recognising that these examples belong to different key principles in performing and completing tasks, one could say that different techniques and understanding of different branches of AI are applied in achieving them. With the broad application and the diverse principles of AI observed in all the three examples, AI can be perceived as a broad field of study consisting of different branches leading to specific applications.

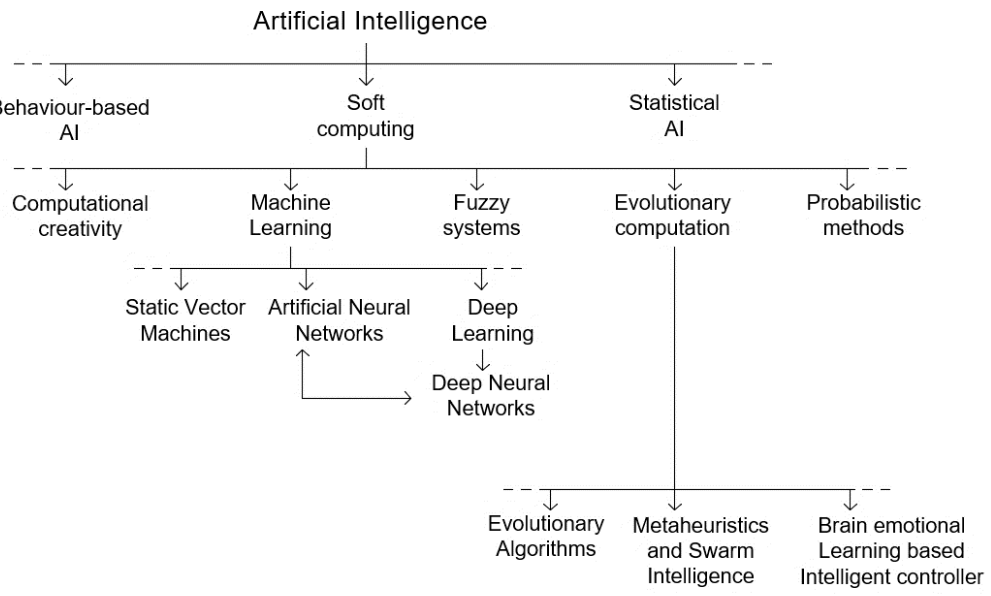


Figure 4 - Classification of Artificial Intelligence (restricted to the project objectives)

It should be noted from Figure 4 that because of the broad classification of AI, only relevant branches, that are anticipated to be applied to the RoboWasp design; These are purely indicated in order to understand the hierarchy and the interrelation of different branches. In further sections, relevant

* https://www.apple.com/uk/siri/

pathways are briefly discussed in understanding their relevance to the project and in defining the objectives of the project.

1.2.4. Different approaches in developing RoboWasps

Two primary approaches were considered further in developing a swarm of RoboWasps - by the means of Machine Learning (ML) or by the means of Swarm Intelligence (SI).

1.2.4.1. Machine Learning

The field of ML is experiencing exponential growth today [33]. The reason to this increased popularity is their rewarding abilities. The tasks that were once performed by humans and could not be performed by machines, ML contributes effectively in training models to perform tasks such as image recognition, text generation or playing games [34]. ML can inculcate four categories of learning of machines – supervised, unsupervised, semi-supervised, and Reinforcement Learning (RL). For a swarm of RoboWasps, where they are expected to build upon their experience with respect to the environment they operate in, working towards dealing with spontaneous events and triggering actions with no human intervention in such situations, ML can prove very helpful in developing such an attribute. The working of ML is based on the idea that it uses the theory of statistics in building mathematical tasks because the core task is to make inference from the model. With the first stage of training, where algorithms are used to solve optimisation problems, to store and process big data, the second stage involves working towards inference efficiency once the model is learned [35]. ML, in itself, can be considered a broad field of study, and is therefore, further branched to sub-fields [35]. From Figure 4, Deep Learning (DL) and Artificial Neural Networks (ANNs) are the two primary sub-fields considered for further discussion.

ANNs go a step further when compared to traditional ML methods on their suitability for handling complex and noisy data sets and their ability to define a good generalisation between input features and output values [36]. To extend these functionalities of ANNs further to DL, various advantages can be listed when compared to traditional ML approaches. In many ways, DL outperforms traditional ML methods by a huge margin in several fields, especially within computer vision, speech recognition, natural language processing, and time series [37]. With DL, more and more complex features can be learned as the layers in the deep neural network increase. Because of this automatic feature-learning, it reduces a significant feature-engineering time, which is a time-consuming activity in traditional ML approaches [37, p.8]. DL works best for unstructured data in the form of images, text, speech, sensor data, and so forth, which when applied to solving pragmatic problems would revolutionise different sectors, such as health care, banking, aviation, e-commerce, and so on [37].

When the RoboWasp problem is considered at hand, the construction of the environment in which a swarm of RoboWasps would operate would be difficult to replicate when ANNs and DL is explored with no prior development. As these exhibit advanced solutions such as optimising the problem further so that the RoboWasps are able to learn within the environment, this could mean jumping straight in the deeper end. As it is important to build upon the problem of RoboWasp in a fundamental environment in the first place, this approach is disregarded at this stage.

1.2.4.2. Swarm Intelligence

> "Fish got to school, birds got to flock, and robots, it seems, got to swarm."
>
> - Chen, A. [38]

A team makes wonders happen and accomplishes more than an individual. Taking this thought forward, a team of robots, or as they are called the "swarm" of robots, can be utilised in accomplishing tasks to reduce individual effort and save significant amount of time. This can be supported by the observation of an experiment of a swarm of drones in exploring a disaster site, where the swarm was much quicker than that possible by a single drone in capturing images and was useful in redundancy, especially when the operation within a single drone fails, yet it is captured by another drone successfully [39]. Thinking of other applications of SR similar to the above experiment of SAR operations, they could be used in multi-sector engineering tasks. Nature reveals this teamwork that leads to achieving results faster – from army ants linking themselves to form rafts and bridges [38] to honeybees collaborating in building their hive [40] through to a flock of migratory birds flying in unison in reducing drag during flight [41]. It can be said by these examples that SI would supplement the pre-established foundation on AI, and if applied to RoboWasps, would improve their operability within a swarm.

Inspired by this teamwork from nature, SI algorithms such as the Ant Colony Optimisation, Particle Swarm Optimisation, and Discrete Firefly Algorithms, enable in various applications within the field of Optimisation [42]. Although the same justification could be provided as for not proceeding with ANNs and DL, the SI methods could be used in establishing criterions to determine the accuracy and effectiveness of the working of RoboWasps within a swarm. These are indicated in section 1.4.

1.2.4.3. Automated Planning System

The planning within AI is rather an application on autonomous systems. With the motivation of combining Nature and Technology in developing a swarm of RoboWasps featuring an intelligent property of planning within an environment, this project unveils another form of AI that enables the swarm performing certain intelligent tasks without human intervention. It is observed that when humans use machines to complete a set of tasks, they use NI in planning the sequence of the tasks to be completed and decide upon it [29]. Planning within autonomous agents by adopting the principles of AI can be categorised in three approaches:

(1) Programming-based approaches: In these approaches, the programmer makes the decision and inputs the sequence of the tasks to be performed, which is then completed by the agent(s). In other words, human intervention is central to such approaches, and therefore, all the burden is on the programmer. A major drawback of such approaches is the anticipation that human error may lead to an inaccurate decision, and therefore, an inaccurate result [43].

(2) Learning-based approaches: In these approaches, the programmer no longer plays a role in arrangement of the sequence of tasks, or in any of the planning and decision-making. Instead, the agent(s) is responsible and the result is induced from experience as in Reinforcement Learning (RL) [43].

(3) Model-based approaches: In these approaches, learning from experience is not involved, however, the result is derived automatically from a model of the actions, sensors and goals. A major drawback for such approaches are that they face a computational problem of solving the model - a problem that is intractable even for the simplest models where information is complete and actions are deterministic [43].

Recognising the planning system to be a good start point rather than starting with learning of RoboWasps, the programming-based approach is considered appropriate in developing a planning system swarm of RoboWasps within an environment. In building upon the problem, learning-based approaches can be introduced in further automation and an independent planning by RoboWasps with no intervention by the programmer (a.k.a domain designer).

1.3. Objectives of the project

Following the fundamental concepts being revised, the objectives of this project can be listed as follows:

(i) To study and critically evaluate on an appropriate planning system for a swarm of RoboWasps.

(ii) To implement the evaluated planning system on a planning problem considering problems from the real-world applications.

1.4. Criterions in fulfilling the objectives

With additional criterions introduced later in this project after theoretical concepts are established (such as the cooperation and collaboration of RoboWasps, discussed in section 2.5, page 22), it is expected that the swarm of RoboWasps would demonstrate the principles of SR, and therefore, the results obtained would be evaluated with these criterions.

The principles of SR [44] are –

- **Robustness** – Robustness is defined as fault tolerance and fail-safety and it is achieved by massive redundancy and the avoidance of single-point of failures.
- **Flexibility** – Flexibility is defined as the ability of the swarm to adapt to a wide range of tasks as this would eliminate the specialisation of hardware for a section of the swarm.
- **Scalability** – Scalability is defined as the ability of the swarm to maintain its function while increasing its size without the need to redefine the way its parts interact.

1.5. Scope of the project

With the project objectives described, the project ignores all other subsystems and sub-domains that would contribute towards the overall 'RoboWasp' design and any subsystem that is believed to be affecting the performance of the current subsystem is deemed to be accurately functional for the purpose of performing an effective analysis. Additionally, further assumptions are described when developing a methodology in Chapter 3, once the literature review is completed, as this would enable reasoning timeframes and the intensity of the problem with available resources.

1.6. Thesis Outline

The thesis outline is structured as follows:

Chapter 2 involves exploring preliminary concepts and performing an extensive literature study on concepts relating to the project objectives. The chapter begins with putting in perspective the fundamental concepts and follows through by extending to the current, state-of-the-art technologies in defining the problem, with relevant resources in combining theory to practice.

Chapter 3 takes forward the concepts from Chapter 2 to structure a methodological approach to designing a domain and structuring corresponding problems. This chapter involves scrutinising different potential approaches by evaluating the strengths and drawbacks of these approaches in relation to fulfilling the project objectives.

Chapter 4 explains how the analyses were performed in achieving the project objectives, abiding to the methodology defined in Chapter 3. This chapter supplements the explanation with graphical representations of the state space and flow charts to give the reader a clear picture of the problem.

Chapter 5 involves generating results for all the cases described in Chapter 4. These are demonstrated with updated graphical representations of the state space and the underlying plans generated. The chapter continues with interpreting the results achieved with reference to the literature research from Chapter 2.

Chapter 6 begins by justifying on the approach, the methodology, the analyses and the results obtained. The chapter proceeds with indicating the strengths and weaknesses of the results achieved and the overall progress of the project for the given timeframe.

Chapter 7 highlights the industrial applications of the swarm of RoboWasps in reference to the results obtained. In addition to the industrial practices indicated, the chapter ends with further research and development towards the development of RoboWasps, with a note to the growing potential of RoboWasps.

CHAPTER TWO

PRELIMINARY CONCEPTS AND RESEARCH

2.1. Chapter overview

In this chapter, an extensive literature study and theoretical concepts on the development of the project and a comprehensive understanding of multiagent planning is provided. This chapter begins with the nature of domain where the automated planning would exist by RoboWasps, along with an extensive literature study of different planning models. In addition to this, different pathways in implementing the planning model are explored. The chapter concludes by summarising on useful indications that are taken forward for design and analysis of the planning system.

2.2. Fundamental concepts from the literature

All through the previous chapter and to put the reader into the perspective of the development of the planning system for a swarm of RoboWasps, there are implicit indications to what attributes are represented by this swarm. To relate these attributes, one can regard the swarm of RoboWasps as a Multi-Agent System (MAS). The planning system for MAS can be regarded as Multi-Agent Planning (MAP).

An agent is described as anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators [45, 46], or a more relevant definition to this project, an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives [47, p.15]. Agents could be categorised primarily by their number count, i.e., single-agent systems and multi-agent systems. The difference between a single-agent system and a multi-agent system is explicitly described in numerous sources of the literature, to an extent that even if different tasks are performed by multiple agents simultaneously and yet these are controlled by a central source, this would be categorised as a single-agent problem [46, p.425]. Therefore, one of the primary attributes demonstrated by RoboWasps would be their decentralised plan execution and their ability to work on different tasks simultaneously.

Planning within MAS could differ significantly according to the environment they operate. MAS can plan for a domain-specific or a domain-independent environment, domain-specific being where specific representations or techniques are used in planning, whereas domain-independent planning is where generic representations or techniques are used [48]. Although there may be some structure in planning for certain and specific environments, it becomes challenging in utilising a generic planning system in multiple environments [29]. An environment can be certain and known, or uncertain. Selecting an appropriate environment is a primary step towards developing a planning system for RoboWasps, and this is explored further in this chapter by relating to different planning models and implementation techniques.

2.3. Domain-independent and Domain-specific planning

In the earlier section, domain-independent and domain-specific planning were discussed briefly. In both of these classifications, To begin with whether the planning system for a swarm of RoboWasps are to be considered in a broader range of domain or towards a specific domain, one would have to differentiate among the two classifications on their advantages and drawbacks, keeping in mind the project objectives and the limitations. One of the crucial advantages of domain-independent planning over domain-specific is that it adopts planning from first principles and can be applied in multiple scenarios and environments – this saves considerable time and hassle in manually applying specific planning techniques to such scenarios [48]. RoboWasps are anticipated to be benefitting across a spectrum of engineering sectors, and therefore, the domain-independent planning system would be ideal. However, there are considerable drawbacks of a domain-independent planning system. With adopting the first principle, a domain designer is expected to have an extensive understanding. These are developed with experience in designing domains and their corresponding characteristics over time. With the given timescales, this would not be achievable. Therefore, the implementation would consider a specific domain to begin with. Despite of the analysis performed targeting a specific domain, many engineering sectors are explored in which the potential solution of this project could be applied, which are briefly explained in section 7.1.

2.4. Planning models

There are different approaches described in the literature when planning for MAS. Although all of these approaches may be valid to applying in some form of MAS, it is important to recognise an appropriate approach, or to utilise characteristics of multiple approaches, for the design and analysis of RoboWasps.

The type of environment in which RoboWasps would operate is of prime importance and forms a defining factor of the planning and decision-making system within the swarm. An environment that is to be explored by RoboWasps can primarily be classified into two categories – open and closed. An open environment is classed as the most complex general class of environment and exhibits inaccessible, non-deterministic, dynamic and continuous environment characteristics [47, p.20]. Therefore, it could be construed that a closed environment would exhibit opposite characteristics of the former, i.e., accessible, deterministic, static and discrete, and therefore, this would be categorised to be the simplest class of environment. The class of models within planning can be classified as – classical, hierarchical, conformant, contingent and epistemic planning. Among these models, the classical and hierarchical planning are the most popular ones [49] because of their simplistic applicability in a broad range of problems. Among the two, classical planning is considered to have more sophisticated techniques than the latter [50]. On the other hand, planning under uncertainty can be accomplished by two of the planning models primarily - contingent and conformant planning [51]. These are explored objectively in the following sections.

2.4.1. Classical Planning

A classical planning problem defines a classical model in compact form through a set of variables [52]. Mauro and Lukáš [53] define a classical model along with its components in their conference paper as:

> "In classical planning, a domain model is specified via sets of predicates and planning operators. A problem model is specified via objects (that are substituted for free variables in predicates and operators), an initial state and a set of goal atoms. A planning task consists of a domain model and a problem instance. A solution plan for a planning task is a sequence of actions in the plan (starting in the initial state) resulting a state in which all goal atoms are true." [53]

The components defined within a classical planning problem are a domain model, a problem model, a planning task and a solution plan. To express a classical problem mathematically in terms of the parameters that define the environment, Geffner and Bonet [43] explicitly list these in their book as –

- A finite and discrete state space, S
- A known initial state, s_0
- A non-empty set, $S_G \subseteq S$, of goal states
- Actions, $A(s) \subseteq A$, applicable in each state, $s \in S$
- A deterministic state transition function, $f(a, s)$, such that $s' = f(a, s)$ stands for state resulting for applying action a in s, $a \in A(s)$
- Positive action costs, $c(a, s)$

Therefore, a classical planning model can be expressed as a 6-tuple $\Upsilon = \langle S, s_0, S_G, A, f, c \rangle$. A similar tuple is defined by Höller *et al.* [49] with modifications to two of the above parameters –

- The deterministic state transition function is replaced by a triple $(prec, add, del)$ of functions that define the pre-conditions and the effect of actions, each mapping actions to a set of state variables $f: A \rightarrow 2^S$. This transition function is denoted by δ.
- An action, a, is applicable when the preconditions are contained in the current state, s, i.e., when $prec(a) \subseteq s$. When it is applicable, the state resulting from the application is defined by the function $\gamma: A \times 2^S \rightarrow 2^S$ with $\gamma(a, s) = (s \setminus del(a) \cup add(a))$.

Additionally, the action cost function has been eliminated in this modification. With these changes, the planning model can now be expressed is reduced to a 5-tuple $\Upsilon = \langle S, s_0, S_G, A, \delta \rangle$.

It is to be noted that due to the nature of the parameters that form the tuple, such as the *discrete* and finite state space S and the *deterministic* state transition function δ, the classical planning model would ideally be applicable to a closed environment. This is confirmed by the definition provided by Ghallab *et al.* [54, p.56] in their book, as the planning model to be imposing simplifying restrictions, namely that it be finite, fully observable, deterministic and static. With this premise, Kuter *et al.* [55] list different algorithms in adopting classical planners towards solving non-deterministic planning problems. This makes it possible to develop a planning system on some characteristics of an open environment through classical methods.

One of the typical examples for a classical planning model is of a gripper robot to have four fundamental set of actions for a goal of moving balls from room A to room B – to pick up a ball from room A, move to room B, put the ball down, and return to room A [56]. These set of actions are iterative until all the balls from room A are moved to room B. In such an environment, there is no uncertainty factor involved. There is a room A and a room B for certain. There is a defined number of balls in room A, which is defined as the initial state of the problem. The goal state is that all of these balls are in room B. Therefore, the only plan that remains would be to iteratively move the balls from room A to room B. This certainty in such a problem would ensure that the plan would have the same sequence of action every time it is executed.

2.4.2. Hierarchical Planning

The composition of hierarchical planning resembles the hierarchical organisational structure seen in most firms - there are several team members led by a set of team leaders, who are supervised by an Assistant Manager, who is monitored by a Manager. This planning model is expressed in the form of a Hierarchical Task Network (HTN), which is a partially-ordered multiset of task names, each representing an activity to be accomplished [57]. Höller *et al.* [49] mathematically define an HTN planning problem as a

7-tuple $\Upsilon = \langle S, C, A, M, s_0, tn, \delta \rangle$. S, A, s_0, and δ are the same as defined in section 2.4.1 for classical planning. C is a set of compound task names. tn is a task network expressed as a triple $(T, \prec, \alpha)$, and M is a set of methods, where $m \in M$. When the triple $(T, \prec, \alpha)$ is incorporated within the 7-tuple, the corresponding tuple would contain 9 parameters. This makes the approach more complex, although their functioning being closely accurate to the classical planning model.

A hierarchical planning structure can prove advantageous in many ways with useful real-world applications. Dempster *et al.* [58] highlight that hierarchical planning is advantageous in reducing complexity by breaking a problem into subproblems which enable in simplifying results and in coping with uncertainty. Additionally, it offers a significant advantage of applications in the real world [59] - for instance in user-centric planning assistance [60-62] and companion systems for human assistance [63].

With major advantages such as enabling simplified results, there are prominent disadvantages of adopting an HTN planning model, such as its difficulty in providing a complete domain knowledge with a complete and a correct set of HTN methods for every task [64]. One of the proposed approaches by Shivashankar *et al.* [64, p.2384] in coping with this drawback is adopting a modification of the HTN planning, GoDel, which expands to Goal Decomposition with landmarks, where the performance improves despite of little domain knowledge. Another related solution in addressing this drawback is modifying the HTN planning to a goal-based hierarchical planning formalism, HGN planning.

Assuming that such modifications may result in including an additional element in the analysis, and therefore, an updated, hybrid planning model can be derived by combining the concepts known from Partial-Order-Casual-Link (POCL) planning [65, 66]. However, despite of this, not being able to provide a correct set of HTN methods for every task in the Hierarchical planning model is seen as a significant drawback. Having difficulty in providing the correct set of HTN methods could lead to an inaccurate analysis, and therefore, an incorrect result. Combining this with the fact that despite of multiple parameters that define the environment, the accuracy is close, or sometimes, not as sophisticated as the good old classical planning model, this adds to the complexity of defining a planning model than a simpler approach of classical planning.

Similar to the classical planning approach, when the nature of the parameters is carefully noted, the hierarchical planning model would ideally be applicable to the closed environment. However, with the modifications present in tackling uncertainties within a domain, although there has not been adequate evidence in the literature, this planning model can also be applied within an open environment to some extent.

2.4.3. Contingent Planning

A contingent planning model can be defined as set of tasks for generating a conditional plan given the uncertainty about the initial state and action effects, also with the ability to observe some aspects of the current world state [67]. Albore *et al.* [68] define the contingent planning model as a tuple $P = \langle F, O, I, G \rangle$, where F stands for fluent symbols, O for the actions, I for the set of clauses over F defining the initial situation, and $G \subseteq F$ is the goal situation. A normal action a is defined to be having preconditions given by a set of fluent literals and L is a literal. As the appropriate environment to use contingent planning has the uncertainty factor, one potential solutions in utilising this planning model would be to introduce stochastic approaches to problem-solving. Majercik and Littman [69] develop a paradigm considering probabilistic planning with a stochastic satisfiability framework. In this approach, the probabilistic planning involves a probabilistic initial state, which is specified by a set of decision trees, one of each proposition. A proposition within a domain is a finite set of distinct result of `True` or `False` at a given (discrete) time. The stochastic satisfiability involves adopting a formula of Boolean variables, where it could be determined whether there is a value for x_1, such that for random values of x_2 (either 0 or 1 with equal probability), there exists a value for x_3, $x = \langle x_1, x_2, x_3, \dots \rangle$ being a set of Boolean variables. Another significant development of a contingent planning model is where the agents' tasks and behaviours are expressed as high-level nondeterministic programs within an Agent Programming Language (APL). Lespérance *et al.* [70] use ConGolog [71] as their APL in their implementation, in which they define the agent's task and behaviour of the agents in the environment as the nondeterministic concurrent programs. Apart from the APL of efficient use in the construction of a nondeterministic domain, the primitives of the model described by them are –

(i) $EnvTrans(\langle \rho, s \rangle, \langle \rho', s' \rangle)$: The agent considers it possible that the environment program ρ in state s can make a transition to state s' with the program ρ' remaining.

(ii) $AgtTrans(\langle \delta, s \rangle, \langle \delta', s' \rangle)$: The agent knows that the agent program δ in state s can make a transition to state s' with the program δ' remaining.

(iii) $AgtTrans(\langle \delta, s \rangle)$: The agent knows that the agent program δ can terminate in state s.

2.4.4. Conformant Planning

Conformant planning is regarded as a special case of contingent planning [67, 72]. Both of these planning models have many similarities, so much so that the conformant planning model is also mathematically expressed as a 4-tuple, just as the contingent planning [73]. However, a closely related to the classical planning domain, yet with a few additions, a nondeterministic planning domain $\mathcal{D}$ is defined as a tuple $\langle \mathcal{X}, \mathcal{V}, \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ [74], where –

- $\mathcal{X}$ is a finite set of state variables
- $\mathcal{V}$ is a range function over $\mathcal{X}$, such that $\mathcal{V}(x)$ is a finite, non-empty set for every variable $x \in \mathcal{X}$
- $\mathcal{S}$ is a set of states, a state being a function over $\mathcal{X}$ that associates to each variable $x \in \mathcal{X}$ an element in $\mathcal{V}(x)$
- $\mathcal{A}$ is a set of actions
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation

Also, a conformant planning problem is defined as a triple $\langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{D}$ is a planning domain, $\mathcal{I}$ and $\mathcal{G}$ are non-empty set of states, called the set of initial states and the set of goal states, respectively [74]; And therefore, a plan π is a solution to a conformant planning problem $\langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ iff –

- It is applicable in $\mathcal{I}$, and
- Every run of π from $\mathcal{I}$ has its final state in $\mathcal{G}$

Considering these definitions, it can be indicated that although main applications of conformant planning are suggested to be within a nondeterministic environment, its roots arise from a classical planning problem, with the similarities that exist in defining the parameters and their function in the planning model.

2.4.5. Epistemic Planning

Epistemic planning is seen as less of a model, but more of a form of planning, for its characteristics indicated below. Epistemic planning is considered as a combination of automated planning with epistemic logic, relying on concepts, ideas and solutions; In the context of MAS planning, this form of planning is particularly helpful in where a successful coordination and collaboration can be expected when agents are able to reason about the knowledge, uncertainty and capabilities of the other agents [75]. If this model were to be used in this project, such a planning would not just enable the application of RoboWasps across different environments, including where uncertainties exist, but would also incorporate the learning attribute during planning, decision-making, and eventually, completing the tasks. Bolander and Andersen [76] describe epistemic logic as a more expressive and a significantly better founded method for representing knowledge and ignorance about the environment.

One of the prominent approaches in implementing the epistemic planning model described by Bolander [75] is the Dynamic Epistemic Logic (DEL) approach, where the initial step starts with the most classical framework for planning, STRIPS, and then is expanded stepwise and generalised until eventually reaching a complete MAP based on DEL. Each of these steps are based on the need to be able to formalise specific planning scenarios. The DEL approach can prove very helpful and can result in an efficient and a structured planning by iterative updating of various attributes of the agents, such as their beliefs. Baltag and Renne [77] define and distinguish between static and dynamic agent beliefs in the context of the DEL approach. Another important notion that the DEL approach accounts for is the knowledge captured by the agents. Pacuit [78] explains that the DEL approach is to formalise a substantial theory of knowledge, in which the agents' knowledge is defined in terms of more primitive concepts.

Cooper *et al.* [79] mathematically express the definition of an epistemic logic, $EL\text{-}O^S$, as a fragment of DEL. These consist of the following components –

- A countable set of propositional variables, $Prop = \langle p_1, p_2, \dots \rangle$
- A finite set of agents, $Agt = \langle 1, 2, \dots, n \rangle$
- A set of visibility operator, $OBS = \langle \boldsymbol{S}_i : i \in Agt \rangle$
- A set of all sequences of visibility operators, OBS^*, where $\langle \sigma, \sigma', \dots \rangle \in OBS^*$
- Any sequence of visibility operators $\boldsymbol{S}_i$ followed by a propositional variable called an atom. These can be expressed mathematically as, $ATM = \langle \sigma p : \sigma \in OBS^*, p \in Prop \rangle$. Also, $\langle \alpha, \alpha', \beta, \beta' \dots \rangle \in ATM$
- Modal operators that capture knowledge K_i, and therefore, $K_{i\varphi}$ would imply that agent i has the knowledge of φ

The language of $EL\text{-}O^S$ is defined by the following grammar –

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_{i\varphi}$$

An epistemic model [76, 80], $\mathcal{L}_k(Prop, Agt)$ and $\mathcal{L}_{kC}(Prop, Agt)$, is mathematically expressed as a triple $M = (W, R, V)$, where –

- W: domain or a finite set of worlds called states.
- $R: Agt \to 2^{W \times W}$ assigns an accessibility relation R_i to each agent $i \in Agt$. (All accessibility relations are equivalence relations).
- $V: Prop \to 2^W$ assigns a set of worlds to each atomic variation
- $\mathcal{L}_k$: language of multiagent epistemic logic
- $\mathcal{L}_{kC}$: common knowledge shared by every agent $i \in Agt$

Such a model is called the Kripke model, which enables describing the agents' abilities to distinguish among actions [81]. This can prove very helpful in context to developing a swarm of RoboWasps, especially to distinguish between what actions is the RoboWasp equipped to perform within a domain.

2.5. Coordination and collaboration of RoboWasps

In the previous section, various planning models were discussed and summarised in order to understand on which environment should the RoboWasps be addressed to. When considering a MAS in comparison to the design of a single-agent system, an added element to devise is the interaction between the agents towards their goals. The agreement, disagreement, cooperation or negotiation – or any other kind of interaction represent a decentralised, multiagent problem-solving ability [82] towards achieving the goals.

The interactions of MAS would depend on the nature of the goal(s) to be achieved. This could be primarily categorised as common goals and conflicting goals, which would then cause a set of various other interactions. Towards achieving a common goal, the agents work together, unlike towards achieving a conflicting goal, the agents either must work against each other or must establish a consensus by means of negotiation. As from the discussions so far, it can be interpreted that the swarm of RoboWasps would work together in achieving a common goal, and therefore, conflicting goals are considered out of scope of this project. Dimopoulos and Moraitis [83] list down the conditions of a plan generated that satisfy coordination and cooperation between MAS towards common goals. These are:

Given two agents $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ with goals $\boldsymbol{G_\alpha}$ and $\boldsymbol{G_\beta}$, initial states $\boldsymbol{I_\alpha}$ and $\boldsymbol{I_\beta}$ and sets of actions $\boldsymbol{D_\alpha}$ and $\boldsymbol{D_\beta}$ respectively, find a pair of plans $(\boldsymbol{P_\alpha}, \boldsymbol{P_\beta})$ such that:

- $\boldsymbol{P_\alpha = P_\alpha^\alpha \cup P_\alpha^\beta}$ *and* $\boldsymbol{P_\beta = P_\beta^\beta \cup P_\beta^\alpha}$
- $\boldsymbol{P_\alpha^\alpha \cup P_\beta^\alpha \vDash_{D_\alpha \cup D_\beta, I_\alpha \cup I_\beta} G_\alpha}$ *and* $\boldsymbol{P_\beta^\beta \cup P_\alpha^\beta \vDash_{D_\alpha \cup D_\beta, I_\alpha \cup I_\beta} G_\beta}$
- *if* $\boldsymbol{P_\alpha^\alpha \vDash_{D_\alpha, I_\alpha} G_\alpha}$ *then* $\boldsymbol{P_\beta^\alpha} = \emptyset$ *and if* $\boldsymbol{P_\beta^\beta \vDash_{D_\beta, I_\beta} G_\beta}$ *then* $\boldsymbol{P_\alpha^\beta} = \emptyset$
- $\boldsymbol{P_\alpha}$ *and* $\boldsymbol{P_\beta}$ *are non-conflicting*

2.5.1. Social laws

A crucial step of a swarm of biological wasps in accomplishing tasks is their interaction, cooperation and collaboration. This interaction enables them in accomplishing collective goals, such as foraging, in a more efficient manner [84]. A similar tendency is found in cooperation of humans when completing tasks as a team. Say for instance, for an engineering project, a team of engineers would formulate a strategy to manage the project with acceptable norms and regulations. Such an attribute is also anticipated from sources of inspiration of this project – RoboBees, subCULTron and CoCoRo, discussed earlier in section 1.1.1.2.

Inspired from this natural process of interaction among wasps and a team of humans, similar normative systems, or more precisely, social laws could be construed as producing additional set of constraints on an agent's existing plan which would result in a cooperative plan generation, and therefore, a structure of interactions and individual actions by each RoboWasp [85]. At this stage, what consists of a social MAS is quite generic and is interpreted differently by various researchers [86-89], however, Shoham [90] provides a concise, logical relation of a social law –

> "A social multiagent system is a collection of social agents that share the set of states, the language for describing states, the set of potential actions, the set of potential social laws, and the transition function."

Taking this as one of the guidelines in forming the social laws, algorithms include these parameters. Many overlapping parameters can be observed within the classical planning model, which also demonstrates the link between the planning model and the constraints applied to it. Where a social law may regulate an operation of RoboWasps, it is important to apply an appropriate social law for efficient use of RoboWasps in a certain environment. Such a law may be characterised as optimal, minimal, or more prominently, useful. A useful social law can be defined as the minimum amount of constraints set that may offer maximum flexibility to agents, yet ultimately achieving their goals [91].

2.5.1.1. Implementing durative actions

Establishing social laws can be beneficial in many ways for MAP problems in ensuring an efficient collaboration among the agents. However, no matter how optimal a social law is, one of the aspects to ensure is the concurrent execution of tasks by agents without significant interference by actions executed by other agents. This could be ensured by introducing durative actions in addition to set of actions A_i [89]. A durative action a is defined by its minimum and maximum duration $d_{min}(a)$ and $d_{max}(a)$, its start condition $cond_s(a) \subseteq F$, invariant condition $cond_i(a) \subseteq F$ and its end condition $cond_e(a) \subseteq F$, as well as its start effect $eff_s(a) \subseteq F$ and end effect $eff_e(a) \subseteq F$, which are both divided into add effects ($add_s(a) \subseteq F$ and $add_e(a) \subseteq F$, respectively) and delete effects ($del_s(a) \subseteq F$ and $del_e(a) \subseteq F$, respectively), where F is the set of facts used to describe the world [89]. As durative actions are implemented by all the agents sharing the environment, these ensure that there are no conflicts whilst actions are being executing by the agents, and therefore, would form an important component of verifying the usefulness of a social law.

2.5.1.2. Implementing the Minimal Social Law Algorithm

One of the other approaches in imposing social laws on MAS is described by Fitoussi and Tennenholtz [87] called the Minimal Social Law Algorithm (MSLA). In this approach, they consider a two-agent system, where an agent would follow a predefined plan P_i, however, their goal/capabilities may change and therefore, the agents are not just restricted to this plan. They show that this is an efficient incremental algorithm that exists for this class of systems [87].

The MSLA approach is structured as –

i. Let s_k^i denote the k^{th} state to be visited in the execution that corresponds to the original goal. Let s_0^i denote that the initial state of agent i. Let $\left(a_1^i, a_2^i, \dots, a_{t_i}^i\right)$ be the sequences of action executed by the agent i in the corresponding plan (P_i).

ii. Initialisation step: Let $k = 0$ and let $A'_i(s) = \emptyset$ for all i and s, intuitively, $A'_i(s)$ represents the set of forbidden actions at s.

iii. Let B_k^j be the set of reachable states at step k for agent j when agent $(3 - j)$ follows its original plan. Initially, B_0^1 contains initial state of agent 1 in C_0, B_0^2 contains initial state of agent 2 in C_0, all other B_k^j are empty sets.

iv. For each state $s \in B_k^1$ for all action $a \in A_1(s) \backslash A'_1(s)$, if $\tau(s, s_k^2, a, a_k^2) = (s^1, s_{k+1}^2)$ where $s^1 \in L_1$, then add $s^1 \in L_1$, then add a to $A'_1(s)$.

v. Increment k by one. While $k < t_i$, go to (iv).

vi. Execute (iv) and (v) again, switching the indexes of agents.

vii. Output the social law whose components are the functions, A'_1, A'_2.

One of the beneficial characteristics presented is the abidance of the laws by MAS. However, this could also be considered as a disadvantage because there may be exceptional situations where the MAS must go for an alternative plan to achieve their goals. In such cases, one would have to account for those exceptional circumstances where the agents would have to break the law. The MSLA approach is, therefore, critical when the possibility of not obeying social laws is considered. Boella and van der Torre [92] extend the MSLA approach to account for this possibility. They reason that one of the potential results of violating a law should be a benefit for the agents in some form, and if an agent does not profit from violating the law, such a social law is known to be quasi-stable [92], and therefore, it is crucial for a social law not to be quasi-stable. Because such a condition would be prominent in an uncertain environment, such an assumption is not considered in the implementation of RoboWasps at this stage.

2.5.2. Organisational structures

A peculiar characteristic observed in biological wasps and bees is that over evolutionary time, they develop preferential relationships with only one or a few plants in performing pollination, but others maintain a more general preference for a wide range of flowering plants. Additionally, a characteristic observed is that female wasps visit the plants mainly to collect enough pollen to feed their young ones, whilst the males need to visit the plants only to feed themselves – this is not even required at times as long as they are fed by the female wasps [93]. Taking this idea forward, a MAP of RoboWasps could be extended in a way such that there exist multiple sections of RoboWasps that are purely equipped for a certain task and are restricted to those tasks alone throughout the plan execution process. Whilst this is the case for one section, the other sections act complementary to this. To put it simply, the other sections would perform other tasks that would complete the execution process in reaching the goal state.

2.6. Tools and Techniques

Different tools are adopted in developing an agent, the organisation it caters to, the mechanism of its interaction between other agents, and the environment in which it operates; Therefore, the programming paradigms are primarily be classified into Agent Oriented Programming [94], Organisation Oriented Programming [95, 96], Interaction Oriented Programming [97] and Environment Oriented Programming [98].

2.6.1. The STRIPS planning

One of the most popular techniques in implementing an automated planning system is the Stanford Research Institute Problem Solver (STRIPS). In STRIPS, a world is represented by a set of well-formed formulae of the first-order predicate calculus [99]. The iteration in STRIPS begins with a successor node tested on whether the first goal is satisfied within the goal list. If so, the corresponding action is applied, generating a new successor node; If not, the difference is stored with the node [99, p.195].

One of the models within STRIPS is the ‘propositional STRIPS planning’, in which an initial state is a finite set of ground atomic formulae indicating that corresponding conditions are initially true and that all the relevant conditions are initially false, the preconditions and postconditions of an operator are ground literals, and the goals are ground literals [100, 101]. An instance of a propositional STRIPS planning is, therefore, defined as a tuple $\langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{P}$ is the set of ground atomic formulas, also known as conditions; $\mathcal{O}$ for the set of operators, classified as *pre* and *post* operators, each having positive and negative conditions (pr_+, pr_-, po_+, po_-); $\mathcal{I} \subseteq \mathcal{P}$ is the initial state; and $\mathcal{G}$ is the goal state with positive $(\mathcal{G}_+)$ and negative goals $(\mathcal{G}_-)$ [100]. If the flow chart in Figure 5 is to be linked to propositional STRIPS planning, the initial world model is equivalent to what is defined as the initial state $\mathcal{I} \subseteq \mathcal{P}$. The set of operators $\mathcal{O}$ function is identical to the set of actions A, described in classical planning (section 2.4.1).

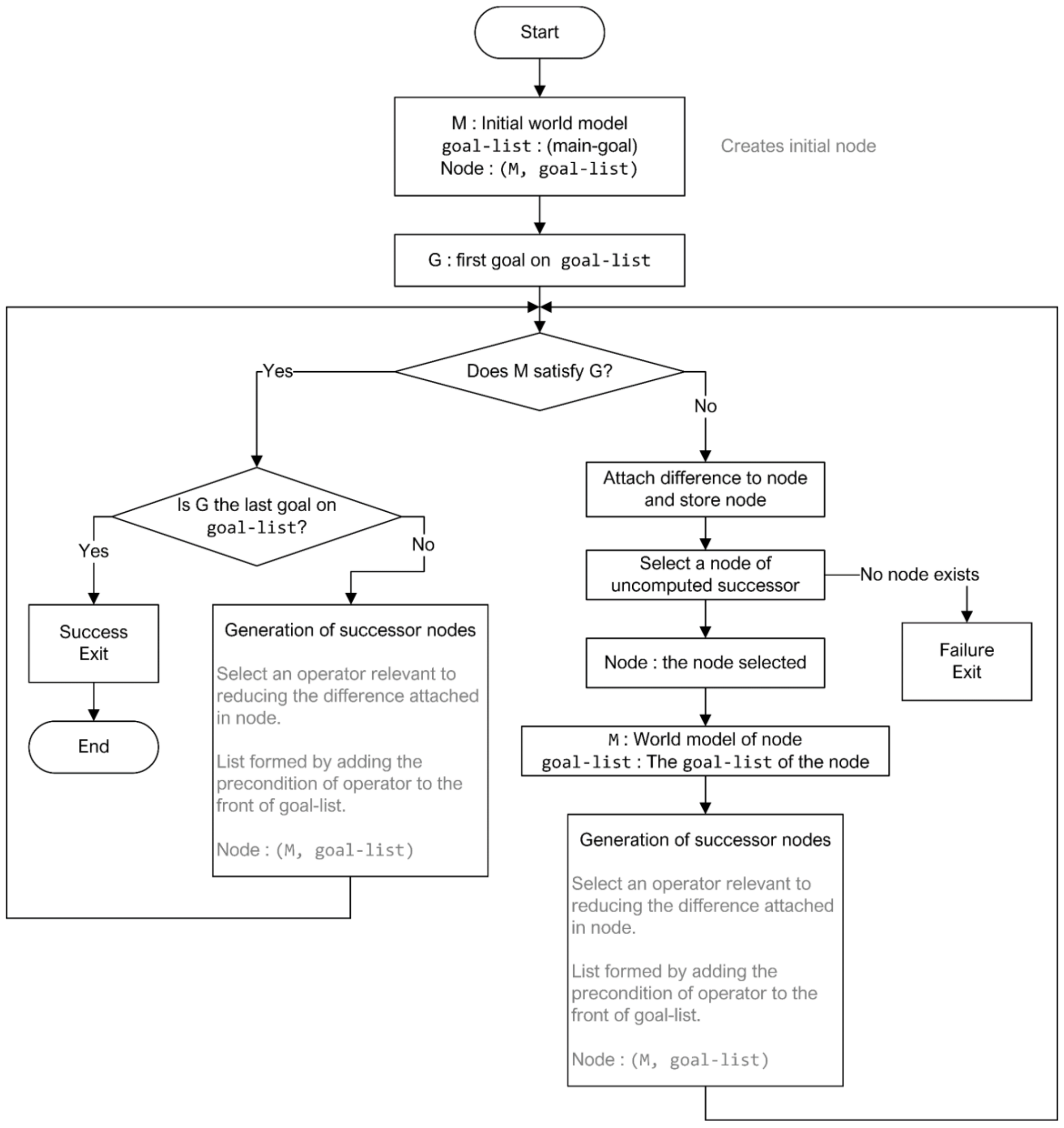

Start
M : Initial world model
goal-list : (main-goal)
Node : (M, goal-list)
Creates initial node
G : first goal on goal-list
Does M satisfy G?
Yes
No
Is G the last goal on goal-list?
Yes
No
Success Exit
End
Generation of successor nodes
Select an operator relevant to reducing the difference attached in node.
List formed by adding the precondition of operator to the front of goal-list.
Node : (M, goal-list)
Attach difference to node and store node
Select a node of uncomputed successor
No node exists
Failure Exit
Node : the node selected
M : World model of node
goal-list : The goal-list of the node
Generation of successor nodes
Select an operator relevant to reducing the difference attached in node.
List formed by adding the precondition of operator to the front of goal-list.
Node : (M, goal-list)

Figure 5 - Flow chart of STRIPS planning

Another parameter that is of interest to a domain designer is the macro-operator, particularly when dealing with an uncertain environment. A macro-operator, macro-action, or simply, a macro is a group of actions selected for application at one time like a single action. Macros could represent high-level tasks comprising of low-level details. Combining several steps in the state space, macros provide extended visibility of the search space to the planner [102]. However, introducing macro-operators can be tricky, especially depending upon the approach adopting in domain formulation. STRIPS is considered to be an easier approach in this regard when compared to the larger PDDL subsets, such as the Action Description Language (ADL) [103].

The propositional STRIPS planning model forms a foundation to many other STRIPS planning models that can be applied to MAP, such as translational STRIPS [104], in which the existing propositional model would be translated to linear programming of mapping conditions and operators in each plan step to variables.

Furthermore, for applying the STRIPS planning model in an uncertain environment for MAPs, inverted STRIPS planning can be adopted. A problem is said to be invertible when there exists a plan to start from the goal state and to reach the initial state [104, p.770], also known as backward search [99, p.195]. Galuszka and Swierniak [105] extend a well-known classical planning problem, Block World, to a conformant planning with an invertible STRIPS planning methodology. For simulations divided into two problems with non-conflicting and conflicting goals, the set of actions were changed to complete opposite functions - such as *stack* was changed to *unstack*, and *pickup* was changed to *putdown*. A plan was achieved when the simulations were run, demonstrating the successful use of inverted STRIPS planning.

2.6.2. The Action Description Language

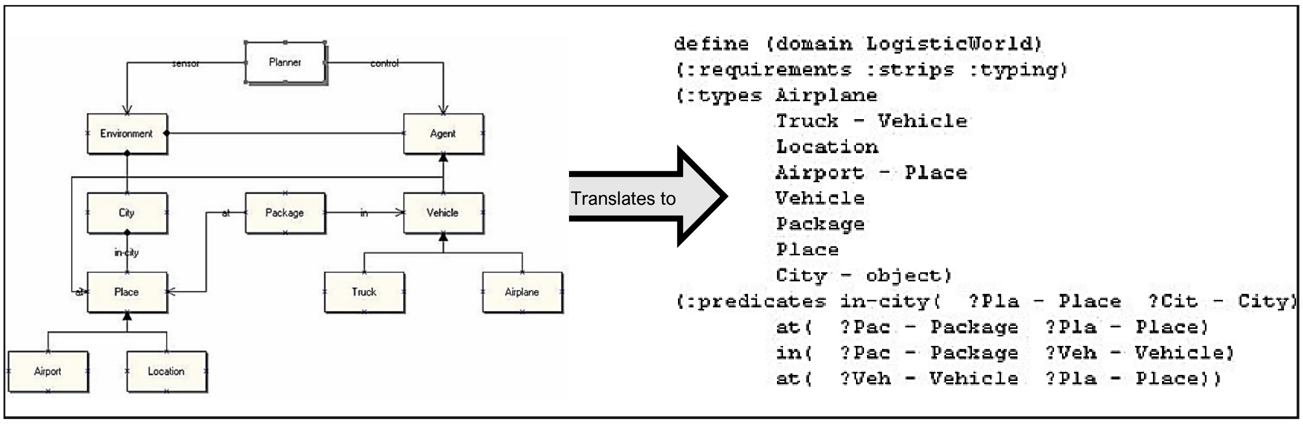
Whilst STRIPS planning considers the deterministic, static and a finite state space, extending further towards open environment problems, an efficient alternative in solving non-deterministic problems is the Action Description Language (ADL). Considering an agent's belief within an epistemic planning model, the existing state transition function within STRIPS is substituted to a transition function ϕ, that incorporates a belief evolution operator, which changes with each cycle of iteration within a dynamic environment [106]. The remaining parameters, such as set of states, set of actions, initial states and goal states, are defined in the same manner as in STRIPS planning.

2.6.3. The Planning Domain Definition Language

A reliable technique to apply STRIPS planning was introduced by Drew McDermott *et al.* [107] in 1998, where the planning domains and the problems are defined concisely, and the syntax is clearly understandable. From problems ranging from pathfinding to space exploration [108], the Planning Domain Definition Language (PDDL) defines the domain and the problem explicitly by adopting parameters from STRIPS planning and ADL planning.

With different versions and extensions of PDDL available for various plan generations to apply to a wide range of environments [109], for a MAS, the extension to PDDL 3.1 enables MAP with the consideration of executing concurrent actions by different agents [110]. One of the features of this extension, MA-PDDL (Multi Agent-PDDL), include the ability to define privacy of each agent, which means that each agent can execute their own tasks without any interference from other agents [111]. Kovacs [111] distinctly lists the added keywords and in-built functions used in MA-PDDL with respect to the earlier version of PDDL, which helps the domain designer(s) to utilise the features of PDDL appropriately towards defining their domain. Alternatively, with earlier PDDL versions that do not support the multiagent feature, a similar planning structure can be developed as demonstrated by Redjaimia [112] for a four-agent planning system. Where additional versions and extensions can be seen as an advantage for the domain designer, Tonidandel *et al.* [113] see this as a complexity in programming and learning its efficient use for a corresponding domain definition. To overcome this complexity, they suggest translating PDDL to an object-oriented model such as the *GIPO* tool and *itSIMPLE*.

The graphic interface in itSIMPLE (shown in Figure 6) offers domain designers define a domain in a friendly manner without going through the complexity of learning to code in PDDL [114].



```
define (domain LogisticWorld)
(:requirements :strips :typing)
(:types Airplane
        Truck - Vehicle
        Location
        Airport - Place
        Vehicle
        Package
        Place
        City - object)
(:predicates in-city(  ?Pla - Place  ?Cit - City)
        at(  ?Pac - Package  ?Pla - Place)
        in(  ?Pac - Package  ?Veh - Vehicle)
        at(  ?Veh - Vehicle  ?Pla - Place))
```

Figure 6 - The graphic interface in itSIMPLE translates the defined objects to PDDL

On the accuracy and the efficiency of PDDL towards accomplishing the project objectives, MA-PDDL could prove helpful in defining a MAS within the planning domain with a specific structure to the planning problem. In addition to this, defining a set of states and a set of actions is fairly straightforward with variables defined as generic, rather than problem-specific. This enables applying the same domain to multiple planning problems that share a common domain, increasing convenience to the domain designer while definition, modifications and testing of the model.

2.6.4. JaCaMo programming

Whilst PDDL is a tailored programming language for plan generation and execution, JaCaMo is a 'multi-agent oriented' programming language as it combines $Jason$ for programming autonomous agents, $CArtAgO$ for programming shared environments and $\mathcal{M}oise^{+}$ for programming agent organisations [94]**.** $Jason$ is a programming language based on belief-desire-intention (BDI) architecture, combining the agents' beliefs, goals and plans [115]. $CArtAgO$ work environments are modelled and engineered in terms of a set of artefacts (agent's viewpoints) and are collected in workspaces [116]. $\mathcal{M}oise^{+}$ considers three dimensions of an agent's organisation – its functionality, its structure and the norms that an agent should obey [117]. JaCaMo integrates these three platforms by defining in particular a semantic link among attributes of the different programming dimensions – agent, environment and organisation – at the meta-model and programming levels, in order to obtain a uniform and consistent programming model aimed at simplifying the combination of those dimensions when programming a MAS [94]. This combination is graphically represented in Figure 7.

In considering the use of JaCaMo towards the various planning models discussed in section 2.4, it can be demonstrated that this programming platform shows its strength in open environments, with nondeterministic, continuous and dynamic characteristics, and therefore, contingent, conforment, and epistemic planning could be implemented effectively. Typical examples where the use of JaCaMo has proved helpful is towards developing a Knowledge Management MAS [118], a smart home model [119], and an intelligent room governance system [120]. When developing a MAS in JaCaMo environment, programming is exploited to define the computational layer encapsulating functionalities and services that agents can use, explore, and share at runtime [121]. This means that the interaction between MAS is well-established during task execution. Another indication to indicate the potential of JaCaMo and its efficiency in MAP is what is stated by Sorici *et al.* [120], following their analysis on a room management tool, that JaCaMo can be considered as a promising solution of conceptual design and implementation of applications that require agile management and adaptation capabilities, especially when agents were to play a certain role. In implementing this to practice, one of the approaches taken by Hübner *et al.* [122] is using $\mathcal{J}ason$ in defining artefacts, that generally provide mechanisms to externalise functions, are implemented as internal actions instead. The shared global goals and global plans are defined in $\mathcal{M}oise^{+}$. The implementation and the deployment of artefacts is done with the $CArtAgO$. This shows that JaCaMo does not just incorporate three platforms, but also offer flexibility in their functionality as per the designer's preference.

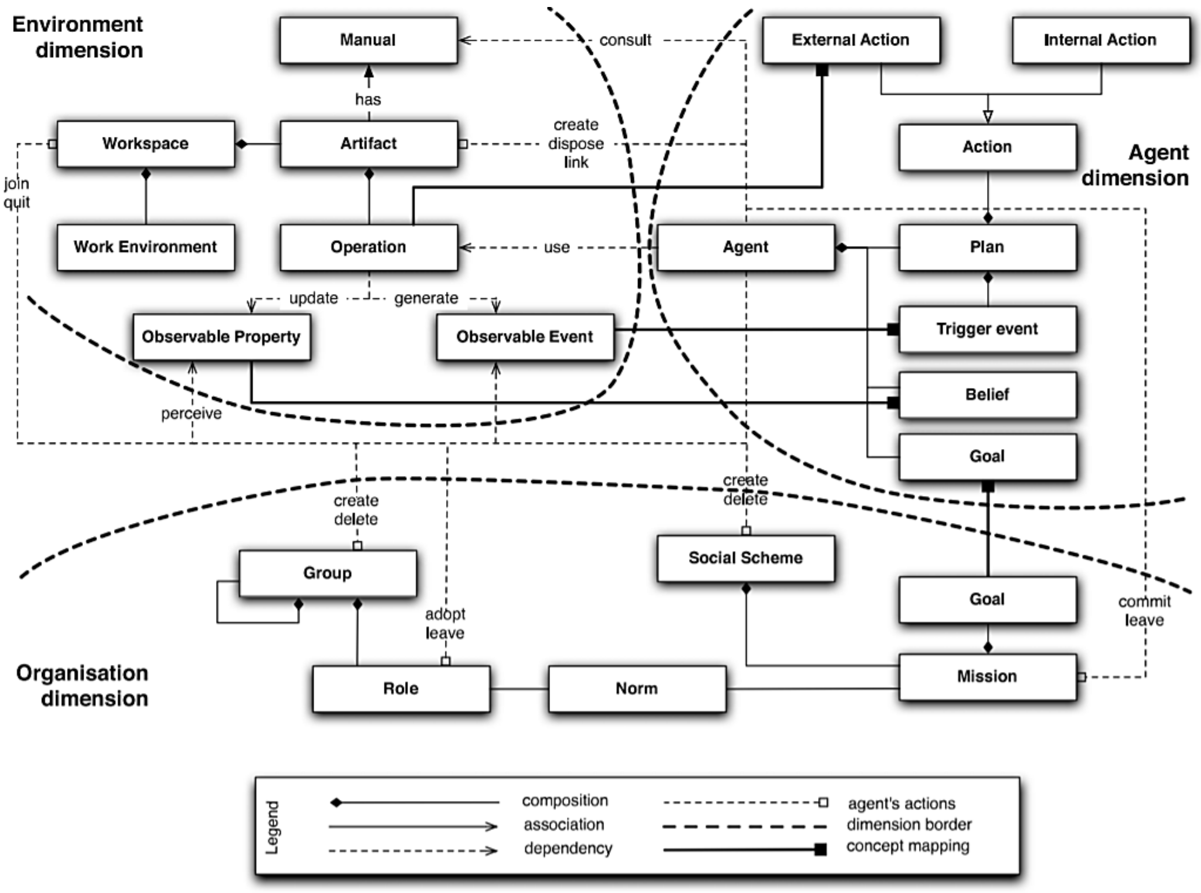


Figure 7 - The integration of three platforms as three dimensions in JaCaMo [94]

On the accuracy and efficiency of JaCaMo towards accomplishing the project objectives, by defining the beliefs and behaviours of RoboWasps, the problem could be extended to an uncertain environment, where the RoboWasps are able to learn from experience and update their belief system with past results achieved. This could particularly prove helpful in open environments, where RoboWasps may be required to adapt with changes arising in their environment.

2.7. Summary

Chapter 2 begins by understanding fundamental concepts such as the definition of an agent in context to RoboWasps and establishing that the swarm of RoboWasps are referred to as a multiagent system throughout the chapter to indicate a generic notion. The chapter progresses further by comparing domain-specific and domain-independent approaches towards achieving the project objectives, gauging them with criterions such as pre-requisite knowledge in model construction, time constraints, and the potential risks of project delays. Domain-specific approaches are selected for the implementation as it is acknowledged that domain-independent approaches would require a significant experience in model construction and working within automated planning systems.

On the nature of environment, these were primarily classified as an open and a closed environment in terms of their characteristics, such as discrete or continuous state space, deterministic or stochastic structure of plans, and their static or dynamic nature. The various planning models - classical, hierarchical, contingent, conformant and epistemic - are compared and their suitability towards the nature of the environment is evaluated, along with their mathematical structure. Once these are recognised, it is explored on the appropriate tools and techniques in implementing these planning models to a domain-specific problem. STRIPS planning, ADL planning, PDDL framework and the JaCaMo framework are studied primarily, with reference to each of the planning models they would support and their problem-solving technique.

CHAPTER THREE

METHODOLOGY

3.1. Chapter Overview

In the previous chapter, different fundamental concepts of MAP were explored and were researched objectively to relate these theoretically to the project objectives and to determine an appropriate approach in implementing an automated planning system on a swarm of RoboWasps.

To recall the project objectives,

(i) To study and critically evaluate on an appropriate planning system for a swarm of RoboWasps.

(ii) To implement the evaluated planning system on a planning problem considering problems from the real-world applications.

This chapter works towards achieving the first objective and going further to the theoretical concepts in understanding how they would be implemented for analysis.

3.2. Planning model

In section 2.4, different planning models were explored and type of environments they could be applied to were recognised. At this stage of comparing and critically evaluating between these planning models, a 'matrix of appropriateness' is tabulated in Table 1.

Despite applications of contingent, conformant and epistemic planning across all the environments, these have apparent drawbacks in the context of this project. The timeframe being a primary drawback, these three types of planning models require a significant amount of domain specification with minute details and a sophisticated structure. A single-agent design within this system might not even be completed within the timeframe planned for the completion of a multiagent planning system using classical or hierarchical planning. Therefore, these three planning models are ruled out for this project.

When scrutinising an appropriate approach among classical and hierarchical planning, it can be found that the HTN planning representation can overcome most of its prominent drawbacks on the domain knowledge by its available variants, however, as the problem dealt in this project involves a completely observable environment, this planning model is not seen to be in any advantage over the classical planning. Additionally, when comparing classical planning and hierarchical planning, the former is simpler to represent and has straightforward terms defined that can be linked to the goal states explicitly – for instance, one of the parameters in classical planning is the goal state S_G, whereas the parameters, C, M, tn_I, are tasks and method-oriented, which would implicitly link to the goal to be achieved by the agents. As this project would consider a problem with clearly defined goal states, S_G is defined. Therefore, the classical planning model would lead to a faster, satisfactorily accurate and simpler analyses, this project considers this model to proceed with the implementation.

Table 1 – Matrix of appropriateness in matching planning models to different types of environments

	Closed			**Open**		
	Deterministic	**Static**	**Discrete**	**Non-deterministic**	**Dynamic**	**Continuous**
Classical	✓	✓	✓	●	✕	✕
Hierarchical	✓	✓	✓	●	✕	✕
Contingent	✓	✓	✓	✓	✓	✓
Conformant	✓	✓	✓	✓	✓	✓
Epistemic	✓	✓	✓	✓	✓	✓

Key:

✓ - can be applied

● - can be applied with modifications/variations

✕ - cannot be applied

Despite that the contingent, conformant and epistemic planning are ruled-out at this stage, these models are best utilised in dynamic and continuous environments and they could potentially be seen as expanding upon the simplistic classical model.

Considering all of the explanations and justifications, it would be reasonable to proceed the analysis with the classical planning model to develop preliminary findings on the planning system for RoboWasps. However, these could further be extended by adopting various problems pertaining to the real-world encounters that may be faced by RoboWasps.

3.3. Programming framework

In section 2.6, four implementation approaches were explored and their application in relation to RoboWasp. Among STRIPS, ADL, PDDL and JaCaMo, the latter two are taken for further consideration, given that PDDL combines the characteristics of STRIPS and ADL within their framework. Table 2 compares PDDL and JaCaMo on various criterions in evaluating their suitability for implementation.

In Table 2, the criterions upon which an appropriate programming framework is decided are their structure, their suitability to the problem and domain, their availability, ease of programming and their accuracy and efficiency in achieving the results.

Table 2 - Distinguishing between PDDL and JaCaMo on different criterions (with their explanation)

Language / **Criterions**	**PDDL**	**JaCaMo**
Structure	In this context, the structure of a programming language would imply its components and features that would contribute towards designing a domain and the problem.	
	PDDL combines the foundation planning frameworks, i.e., STRIPS and ADL, indicating that it would support combined features of these frameworks. This would offer the domain designer to utilise an extended range of functions in defining their domain and problem.	JaCaMo combines three different planning frameworks, i.e., $Jason$, $CArtAgO$ and $\mathcal{M}oise^{+}$, each of which address the three exclusively mutual attributes that would define the domain and the problem in a holistic manner.
Suitability to problem and domain definition	A closed environment is considered for the implementation. This criterion is differentiated is in reference to the suitability for a classical planning problem.	
	PDDL offers consideration of all the parameters that define the classical planning model. Not just this, it offers a direct way in defining the set of states and actions in the planning domain, whilst the initial state and the goal state is defined separately. This offers easier modification and the use of a common domain to multiple problems.	As the problem considered is a classical problem, most of the parameters such as the agents' beliefs and behaviours, internal actions, and definition of events are not utilised. The remaining parameters that are to be utilised are available in PDDL. In such a case, incorporating three distinct platform may be more time-consuming, when a single framework is available.

Availability	When a programming framework is easily available, it increases the likelihood of using that framework. Therefore, availability is an important criterion.	
	PDDL v2.15.2 is available to download from the Visual Studio Marketplace to be used on VS Code [**]. However, for this problem, an ideal version required is PDDL v3.1 with an extension of MA-PDDL.	The JaCaMo framework is available to download from the Internet [††], with JaCaMo being supported on various IDEs - Eclipse, shell script commands, Gradle, or Docker.
Ease of programming	For a domain designer with no prior experience in defining a domain and a problem, it is crucial to start with an easier programming framework that has a simpler syntax and troubleshooting options, along with being compatible with the preferred editor.	
	The programming syntax is simple to understand and interpret. At the same time, most predicates and actions use simple English words, making programming these convenient for the domain designer.	For different parameters that exist in JaCaMo, it is important to understand how each parameter is linked to which framework, and towards which aspect does it address – the environment, the agent, or the organisation, which makes PDDL simpler.
Accuracy/Efficiency	Accuracy, in this context, can be defined as the property of generating plans along with an indication to its execution and evaluation of how compliant the plan is for a swarm of RoboWasps.	
	PDDL considers crucial parameters that define a classical planning model – set of predicates, set of states, set of actions and a goal state. Therefore, the anticipated planning system is expected to indicate results in reference to these values. Moreover, the efficiency of PDDL has been recognised by many recent research papers, showing its efficiency in the present time.	JaCaMo considers additional elements such as beliefs, organisational role, and internal and external actions. However, because all these characteristics are efficient in an open environment, and the implementation is only limited to a closed environment, beliefs and internal actions play a negligible role, resulting in an efficiency not to be much different than PDDL.

[**] https://marketplace.visualstudio.com/items?itemName=jan-dolejsi.pddl
[††] http://jacamo.sourceforge.net/

Considering these criterions, PDDL is considered for analysis over JaCaMo. Despite a robust structure, accuracy and suitability of JaCaMo, PDDL offers a greater advantage with respect to its simpler syntax and troubleshooting options, along with an option to use GIPO tool or itSIMPLE if the domain designer does not have adequate programming experience and prefers a graphic interface. Additionally, although PDDL may have a sophisticated extension for MAP, MA-PDDL, it offers a possibility of temporarily detaching the communication between RoboWasps in certain conditions. This means that the RoboWasps would not be reliant on each other at all times and can function effectively by their own when required.

3.4. Planning domain

The parameters that define a classical planning model were the set of states, set of actions, initial state(s) and goal state(s). In PDDL, two separate files – one consisting the former two are defined within the `domain.pddl` file and the latter two are defined within the `problem.pddl`. The domain considered is a garden with flowers and the goal is to pollinate these by RoboWasps.

The set of states or predicates are primarily defined with reference to the number of RoboWasps, flowers and their respective locations. Adding to that, for later simulations, the number of biological wasps would be defined within the predicates as well. Once the location of fixed types (i.e., flowers) and the initial state of dynamic types (i.e., RoboWasps and biological wasps) are established, the tasks they are equipped to execute are defined as a set of actions and the possible effect they would have is defined by a new set of predicates.

The set of actions begin with the most fundamental action for any RoboWasp, irrespective of their operation, i.e., `navigate`. Within the state space, this action is reiterated until it reaches the flower to perform further action assigned to them according to the problem. Once their respective operation on the flower are completed, the RoboWasps move to another flower, where the next action iterated would be `navigate` again. In addition to the first action defined, the goal state is also deterministic, irrespective of the problem, i.e., `pollinate-flower`. The intermediate set of actions are defined according to the problem considered. For instance, if an organisational structure is considered, where the roles of RoboWasps would depend on what they are equipped to perform, the set of actions would have to be defined in accordance to their ability to perform a task.

CHAPTER FOUR

DESIGN AND ANALYSIS

4.1. Chapter Overview

Further to Chapter 3, where the methodology for analyses was explored and decided upon, this chapter implements the methodology in the selected programming framework and different planning domains and problems are designed. Once the design is complete, these are analysed on different conditions and situations that demonstrate an efficient operation of the swarm of RoboWasps.

4.2. Domain definition

Biological wasps make an essential contribution to natural processes by assisting in pollination of flowering plants. Inspired from the biological wasps, the swarm of RoboWasps would assist wasps and bees in performing pollination by splitting their work by a significant amount.

Questions may arise such as, why interfere in the works of natural systems? Aren't honeybees and wasps designed to pollinate the flowers and aren't they doing this well enough already? – The answer to all these questions is that instead of proving to be disturbing the natural system the way it currently is, RoboWasps would rather prove as helping hands. In the event of changes due to external and environmental factors, the ability of biological wasps and honeybees could be affected adversely. These include –

i. <u>Mechanisms such as heliotropism and thermogenesis</u> – Heliotropism and thermogenesis could result in changes of morphological features of flowers, such as the colour and shape, which can also increase intra-floral temperature to levels up to 11 °C above ambient conditions. These factors hold implications for wasps and bees that contribute in the pollination process [123, 124].

ii. <u>Climate change</u> – In addition to the physical appearance that impacts on the ability of pollinators to perform an efficient pollination, climate change can pose a risk to a timely greening, flowering and senescence, or more precisely, the plant maturation. Dixon [125] specifies that climatic factors such as decrease in precipitation and a seasonality shift of rainfall, particularly in Mediterranean regions, could result in a reduced plant vigour, a delayed plant maturation and a decline in nectar production capacity. He also warns that global warming may lead to partial or total asynchrony between pollinator life cycles and flowering phenologies [126].

With RoboWasps effectively acting as artificial pollinators, it can be ensured that the pollination takes place without placing much reliance on natural systems due to factors such as their ability to locate mature flowers and their own life cycle.

In PDDL, the domain starts by defining the requirements

Code Listing 1 – Invoking requirements in PDDL

```
;domain definition
(define (domain garden0)
(:requirements :strips :fluents :typing :equality)
```

It can be noticed from Code Listing 1 that the requirements invoked in PDDL are strips, fluents, typing and equality. The functions of these requirements in this specific domain is –

Table 3 - Functions of requirements invoked in PDDL

`:strips`	Invoking STRIPS (planning approach)
`:fluents`	Invoking fluents would enable plan validation tool that PDDL offers
`:typing`	Invoking typing indicates that the domain uses `:types`
`:equality`	Invoking equality would enable checking whether a predicate would be made true, and therefore, an action is triggered.

4.2.1. Definition of types

Within a domain defined in PDDL, types, or particularly, object types are essential for defining the environment in which the RoboWasps would operate in a straightforward manner. Generally, these are not mandatory depending upon the domain and the problem. The three most important types for the RoboWasp problem are `robowasp`, `flower`, and `location`, irrespective of the nature of the problem, although additional types may be added for further experimentation within the domain.

Code Listing 2 – Primary definition of types in PDDL

```
(:types
    robowasp
    flower
    location
)
```

The graphical representation of types is generated by the PDDL in VS code is –

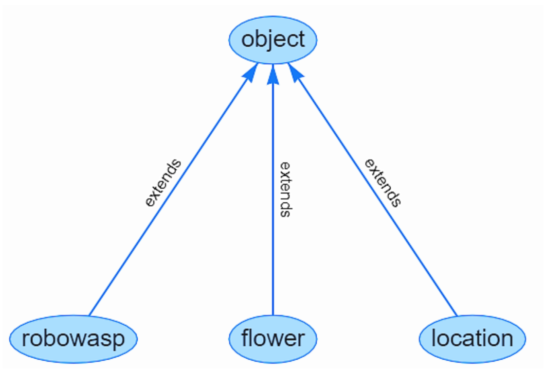


Figure 8 - A graphical representation of types generated in VS code

4.2.2. Definition of predicates

Within a domain defined in PDDL, predicates are the set of states that are achieved when a particular action or a set of actions are triggered. A set of predicates is defined as –

Code Listing 3 – The set of predicates defined in PDDL

```
(:predicates
    (at ?rw - robowasp ?x - location)
    (at0 ?fl ?x)
    (sample-for-maturity ?rw ?fl)
    (is-mature ?fl)
    (is-immature ?fl)
    (ignore-flower ?rw ?fl)
    (draw-pollen ?rw ?fl)
    (transfer-pollen-to-pistil ?rw ?fl)
)
```

It can be noticed from Code Listing 3 that the location of RoboWasp is declared by the predicate `at` and for the location of flower is defined by `at0`. This is because in PDDL, predicates considering different objects cannot be identical, although their function may be the same.

4.2.3. Definition of actions

The set of actions are defined within the domain and differ for each problem. The problems considered for analysis are classified as problem 0, 1, 2, and 3. The set of actions defined in Table 4.

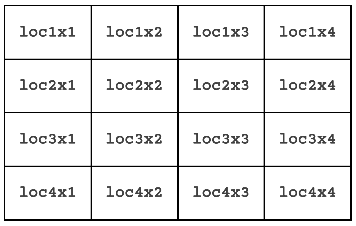
It can be noticed from Table 4 that two different actions, `draw-pollen` and `transfer-pollen-to-pistil`, in problem 0 are combined to one action, `pollinate-flower`, in problem 1 because other object types increase in number. This would enable a clear representation of results, especially when these actions occur in sequence at all times and have no conditions between them.

Table 4 - Set of actions defined in accordance to each problem

Problem 0	`{navigate, sample-flower, ignore-flower, draw-pollen, transfer-pollen-to-pistil}`
Problem 1	`{navigate, sample-flower, pollinate-flower}`
Problem 2	`{navigate, sample-flower, communicate-maturity, navigate0, pollinate-flower}`
Problem 3	`{biological-wasp-navigates, navigate, biological-wasp-pollinate-flower, monitor-biological-wasps, communicate-location-of-biological-wasp, ignore-flower-due-to-biological-wasp, navigate0, sample-flower, ignore-flower, pollinate-flower}`

4.3. Problem definition

In the problem file, `problem_0.pddl`, the finite and discrete environment considered for plan generation in the first round of simulation is a 4×4 matrix, with rows and columns labelled as shown in Figure 9. In the problem file, `problem_1.pddl`, the state space is expanded to a 7×7 matrix with more flowers and expanding the problem to a MAP problem. As further conditions are added, the state space is balanced with respect to the increase in the computational cost.



loc1x1	loc1x2	loc1x3	loc1x4
loc2x1	loc2x2	loc2x3	loc2x4
loc3x1	loc3x2	loc3x3	loc3x4
loc4x1	loc4x2	loc4x3	loc4x4

Figure 9 - Environment definition for the first round of simulation

4.3.1. Problem 0: Plan generation for a single-agent system

For a single-agent problem, an initial position of the RoboWasp, `rw1`, is defined to be `loc1x1` and the initial position of the flowers `fl1`, `fl2`, `fl3`, and `fl4`, is defined to be `loc1x2`, `loc3x1`, `loc2x3`, `loc4x3`, respectively. The goal state in this simulation is for the RoboWasp to transfer the pollen to the pistil, which indicates a successful pollination. However, this is true only for flowers that were found to be mature when sampling. If the flower is immature when sampled, the RoboWasp ignores it. The PDDL code for this problem is shown in Code Listing 4 and Code Listing 5.

Code Listing 4 - Object definition for Problem 0

```
(:objects rw1 - robowasp
          fl1 fl2 fl3 fl4 - flower
          loc1x1 loc1x2 loc1x3 loc1x4 - location
          loc2x1 loc2x2 loc2x3 loc2x4 - location
          loc3x1 loc3x2 loc3x3 loc3x4 - location
          loc4x1 loc4x2 loc4x3 loc4x4 - location
)
```

Code Listing 5 - Set of initial states for Problem 0

```
(:init
    (at rw1 loc1x1)
    (at0 fl1 loc1x2) (is-mature fl1)
    (at0 fl2 loc3x1) (is-immature fl2)
    (at0 fl3 loc2x3) (is-mature fl3)
    (at0 fl4 loc4x3) (is-immature fl4)
)
```

With the PDDL code indicated, the state space is graphically represented as shown in Figure 10.

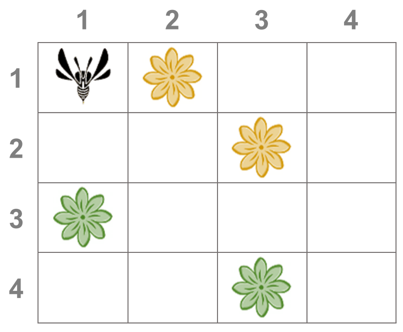


Figure 10 - State space representation for Problem 0. The yellow flowers are mature, and the green ones are immature.

4.3.2. Problem 1: Plan generation with no constraints

For a multi-agent problem, multiple RoboWasps are introduced in initial positions `loc1x1`, `loc7x7`, `loc7x1` and `loc1x7`. With the same goals to be achieved, i.e., pollinating all the flowers in the environment, the PDDL code for this problem in indicated in Code Listings 6 and 7. As there exists a central flower in the problem, `fl11`, this is assigned to `rw4` to perform pollination..

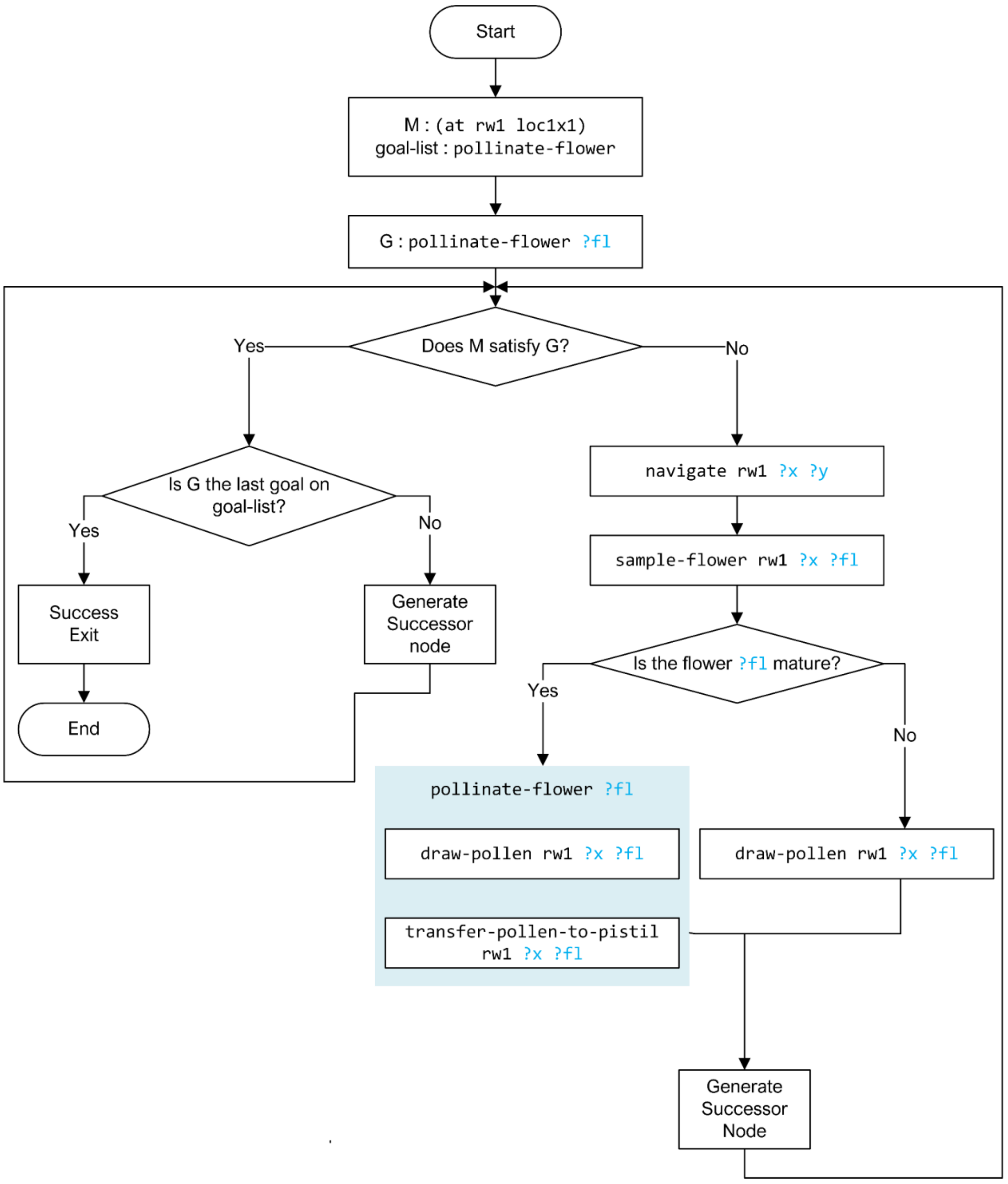


Figure 11 - STRIPS-like flow chart for Problem 0 for set of actions defined in PDDL

Code Listing 6 - Object definition for Problem 1

```
(:objects rw1 rw2 rw3 rw4 - robowasp
          fl1 fl2 - flower                  ;row 1
          fl3 fl4 fl5 - flower              ;row 2
          fl6 fl7 fl8 fl9 - flower          ;row 3
          fl10 fl11 fl12 - flower           ;row 4
          fl13 fl14 fl15 fl16 - flower      ;row 5
          fl17 fl18 fl19 - flower           ;row 6
          fl20 fl21 - flower                ;row 7
          loc1x1 loc1x2 loc1x3 loc1x4 loc1x5 loc1x6 loc1x7 - location ;row 1
          loc2x1 loc2x2 loc2x3 loc2x4 loc2x5 loc2x6 loc2x7 - location ;row 2
          loc3x1 loc3x2 loc3x3 loc3x4 loc3x5 loc3x6 loc3x7 - location ;row 3
          loc4x1 loc4x2 loc4x3 loc4x4 loc4x5 loc4x6 loc4x7 - location ;row 4
          loc5x1 loc5x2 loc5x3 loc5x4 loc5x5 loc5x6 loc5x7 - location ;row 5
          loc6x1 loc6x2 loc6x3 loc6x4 loc6x5 loc6x6 loc6x7 - location ;row 6
          loc7x1 loc7x2 loc7x3 loc7x4 loc7x5 loc7x6 loc7x7 - location ;row 7
```

Code Listing 7 - Set of initial states for Problem 1

```
(:init
    (at rw1 loc1x1) (at rw2 loc7x7) (at rw3 loc7x1) (at rw4 loc1x7)
    (at0 fl1 loc1x3) (at0 fl2 loc1x5) (at0 fl3 loc2x2) (at0 fl4 loc2x4)
    (at0 fl5 loc2x6) (at0 fl6 loc3x1) (at0 fl7 loc3x3) (at0 fl8 loc3x5)
    (at0 fl9 loc3x7) (at0 fl10 loc4x2) (at0 fl11 loc4x4) (at0 fl12 loc4x6)
    (at0 fl13 loc5x1) (at0 fl14 loc5x3) (at0 fl15 loc5x5) (at0 fl16 loc5x7)
    (at0 fl17 loc6x2) (at0 fl18 loc6x4) (at0 fl19 loc6x6) (at0 fl20 loc7x3)
    (at0 fl21 loc7x5)
)
```

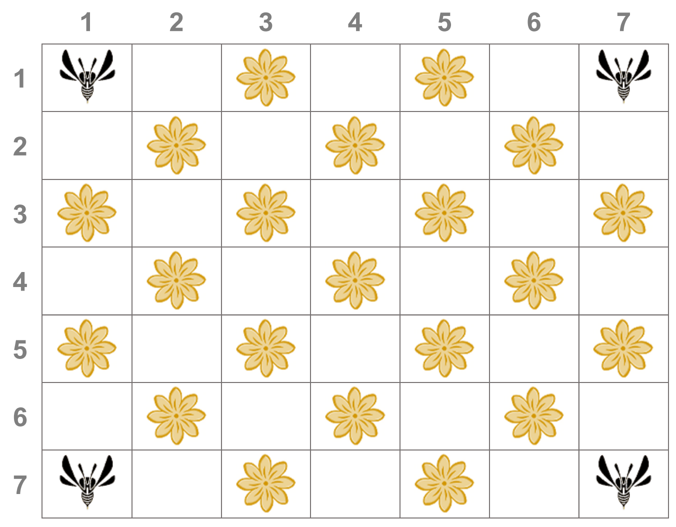


Figure 12 – State space representation for Problem 1a

To extend the problem further from a four-agent problem to a five-agent problem, one more RoboWasp is added to the state space at location `loc3x4`. The state space representation is as shown in Figure 13. The analysis is performed to verify whether a valid plan is generated.

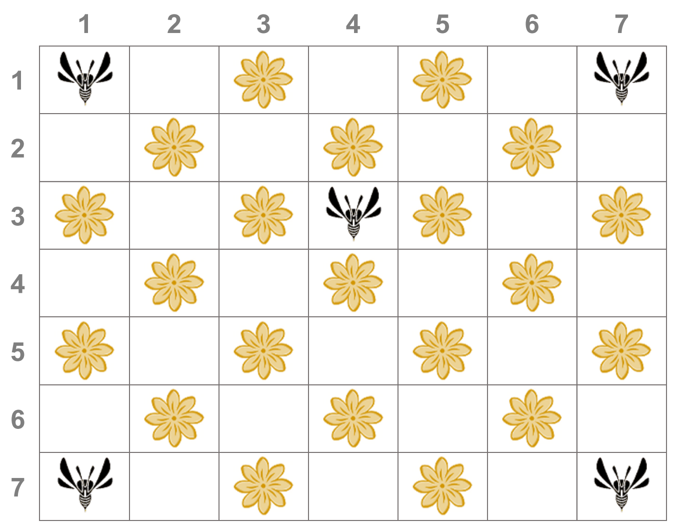


Figure 13 - State space representation for Problem 1b

The problem is further extended to a six-agent problem, where the newly introduced RoboWasps are located at `loc4x1` and `loc4x7`. The state space representation for the updated problem is shown in Figure 14. The analysis is performed in verifying the accuracy of the plan generated when compared to the initial, four-agent problem.

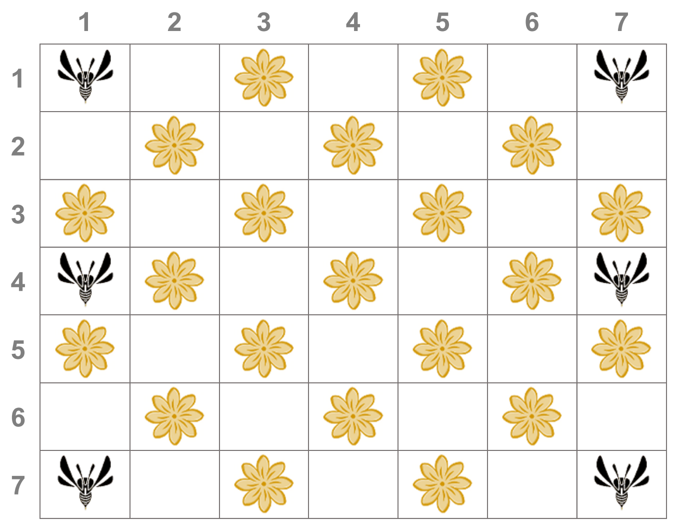


Figure 14 - State space representation for Problem 1c

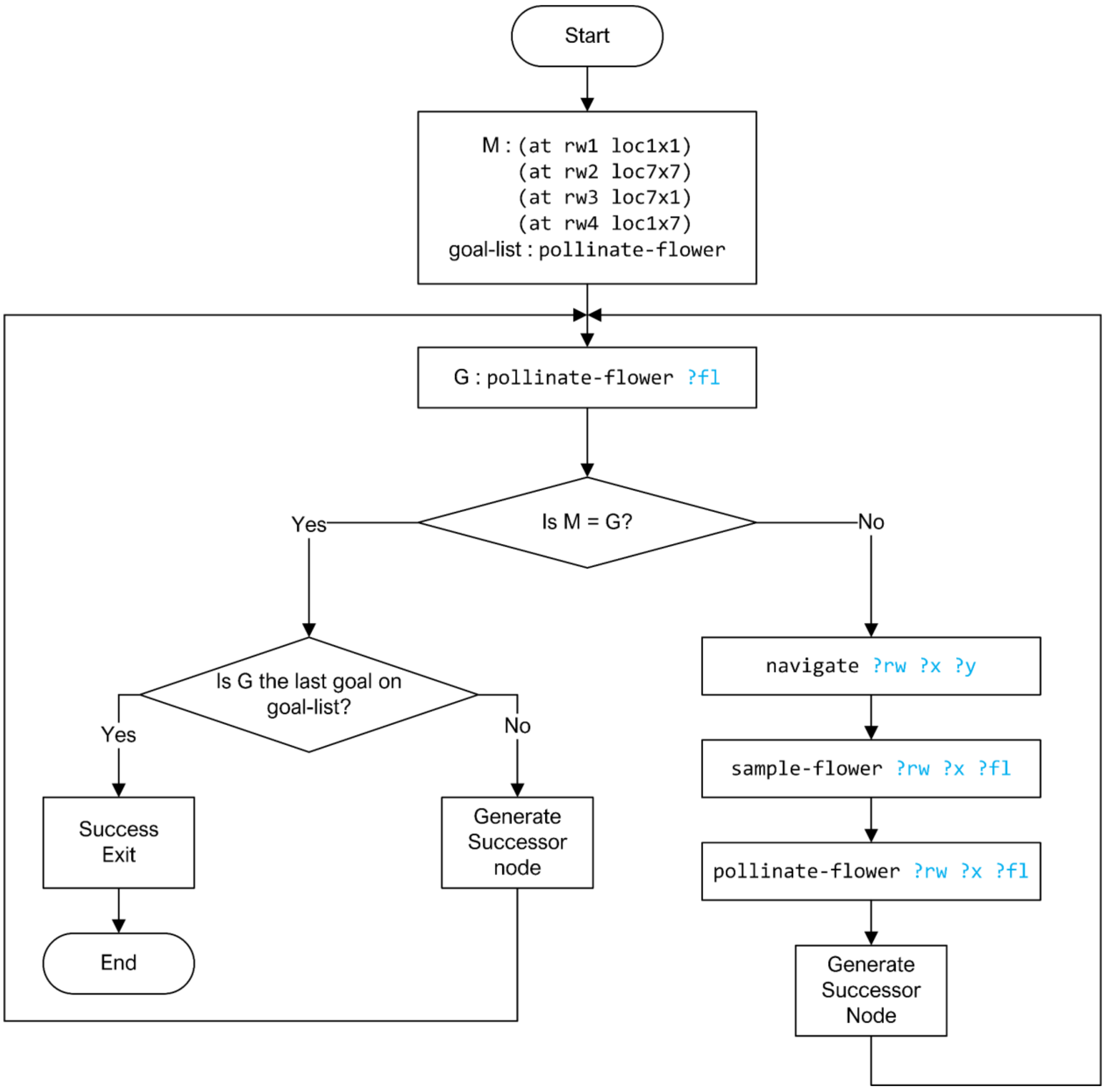


Figure 15 - STRIPS-like flow chart for problem 1 for set of actions defined in PDDL

4.3.3. Problem 2: Plan generation with organisational structure

Unlike the first two problems, this problem differs adversely in its structure. In the previous problem, all the 4 RoboWasps are equipped to perform the sampling and the pollination. As they behave in the same manner, the plan generation would be anticipated to be uniform for a given RoboWasp. Elevating this problem further, an organisational structure is established. As seen in section 2.5.2 for the tasks performed by the female wasps with respect to the male wasps, the female ones visit the flowers perpetually, whereas the male ones visit occasionally. Therefore, in this problem, two of the RoboWasps are purely operated as inspectors, whose primary task is to verify the maturity of the flower and communicate these to the other two RoboWasps, whose primary task is to transfer the pollen for the recognised mature flowers. The role of the Inspector RoboWasps is perpetual and iterative, whereas the role of other two RoboWasps only comes in when there is a need to transfer pollen in a mature flower. In implementing this, the PDDL code is shown in Code Listing 8.

Running this code yields to a plan generated with organisational structures in conjunction to the social laws and are presented as results.

Code Listing 8 - Object definition for Problem 2

```
(:objects rw1 rw2 - inspector_robowasp
          rw3 rw4 - pollinator_robowasp
          fl1 fl2 - flower                 ;row 1
          fl3 fl4 fl5 - flower             ;row 2
          fl6 fl7 fl8 fl9 - flower         ;row 3
          fl10 fl11 fl12 - flower          ;row 4
          fl13 fl14 fl15 fl16 - flower     ;row 5
          fl17 fl18 fl19 - flower          ;row 6
          fl20 fl21 - flower               ;row 7
          loc1x1 loc1x2 loc1x3 loc1x4 loc1x5 loc1x6 loc1x7 - location ;row 1
          loc2x1 loc2x2 loc2x3 loc2x4 loc2x5 loc2x6 loc2x7 - location ;row 2
          loc3x1 loc3x2 loc3x3 loc3x4 loc3x5 loc3x6 loc3x7 - location ;row 3
          loc4x1 loc4x2 loc4x3 loc4x4 loc4x5 loc4x6 loc4x7 - location ;row 4
          loc5x1 loc5x2 loc5x3 loc5x4 loc5x5 loc5x6 loc5x7 - location ;row 5
          loc6x1 loc6x2 loc6x3 loc6x4 loc6x5 loc6x6 loc6x7 - location ;row 6
          loc7x1 loc7x2 loc7x3 loc7x4 loc7x5 loc7x6 loc7x7 - location ;row 7
```

The state space representation for Problem 2a is shown in Figure 16. It can be noticed that the RoboWasps are coloured ‘blue’ and ‘orange’ to recognise their distinct operations. The ‘blue’ RoboWasps are the Inspectors and the ‘orange’ ones are the Pollinators.

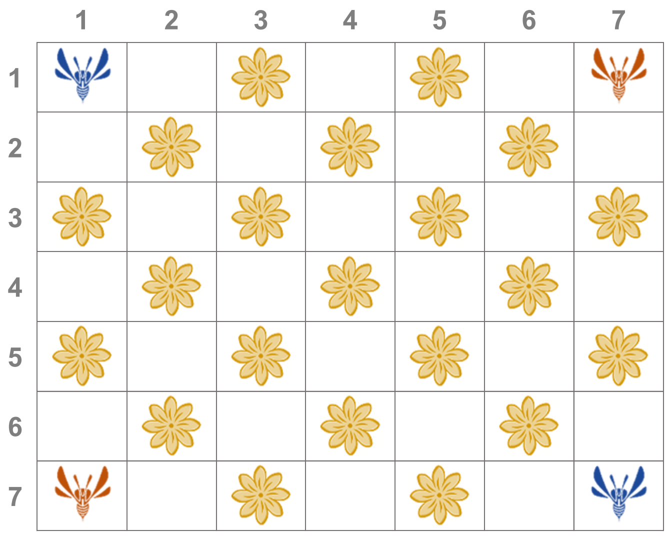


Figure 16 - State space representation for Problem 2a

To extend this problem further, where the ratio of number of RoboWasps for inspecting to pollinating is $2:2$, Problem 2b is considered to be having a ratio of $3:1$ to show how this asymmetric task delegation is managed by the swarm of RoboWasps. The state space representation would, then, be shown as in Figure 17.

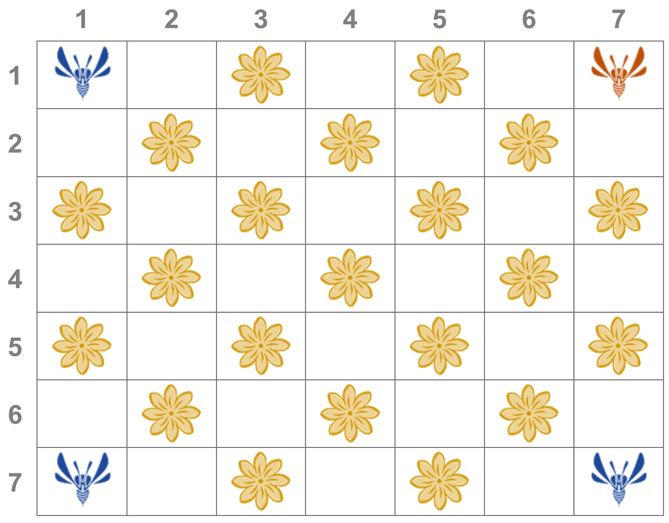


Figure 17 - State space representation for Problem 2b

In consideration of Problem 2c, the factors from Problem 2b are adopted, where a new RoboWasp is introduced in the problem. In Problem 2c, a fifth RoboWasp is introduced to operator as a pollinator. The fifth RoboWasp, `rw5` is located at `loc3x4`, as in Problem 1b, and is appointed to assist `rw4` in performing pollination whilst RoboWasps `rw1`, `rw2`, and `rw3` reprise their role of communicating the maturity of flowers. The ratio of Inspectors to Pollinator is $3:2$. The state space representation for Problem 2c is shown in Figure 19.

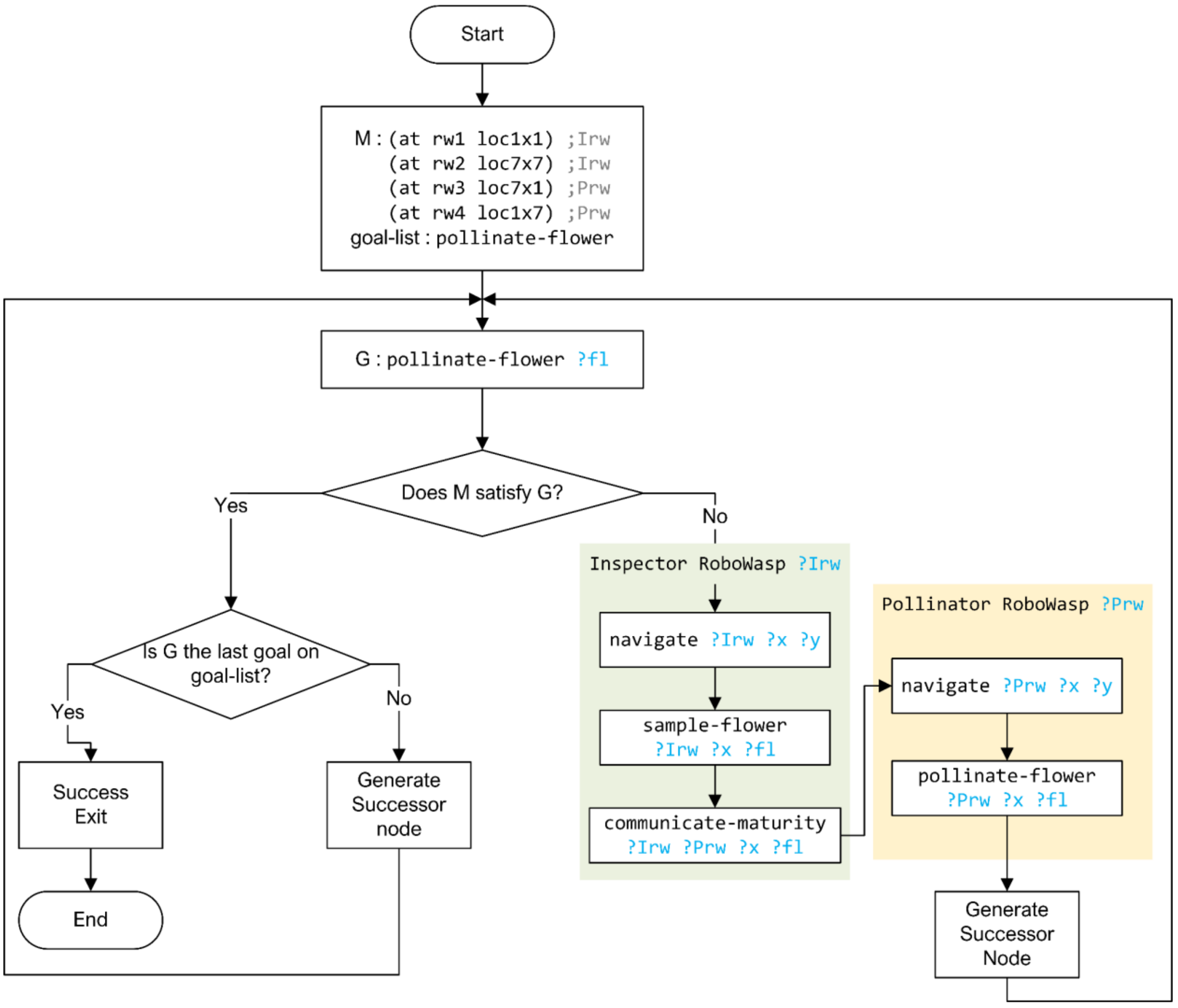


Figure 18 - STRIPS-like flow chart for problem 2 for the set of actions defined in PDDL

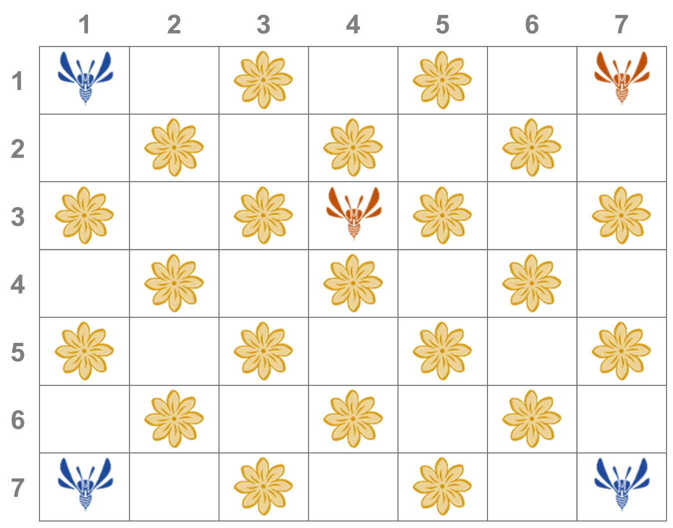


Figure 19 - State space representation for Problem 2c

4.3.4. Problem 3: Plan generation under interference by external agents

A special case is considered replicating a scenario that may arise in the real world. Assuming that there exist biological wasps in the same world where RoboWasps are operated, in such a scenario, there is a combination of natural and artificial pollination processes taking place simultaneously. It is important that RoboWasps execute a plan that runs in parallel to that executed by biological wasps. This special case demonstrates that the efficiency of plan generation and execution of RoboWasps is not compromised by the interference of external factors. In implementing this, a new object type is introduced called `biowasp`. There are several set of actions introduced with existing set of actions from the former problems (shown in Table 4), making this problem to be having the most number of possible actions to select from. The set of predicates are updated and shown in Code Listing 9.

Code Listing 9 - Set of predicates for Problem 3

```
(:predicates
    (at ?rw ?x)
    (at0 ?fl ?x)
    (at1 ?bw ?fl)
    (monitor-biological-wasps ?rw ?bw)
    (communicate-location-of-biological-wasp ?rw ?bw ?x)
    (ignore-flower-due-to-biological-wasp ?rw ?wsp ?bw ?fl)
    (sample-for-maturity ?rw ?fl)
    (is-mature ?fl)
    (is-immature ?fl)
    (ignore-flower ?rw ?fl)
    (pollinate-flower ?rw ?fl)
    (pollinate-flower0 ?bw ?fl)
)
```

This problem is special not only for the reason of interference of the biological wasps, but because it also brings in the conditions of maturity and immaturity of flowers with their underlying varied set of actions from Problem 0, which were discontinued in the former problems. The first and foremost reason to do this is to replicate this problem to a real-world scenario as much as possible, and secondly, to include characteristics of domains from all the former problems. Then, the state space can be represented as shown in Figure 20.

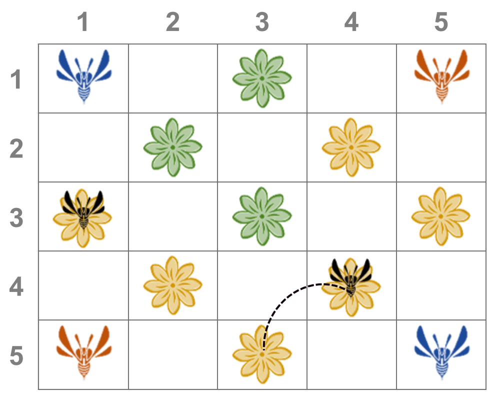


Figure 20 - State space representation for Problem 3

Code Listing 10 - Problem definition in PDDL for Problem 3

```
(:objects rw1 rw2 - inspector_robowasp
          rw3 rw4 - pollinator_robowasp
          bw1 bw2 - biowasp
          fl1 - flower                  ;row 1
          fl2 fl3 - flower              ;row 2
          fl4 fl5 fl6 - flower          ;row 3
          fl7 fl8 - flower              ;row 4
          fl9 - flower                  ;row 5
          loc1x1 loc1x2 loc1x3 loc1x4 loc1x5 - location ;row 1
          loc2x1 loc2x2 loc2x3 loc2x4 loc2x5 - location ;row 2
          loc3x1 loc3x2 loc3x3 loc3x4 loc3x5 - location ;row 3
          loc4x1 loc4x2 loc4x3 loc4x4 loc4x5 - location ;row 4
          loc5x1 loc5x2 loc5x3 loc5x4 loc5x5 - location ;row 5
)
(:init
    (at rw1 loc1x1) (at rw2 loc5x5) (at rw3 loc5x1) (at rw4 loc1x5)
    (at0 fl1 loc1x3) (at0 fl2 loc2x2) (at0 fl3 loc2x4) (at0 fl4 loc3x1)
    (at0 fl5 loc3x3) (at0 fl6 loc3x5) (at0 fl7 loc4x2) (at0 fl8 loc4x4)
    (at0 fl9 loc5x3) (is-immature fl1) (is-immature fl2) (is-mature fl3)
    (is-mature fl4) (is-immature fl5) (is-mature fl6) (is-mature fl7)
    (is-mature fl8) (is-mature fl9)
    (at1 bw1 fl4) (at1 bw2 fl8)
)
```

As it can be observed, the state space is reduced to a 5×5 matrix. This is because of the high number of actions introduced in this problem. More the actions, the solution would comprise of a broader sequence of this action, making it difficult to interpret the results. By reducing the state space from a 7×7 matrix to a 5×5 matrix, a significant number of flowers are reduced, whilst keeping the number of RoboWasps to be the same, i.e., four RoboWasps – two of which are the Inspectors and the other two are the Pollinators. In this problem, the two Inspectors monitor the activity of biological wasps and communicate these to the Pollinators, whilst the Pollinators sample the flowers, determine their maturity, and take the decision to either pollinate them or ignore them. As from Figure 16, the Inspectors are coloured 'blue' and the Pollinators are coloured 'orange'. The black wasps on the flower signify biological wasps. The dotted arc arising from one of the biological wasps indicates that it navigates to the next flower after pollinating its current one. This dynamic behaviour is monitored by the Inspector RoboWasp and communicated to the Pollinator RoboWasp that operates in that sub-domain. The maturity of the flower is as shown in Figure 10 – the yellow flowers are mature, and the green ones are immature. The PDDL code for the problem definition is given in Code Listing 10.

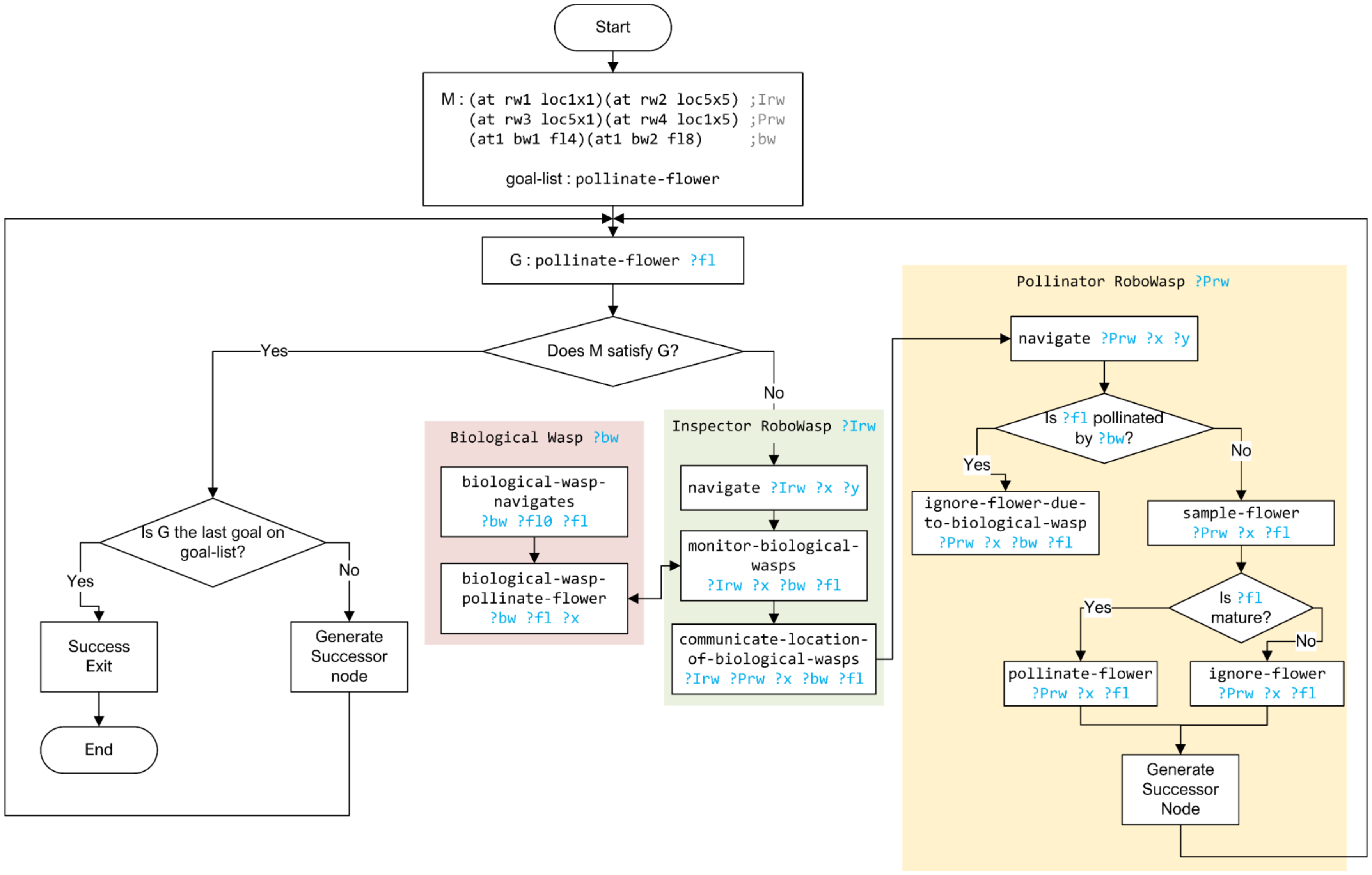

Start
M : (at rw1 loc1x1)(at rw2 loc5x5) ;Irw
(at rw3 loc5x1)(at rw4 loc1x5) ;Prw
(at1 bw1 fl4)(at1 bw2 fl8) ;bw
goal-list : pollinate-flower
G : pollinate-flower ?fl
Does M satisfy G?
Yes
No
Is G the last goal on goal-list?
Yes
No
Success Exit
End
Generate Successor node
Biological Wasp ?bw
biological-wasp-navigates ?bw ?fl0 ?fl
biological-wasp-pollinate-flower ?bw ?fl ?x
Inspector RoboWasp ?Irw
navigate ?Irw ?x ?y
monitor-biological-wasps ?Irw ?x ?bw ?fl
communicate-location-of-biological-wasps ?Irw ?Prw ?x ?bw ?fl
Pollinator RoboWasp ?Prw
navigate ?Prw ?x ?y
Is ?fl pollinated by ?bw?
Yes
No
ignore-flower-due-to-biological-wasp ?Prw ?x ?bw ?fl
sample-flower ?Prw ?x ?fl
Is ?fl mature?
Yes
No
pollinate-flower ?Prw ?x ?fl
ignore-flower ?Prw ?x ?fl
Generate Successor Node

Figure 21 - STRIPS-like flow chart for Problem 3 for set of actions defined in PDDL

CHAPTER FIVE

RESULTS

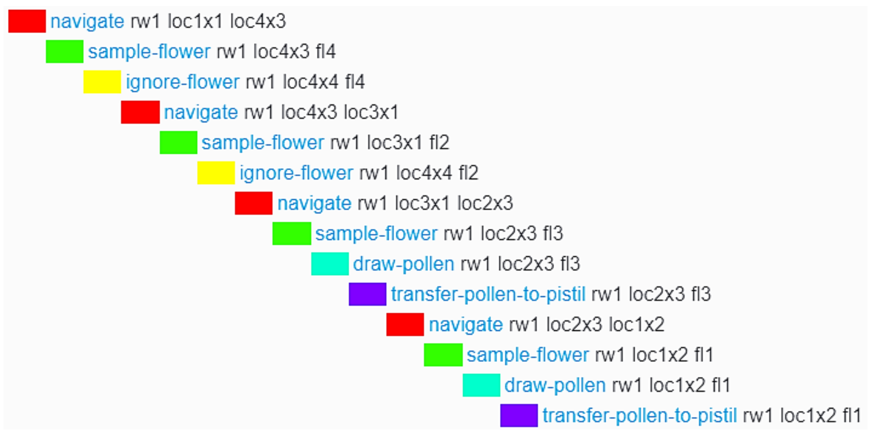
5.1. Chapter Overview

Further to the previous chapter, where the design and analysis were described and performed accordingly, this chapter shows results for each of the problems considered and interprets the plan generated, reflecting back to the theoretical concepts from Chapter 2.

5.2. Plan generation

5.2.1. Problem 0: Planning for a single-agent system

When the simulations are run for Problem 0, the results are as follows:



🟥 navigate rw1 loc1x1 loc4x3
🟩 sample-flower rw1 loc4x3 fl4
🟨 ignore-flower rw1 loc4x4 fl4
🟥 navigate rw1 loc4x3 loc3x1
🟩 sample-flower rw1 loc3x1 fl2
🟨 ignore-flower rw1 loc4x4 fl2
🟥 navigate rw1 loc3x1 loc2x3
🟩 sample-flower rw1 loc2x3 fl3
🟦 draw-pollen rw1 loc2x3 fl3
🟪 transfer-pollen-to-pistil rw1 loc2x3 fl3
🟥 navigate rw1 loc2x3 loc1x2
🟩 sample-flower rw1 loc1x2 fl1
🟦 draw-pollen rw1 loc1x2 fl1
🟪 transfer-pollen-to-pistil rw1 loc1x2 fl1

Figure 22 – Tasks and their colour codes in Problem 0

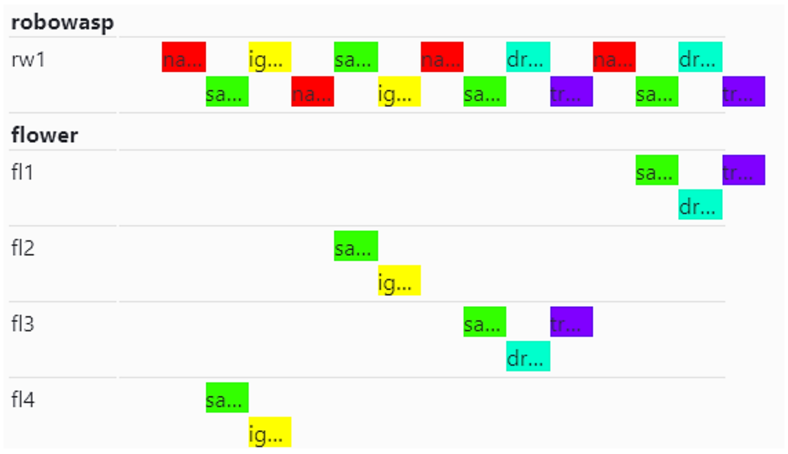


Figure 23 – Plan generated by the RoboWasp for Problem 0

5.2.2. Problem 1: Planning with no constraints

5.2.2.1. Problem 1a: Planning for a four-agent system

The plan generated by the four RoboWasps, `rw1`, `rw2`, `rw3`, and `rw4`, is shown in Figure 24.

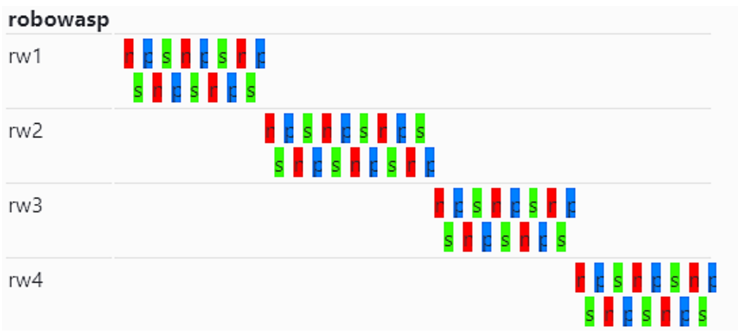
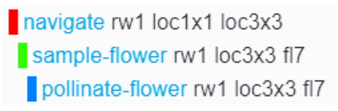


Figure 24 – Plan generated for Problem 1



🟥 navigate rw1 loc1x1 loc3x3
🟩 sample-flower rw1 loc3x3 fl7
🟦 pollinate-flower rw1 loc3x3 fl7

Figure 25 – Tasks and their colour codes in Problem 1

Figure 25 indicates to what each colour means in Figure 24. From those results, one could link the plan generated (Figure 24) for each RoboWasp. From this sequence, it can be interpreted that the 4 RoboWasps equally delegate their tasks of navigating their way through flowers, sampling their maturity, and then, pollinating them. When the sequence is to be related to the state space representation (Figure 12), the results obtained can be shown as:

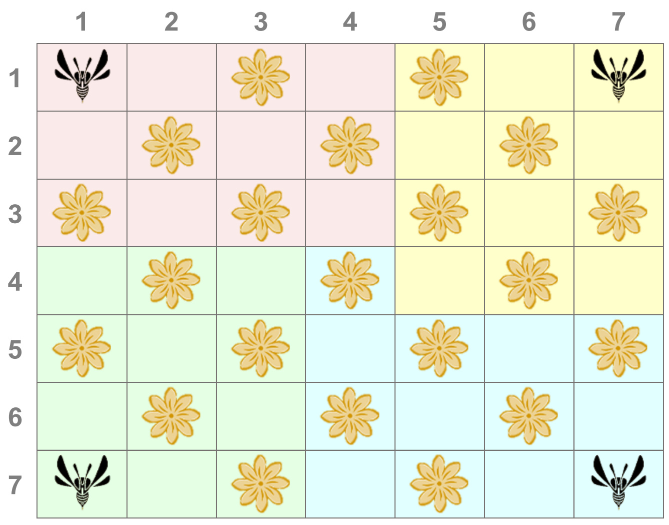


Figure 26 – State space representation of results obtained for Problem 1a

5.2.2.2. Problem 1b: Planning for a five-agent system

The plan generated by the five RoboWasps `rw1`, `rw2`, `rw3`, `rw4`, and `rw5`, is:

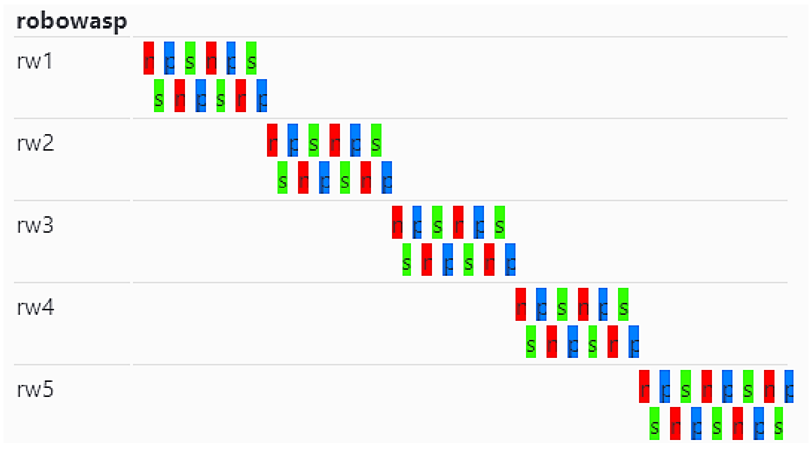


Figure 27 - Plan generated for Problem 1b

The state space representation for the result obtained for Problem 1b are shown as:

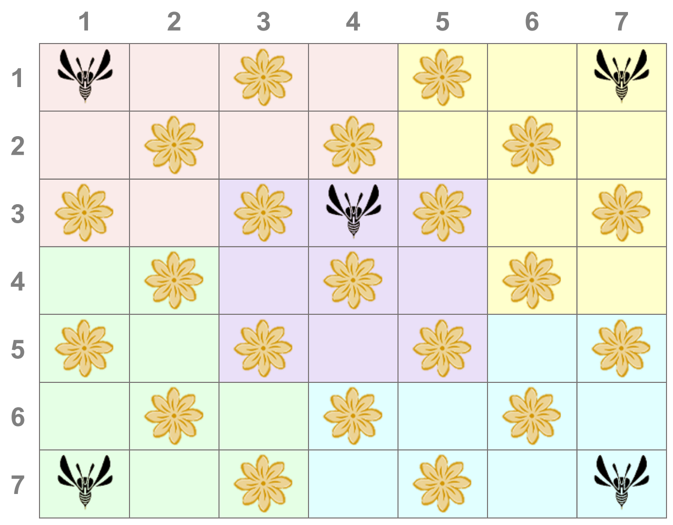


Figure 28 - State space representation of results obtained for Problem 1b

5.2.2.3. Problem 1c: Planning for a six-agent system

The plan generated by the six RoboWasps `rw1`, `rw2`, `rw3`, `rw4`, `rw5`, and `rw6`, is:

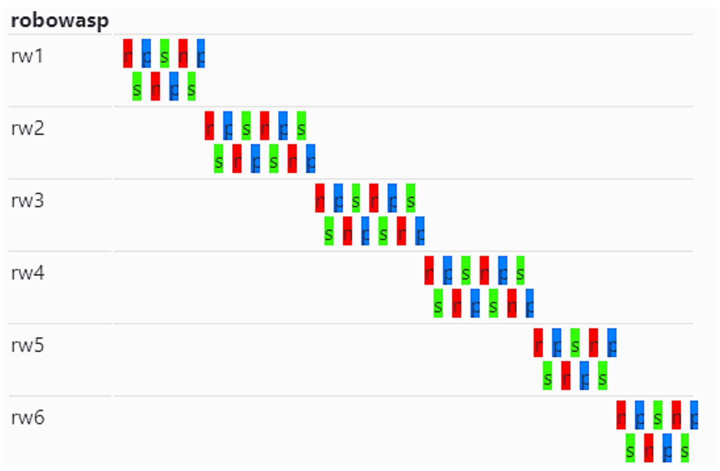


Figure 29 - Plan generated for Problem 1c

The state space representation for Problem 1c can be shown as:

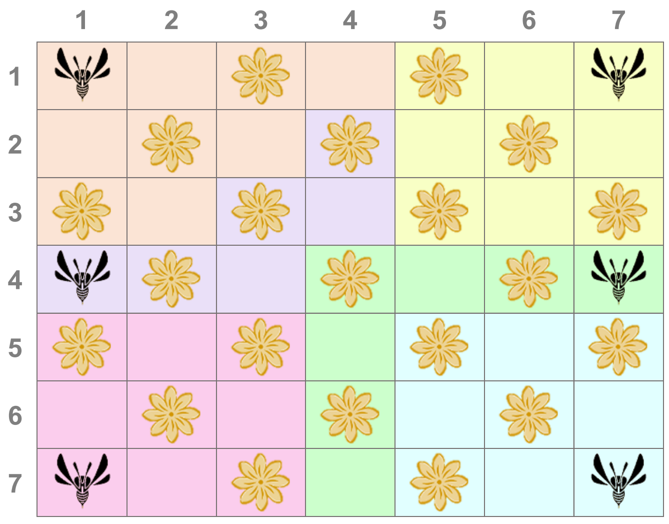


Figure 30 - State space representation of results obtained for Problem 1c

5.2.3. Problem 2: Planning with social laws and organisational structure

5.2.3.1. Problem 2a: Planning for RoboWasps when ratio of inspector to pollinator is 2:2

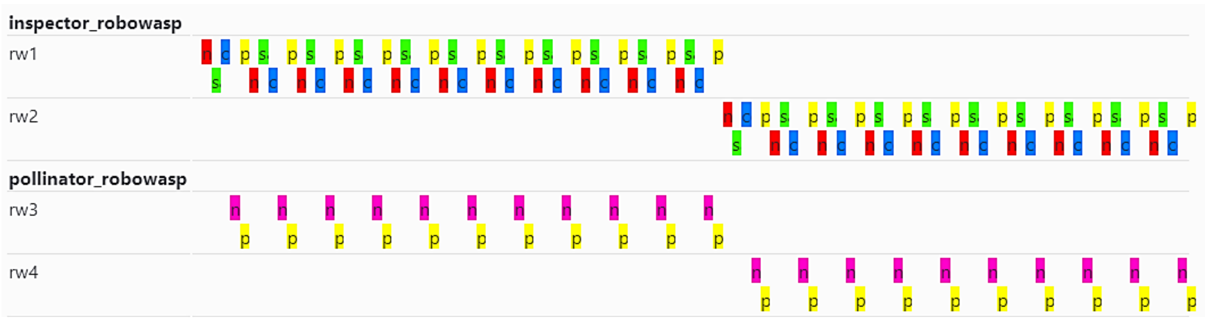


Figure 31 - Plan generated for Problem 2a

For Problem 2a, the ratio of number of Inspector RoboWasps to Pollinator RoboWasps is 2:2. From Figure 31, it can be observed that the inspectors perform sampling and communicate the maturity of the flower, whilst the pollinators navigate their way through the communicated flower and pollinate them. It can be observed that the Inspectors, `rw1` and `rw2`, have an iterative sequence of red, green and blue tasks, whilst the pollinators, `rw3` and `rw4`, have an iterative sequence of pink and yellow tasks. The state space representation is shown as:

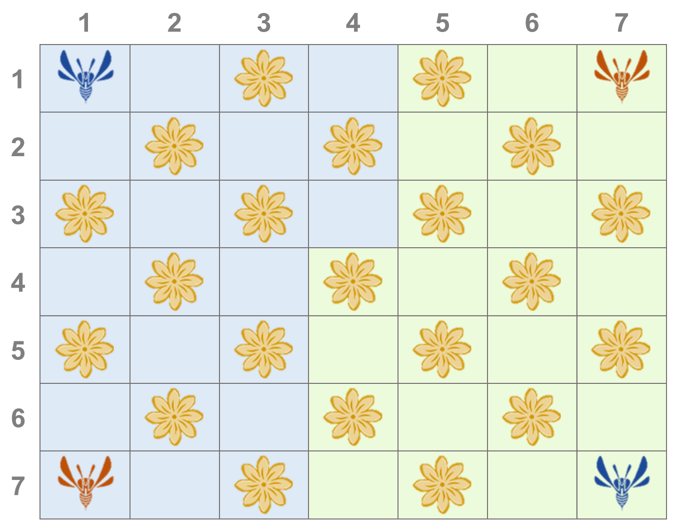


Figure 32 - State space representation of results obtained for Problem 2a

In Figure 32, the blue area is operated by RoboWasps `rw1` and `rw3`, whereas the green area is operated by the RoboWasps `rw2` and `rw4`.

5.2.3.2. Problem 2b: Planning for RoboWasps when ratio of inspector to pollinator is 3:1

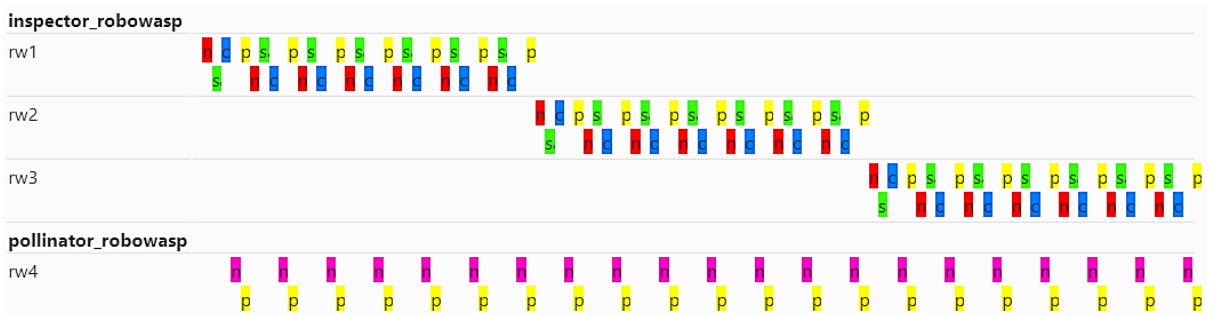


Figure 33 – Plan generated for Problem 2b

The state space representation for Problem 2b is shown in Figure 34. It can be observed from the figure that the Inspector RoboWasps, `rw1`, `rw2`, and `rw3`, delegate their tasks within their own sub-domains and the pollination is performed by the Pollinator RoboWasp, `rw4`, for the entire state space.

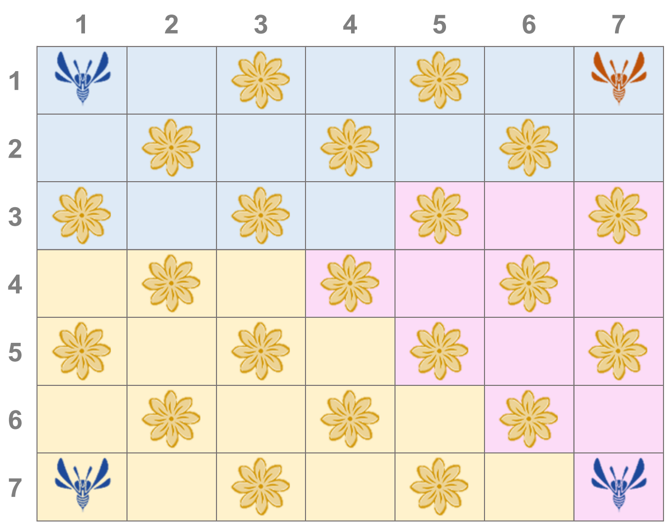


Figure 34 - State space representation for Problem 2b

5.2.3.3. Problem 2c: Planning for RoboWasps when ratio of inspector to pollinator is 3:2

The plan generated for Problem 2c is represented as:

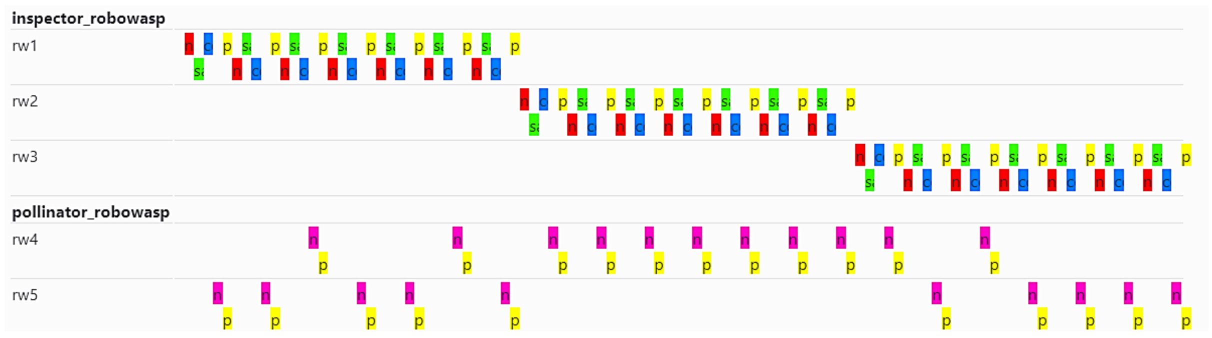


Figure 35 - Plan generation by each RoboWasp for Problem 2c

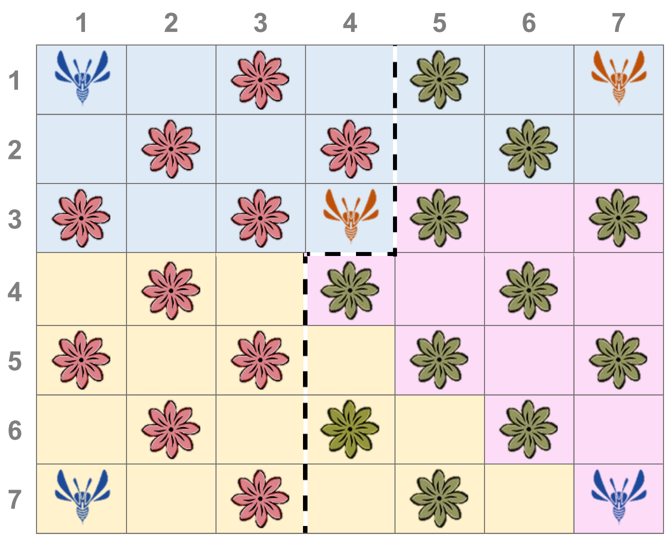


Figure 36 - State space representation for Problem 2c

5.2.4. Problem 3: Planning with interference from external agents

The plan generates for Problem 3 is as follows:

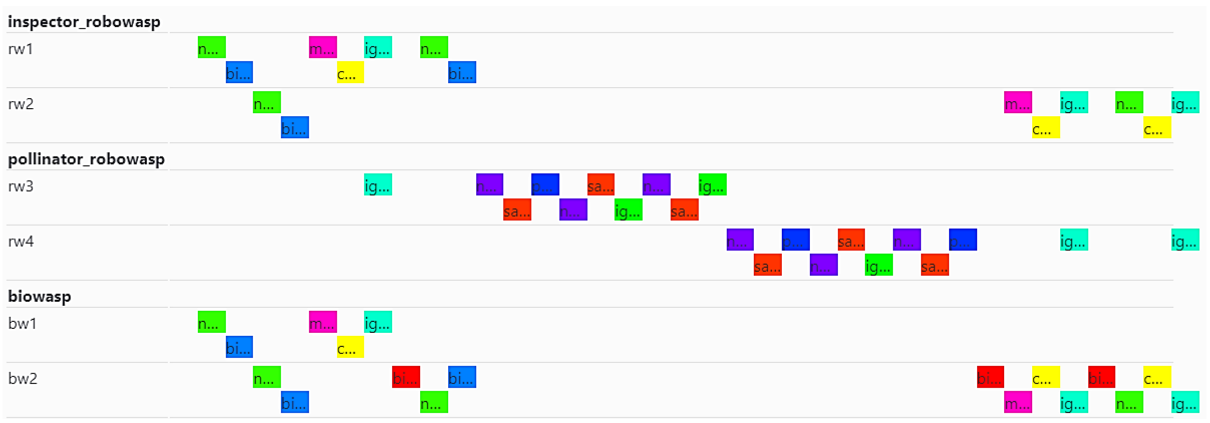
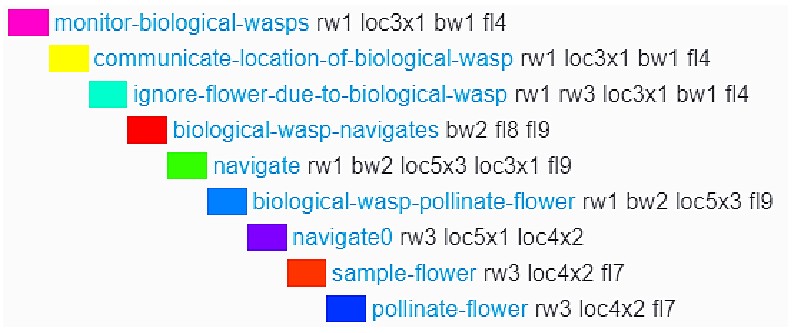


Figure 37 - Plan generated for Problem 3

To indicate what actions are represented by distinct colour bands, Figure 38 is included for reference.



🟪 monitor-biological-wasps rw1 loc3x1 bw1 fl4
🟨 communicate-location-of-biological-wasp rw1 loc3x1 bw1 fl4
🟦 ignore-flower-due-to-biological-wasp rw1 rw3 loc3x1 bw1 fl4
🟥 biological-wasp-navigates bw2 fl8 fl9
🟩 navigate rw1 bw2 loc5x3 loc3x1 fl9
🟦 biological-wasp-pollinate-flower rw1 bw2 loc5x3 fl9
🟪 navigate0 rw3 loc5x1 loc4x2
🟥 sample-flower rw3 loc4x2 fl7
🟦 pollinate-flower rw3 loc4x2 fl7

Figure 38 - Tasks and their colour codes for Problem 3

Referring the generated plan from Figure 37, the state space can be represented as:

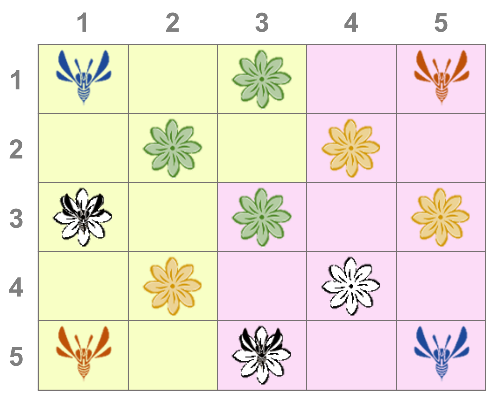


Figure 39 - Resulting state space representation for Problem 3

From Figure 39, it can be observed that the white flowers are pollinated by the biological wasps, `bw1` and `bw2`, and other mature flowers are pollinated by the Pollinator RoboWasps, `rw3` and `rw4`, according to the sub-domain they operate in. The immature flowers are ignored by the Pollinator RoboWasps. Inspector RoboWasps, `rw1` and `rw2`, monitor the activity of biological wasps and communicate this to the Pollinator RoboWasps.

5.3. Interpretation of results – accuracy and scientific evidence

5.3.1. Problem 0: Planning for a single-agent problem

As this simulation does not directly contribute towards the objectives of the project, this simulation is referred to as 'Problem 0'. By performing this simulation, the working of PDDL towards the application of the problem, i.e., artificial pollination, is demonstrated. The problem is defined with a single RoboWasp operating in the environment of multiple flowers, both mature and immature. The RoboWasp `rw1` recognises the maturity of the flower and as per the conditions satisfied, the RoboWasp either chooses to pollinate the flower or ignore it.

The structure of the problem within PDDL includes all the parameters of a classical planning model (discussed in section 2.4.1) – set of states (`:predicates` in PDDL), set of actions (`:action` in PDDL), initial states (`:init` in PDDL), and goal states (`:goal` in PDDL). Additionally, the cost factor can also be shown when the results are achieved. Although the action cost has not been considered in this project, PDDL offers an option to a domain designer to generate an automated plan considering action costs. Not just this, it is observed when running the analysis for Problem 0, the STRIPS planner, or more precisely, the PDDL framework automatically chooses among the two courses of action depending upon the conditions fulfilled, ultimately leading to either ignoring the flower or to pollinate it. The results demonstrate that after the sampling is done and the maturity is recognised, two flowers, `fl1` and `fl3`, are pollinated and the other flowers, `fl2` and `fl4`, are ignored.

5.3.2. Problem 1: Planning for a multiagent system with no constraints

Section 1.4 holds of integral importance in interpreting the results and justifying on the variations of the simulations performed from Problem 1 onwards. The three principles of SR are Robustness, Flexibility and Scalability. Problem 1a (Figure 12) considers a four-agent system in a finite, 7×7 matrix environment, to pollinate multiple flowers. It is observed that the tasks are equally distributed within the environment. When the resulting state space representation (Figure 26) was shown, each RoboWasp seems to have allotted their own sub-domain peripheral to themselves, supplementing with the 'navigate' action, that enables a RoboWasp to move to another flower after pollinating a flower that they are currently on.

With Problems 1b and 1c extended to a five-agent and a six-agent problem, respectively, the results (Figure 24, Figure 27, Figure 29) show that the tasks were equally distributed among the five RoboWasps and the six RoboWasps for Problems 1b and 1c, respectively. With an increase in the number of RoboWasps, the number of flowers per RoboWasp is decreased. The equal distribution of tasks indicate that despite the number of RoboWasps are increased, the tasks are as efficiently performed as seen in the four-agent problem. This verifies Robustness and Scalability of RoboWasps, fulfilling two of three principles of SR.

5.3.3. Problem 2: Planning with social laws and organisational structures

Considering the principles of SR – Robustness, Scalability and Flexibility, Problem 2a begins with two Inspector RoboWasps, `rw1` and `rw2`, and two Pollinator RoboWasps, `rw3` and `rw4`. With distinct operations performed by each RoboWasp within an almost symmetrically-divided sub-domain, as represented in the state space representation for Problem 2a (Figure 32), the problem is extends further to the number of Inspectors to the number of Pollinators to be $3:1$. Problem 2b considers `rw1`, `rw2` and `rw3`, to be Inspector RoboWasps, whilst `rw4` is the only Pollinator RoboWasp in the problem. Problem 2c extends Problem 2b further and introduces `rw5` as a new Pollinator RoboWasp. In this problem, whilst the Inspector RoboWasps, `rw1`, `rw2` and `rw3`, delegate their tasks equally within three almost equal sub-domains, the Pollinator RoboWasps, `rw4` and `rw5`, delegate their tasks to an almost symmetrically-divided sub-domain along with seeking the maturity data from the Inspectors.

Flexibility is the ability of the swarm of agents to be flexible and equipped in taking up different tasks to achieve the goals. When the initial ratio of Inspectors to Pollinators of $2:2$ in Problem 2a is changed to a $3:1$, `rw3` turns an Inspector from Pollinator, the results (Figure 33) show that the inspection within the environment is divided almost equally by the three RoboWasps, whilst the pollination is solely performed by `rw4`. Scalability is the ability of the swarm of agents to adapt to tasks and achieve goals despite of an increase or a decrease in their number. In Problem 2c, when a new RoboWasp, `rw5`, is introduced to assist `rw4` on the pollination, the tasks are equally delegated within the two RoboWasps, whilst seeking the communicated maturity data from the Inspectors.

Further, an additional element that is different for Problem 2 is the cooperation and collaboration among RoboWasps, defined by the conditions (also on page 22) in Table 5.

Table 5 - Interpretation of results in context to the cooperation and collaboration by RoboWasps for Problem 2

$\boldsymbol{P_\alpha = P_\alpha^\alpha \cup P_\alpha^\beta}$ *and* $\boldsymbol{P_\beta = P_\beta^\beta \cup P_\beta^\alpha}$	In Problem 2, in all the simulations, with the Inspector-Pollinator ratio of $2:2$, $3:1$, and $3:2$, where let α be the Inspectors and β be the Pollinators, the plan generated by α, P_α^α, is the plan generated with a goal to communicate the maturity data, whereas the plan generated by β, P_β^β is the plan generated with a goal to pollinate the mature flowers. P_α^β and P_β^α are the plans generated for sharing maturity data that would initiate β agents to pursue the plan P_β^β. The ultimate plan P_α and P_β is, therefore, towards pollinating the flowers.
$\boldsymbol{P_\alpha^\alpha \cup P_\beta^\alpha \vDash_{D_\alpha \cup D_\beta, I_\alpha \cup I_\beta} G_\alpha}$ *and* $\boldsymbol{P_\beta^\beta \cup P_\alpha^\beta \vDash_{D_\alpha \cup D_\beta, I_\alpha \cup I_\beta} G_\beta}$	For this problem, the goal to be achieved for agents α, G_α, is to sample the flower with communicating the maturity data, and the goal to be achieved for agents β, G_β, is to pollinate the flowers when received with maturity data. With the results (Figure 31, 34, 36) achieved, this mathematical statement can be proved to be true.
if $\boldsymbol{P_\alpha^\alpha \vDash_{D_\alpha, I_\alpha} G_\alpha}$ *then* $\boldsymbol{P_\beta^\alpha = \emptyset}$ *and* *if* $\boldsymbol{P_\beta^\beta \vDash_{D_\beta, I_\beta} G_\beta}$ *then* $\boldsymbol{P_\alpha^\beta = \emptyset}$	Taking the previous mathematical statement and the parameters, the plan P_α^α is limited to sampling the flowers with the assumption that the β agent group does not exist and the plan P_β^β is limited to solely pollinating the flowers with the assumption that the α agent group does not exist, then the possibility of the plans P_β^α and P_α^β for sharing the sampling data are eliminated, and therefore, $P_\beta^\alpha = P_\alpha^\beta = \emptyset$.
$\boldsymbol{P_\alpha}$ *and* $\boldsymbol{P_\beta}$ *are non-conflicting*	From the first condition fulfilled, where P_α and P_β share common goals, this condition is fulfilled. This is shown to be true from the results obtained (Figure 16, 17, 19).

5.3.4. Problem 3: Planning with interference of external agents

This problem is regarded as a special case. It is mentioned in section 4.3.4 that one of the reasons to call it so is because it combines all the characteristics from the former problems – the sampling of maturity of flowers leading to distinct set of underlying actions, introducing an organisational structure for RoboWasps, and determining their performance by adding more agents to the environment.

Recalling the principles of SR - Robustness, Flexibility and Scalability - Problem 3 fulfils all these principles. Robustness is demonstrated by the efficient generation of the plan with the given set of actions and working towards achieving the goal state by utilising these effectively. Flexibility of Inspector RoboWasps in adapting to new set of actions, i.e., monitoring the external agents (biological wasps) whilst their existing set of actions are performed by Pollinator RoboWasps, demonstrates the second principle. Scalability is accomplished in a different way than usual. The fact that the number of flowers are reduced due to the intervention of biological wasps, RoboWasps perform the pollination of the remaining flowers, i.e., 6 rather than 9 flowers, and yet achieve an equally-distributed plan, demonstrates the third principle of SR.

Additionally, as cooperation and collaboration among RoboWasps is an integral part in this problem, so as the previous one, the conditions are verified for Problem 3 in Table 6.

Table 6 - Interpretation of results in context to the collaboration and cooperation by RoboWasps for Problem 3

$\boldsymbol{P_\alpha = P_\alpha^\alpha \cup P_\alpha^\beta}$ *and* $\boldsymbol{P_\beta = P_\beta^\beta \cup P_\beta^\alpha}$	In Problem 3, where let α be the Inspectors and β be the Pollinators, the plan generated by α, P_α^α is the plan generated with a goal to communicate the location of biological wasps, whereas the plan generated by β, P_β^β is the plan generated with a goal to avoid the flowers pollinated by biological wasps whilst ignoring the immature flowers and pollinating the remaining mature flowers. P_α^β and P_β^α is the plan generated for sharing the current location of biological wasps that would initiate β agents to pursue the plan P_β^β. The ultimate plan P_α and P_β is, therefore, pollinating the flowers with underlying conditions of avoiding the flowers pollinated by the biological wasps in the domain.
$\boldsymbol{P_\alpha^\alpha \cup P_\beta^\alpha \vDash_{D_\alpha \cup D_\beta, I_\alpha \cup I_\beta} G_\alpha}$ *and* $\boldsymbol{P_\beta^\beta \cup P_\alpha^\beta \vDash_{D_\alpha \cup D_\beta, I_\alpha \cup I_\beta} G_\beta}$	For this problem, the goal to be achieved for agents α, G_α, is to monitor the activities of biological wasps with communicating their location and the goal to be achieved for agents β, G_β, is to either ignore or pollinate the remaining flowers when the monitoring information is received. With the

	results (Figure 37, 39) achieved, this mathematical statement can be proved to be true.

if $\boldsymbol{P_\alpha^\alpha} \vDash_{\boldsymbol{D_\alpha,I_\alpha}} \boldsymbol{G_\alpha}$ *then* $\boldsymbol{P_\beta^\alpha} = \emptyset$ *and*
 	
 if $\boldsymbol{P_\beta^\beta} \vDash_{\boldsymbol{D_\beta,I_\beta}} \boldsymbol{G_\beta}$ *then* $\boldsymbol{P_\alpha^\beta} = \emptyset$
	Taking the previous mathematical statement and the parameters, the plan P_α^α is limited to monitoring the biological wasps with the assumption that the β agent group does not exist and the plan P_β^β is limited to solely ignoring/pollinating the immature/mature flowers with the assumption that the α agent group does not exist, then the possibility of the plans P_β^α and P_α^β for sharing the monitoring information is eliminated, and therefore, $P_\beta^\alpha = P_\alpha^\beta = \emptyset$.

$\boldsymbol{P_\alpha}$ *and* $\boldsymbol{P_\beta}$ *are non-conflicting*	From the first condition fulfilled, where P_α and P_β share common goals, this condition is fulfilled. Moreover, this is shown to be true from the results obtained (Figure 37).

CHAPTER SIX

DISCUSSION AND REFLECTION

6.1. Reflection

6.1.1. On the idea

The development of RoboWasps is an open-ended problem with multiple subsystems that could be studied and implemented. With Aerodynamic, propulsion system, wireless communication system, and control system, the project focusses on one subsystem within which the planning system for a swarm of RoboWasps is developed. Combining Nature and Technology as the core foundations towards the development of this project, examples of how biological wasps interact with each other is studied and their behaviour in achieving tasks collectively is applied towards an artificial pollination problem.

The project is developed with certain themes such as bio-inspiration, AI, automated planning, and SR. In addition to these themes being at the forefront of research in the present time, such a project would lead to an independent and a unique research analysis that is anticipated to have promising contribution towards the engineering sector. The possibility of such a project was confirmed from the sources of inspiration (section 1.2.1, page 4) – RoboBees, SubCULTron and CoCoRo, which were inspired from natural groups such as swarm of bees and school of fishes. Their applications were related to finding an appropriate problem to solve within this project.

Not just this project is seen as a futuristic technological solution, but closely-related projects have also been at the forefront of research in present time. Typical examples include the development of social drones by WASP-HS research group at Chalmers University of Technology in Sweden [127, 128], the development of a swarm of DargonFlEye by TU Delft in the Netherlands [129], and the development of Inspectrone project at DTU in Denmark [130]. With all these factors in consideration, the idea of RoboWasps could provide a robust starting point towards solving ‘real world’ problems with pragmatic approaches.

6.1.2. On the methodology

The research begins with an extensive literature study of different parameters that contribute towards implementation of the problem. The nature of domains, prominent planning models and various implementation techniques were studied and scrutinised with respect to their structure, parameters and mathematical composition towards fulfilling the project objectives. All of these criterions were considered further in Chapter 3 in developing a methodology with logical justifications. With selecting domain-specific problems, it was justified that this was a better starting point over domain-independent problems as the latter would require significant experience on the field of study. When the planning models were critically evaluated, primarily to justify starting from a simplistic model yielding to accurate results, classical planning was chosen to be considered for further analyses.

Additionally, further elements were considered in designing a domain and defining a problem by considering attributes of biological wasps. One such element which was of interest was the introduction of social laws. Two potential approaches - durative actions and the MSLA approach - were explored. Although not forming an integral part of the analysis, durative actions were simpler to apply within the problem considered, with an anticipation of performing concurrent actions. This step would not have been necessary, was the MA-PDDL version available for implementation. The features of MSLA, such as the dynamic nature of goal and capabilities, which would generate the stochastic structure of a plan, shows that this approach is more useful for an open environment. Whilst they were studied, these were not used for implementation as no significant use was seen for the Problems considered in this project.

In selecting an appropriate implementation tool, two programming frameworks were compared. Among PDDL and JaCaMo, the former was considered for implementation with respect to the criterions indicated in Table 2. Despite the apparent benefits of using PDDL for analyses, it came with challenges. One of the primary challenges were the learning of a new programming framework in a limited amount of time whilst working towards achieving results. Secondly, with restricted amount of resources available and for a niche programming language, solving syntax and runtime errors could be time-consuming. However, with persistent research and examples found on github, the programming framework was studied and applied on the problem within a few weeks, whilst reflecting back to the literature review, to relate the implementation to theoretical concepts.

For the given timeframe, this methodology seems justified in light of the results obtained. Where previous projects, such as the design of Arthrobot [131], demonstrated closely related methodology to the development of RoboWasps, the methodology was unique and independent to the reference materials in development of this project. Not just this, the idea of aiming towards achieving the goal state with no robust representation of the planning structure could prove challenging in reflecting the theoretical concepts to the results obtained. This was not the case when using PDDL.

6.1.3. On the analyses

A classical planning model is selected for implementation. The parameters that define classical planning are clearly defined in PDDL. Parameters within the defined set of actions, such as preconditions and effects, confirm that all the features of the classical planning tuple are addressed when constructing a domain and defining a problem, including the inclusion of the state transition function within the definition of actions. As the classical planning model is used within a closed environment, STRIPS solving technique is invoked in PDDL. The flow-chart of iteration can be recognised by the sequence of actions represented in the plans generated for all the problems (Figure 11, 15, 18, 21). With these indications, the accuracy of analyses can be related to the theoretical concepts discussed in Chapter 2.

In Chapter 4, the problem was primarily classified to Problem 0: Single-agent problem, Problem 1: Multiagent problem, Problem 2: MAP with organisational structure, and Problem 3: MAP with interference by external agents. Problems 1 and 2 were further expanded to three distinct cases. Problem 1a began with a four-agent system, extending to Problems 1b and 1c with a five-agent and six-agent system, respectively. Problem 2a began with an organisational structure with a ratio of $2:2$, extending to Problems 2b and 2c with a ratio of $3:1$ and $3:2$, respectively. Problem 3 combines all the attributes from the former problems, along with extending to generating a plan with interference of external agents. All of the problems were structured to demonstrate that the planning system for RoboWasps fulfils the three principles of SR - Robustness, Flexibility and Scalability.

In addition to this, Problem 3 deals with dynamic behaviour of external agents, i.e., their positions change with time, however, as this behaviour is declared within the problem, these factors are deterministic. However, the computation gets more complex and expensive when their positions change further whilst the goal states are achieved, so much so that a suspected timeout occurs due to use of excess memory. As this was a special case considered within the classical planning model, there could be other alternatives in solving this dynamic behaviour of external agents, such as adopting contingent, conformant, or epistemic planning models.

6.1.4. On the results

In Chapter 5, corresponding results obtained for Problems 0, 1, 2, and 3 are presented. With different simulations performed, the problem at hand improves progressively in terms of the number of agents, their operation and abilities, and the restricted roles they would perform when assigned to a specific operation. With considerable time effort put in defining a domain, if such problems were to be extended to a 100-agent problem or a 1000-agent problem, where the RoboWasps would operate in a large environment, PDDL could prove to be an efficient tool in developing a planning system for a larger swarm of RoboWasps, yet could be time-consuming and computationally expensive.

6.2. Strengths

In Chapter 1, working towards the planning system was considered to be the objectives of this project over ML and SI approaches in developing RoboWasps. The reason in doing so was to build upon the fundamentals of designing an artificial swarm of wasp-sized drones, inspired by operations observed in biological wasps. When comparing this justification with the number of simulations performed, along with the sequence of which they were developed, it can be said that these were built upon progressively. At the same time, the simulations work towards satisfying the criterions of SR, and these were proven to be satisfying of the results.

When these are to be applied to any other domain where the RoboWasps would operate, structuring problems in a progressive manner would enable the domain designer in correcting errors in an efficient manner rather than building just one realistic simulation straight-away. This methodological and stepwise approach is considered to be one of the apparent strengths, as these would enable a perpetual development of the project whilst enabling a better understanding and easier troubleshooting of the code. Although it can also be said that the scope of an automated planning system for a swarm of RoboWasps is enormous, this approach proves a good starting point if this project was to progress in adding other attributes, such as ‘learning from experience’ of RoboWasps.

6.3. Weaknesses

Although it can be justified that the simulations performed were structured considering the timeframe and the computational facility available, one could list a few limitations in the approach taken and the results achieved. One of the major limitations of the analyses is the lack of goal preference. In the real-world, as observed in humans, in situations where we deal with multiple tasks simultaneously, we prioritise executing the most important task first and then deal with lesser important tasks. In the same manner, when RoboWasps are operating in an environment, the ignored flowers may be considered as less important tasks, but should still be addressed when they are mature. Moreover, in the simulations performed, the pollination of the flower is considered to be the most important action by default, however, when there is interference from external agents, their monitoring needs to be considered of utmost important over pollinating flowers. This is seen as a limitation in Problem 3. Moreover, it can also be seen that the PDDL solver considers completing actions that have lesser number of corresponding actions by default, and then completes actions that are more complex. This indicates the lack of goal preference by RoboWasps. This is seen in the plan generated in Problem 0 (Figure 23). The immature flowers are addressed first as they have a straightforward corresponding action, i.e., to ignore them. Once all of the immature flowers are addressed, the mature flowers are addressed as they have two further actions to achieve the goal state, i.e., to draw pollen and then, to transfer these to the pistil.

Secondly, although Problem 3 was considered as a special case, this possibility is more likely to occur in the real world, and it is important to extend this problem further. Considering a case where the biological wasps may recognise RoboWasps as a threat, they have to be equipped with a mechanism to communicate to biological wasps that they are in the environment to assist rather than attack. This is shown by the waggle dance mechanism in RoboBees (section 1.2.1.1) to communicate with honeybees [132].

Both of these weaknesses exhibit dynamic behaviour within an environment, and if these were to be addressed, a classical planning model may prove to be insufficient. In such a situation, either a different planning model should be adopted, or additional attributes should be included in the design of RoboWasps. As these were considered outwith the scope of this project, these weaknesses are not seen as affecting the project deliverables in any way.

6.4. Recommendations

In addition to overcoming the weaknesses indicated in section 6.3, other practical aspects of the design could be explored. Whilst the theory in dealing with the uncertainties in the real world, a more sophisticated planning model could be helpful. Epistemic planning model (section 2.4.5) considers uncertainties at the core of its structure. If the artificial pollination problem was to be extended in an uncertain environment,

Along with a sophisticated planning model, a powerful implementation tool could supplement on further improvements. JaCaMo is a programming framework that combines three distinct programming languages in designing the environment, the agents' beliefs and behaviour, and the organisation (section 2.6.4). This would demonstrate a robust planning system in uncertain environments, where the beliefs and behaviour of the agents could lead to logical reasoning in solving problems within an environment. A typical example of how beliefs in a MAP contribute with an epistemic planning model (DEL approach), in solving the Muddy Children Puzzle. As shown by Kominis and Geffner [133], three agents work collaboratively in recognising whether a child is muddy. A combination of joint beliefs by the agents and conditional beliefs by individual agents enable them in reaching the goal state.

CHAPTER SEVEN

CONTRIBUTION AND CONCLUSION

7.1. Industrial Applications

In this section, various examples are explored where RoboWasps can be used with the existing design implemented in this project. In addition to this, factors of uncertainties are explored in these examples and modifications of the RoboWasp design is recommended to address these uncertainties. The list is not exhaustive and the application of RoboWasps is considered to be extensive across multiple engineering sectors.

7.1.1. Smart indoor monitoring

Figure 40 – An illustration of monitoring by RoboWasps inside an Airport

Consider a high-security indoor venue, say an International Airport, where hundreds of CCTV cameras are mounted for a thorough surveillance. The reason to install a multiple CCTV camera setup in different angles is to cover the maximum field of view within a space, especially when the camera is mounted at a fixed point [134]. Now consider that these cameras were free to move through the space to cover this field of view. This could be possible by replacing these CCTV cameras with a swarm of RoboWasps. Where a swarm of RoboWasp that are equipped with cameras and recognising objects in an environment, they could move through the space, which could eventually lead to using a significantly lesser number of cameras for a more precise surveillance.

In this example, consider a case of detecting adequate lighting in an environment, a swarm of RoboWasps is appointed in recognising this and reporting the exact location where the lighting is inadequate or in excess. The set of predicates, can then, be anticipated as:

Code Listing 11 - Anticipated set of predicates for the lighting adequacy problem

```
(:predicates
    (at ?rw ?loc)
    (mounted-with-camera ?rw)
    (equipped-with-ability-to-detect-objects ?rw)
    (monitor-location ?rw ?loc)
    (is-too-dark ?loc)
    (is-too-bright ?loc)
    (lighting-is-adequate ?loc)
    (mark-as-ADEQUATE ?rw ?loc ?lt)
    (mark-as-TOO-DARK ?rw ?loc ?lt)
    (mark-as-TOO-BRIGHT ?rw ?loc ?lt)
    (communicate-location-status ?rw ?wsp ?loc)
)
```

The uncertainty factor is a common challenge that would be faced in such environments. For a state space being too bright or too dark, this would also depend on the brightness due to sunlight, the frequency of using specific areas with respect to the time of the day, and variations in weather and climatic conditions. All of these dynamic factors would call for a swarm of RoboWasps to recognise such changes and develop a routine to make accurate changes to the lighting of indoor venues by artificial learning of the repetition of these factors. Recalling the former example of replacing CCTV cameras by a swarm of RoboWasps, when the uncertainty factor and the dynamics of the environment are addressed, the problems could be transpired to monitoring of dynamic entities such as movement of people within the venue and recognising suspicious activities with alerting the concerned surveillance teams. This is specifically beneficial in high-security venues such as an Airport. When this is developed, there are many approaches that are described in this project that could be explored for this specific environment.

The automated planning approaches described in section 1.2.4.3 are - programming-based, learning-based and model-based. With programming-based approach being a starting point (as seen in the artificial pollination problem) for automated planning, where plans are highly reliant on the programmer, these could be progressed further by other planning approaches. For the adequate lighting problem, when learning-based approaches are adopted, involving ML and RL, an ANN can be built by using Python frameworks such as Tensorflow and Keras, as built by Milani *et al.* [135] towards their planning model. They highlight that when compared to classical planning, the preconditions for defining actions are not defined by the programmer, and are instead learned through the trained ANN. Such an approach would prove extremely helpful when the programmer is unable to determine preconditions for an uncertain environment, whereas the swarm of RoboWasps can.

7.1.2. Construction

"Drones make the identifying of any defects easier, safer and quicker. Engineers can [then] carry out repairs without delay."

- McEachran, R.[136]

Within the construction industry, drones can prove very helpful in monitoring, inspecting and mapping construction sites [137]. Where minute human factors can lead to catastrophic consequences, including threat to human lives, drones overcome this limitation by regular monitoring and inspection of sites. Ashour *et al.* [138] propose inspection by drones for three procedures - violation inspections, progress of work inspections, and complaints assistance.

Figure 41 - An illustration of RoboWasps in the construction sector

In this example, consider an inspection procedure performed by a swarm of RoboWasps to track progress of work done. This problem can be reduced to a 'spot the differences' problem. Along with an efficient data transfer by the swarm of RoboWasps of different captured perspectives of the construction site, additional attributes such as pattern recognition, image processing and computer vision are anticipated to be introduced as per the intensity of the problem. Therefore, it can be said that the domain design and problem-solving would go beyond the programming-based planning.

Considering the learning-based approach and by using JaCaMo instead of the PDDL framework, the three dimensions to be defined are: the agent, the environment, and the organisation. These are anticipated as described in Table 7.

Table 7 - Anticipated parameters for the work progress problem within the construction sector

Dimension	Framework	Parameters
Agent	$\mathcal{J}ason$	• <u>Belief</u>: Each monitoring iteration stores the output achieved within agents' belief. • <u>Action</u>: An agent captures different views of the construction site for a given monitoring iteration i, recognise the difference with the previous iteration $(i-1)$, and returns the value as 'changes made'. • <u>Goal</u>: The image captured matches the design plan produced by the Project Manager/Site Engineer.
Environment	$CArtAgO$	• <u>Artefacts</u>: The difference between the initial processed image to the current processed image is the changes made between i and $(i+1)$ monitoring routine. • <u>Workspaces</u>: Each iteration of monitoring routine $i \in [1, n]$ is stored within agents' belief, where n is the routine attaining to goal state.
Organisation	$\mathcal{M}oise^{+}$	• <u>Norm</u>: Any duplication of image captured is automatically deleted. This would enable efficient use of memory and intelligent acquisition of data.

With the parameters anticipated to be as tabulated, when the image is captured and the difference is returned for each monitoring iteration, this could be verified with the project plan to check the progress of the construction completed. Additionally, features such as automatic deletion of duplicate images would ensure avoiding overuse of the memory than expected and eliminate the possibilities of any confusion that may arise when verifying the progress of the project.

7.1.3. Healthcare

Many useful applications of drones in healthcare have been recognised - from autonomous delivery of food, water and first aid kits to remote locations through to performing CPR by an automatic defibrillator [139]. In fact, in the current situation, where the world is fighting a global pandemic and the healthcare sector is fragile and critical, drones can prove extremely helpful, especially in ensuring a safe and zero-human contact. Manna Aero, an Irish drone company, is working towards this and is now at its experimentation stage, reports Weckler [140]. In addition to this, drone applications within healthcare are precisely explained by Rosser et al. [141] and classified as shown in Figure 42.

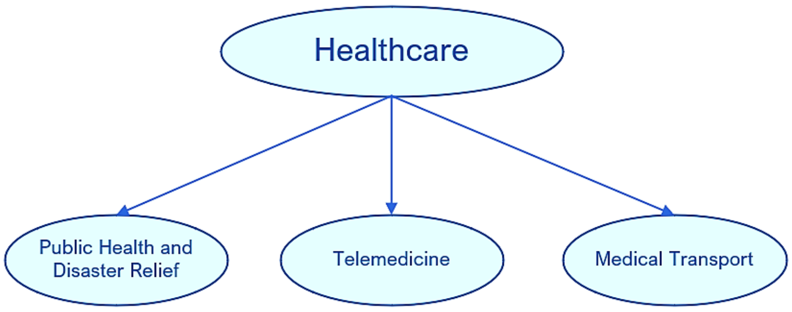


Figure 42 - Classification of drone applications within healthcare

Consider a blood sample is to be transported within a Hospital. A swarm of RoboWasps are appointed to perform the task. Such a problem could be reduced to a path-finding problem within a MAS. Considering a programming-based problem where labelled blood samples are available within a specific room and is requested by a medical department. This is communicated to a specific RoboWasp which is available to perform the task. The RoboWasp is equipped to recognise labelled blood samples and match the requested blood sample data to the labelled samples. Once this match is found, the RoboWasp is equipped to carry the blood sample and navigates from the blood sample room to the requested location.

Figure 43 - An illustration of RoboWasp carrying the blood sample

Considering this problem within a classical planning model in PDDL, the set of predicates can be anticipated as shown in Code Listing 12. The uncertainty factor in this problem is minimal, except the fact that when multiple RoboWasps are available in performing the tasks and when a certain task is not assigned to a specific RoboWasp, there may be a squabble between them on whether who should perform the task. Instead of being cooperative, RoboWasps may be required to negotiate in meeting an acceptable agreement. Whilst negotiation can be introduced into the problem, another alternative is to assign numbers to RoboWasps and initiate an iterative loop of check availability in numerical order to `rw1` to `rwN`, where there exist N number of RoboWasps. In addition to this, the matching of request sample to searched blood sample, this could be done by using `:fluents`, where the ASCII values of the text of the samples are compared and matched. In this comparison, should the value return a zero, this would indicate that the searched blood sample matches the requested blood sample. All of this is defined as an action within the domain definition file. When these challenges are overcome within a programming-based approach, neither may it be required to adopt learning- or model-based approaches for this specific problem, nor required to extend the problem with an epistemic planning model.

Code Listing 12 - Anticipated set of predicates for the blood sample problem

```
(:predicates
    (at ?rw ?samp_room)
    (at0 ?rw ?req_loc)
    (mounted-with-camera ?rw)
    (read-requested-sample-data ?rw ?req_samp)
    (equipped-with-ability-to-detect-text ?rw)
    (equipped-to-carry-blood-samples ?rw ?bld_samp)
    (match-samples ?bld_samp ?req_samp)
    (carry-sample ?rw ?bld_samp)
    (navigate ?rw ?samp_room ?req_loc)
    (store-transported-sample-data ?rw ?bld_samp ?req_samp)
)
```

7.1.4. Hazardous and restrictive spaces

One of the apparent benefits highlighted towards the development of RoboWasps was its significantly small size. This benefit can lead to impressive applications of RoboWasps, such as using them in restricted spaces such as pipelines, ducts, and restricted spaces such as the inside of an oil tanker. Consider that the inside of an oil tanker's hull is to be inspected, and this is recognised as a finite, yet continuous state space. A swarm of RoboWasps is appointed to detect damage such as cracks or major dents that may compromise on the structural reliability of the hull. Each of the RoboWasps is equipped with a camera. They are then trained towards detecting a crack or a dent. Therefore, this problem could be categorised as a learning-based planning system, and because the preconditions are not known during the design stage, this would have to be decided upon by the RoboWasps. Once the cracks or dents are detected, these are stored along with the specific RoboWasp that found this damage and the exact location within the state space where the damage exists. This is, then, reported to the site engineer so that the damage could be dealt with. The planning attribute of RoboWasp within this problem could be their cooperative working towards goals. Given the number of RoboWasps available, the subdomains could either be equally distributed and decided upon by RoboWasps randomly, or could be assigned with a sequence by labelling the RoboWasps from `rw1` to `rwN`, where N is the total number of RoboWasps present.

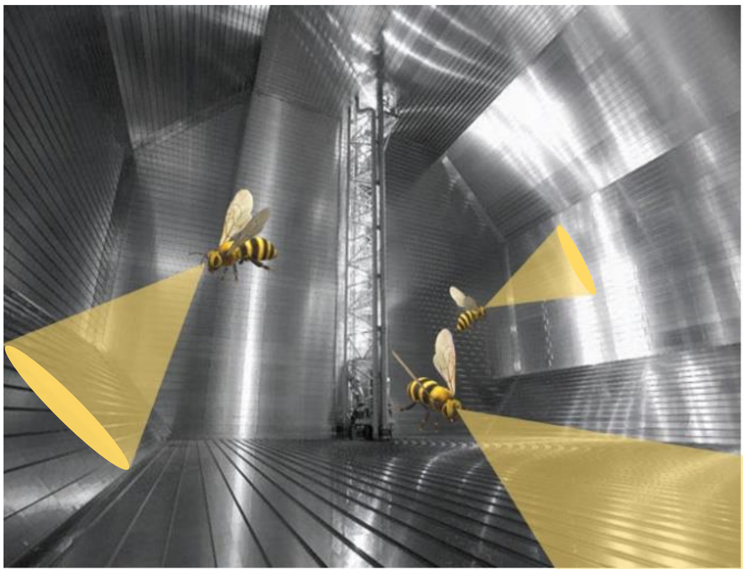


Figure 44 – An illustration of RoboWasps inspecting an oil tanker

7.1.5. Agricultural practices

Figure 45 - An illustration of a swarm of RoboWasps in an indoor farm

Although this project has considered one of the examples towards the implementation of RoboWasps, there are additional applications within agricultural practices where RoboWasps can prove helpful. Consider an indoor farm, where a large number of crops, herbs and vegetables are grown within a confined space. Towards agricultural practices, tasks may range in nature such as detecting unhealthy crops, weeds and infested elements in the environment, along with detecting whether a crop is ready for harvesting. If all of these tasks were to be performed by the swarm of RoboWasps, one of the straightforward approaches could be introducing an organisational structure, as in Problem 2 of the research analysis. The first group of RoboWasps could detect unhealthy and infested crops, whilst the other group could detect crops ready for harvesting. However, it can be observed that there is no requirement of collaboration or negotiation among the two groups, and therefore, they would not fulfil the criterions of cooperative attributes of MAS, as described in section 2.5.

7.2. Challenges

The RoboWasp project has many applications and effective potential contributions to the industry, however, certain challenges exist. Two of the apparent challenges are described below.

7.2.1. Public impression

When I described the idea of RoboWasps to friends, family, colleagues and potential employers, most seemed impressed, however, a few worried about the uncanny consequences of the *dark* side of RoboWasps. A few inquired – “Would your RoboWasps sting?” – to which my chuckling response was – “No, certainly not!”. This fear is not just restricted to RoboWasps, but is instilled for biological wasps as well. A survey conducted on people across 46 countries in 2018 indicated that wasps are considered nuisances rather than ecological assets [142]. One of the challenges in developing RoboWasps, to ensure that they are embraced, would be to overturn this public perspective and to convince people that they would prove to be helpful rather than harmful. There is so much to learn from RoboWasps – their birth from Nature, their robust teamwork skills and their uniform work efficiency at all times. When these are emphasised to people who fear them, this perspective could be overturned.

7.2.2. Risk assessment

A pragmatic challenge that holds for the swarm of RoboWasps is to analyse their potential risks and determine their reliability. As the swarm of RoboWasps are machines after all, they are prone to damage like any other machine. If programming-based approaches to automated planning were involved, it can be simpler and straightforward to control and troubleshoot the RoboWasps, however, in learning-based and model-based approaches, where a programmer would have minimal intervention in their operation, this may pose a complex challenge. So how would we confront AI? Would it be beyond our ken to stop it when required?

Although there may be some unpredictability on assessing risks for such a technology at this stage, Cheatham *et al.* [143] list three potential risk mitigations - clarity, breadth and nuance. Clarity can be defined as a structured identification approach in detecting critical risks. Breadth can be defined as sharpening the understanding on the models and training them on a regular basis to detect failures. Nuance can be defined as reinforcing specific controls depending upon the nature of risks. Although potential mitigations are known, this article cannot be relied as these are not tested and verified. This may prove as a complex challenge that one would have to overcome when taken in context to the development of RoboWasps.

7.4. Conclusion

Nature and Technology are the two foundations towards the development of a planning system for a swarm of RoboWasps. With sources of inspiration such as RoboBees and SubCULTron, the concept of RoboWasps is put in perspective and related to different branches within AI to determine the project objectives. Once these are established, different approaches within automated planning - programming-based, learning-based, and model-based - were explored, with further research on different parameters that would contribute towards implementation. The nature of domain, planning models and tools for implementation were studied, compared and critically evaluated to develop a methodology with justifications to the selected approach with respect to multiple criterions. With the methodology given, analysis is performed in PDDL with progressive problems, starting from a single-agent problem through to tackling external factors in achieving goals. With results demonstrated, further recommendations are made of including dynamic and nondeterministic factors into the environment and improving the analyses by adding agents' attributes such as beliefs and learning from experience. The project concludes with a comprehensive description of applications of RoboWasps across multiple engineering sectors.

REFERENCES

1. Copeland BJ. Artificial Intelligence. *Encyclopædia Britannica*. https://www.britannica.com/technology/artificial-intelligence: Encyclopædia Britannica, inc., 2019.

2. Marr B. What Are Artificial Neural Networks - A Simple Explanation For Absolutely Anyone. *Forbes*. https://www.forbes.com/sites/bernardmarr/2018/09/24/what-are-artificial-neural-networks-a-simple-explanation-for-absolutely-anyone/#205861a71245: Enterprise & Cloud, 2018.

3. Pattanayak S. Mathematical Foundations. *Pro Deep Learning with TensorFlow: A Mathematical Approach to Advanced Artificial Intelligence in Python*. Berkeley, CA: Apress, 2017, p.1-87.

4. Moolayil JJ. A Layman's Guide to Deep Neural Networks. Towards Data Science 2019.

5. Simos DE. Genetic Algorithms for the Construction of 2^2 and 2^3 -Level Response Surface Designs. In: Lagaros ND and Papadrakakis M, (eds.). *Engineering and Applied Sciences Optimization: Dedicated to the Memory of Professor MG Karlaftis*. Cham: Springer International Publishing, 2015, p.207-15.

6. Eremia M, Liu C and Edris A. Multiagent Systems. *Advanced Solutions in Power Systems: HVDC, FACTS, and Artificial Intelligence*. IEEE, 2016, p.903-30.

7. Hosch WL. Machine Learning. *Encyclopædia Britannica*. https://www.britannica.com/technology/machine-learning: Encyclopædia Britannica, inc., 2019.

8. Sternberg RJ. Human intelligence. *Encyclopædia Britannica*. https://www.britannica.com/science/human-intelligence-psychology: Encyclopædia Britannica, inc., 2017.

9. Tzorakoleftherakis E. Reinforcement Learning: A Brief Guide. United Kingdom: MathWorks, 2019.

10. Connor S. The core of truth behind Sir Issac Newton's apple. *The Independent*. United Kingdom: The Independent, 2010.

11. Salata M. How taking a bath led to Archimedes' principle. YouTube: Ted-Ed, 2012.

12. Moynihan R and Walkow M. Airbus unveils a 'Bird of Prey' concept plane inspired by eagle's wings. *Business Insider*. Poland 2019.

13. Professional Engineering. The Bird of Prey: Following the flight path of our feathered friends. *Professional Engineering by Institution of Mechanical Engineers (IMechE)*. 2019: 64.

14. Jafferis NT, Helbling EF, Karpelson M and Wood RJ. Untethered flight of an insect-sized flapping-wing microscale aerial vehicle. *Nature*. 2019; 570: 491-5.

15. Williams A. Real-life Transformers are being inspired by insects, not cars. *Professional Engineering*. IMechE, 2019, p.41-5.

16. Wood R. RoboBees project. *Harvard University*. 2015.

17. Harvard University. RoboBees: Robotic insects make first controlled flight (w/ video). Phys.org, 2013.

18. Landgraf T. RoboBee: a biomimetic honeybee robot for the analysis of the dance communication system. 2013.

19. Soffel J. These RoboBees could pollinate crops and save disaster victims. World Economic Forum, 2016.

20. Breuer K. Flight of the RoboBee. Nature Publishing Group, 2019.

21. Leiber N. Subcultron: Swarming Robots that keen an Eye on Waterways. Bloomberg, 2016.

22. Bonner W. Robot swarms will explore the waterways of Venice. Fox News, 2015.

23. University of Graz. SubCULTron: Project Description. University of Graz, 2015.

24. Riva G. SubCULTron: Robotic Exploration of Unconventional Environmental Niches. *CyberPsychology, Behavior & Social Networking*. 2017; 20: 775.

25. CORDIS. Robot swarms use collective cognition to perform tasks. Phys.org, 2015.

26. Clauser G. What Is Alexa (and What's the Best Alexa Speaker)? *Electronics > Accessories*. The Wire Cutter, 2019.

27. Kavanagh D. The Future of Personalization in Ecommerce. *Global Web Index*. 2019.

28. Johnson K. How Google Maps uses machine learning to predict bus traffic delays in real time. *AI*. Venture Beat, 2019.

29. Ketkar N. Introduction to Deep Learning. *Deep Learning with Python: A Hands-on Introduction*. Berkeley, CA: Apress, 2017, p.1-5.

30. Aron J. How innovative is Apple's new voice assistant, Siri? *New Scientist*. 2011; 212: 24.

31. Putrevu J, Subramanya SB, Mathihalli M, Natarajan G, Yuan G and Guo F. System for page type based advertisement matching for sponsored product listings on e-commerce websites and method of using same. Google Patents, 2019.

32. Castrodale J. This is how Google Maps knows which route is the fastest at any given moment. USA Today, 2015.

33. Neo B. The Future of Machine Learning. Online: Towards Data Science, 2019.

34. Tanner G. What is Machine Learning and why is it important. Online: Medium, 2018.

35. Alpaydin E. *Introduction to machine learning*. 2nd ed.. ed. Cambridge, Mass.: Cambridge, Mass. : MIT Press, 2010.

36. Rosa JPS, Guerra DJD, Horta NCG, Martins RMF and Lourenço NCC. Overview of Artificial Neural Networks. *Using Artificial Neural Networks for Analog Integrated Circuit Design Automation*. Cham: Springer International Publishing, 2020, p.21-44.

37. Pattanayak S. Introduction to Deep-Learning Concepts and TensorFlow. *Pro Deep Learning with TensorFlow: A Mathematical Approach to Advanced Artificial Intelligence in Python*. Berkeley, CA: Apress, 2017, p.89-152.

38. Chen A. Heads up for the gathering robot swarm. *Science Magazine*. ScienceMag.org, 2014.

39. Professional Engineering. Swarm of tiny drones finds 'disaster victims' with minimal computing power. *Professional Engineering by IMechE*. 2019.

40. Chittka L and Mesoudi A. Insect Swarm Intelligence. *Science*. 2011; 331: 401.

41. Hein AM, Rosenthal SB, Hagstrom GI, Berdahl A, Torney CJ and Couzin ID. The evolution of distributed sensing and collective computation in animal populations. *Elife*. 2015; 4.

42. Yang X-S. *Nature-Inspired Computation in Engineering [internet resource]*. 1st ed. 2016.. ed.: Cham : Springer International Publishing, 2016.

43. Geffner H. Planning and Autonomous behavior. In: Bonet B, (ed.). *A concise introduction to models and methods for automated planning*. San Rafael, Calif. (1537 Fourth Street, San Rafael, CA 94901 USA): IEEE Xplore: Morgan & Claypool, 2013.

44. Hamann H. Introduction to Swarm Robotics. *Swarm Robotics: A Formal Approach*. Cham: Springer International Publishing, 2018, p.1-32.

45. Vlassis N. Rational Agents. *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*. Morgan & Claypool, 2007, p.7.

46. Russell SJ and Norvig P. *Artificial intelligence: a modern approach*. 3rd ed.. ed. Upper Saddle River, N.J.: Upper Saddle River, N.J. : Prentice Hall, 2010.

47. Wooldridge MJ. *An introduction to multiagent systems*. New York: J. Wiley, 2002.

48. Wickler G and Tate A. What is Planning? *AIPLAN*. United Kingdom: Open Education Edinburgh, 2014.

49. Höller D, Bercher P, Behnke G and Biundo S. On guiding search in HTN planning with classical planning heuristics. *Proc of the IJCAI*. 2019.

50. Behnke G, Höller D, Bercher P and Biundo S. More Succinct Grounding of HTN Planning Problems-Preliminary Results. 2019.

51. Smith DE and Weld DS. Conformant graphplan. *AAAI/IAAI*. 1998, p.889-96.

52. Geffner H and Lipovetzky N. Width and serialization of classical planning problems. 2012.

53. Vallati M and Chrpa L. On the Robustness of Domain-Independent Planning Engines: The Impact of Poorly-Engineered Knowledge. *Proceedings of the 10th International Conference on Knowledge Capture*. Marina Del Rey, CA, USA: Association for Computing Machinery, 2019, p.197–204.

54. Ghallab M, Nau D and Traverso P. *Automated Planning: theory and practice*. Elsevier, 2004.

55. Kuter U, Nau D, Reisner E and Goldman RP. Using classical planners to solve nondeterministic planning problems. *Proceedings of the Eighteenth International Conference on International Conference on Automated Planning and Scheduling*. Sydney, Australia: AAAI Press, 2008, p.190–7.

56. Celorrio SJ and Jonsson A. Computing plans with control flow and procedures using a classical planner. *Eighth Annual Symposium on Combinatorial Search*. 2015.

57. Alford R, Shivashankar V, Roberts M, Frank J and Aha DW. Hierarchical Planning: Relating Task and Goal Decomposition with Task Sharing. *IJCAI*. 2016, p.3022-9.

58. Dempster MAH, Fisher M, Jansen L, Lageweg B, Lenstra JK and Rinnooy Kan A. Analytical evaluation of hierarchical planning systems. *Operations Research*. 1981; 29: 707-16.

59. Alford R, Shivashankar V, Roberts M, Frank J and Aha DW. Hierarchical planning: relating task and goal decomposition with task sharing. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. New York, New York, USA: AAAI Press, 2016, p.3022–8.

60. Bercher P, Höller D, Behnke G and Biundo S. User-Centered Planning. *Companion Technology*. Springer, 2017, p.79-100.

61. Bercher P, Richter F, Hörnle T, et al. A planning-based assistance system for setting up a home theater. *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

62. Behnke G, Höller D, Bercher P, et al. *Hierarchical Planning in the IPC*. 2019.

63. Honold F, Bercher P, Richter F, et al. Companion-Technology: Towards User- and Situation-Adaptive Functionality of Technical Systems. *2014 International Conference on Intelligent Environments*. 2014, p.378-81.

64. Shivashankar V, Alford R, Kuter U and Nau D. The GoDeL planning system: a more perfect union of domain-independent and hierarchical planning. *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. Beijing, China: AAAI Press, 2013, p.2380–6.

65. Bercher P, Keen S and Biundo S. Hybrid planning heuristics based on task decomposition graphs. *Seventh Annual Symposium on Combinatorial Search*. 2014.

66. Bercher P, Behnke G, Höller D and Biundo S. An Admissible HTN Planning Heuristic. *IJCAI*. 2017, p.480-8.

67. Hoffmann J and Brafman R. Contingent planning via heuristic forward search with implicit belief states. *Proc ICAPS*. 2005.

68. Albore A, Palacios H and Geffner H. A translation-based approach to contingent planning. *Twenty-First International Joint Conference on Artificial Intelligence*. 2009.

69. Majercik SM and Littman ML. Contingent planning under uncertainty via stochastic satisfiability. *AAAI/IAAI*. 1999, p.549-56.

70. Lespérance Y, De Giacomo G and Ozgovde AN. A model of contingent planning for agent programming languages. *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*. 2008, p.477-84.

71. De Giacomo G, Lespérance Y and Levesque HJ. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*. 2000; 121: 109-69.

72. Hoffmann J and Brafman RI. Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*. 2006; 170: 507-41.

73. Cashmore M. Planning as Quantified Boolean Formulae. In: University of Strathclyde. Dept. of Computer and Information S, (ed.). Thesis [Ph. D] -- University of Strathclyde, 2013, 2013.

74. Cimatti A, Roveri M and Bertoli P. Conformant planning via symbolic model checking and heuristic search. *Artificial Intelligence*. 2004; 159: 127-206.

75. Bolander T. A gentle introduction to epistemic planning: The DEL approach. *arXiv preprint arXiv:170302192*. 2017.

76. Bolander T and Andersen MB. Epistemic planning for single-and multi-agent systems. *Journal of Applied Non-Classical Logics*. 2011; 21: 9-34.

77. Baltag A and Renne B. Dynamic epistemic logic. 2016.

78. Pacuit E. Dynamic Epistemic Logic I: Modeling Knowledge and Belief. *Philosophy Compass*. 2013; 8: 798-814.

79. Cooper MC, Herzig A, Maffre F, Maris F and Regnier P. A simple account of multi-agent epistemic planning. *Proceedings of the Twenty-second European Conference on Artificial Intelligence*. The Hague, The Netherlands: IOS Press, 2016, p.193–201.

80. Engesser T, Bolander T, Mattmüller R and Nebel B. Cooperative epistemic multi-agent planning for implicit coordination. *arXiv preprint arXiv:170302196*. 2017.

81. Huang X, Fang B, Wan H and Liu Y. A General Multi-agent Epistemic Planner Based on Higher-order Belief Change. *IJCAI*. 2017, p.1093-101.

82. Clement BJ and Barrett AC. Continual coordination through shared activities. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. Melbourne, Australia: Association for Computing Machinery, 2003, p.57–64.

83. Dimopoulos Y and Moraitis P. Multi-Agent Coordination and Cooperation through Classical Planning. *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. 2006, p.398-402.

84. Dukas R and Real LA. Learning foraging tasks by bees: a comparison between social and solitary species. *Animal Behaviour*. 1991; 42: 269-76.

85. Ågotnes T and Wooldridge M. *Optimal social laws*. Toronto, Canada: International Foundation for Autonomous Agents and Multiagent Systems, 2010, p.667–74.

86. Borrajo D and Fernández S. Efficient approaches for multi-agent planning. *Knowledge and Information Systems*. 2019; 58: 425-79.

87. Fitoussi D and Tennenholtz M. Choosing social laws for multi-agent systems: Minimality and simplicity. *Artificial Intelligence*. 2000; 119: 61-101.

88. Wu J, Zhang L, Wang C and Xie J. *Mechanism Design for Social Law Synthesis under Incomplete Information*. São Paulo, Brazil: International Foundation for Autonomous Agents and Multiagent Systems, 2017, p.1757–9.

89. Nir R and Karpas E. Automated Verification of Social Laws for Continuous Time Multi-Robot Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI, 2019.

90. Shoham Y and Tennenholtz M. On the synthesis of useful social laws for artificial agent societies (preliminary report). 1992.

91. Fitoussi D and Tennenholtz M. Minimal Social Laws. *Association for the Advancement of Artificial Intelligence*. 1998; AAAI-98.

92. Boella G and Torre Lvd. *Enforceable social laws*. The Netherlands: Association for Computing Machinery, 2005, p.682–9.

93. James R, James RR and Pitts-Singer TL. *Bee pollination in agricultural ecosystems*. Oxford University Press on Demand, 2008.

94. Boissier O, Bordini RH, Hübner JF, Ricci A and Santi A. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*. 2013; 78: 747-61.

95. Boissier O, Hübner JF and Sichman JS. Organization Oriented Programming: From Closed to Open Organizations. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, p.86-105.

96. Pynadath DV, Tambe M, Chauvat N and Cavedon L. Toward Team-Oriented Programming. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, p.233-47.

97. Boissier O. Multi-agent oriented programming and intelligent environments. *2013 19th International Conference on Control Systems and Computer Science*. IEEE, 2013, p. 457-64.

98. Ricci A, Piunti M and Viroli M. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*. 2011; 23: 158-92.

99. Fikes RE and Nilsson NJ. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*. 1971; 2: 189-208.

100. Bylander T. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*. 1994; 69: 165-204.

101. Allen J, Hendler JA and Tate A. *Readings in planning*. Morgan Kaufmann Pub, 1990.

102. Newton MAH, Levine J, Fox M and Long D. Learning Macro-Actions for Arbitrary Planners and Domains. *ICAPS*. 2007, p.256-63.

103. Botea A, Enzenberger M, Müller M and Schaeffer J. Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research*. 2005; 24: 581-621.

104. Galuszka A and Swierniak A. Translation STRIPS Planning in Multi-robot Environment to Linear Programming. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, p.768-73.

105. Gałuszka A and Świerniak A. Planning in multi-agent environment as inverted STRIPS planning in the presence of uncertainty. *Recent Advances In Computers, Computing and Communications (N Mastorakis and V Maldenov, Eds)—Athens: WSEAS Press*. 2002: 58-63.

106. Hunter A and Delgrande JP. An Action Description Language for Iterated Belief Change. *IJCAI*. 2007, p.2498-503.

107. McDermott D, Ghallab M, Howe A, et al. PDDL-the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

108. Dolejsi J. vscode-pddl-samples. Online: Github, 2020.

109. Pellier D and Fiorino H. PDDL4J: a planning domain description library for java. *Journal of Experimental & Theoretical Artificial Intelligence*. 2017: 1-34.

110. Kovács DL. A multi-agent extension of PDDL3.1. 2012.

111. Kovacs DL. Complete BNF definition of MA-PDDL with privacy. Czech Technical University.

112. Redjaimia A. Multi-Agent System and Events Plan Construction Using PDDL. *Journal of Computer Science & Systems Biology*. 2015; 8: 333.

113. Tonidandel F, Vaquero TS and Silva JR. Reading PDDL, Writing an Object-Oriented Model. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, p.532-41.

114. Vaquero TS, Tonidandel F and Silva JR. The itSIMPLE tool for modeling planning domains. *Proceedings of the First International Competition on Knowledge Engineering for AI Planning, Monterey, Califormia, USA*. 2005.

115. Bordini RH, Hübner JF and Wooldridge M. *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

116. Ricci A, Piunti M, Viroli M and Omicini A. Environment Programming in CArtAgO. In: El Fallah Seghrouchni A, Dix J, Dastani M and Bordini RH, (eds.). *Multi-Agent Programming: Languages, Tools and Applications*. Boston, MA: Springer US, 2009, p.259-88.

117. Hübner JF, Sichman JS and Boissier O. Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*. 2007; 1: 370-95.

118. Toledo CM, Bordini RH, Chiotti O and Galli MR. Developing a Knowledge Management Multi-Agent System Using JaCaMo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p.41-57.

119. Martins R and Meneguzzi F. A smart home model using JaCaMo framework. *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*. 2014, p.94-9.

120. Sorici A, Boissier O, Picard G and Santi A. Exploiting the JaCaMo framework for realising an adaptive room governance application. *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11*. Portland, Oregon, USA: Association for Computing Machinery, 2011, p.239–42.

121. Weyns D, Omicini A and Odell J. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*. 2007; 14: 5-30.

122. Hubner JF, Bordini RH, Gouveia GP, Pereira RH, Piunti M and Sichman JS. Using jason, moise+, and cartago to develop a team of cowboys. *Proceedings of the 10th International Workshop on Computational Logic in Multi-Agent Systems*. 2009.

123. Shrestha M, Garcia JE, Bukovac Z, Dorin A and Dyer AG. Pollination in a new climate: Assessing the potential influence of flower temperature variation on insect pollinator behaviour. *PLoS ONE*. 2018; 13: e0200549.

124. Shrestha M, Dyer AG, Boyd-Gerny S, Wong BBM and Burd M. Shades of red: bird-pollinated flowers target the specific colour discrimination abilities of avian vision. *New Phytologist*. 2013; 198: 301-10.

125. Dixon KW. Pollination and Restoration. *Science*. 2009; 325: 571.

126. Hegland SJ, Nielsen A, Lázaro A, Bjerknes A-L and Totland Ø. How does climate warming affect plant-pollinator interactions? *Ecology Letters*. 2009; 12: 184-95.

127. Ljungblad S. The rise of social drones: A constructive design research agenda. Online: Chalmers University of Technology, 2020.

128. Baytas MA, Çay D, Zhang Y, Obaid M, Yantaç AE and Fjeld M. The Design of Social Drones: A Review of Studies on Autonomous Flyers in Inhabited Environments. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Glasgow, Scotland, UK: Association for Computing Machinery, 2019, p. Paper 250.

129. Schaft Pvd. Pollination Drones Seen as Assistants for Ailing Bees. Online: Robotics Business Review, 2018.

130. Boukas E and Rosenberg M. Intelligent drones must inspect the safety of Danish ships. Online: Denmark Technical University, 2019.

131. Lilleyman A, Melens C, Perella B, Stewart A, Watson L and Wright A. Arthrobot: the design and analysis of an insect-flying drone. *Department of Mechanical and Aerospace Engineering*. University of Strathclyde, 2019.

132. Landgraf T, Bierbach D, Kirbach A, et al. Dancing honey bee robot elicits dance-following and recruits foragers. *arXiv preprint arXiv:180307126*. 2018.

133. Kominis F and Geffner H. Beliefs in multiagent planning: From one agent to many. *Twenty-Fifth International Conference on Automated Planning and Scheduling*. 2015.

134. Lovie-Kitchin JE and Woo GC. Effect of magnification and field of view on reading speed using a CCTV. *Ophthalmic and Physiological Optics*. 1988; 8: 139-45.

135. Milani A, Niyogi R and Biondi G. Neural Network Based Approach for Learning Planning Action Models. Cham: Springer International Publishing, 2019, p.526-37.

136. McEachran R. Going beyond 'eye in the sky'. *Professional Engineering*. nn: IMechE, 2020, p.53.

137. Loveless C. Drones in Construction. *Available at SSRN 3111338*. 2017.

138. Ashour R, Taha T, Mohamed F, et al. Site inspection drone: A solution for inspecting and regulating construction sites. *2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2016, p.1-4.

139. Dragolea N. 9 Drones That Will Revolutionise Healthcare. Doctorpreneurs.com, 2020.

140. Weckler A. Lockdown survival: medicine delivery trial by drone greenlighted for Offaly town. Online: Independent.ie, 2020.

141. Rosser J, Vudatha V, Terwilliger B and Parker B. Surgical and Medical Applications of Drones: A Comprehensive Review. *JSLS : Journal of the Society of Laparoendoscopic Surgeons*. 2018; 22: e2018.00018.

142. Ghosh P. Why do we hate wasps and love bees? *BBC News*. BBC News, 2018.

143. Cheatham B, Javanmardian K and Samandari H. Confronting the risks of artificial intelligence. Online: McKinsey, 2019.