The Impact of Behavioural Diversity in the Evolution of Multi-Agent Systems Robust to Dynamic Environments

by

Scott Hallauer

A minor dissertation presented in partial fulfilment of the requirements
for the Master of Science degree in Computer Science

Supervisors

A/Prof. Geoff Nitschke
Prof. Emma Hart

July 2023

Author's Declaration

I know the meaning of plagiarism and declare that all of the work in this document, save for that which is properly acknowledged, is my own.

Acknowledgements

I would like to thank my supervisor, Geoff Nitschke, for his invaluable guidance and support throughout the conception, investigation, compilation and revision of this research project. His timeous response to queries, feedback on drafts and assistance with difficulties contributed significantly to the successful completion of this thesis.

My co-supervisor, Emma Hart, is also owed a great debt of gratitude for her primary role in enabling my research visit to Edinburgh Napier University during October and November 2022. This trip fulfilled one of my big dreams for international study and solo travel experience. I learned so much during a fascinating swarm robotics workshop in York, met so many wonderful friends in the Napier student societies, explored myriad parts of the incredible city of Edinburgh and discovered greater depths to myself through all of it.

I would also like to thank my parents for their endless support, both emotionally and financially, throughout my university journey. My appreciation extends to my sister and the inspiration she provides by following and excelling at her passions. There is no doubt that I would not have reached this point without the love and light of my precious family.

Furthermore, I want to thank my roommate, Emma Marriott, for simply being the best person to live with over this final stretch of my studies. Her special companionship, impeccable sense of humour and unsurpassed cooking abilities have warmed my soul and brought countless smiles to my face.

My heart also goes out to Lesanne Brooke and the radiant energy she has cast on my life. Her curious and creative mind has always encouraged me to think more broadly and find connections in everything.

I would also like to express my gratitude to Anna Brooke for all the unforgettable adventures we have shared, for all the lessons she has taught me and for pushing me to become the person I am today. You will always be my first star.

There are so many other people who have touched my life and contributed to my growth, happiness and well-being over the past few years. Although there is not enough space to properly mention everyone here, I would like to especially extend my recognition and heartfelt appreciation to Benjamin Botha, Natalie Gelderblom, Sam Cohen, Sasha Czech, Michael Scott, Marius Lang, David Newton, Callum Barry, Jess Bourn, Josh Selfe, Liam Carew, Rhys Willmore, Allison Brennan, Kayla Welsh, Rory Wightman, Emily Schneider, Sarina Lincoln, Sammy Pan, Shilpa Ramasar and Marco Donlic.

Finally, I must also acknowledge the *Royal Society* and the South African *National Research Foundation* for funding this work through grants, as well as the South African *Centre for High Performance Computing* for providing computational resources to this research project.

Dedication

For Tomi, Monty and all the good boys that came before.

Tomi

Monty

Abstract

Behavioural diversity has been shown to be beneficial in biological social systems, such as insect colonies and human societies, as well as artificial systems such as large-scale swarm robotics applications. Evolutionary swarm robotics is a popular experimental platform for demonstrating the emergence of various social phenomena and collective behaviour, including behavioural diversity and specialisation. However, from an automated design perspective, the evolutionary conditions necessary to synthesise optimal collective behaviours that function across increasingly complex environments remains unclear. Thus, we introduce a comparative study of behavioural diversity maintenance methods (based on the MAP-Elites algorithm) versus those without behavioural diversity mechanisms (based on the steady-state genetic algorithm), as a means to evolve suitable degrees of behavioural diversity over increasingly difficult collective behaviour tasks. For this purpose, a collective sheep-dog herding task is simulated which requires the evolved robots (dogs) to capture a dispersed flock of agents (sheep) in a target zone. Different methods for evolving both homogeneous and heterogeneous swarms are investigated, including a novel approach for optimising swarm allocations of pre-evolved, behaviourally diverse controllers. In support of previous work, experiment results demonstrate that behavioural diversity can be generated without specific speciation mechanisms or geographical isolation in the task environment. Furthermore, we exhibit significantly improved task performance for heterogeneous swarms generated by our novel allocation evolution approach, when compared with separate homogeneous swarms using identical controllers. The introduction of this multi-step method for evolving swarm-controller allocations represents the major contribution of this work.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AMHET	Allocate MAP-Elites Heterogeneous
ANN	Artificial Neural Network
ASHET	Allocate SSGA Heterogeneous
EA	Evolutionary Algorithm
EDQD	Embodied Distributed Quality Diversity
ER	Evolutionary Robotics
ES	Evolutionary Strategy
FOV	Field of View
GA	Genetic Algorithm
GGA	Generational Genetic Algorithm
GP	Genetic Programming
MAP-Elites	Multi-dimensional Archive of Phenotypic Elites
mEDEA	Minimal Environment-driven Distributed Evolutionary Adaptation
MHET	MAP-Elites Heterogeneous
MHOM	MAP-Elites Homogeneous
PFSM	Probabilistic Finite State Machine
SR	Swarm Robotics
SHET	SSGA Heterogeneous
SHOM	SSGA Homogeneous
SSGA	Steady-State Genetic Algorithm
QD	Quality Diversity
QED	Quality-Environment Diversity

Chapter 1

Introduction

The field of *swarm robotics* (SR) has progressed greatly since its inception approximately 30 years ago [11, 12]. Research in this area is primarily focused on the development of artificial, multi-agent systems [27] that can effectively collaborate to achieve a common goal. It has been defined as *"the study of how large numbers of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment"* [121]. Inspiration for these behaviour models traditionally come from natural collective systems, such as ant and other insect colonies.

Developing such systems requires careful consideration of the controller algorithms employed for individual robots to interact cohesively. *Artificial neural networks* (ANNs) are commonly used for this purpose, which attempt to emulate the functioning of biological brains [140]. A graph of nodes is structured in sequential layers whereby sensory information is passed into the first layer of input nodes and propagated through several so-called "hidden" layers, finally resulting in behaviour directives being passed out the final layer of output nodes. The transformation from input values to output values is chiefly governed by weighted connections between nodes. Rather than programming these manually, a popular approach is to use evolutionary techniques in a process known as "neuro-evolution". This is implemented by simulating a task environment where, through the process of natural selection over multiple repetitions, robots have their ANN connection weights adjusted (by mutation and crossover events) and gradually improve at achieving their assigned goals.

Successful swarm robotic systems should also be designed to cope with the unpredictable nature of real-world environments and to adapt accordingly. Recent research has highlighted the potential of evolving behaviourally diverse populations that demonstrate robustness [15, 61]. This approach is intuitive to understand in the context of biological populations, where groups of similar individuals are more susceptible to environmental pressures and catastrophic events than diverse groups of individuals. In the latter case, it is likely that subsets of a diverse population will display the ability to adapt to new circumstances even if the remainder of the population cannot. It has also been shown that diverse groups of individuals tend to perform better at solving problems and collective tasks than homogeneous groups [63, 66, 89]. As a result of these purported benefits,

research interest has grown surrounding different approaches for evolving behavioural (or functional) diversity in robot swarms.

Traditional evolutionary algorithms, however, do not explicitly promote the evolution of behavioural diversity. One example is the collection of "steady state" approaches, which was originally introduced by Whitley and Kauth with the GENITOR algorithm in 1988 [138]. The key factor with steady-state genetic algorithms (SSGAs) is that the population size remains constant over time, with fitter offspring individuals replacing others in the parent population. Another example is the minimal Environment-driven Distributed Evolutionary Adaptation (mEDEA) algorithm, which was introduced in the 2010 study by Bredeche and Montanier [19]. This is an embodied (or distributed) evolutionary algorithm [57] that functions based solely on local interactions between robots, as opposed to a centralised approach in which there is a global coordination of reproduction and evolution. The mEDEA algorithm aims to provide long-term adaptation to a robot population in the face of unpredictable environment changes and operates via genome broadcasting between nearby robots that, at the end of their lifetime, randomly select one of their received genomes, mutate it and use it for the next generation.

Since focus has shifted towards heterogeneous robot swarms in recent years, a number of evolutionary algorithms have been developed that optimise evolved populations for both high task performance and greater behavioural diversity. These are often referred to as "illumination algorithms" because of their ability to illuminate the search space defined by specific behavioural features of solutions [132]. Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) is a popular approach that underlies many of these algorithms. Introduced in 2015 by Mouret and Clune [97], MAP-Elites operates by storing solutions in an archive which is apportioned into bins along multiple feature dimensions. For example, designs of a robot body evolved for a walking task might placed in an archive based on the solution features of leg length and body weight. As evolution proceeds, novel solutions that fall into bins containing existing solutions are only stored if they demonstrate superior fitness, resulting in an archive of elite solutions being produced.

Some extensions to MAP-Elites include the Embodied Distributed Quality Diversity (EDQD) and Quality-Environment Diversity (QED) algorithms. EDQD was introduced in 2018 by Hart, Steyven and Paechter [61] and presents a hybrid approach incorporating both MAP-Elites and mEDEA. This involves the transmission of a local MAP-Elites archive between robots rather than just a single genome. QED was introduced in 2020 by Bossens and Tarapore [15] and makes use of feature dimensions that describe the solution's evaluation environment rather than the solution itself.

The simulation environments used in the evolution of SR systems are usually task-specific and can be either static or dynamic in nature. Static environments are those in which changes are only brought about through the direct manipulation by robots of objects in the arena. In other words, the absence of robots would result in an unchanging environment over time. For example, collective construction tasks [113] are often simulated in static environments where the position of objects (or building blocks) are only affected by robots directly moving them. Dynamic environments, on the other hand, are those in which changes occur independent of robot interaction. For example, a collective herding

task [86] is inherently dynamic due to the presence of non-robot agents that move freely through the environment, thereby changing the complexity of the task over time.

Our aim, in this thesis, is to investigate methods for generating behavioural diversity in a collective herding task with the end goal of creating a heterogeneous robot swarm. This is approached using extensions of the SSGA and MAP-Elites evolutionary algorithms. Both are implemented in a centralised manner (as opposed to distributed) with the key difference being that MAP-Elites explicitly promotes diversity, whereas SSGA does not. We compare the effectiveness of these algorithms from the two perspectives of either evolving behaviours for a homogeneous swarm (which could be combined in future to make a heterogeneous swarm) or directly evolving a heterogeneous swarm. We then investigate the potential of a novel, multi-step procedure in which the pre-evolved solutions for a homogeneous swarm are used for evolving an optimal allocation in a heterogeneous swarm.

1.1 Research Questions

This research focuses on the evolution of behavioural diversity for SR systems in dynamic task environments. Therefore, we are concerned with algorithms capable of evolving both high-performing robot controllers and effective allocations of these controllers to individuals within a behaviourally diverse swarm. We aim to investigate and compare the purported performance benefits of heterogeneous swarms with those of homogeneous swarms, as well as different methods for generating heterogeneous swarms.

There are three primary approaches that are implemented and analysed for the above purposes. Firstly, we look at standard evolutionary algorithms (EAs), including SSGA and MAP-Elites, that produce a homogeneous swarm by evolving the same controller for each robot (Method 1). Secondly, we look at variations of these EAs that produce a heterogeneous swarm by simultaneously evolving different controllers (as part of a single swarm genome) for each robot (Method 2). Finally, we look at a novel approach that produces a heterogeneous swarm by evolving an allocation of behaviourally diverse controllers (generated by Method 1) for each robot (Method 3).

These methods are each divided into sub-methods based on the underlying EA used for controller evolution, where the first sub-method uses SSGA and the second sub-method uses MAP-Elites. In this report, both Method 1 and Method 2 will be referred to as the *behaviour evolution* experiments (since the robot controllers themselves are being evolved) and Method 3 will be referred to as the *allocation evolution* experiments (since the allocation of robot controllers is being evolved).

The specific questions that this research seeks to answer are therefore:

1. Does the use of MAP-Elites for the evolution of a *homogeneous* swarm (Method 1.2) result in higher task performance compared with SSGA (Method 1.1)?

2. Does the use of MAP-Elites for the evolution of a *heterogeneous* swarm (Method 2.2 and Method 3.2) result in higher task performance compared with SSGA (Method 2.1 and Method 3.1)?

3. Does a *heterogeneous* swarm (Method 2 and Method 3) elicit any advantage over standard *homogeneous* swarm-behaviour evolution (Method 1) for a collective herding task, given increasing task complexity?

4. When evolving a *heterogeneous* swarm for a collective herding task with increasing task complexity, does a multi-step evolutionary process that optimises the allocation of pre-evolved controllers to robots (Method 3) produce higher-performing swarms than the single-step direct evolution of a diverse swarm (Method 2)?

In answering these questions, several standard metrics are measured to evaluate swarm performance and behavioural diversity. These include archive size, maximum fitness and quality diversity score. Details of each metric are provided in Section 3.4.

1.2 Motivation

These questions were motivated by previous results and research in the literature, as described in Chapter 2. Specifically, we aim to extend on the results presented in the 2018 study by Hart, Steyven and Paechter [61].

In ecology, there are hypotheses that suggest behavioural diversity amongst individual organisms is positively related to the robustness of the overall population to changes in the environment [26, 134]. It is proposed that this same principle can be applied to the design of SR systems, thereby providing similar benefits to population performance and adaptability [61]. This research seeks to investigate and yield further evidence for this theory.

Moreover, a collective herding task has been selected as it represents a uniquely dynamic interaction between swarm, task and environment. This task not only encourages robots to adapt to a changing environment with moving entities, but also to an unpredictable task with differing solutions. Since the herd's movement is randomly generated for each simulation, the task environment promotes the evolution of robots equipped with more general herding approaches rather than specialised robots attuned to a few specific herd configurations. Additionally, this task environment allows us to investigate the general applicability of previous hypotheses in different environments.

The two underlying evolutionary algorithms, SSGA and MAP-Elites, have been chosen for this investigation since they both represent popular, elementary algorithms. SSGA is a standard approach that maintains a regularly-sized population throughout the evolutionary process but only seeks to optimise task performance rather than behavioural diversity. MAP-Elites, on the other hand, is a recently introduced approach that seeks to optimise both task performance and behavioural diversity in a variable-sized population during evolution.

Finally, the implementation of our novel approach for generating heterogeneous swarms via the evolution of behaviour allocations offers the opportunity to compare performance with previous approaches from the literature, such as Embodied Distributed Quality Diversity (EDQD) [61] and Quality-Environment Diversity (QED) [15].

1.3 Contributions

In addressing the questions stated above, this research provides the following contributions:

1. Replication of previous research investigating swarm behavioural diversity in a different, dynamic environment for a collective herding task.
2. A novel evolutionary approach based on the SSGA and MAP-Elites algorithms for efficiently generating heterogeneous swarms via the allocation of pre-evolved, behaviourally diverse robot controllers.
3. An extended simulation framework, which enables the simultaneous control of multiple agent types as well as parallelised performance evaluation.

1.4 Overview

This thesis is divided into five chapters: Related Work (Chapter 2), Methods (Chapter 3), Experiments (Chapter 4), Results and Discussion (Chapter 5) and Conclusion (Chapter 6).

Chapter 2 reviews related work in the fields of swarm robotics and neuro-evolution covering, inter alia, collective behaviour, artificial neural networks, evolutionary algorithms, behavioural diversity and population adaptability.

Chapter 3 describes the various aspects of the methodology implemented in this research. This includes the simulation framework, task environment, agent types, controller design, evolutionary algorithms and solution evaluation.

Chapter 4 outlines the experimental design used to evaluate the various evolutionary strategies being investigated in this research. The simulator configuration and parameter tuning for these experiments are also presented here.

Chapter 5 presents the experimental results and discusses performance differences between alternative methods. There is a specific focus on reviewing and addressing our research questions, as described in Section 1.1.

Finally, Chapter 6 distils the key conclusions from the discussion, summarises the contributions of this research and provides suggestions for future work.

Chapter 2

Related Work

This chapter reviews the literature covering the fields of collective robotics and neuro-evolution, as well as work investigating the incorporation of behavioural diversity into the evolution of adaptable populations. Firstly, a high-level explanation is given for how collective robotics systems are simulated and evolved. Thereafter, an in-depth analysis of the strategy and process of neuro-evolution is provided. This includes an introduction to the concepts of artificial neural networks and evolutionary algorithms. Finally, the concept of population adaptability in changing environments is outlined and previous work on state-of-the-art algorithms for evolving behavioural diversity is reviewed.

2.1 Swarm Robotics

Swarm robotics (SR), a sub-field of *collective robotics*, is the problem-solving approach whereby a homogeneous, multi-robot system accomplishes specific task goals through collaborative, *collective behaviour* [80]. In this context, the term "homogeneous" is referring to groups of robots with highly similar, if not identical, neural controllers and/or body plans. There are many real-world applications for SR systems, such as hazardous waste clean-up and oceanic environmental monitoring [121]. This section reviews previous work in the SR field and explores the connection with *evolutionary robotics*.

2.1.1 Collective Behaviour

The 1990s saw a surge of interest in the application of social insect group behaviour patterns to the field of robotics [36, 79, 80, 90]. Similar to how ant colonies consisting of many simple individuals can achieve tasks far beyond the capabilities of any single ant, it was envisaged that simple robot controllers could collectively produce complex behaviour. The successful implementation of such a SR system requires a form of decentralised control similar to that employed by social insects [80]. This requires robot controllers to be designed in such a way that the local interactions produced by their behaviour result in global coordination [18]. Modelling such control architectures is not an elementary computational task and so the

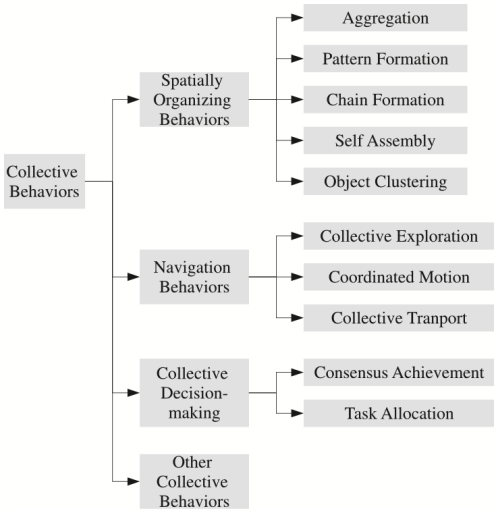

Figure 2.1: Taxonomy for collective behaviours [101]. Diagram representing classes of collective behaviours relevant to swarm robotics applications, as proposed by Brambilla *et al.* [17].

use of *artificial neural networks* and *evolutionary algorithms* (elucidated in Section 2.2), both inspired by biological systems, has been adopted as a standard approach [80, 90, 131].

Several classes of collective behaviour have been previously defined, including spatially organising behaviours, navigation behaviours and collective decision-making (see Figure 2.1). The following subsections review each of these behaviour classes and the related work in the literature.

2.1.1.1 Spatially Organising Behaviours

This class of behaviours is related to tasks in which robots and objects are arranged and distributed in space. Such behaviours include aggregation, pattern formation, chain formation, self-assembly and object clustering.

Aggregation represents a simple form of collective behaviour whereby robots gather in a specific region of the environment, allowing for nearby interaction. Implementation is usually achieved through the use of either probabilistic finite state machines (PFSMs) or artificial evolution [17]. A popular probabilistic approach is based on the aggregation dynamics of cockroach populations, as originally proposed in 2007 [33]. Similar approaches using slightly altered PFSMs have also been successfully demonstrated [10, 123]. Alternatively, evolutionary methods have been applied to evolving suitable robot controllers for aggregation tasks [50, 130].

Pattern formation behaviours involve positioning robots in relation to each other at regular and repetitive intervals [17]. Usually, these robot models are developed using virtual physics-based design. One of the first studies implementing such a design was published in 2004, where robots used virtual forces to form a hexagonal lattice [127].

Chain formation is a class of behaviours in which robots position themselves to connect target points. The most common approaches used in this category are PFSMs [40, 107], virtual physics-based design [91] and artificial evolution [128].

Self assembly is a unique form of collective behaviour where robots physically connect to each other. There are two primary, associated challenges – morphogenesis (how to form the target structure) and control (how to command the target structure) – that are approached differently [17]. The former is usually addressed using PFSMs [109] with the latter using either PFSMs or artificial evolution [110].

Object clustering includes behaviours in which robots move objects in the environment. This can result in either the gathering (non-connected) or assembling (connected) of those objects [17]. The most common approach is to utilise PFSMs, such as in the seminal work by Beckers *el al.* [13].

2.1.1.2 Navigation Behaviours

In these types of collective behaviours, the coordinated movement of robot swarms is developed and simulated. Included behaviours are collective exploration, coordinated motion and collective transport.

Collective exploration includes behaviours in which robots cooperate to explore and navigate their surrounding environment. To achieve these goals, the collective behaviours of *area coverage* and *swarm-guided navigation* are often used. Area coverage aims to distribute robots throughout an environment forming a grid of communicating robots, whereas swarm-guided navigation aims to have robots direct the movement of other robots. Virtual physics-based design is usually employed to address area coverage, while swarm-guided navigation focuses on communication protocols implemented with PFSMs [17]. One study introduced the concept of "virtual pheromones" where, through communication, robots created a gradient between source and destination that could be exploited for navigation [112].

Coordinated motion, also known as *flocking*, refers to collective behaviours in which robots harmoniously move together in a manner similar to that of birds or fish. These behaviour models are often based on virtual physics-based design, but artificial evolution has also been used successfully [17]. The first application of coordinated motion developed a virtual flock of birds and was published in 1987 under the computer graphics domain [119]. In this paper, the concept of a "bird-oid" object (or "boid") was introduced which has since come to refer to the popular flocking algorithm employed.

Collective transport tasks involve robots collaborating to move objects that are too large or heavy to be handled by individual robots. These behaviours are obtained through the implementation of PFSMs or artificial evolution [17]. One of the first studies into distributed, collective transport was published in 1997 and proposed behaviours based on force sensing, position sensing and orientation sensing [38].

2.1.1.3 Collective Decision-Making

These types of collective behaviour cope with the problem of how robots should influence other robots when deciding between different options. Such behaviours include consensus achievement and task allocation.

Consensus achievement is associated with achieving agreement on decisions between all robots in a swarm. There are two subcategories of implementation based on the type of communication model employed, namely direct and indirect communication [17]. A 2010 study used direct communication in a swarm of foraging robots to choose between two foraging zones [59]. Indirect communication was used in a 2009 study where a swarm of cockroach-like robots sought consensus agreement [55].

Task allocation behaviours attempt to optimise performance by delegating available tasks between robots in a swarm. The primary approach used here is PFSMs, where probabilities for selecting different tasks are altered amongst robots [17]. A 2000 study, representing one of the first works in task allocation, implemented a simple threshold-based model governing how robots would either leave a nest to collect prey or remain in the nest [78].

2.1.1.4 Other Collective Behaviours

Besides these major classes of collective behaviour described above, there are many works in swarm robotics that do not fall neatly into this classification system. These include behaviour types such as collective shepherding [86], collective fault detection [30], group size regulation [94] and human-swarm interaction [93].

Collective shepherding, specifically, represents a form of dynamic task environment for SR applications not covered by the previous classification. Unlike most of the behaviours previously described in this section, collective shepherding (or simply *collective herding*) involves interactions with other autonomous agents in the environment rather than just static objects. In 2020, Long *et al.* [86] compiled a comprehensive review of prior research in swarm shepherding behaviour.

2.1.2 Environment Simulation

There are a variety of software packages available for simulating swarm robotics systems, which have developed in terms of increasing complexity and realism over the years. These broadly fall into the two categories of 2-dimensional (2D) and 3-dimensional (3D) simulators.

2.1.2.1 2D Simulation

SimbotCity represents one of the early robot population simulators, developed by Kube and Zhang for their related work in the 1990s [79, 80]. In their implementation, a modified fixed-priority *subsumption* architecture [21] was used for managing behaviour arbitration.

Many modern alternatives have been developed in recent years. ARGoS was introduced in 2011 with the primary objective of providing a simulation environment for large heterogeneous robot swarms [114]. In 2013, a C++ based simulator named Roborobo! was released which specialises in evolutionary swarm robotics [20]. In 2015, Kilombo was announced as virtual simulator for the popular Kilobot robot [120] (traditionally used in physical simulations) which greatly expedites development and allows for pre-screening of potential controller algorithms [72]. Even more recently, in 2019, a paper was published detailing a massive multi-agent simulation environment, SCRIMMAGE, for simulating collaborative robots [35].

2.1.2.2 3D Simulation

In support of the popular Khepera physical robots [125] (which are simple, circular "e-puck" robots), the Webots simulator was introduced in 1998 [95]. This project sought to improve an original 2D simulator for the Khepera platform by adding 3D rendering and extensibility for other robotic architectures.

Gazebo was then introduced in 2004 as one of the first freely-available 3D simulators specific to robotics applications [76]. A major intention behind its development was to fill a gap in simulators of the time by producing one capable of outdoor environment simulation. Despite representing a significant step forward in robotics simulation, the simulator possessed many limitations such as a lack of physics models for pliable surfaces (e.g. soil, sand and grass) which are prevalent in outdoor environments.

In 2006, an open source Java 3D robotics simulator named Simbad was introduced [68]. This simulator was designed to be simple and fast to adopt for projects in either research or education. It was specifically intended to be used in evolutionary robotics applications and was distributed with pre-installed packages for both neural network and artificial evolution implementations.

The distributed robotics simulator V-REP (Virtual Robot Experimentation Platform) was first released in 2010 [51]. A versatile, modular architecture was incorporated that maintains a high level of performance by only applying intensive physics models to the necessary parts of the system. The processes of control, actuation, sensing and monitoring are concurrently simulated, allowing for complex configurations of different sensors and/or actuators. Still under active development, this 3D simulator is widely used today and has since been renamed to CoppeliaSim.

2.1.3 Evolutionary Robotics

Evolutionary robotics (ER) is the field of research into the automatic design of autonomous robot controllers and morphologies through processes that mimic *natural evolution* [52]. The basic premise of natural evolution is that a population of genetically similar, yet unique, organisms compete in a common environment and, through the process of *natural selection*, the "fittest" individuals survive and reproduce [139]. In this manner, superior

traits are inherited over successive generations and the population as a whole tends to become better adapted to its environment.

2.1.3.1 Automated Robot Design

Traditionally, robotic controllers and morphologies are manually, and often laboriously, designed by human engineers [102]. This is an expensive endeavour that has hindered potential niche applications of robotics, which are not feasible without the economies of scale [85]. ER provides an alternative, cost-effective approach to robot design whereby variations are automatically assessed for functionality through simulation and iteratively improved over multiple cycles [42]. Using this method of *artificial evolution*, new models of cognition can be developed that, through experimental analysis, may prove better than existing hand-designed solutions [105]. Furthermore, many real-world environments are inaccessible (e.g. deep ocean, outer space and extraterrestrial planets) and necessitate fully automated robot design and evaluation approaches, where it is not feasible to have pre-engineered solutions [69, 103].

2.1.3.2 Fitness Functions

A key aspect of the artificial evolution process used in ER is the measurement of fitness for the developing robotic components. During each cycle, or *generation*, robots (consisting of the current state of the evolving morphology and/or controller) are tasked with engaging in an evaluation period. Each robot morphology and/or controller (whichever is being evolved) is then assessed based on its performance using a *fitness function*. As the final step in each cycle, an *evolutionary algorithm* (see Section 2.2.2) is applied that uses the results from the fitness function to selectively propagate the fittest robots for the next generation [102].

There are seven broad classes of fitness functions as defined by Nelson, Barlow and Doitsidis [102]. This classification system is based on the level of *a priori* knowledge incorporated by the fitness functions (see Figure 2.2). Firstly, there are *training data fitness functions* which use datasets for comparing behaviour and scoring fitness. These are commonly used in mimetic learning, where a robot attempts to replicate the behaviour of a trainer [37]. Secondly, there are *behavioural fitness functions* which assess fitness based on how a robot goes about performing its tasks rather than what it ultimately achieves [8]. In contrast, there are also *aggregate fitness functions* which select based only on high-level ability to accomplish a task, without consideration of the behaviour involved [67]. Combining characteristics of both behavioural and aggregate fitness functions, there are *tailored fitness functions* which contain behaviour-measuring terms and aggregate terms in the fitness calculation [117]. Then there are *functional incremental fitness functions* which are used to iteratively select for more complex abilities by progressively adapting the fitness function over the course of robot evolution [82]. Similarly, there are *environmental incremental fitness functions* which gradually increase the difficulty of the environment in which the evolving robots interact [100]. Finally, there is *competitive* and *co-competitive*

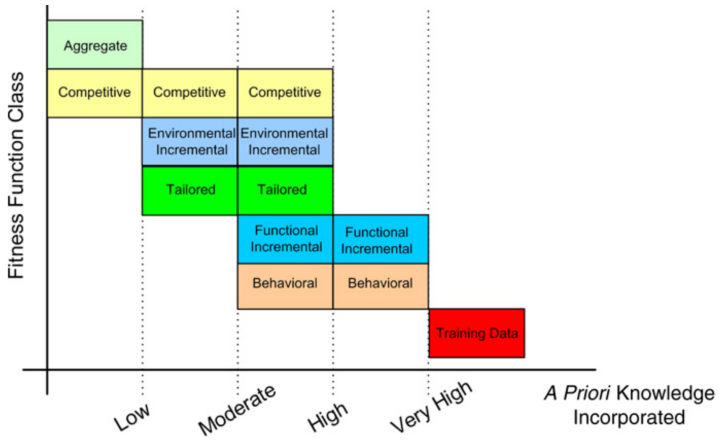

Figure 2.2: Taxonomy for fitness functions [102]. Chart relating classes of fitness function to levels of incorporated *a priori* knowledge.

fitness selection which involves direct intra-population competition between individuals, where interactions influence other robots' behaviours and their resulting fitness evaluation [23].

2.1.3.3 Evolving Collective Behaviour

ER enables the development of robot controllers that facilitate the self-organising, collective behaviour patterns required in SR systems. This is possible by simulating environments that require collaboration between individuals for successful task completion and the subsequent selection and propagation of effective controllers.

Baldassarre, Nolfi and Parisi [7] demonstrated that simulated robots evolved for the ability to move together toward a light target were able to display collective behaviours with interesting properties. Forms of *situated specialisation* were observed where robots with identical controllers expressed varying behavioural roles dependent on their specific circumstances in the group.

Similar forms of specialised behaviour (albeit in a heterogeneous population) were specifically selected for in research by Nitschke, Schut and Eiben [104] where the performance of three cooperative co-evolution methods was compared. In this study, robot controllers were evolved for a simulated collective construction task. The results indicated that the method of *collective neuro-evolution* (CONE) outperformed the other two tested methods: *multi-agent enforced sub-populations* (MESP) and *cooperative co-evolutionary algorithm* (CCGA).

Distinguishing it from the majority of studies in the field, research in 2002 by Quinn *et al.* [117] explored the evolution of controllers for real robots. Here, robots, minimally equipped with infrared sensors, were evolved for a formation movement task. It was observed that a team of robots successfully evolved to adopt individual, specialised roles in order to complete the assigned task.

More recent work by Duarte *et al.* [39] similarly investigated the evolution of collaborative control systems for a physical swarm of robots. The neural controllers were initially evolved, in simulation, for various swarm robotics tasks (e.g. homing, dispersion, clustering and monitoring) before being transferred to real aquatic surface robots. Results demonstrated that the controllers successfully achieved similar task performance in a real-world, uncontrolled environment as they did in simulation.

2.2 Neuro-Evolution

Neuro-evolution is a broad term encompassing an array of approaches to training *artificial neural networks* (ANNs) for specialised behaviours, all of which draw on insights from neuroscience and evolutionary biology [45]. ANNs are used in many different problem spaces including pattern recognition, prediction, optimisation and associative memory, as well as robot control [70, 131]. This section introduces the concept of an ANN, explains their applicability in *evolutionary algorithms* (EAs) and describes the method of embodied evolution in the field of ER.

2.2.1 Artificial Neural Networks

ANNs are structured as weighted directed graphs with nodes representing artificial neurons and edges representing the connections between neuron inputs and outputs [70]. An example of the basic composition of an ANN is presented in Figure 2.3, where environmental information flows from *input nodes* (or units) to *output nodes* via weighted connections which may pass through *hidden nodes* within the network. The original inspiration for this architecture, as evidenced by the name, can be traced back to the fundamental structure and operation of biological nervous systems [92]. This alternative approach to computation makes ANNs far more effective in solving certain problems, such as pattern recognition, which traditional computational architectures struggle to handle efficiently.

2.2.1.1 Classification

There are two major types of ANNs: *feed-forward* and *recurrent* networks [70]. In feed-forward networks, the graph is acyclic, and data only moves from input nodes to output nodes. Alternatively, in recurrent (or *feedback*) networks, cycles occur in the graph whereby information from previous activations can feed back into the network and alter the inputs to each node. As a result of this operation, feed-forward networks are considered to be *static* (i.e. only produce a single set of output values for a given input value) whereas recurrent networks are considered to be *dynamic* (i.e. produce a sequence of output values for a given input value).

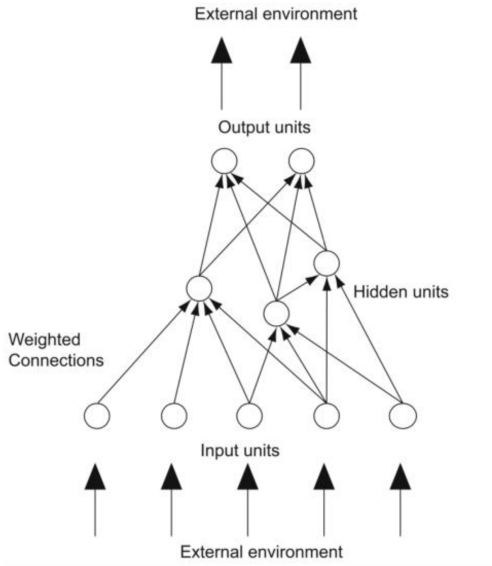

Figure 2.3: Generic neural network architecture [45]. Input units and output units are connected to the external environment and hidden units which connect to other neurons but are not directly connected to the environment.

2.2.1.2 Encoding

ANN architectures can be represented using either *direct* or *indirect* encoding schemes [74, 141]. In the former approach, all the structural information (i.e. every node and connection) of the ANN is encoded in an unabridged form. This representation results in a one-to-one mapping from the *genotype* (or encoding) to the *phenotype* (or architecture). At the other extreme, with indirect encoding, only the most pertinent aspects of the network (e.g. the number of hidden layers) are encoded such that a variation of the phenotype can be derived from the genotype [141].

2.2.1.3 Training Methods

Training ANNs involves iteratively updating connection weights to achieve greater task performance. The ability of ANNs to automatically learn through experience of examples, rather than following prescriptive rules, is one of the major advantages over conventional expert systems [70].

Traditionally, ANNs are trained using methods such as *supervised* and *reinforcement* learning. Supervised learning techniques, such as *backpropagation* [5], make use of labelled data sets (i.e. correct input-output pairs) to incrementally revise connection weights in the network so as to match the mapping between inputs and expected outputs. Reinforcement learning techniques, on the other hand, do not use labelled training data and are commonly implemented in situations where such data is difficult or impossible to compile. In the standard model, an agent (or robot) is placed in an environment where it can both sense its surroundings and control the actions of its body. The agent is then incentivised, through

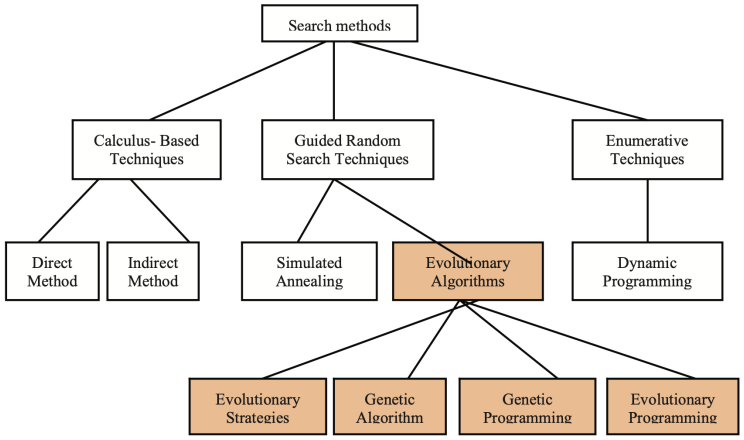

Figure 2.4: Taxonomy for evolutionary algorithms [133]. Diagram representing the classes of evolutionary algorithm as well as their placement in the broader field of search methods.

a system of reward and punishment, to favour behaviour that achieves specific goal-oriented objectives [73].

An alternative approach, known as *unsupervised* learning, has also proven successful in training ANNs. Unsupervised learning techniques provide none of the exemplar solutions or reward and punishment feedback used in supervised and reinforcement learning, respectively [9]. Instead, simply the correlations between input data are considered. Neuro-evolution algorithms (see Section 2.2.2) typically fall under this category.

2.2.2 Evolutionary Algorithms

Evolutionary algorithms (EAs) represent a class of probabilistic optimisation algorithms which are based on the model of natural evolution [6]. These entail a collective learning process whereby individuals in a population, representing potential solutions to a problem, undergo repetitive cycles of *selection*, *mutation* and *recombination* converging on an optimal solution [6]. When EAs are applied to training ANNs (especially for complex control tasks), the process is referred to as neuro-evolution.

There are four main streams of EAs (as shown in Figure 2.4) which are differentiated based on the data structures used in the encoding of candidate solutions: *genetic algorithms*, *genetic programming*, *evolution strategies* and *evolutionary programming* [6, 41, 133]. The following subsections review each of these categories and their related works.

2.2.2.1 Genetic Algorithms

The concept of genetic algorithms (GAs) was first introduced by Holland in the early 1960s [65] and remains the most popular type of EA in use today [60, 133]. Sexual recombination acts as the primary operator for this method, with mutation as a secondary operator [31].

GAs use strings over a finite alphabet (traditionally binary [141]) to represent the genotype of solutions. Applications have been explored in the areas of machine learning [58], pattern recognition [3, 87] and optimisation problems [81, 108].

2.2.2.2 Genetic Programming

The genetic programming (GP) approach was originally introduced by Koza in 1992 [77], using a genetic algorithm with a tree-based encoding [31]. It presented a solution to the problem of how to use an artificial intelligence system to automatically generate a desired computer program [31]. Unlike the binary encoding employed by GAs, GP uses programs or instruction sets as attributes [133]. This method has been applied to arithmetic operations [99], boolean operations [96] and recursive functions [1].

2.2.2.3 Evolutionary Strategies

The class of evolutionary strategies (ESs) was first developed by Rechenberg during his PhD studies in 1973 [118]. This was proposed as an optimisation method for difficult problems in hydrodynamics [31]. In this approach, solutions are directly represented using vectors of real numbers and no intermediary encoding is used as with GAs and GP [133]. Relevant application areas include networking, biochemistry and optics [31].

2.2.2.4 Evolutionary Programming

Evolutionary programming was introduced by Fogel in the 1960s [46] and bears close resemblance to ESs. This method attempts to consider intelligence in the light of being an adaptive behaviour [31]. Similar to ESs, no particular encoding is used and representation is selected based on the appropriate format for decision variables [31]. Some well-known applications of evolutionary programming include forecasting [47], games [25] and automatic control.

2.2.3 Embodied Evolution

The approach of *embodied evolution* (see Figure 2.5) was first introduced in 1999 [43]. This method is based on a probabilistic version of Harvey's microbial GA [62]. It sought to address the need for ER systems to autonomously evolve while simultaneously performing tasks in the environment. The proposed method is defined as *"evolution taking place within a population of real robots where evaluation, selection and reproduction are carried out by and between the robots in a distributed, asynchronous and autonomous manner"* [43].

Therefore, since there is no centralised locus of control in such an algorithm, the fundamental evolutionary operations of evaluation, selection and reproduction must be executed locally by each robot. Performing *evaluation* locally necessitates programming some metric of fitness measurement into the robots. *Selection* involves differentiating between the more

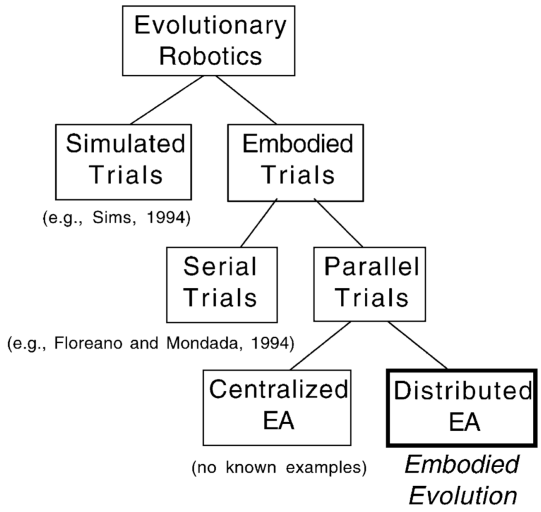

Figure 2.5: Embodied evolution as an evolutionary robotics methodology [135]. Tree diagram illustrating the relative classification of embodied evolution alongside other methods.

and less fit individuals in the population and ensuring that the more fit robots propagate genes while the less fit have their genes replaced. Finally, *reproduction* usually entails using the previously selected, less fit robots to be used as bodies for new offspring.

2.3 Behavioural Diversity

Recently, focus has intensified in the investigation of the effect of behavioural (or functional) diversity on robot task performance. There are practical benefits to the development of functionally diverse robot swarms, including increased robustness to environmental changes [61]. In this section, we review the implications of changing environments, the relevance of behavioural diversity to population adaptability and, finally, some evolutionary approaches for developing performant and diverse robotic populations.

2.3.1 Changing Environments

Most research in the field of evolutionary robotics to date has been conducted in static and predictable simulated environments. Real-world environments, however, are constantly undergoing gradual changes and occasionally experience sudden periods of major disruption. In natural ecosystems, such phases of rapid change are often accompanied by the extinction of some species and the adaptation of others [24]. This represents a fundamental challenge for the field of swarm robotics, in which multi-agent systems must be developed for prolonged use in dynamic, unpredictable environments. Future swarm robotics applications envision sending robot swarms into remote, hazardous environments for tasks such

as nuclear waste removal [2, 14]. These swarms must be able to continuously adapt their behaviour to survive and perform tasks successfully.

In the field of machine learning, overfitting is an established problem whereby candidate solutions perform effectively on training data but fail to generalise well to unseen examples [136]. Overfitting has been demonstrated in reinforcement learning applications, such as maze-finding tasks where trained agents perform poorly when placed in new environments [126]. Furthermore, the concept of "the reality gap" (or the "Sim2Real" problem [64]) has been coined in evolutionary robotics to refer to systems that work appropriately in simulation but not in the real world [71]. This problem relates to the trade-offs that have to be made between efficiency in virtual simulations and accuracy in physical trials. In this regard, solutions which are adaptive to varying operating conditions in simulation (i.e. are not overfitted) are also more likely to perform well in the real world. Therefore, evolutionary approaches that optimise for population adaptability in changing environments (as opposed to solely static task performance) are needed.

2.3.2 Population Adaptability

To combat the negative consequences of changing environments, natural populations have developed mechanisms to adapt to new circumstances. Natural selection is the elementary process whereby species evolve over time to become better suited to their environments [139]. While evolutionary robotics attempts incorporate this theory, most approaches have distinct training phases (where the robots are evolved) and application phases (where the evolved robot controllers remain fixed and are indefinitely applied to a task). This shortcoming has driven the development of new evolutionary methods that adopt a lifelong learning process of continual adaptation, even after the initial training phase [122].

Behavioural (or functional) diversity in natural populations has also been shown to improve both their robustness and overall performance. For example, functional diversity in bee populations has been demonstrated to increase pollination rates [63]. Cultural diversity amongst humans in the workplace is related to better problem solving with different ways of approaching the same tasks [89]. A more general study using agent-based modelling also found that a diverse team of problem solvers is more likely to outperform a team of high-ability problem solvers [66]. These results have led to recent research investigating the impacts on task performance of evolving robotic populations that demonstrate behavioural diversity.

2.3.3 Optimising Task Performance

Traditional evolutionary algorithms are designed to primarily optimise a solution's fitness or task performance. In these cases, the only selective pressure applied to an evolving population is the relative fitness scores of the individuals. Below, we review two such algorithms: one that is implemented in a centralised manner (SSGA) and another that is distributed (mEDEA).

Algorithm 1 Example of a SSGA [75]

procedure STEADY-STATE GA
  Randomly generate N solutions $s_i, i = 1, 2, ..., N$
  repeat
    Choose parents p_1 and p_2 from the population by proportional selection
    $(o_1, o_2) \leftarrow$ crossover(p_1, p_2)
    $o \leftarrow$ fitter of o_1 and o_2
    Replace an individual in the population with offspring o using preselection and Genitor-style replacement
  until the given number T of generations is reached
return the fittest solution found

Algorithm 2 Example of a GGA [75]

procedure GENERATIONAL GA
  Randomly generate N solutions $s_i, i = 1, 2, ..., N$
  repeat
    for each $k \leftarrow 1, 2, ..., N/2$ **do**
      Select parents s_{k_1} and s_{k_2} from the population using tournament selection with size t
      $(o_{2k-1}, o_{2k}) \leftarrow (s_{k_1}, s_{k_2})$
      $(o_{2k-1}, o_{2k}) \leftarrow$ crossover(s_{k_1}, s_{k_2}) with probability p_c
      $o_{2k-1} \leftarrow$ mutation(o_{2k-1}) with probability p_m
      $o_{2k} \leftarrow$ mutation(o_{2k}) with probability p_m
    Replace the population with offspring $o_1, o_2, ..., o_N$, keeping the fittest individuals
  until the given number T of generations is reached
return the fittest solution found

2.3.3.1 SSGA: Steady-State Genetic Algorithm

Evolutionary algorithms incorporating a "steady state" approach were first introduced in 1988 by Whitley and Kauth with the GENITOR algorithm [138]. In a Steady-State Genetic Algorithm (SSGA), the population being evaluated remains a constant size throughout evolution. Reproduction takes place one individual at a time, with the parents selected from the current population and the offspring added directly back into the population. Individuals are sorted by fitness and, for every new individual generated, the least fit individual is removed from the population [137].

It should be noted that SSGA is considered very similar to another approach known as the Generational Genetic Algorithm (GGA) [129]. Whereas SSGA is traditionally implemented so that two parent individuals are selected to produce one offspring individual in each iteration step, GGA rather replaces the entire population with new offspring individuals in each iteration step (referred to as a "generation") [124]. This difference in implementation is clarified in the pseudo-code examples for both SSGA (see Algorithm 1) and GGA (see Algorithm 2).

Algorithm 3 The mEDEA algorithm [19]

```
genome.randomInitialize()
while forever do
    if genome.notEmpty() then
        agent.load(genome)
    for iteration = 0 to lifetime do
        if agent.energy > 0 and genome.notEmpty() then
            agent.move()
            broadcast(genome)
    genome.empty()
    if genomeList.size > 0 then
        genome = applyVariation(select_random(genomeList))
    genomeList.empty()
```

In this thesis, the method that we refer to as SSGA is implemented in a manner more closely resembling the traditional GGA approach. However, since these terms are used interchangeably in recent literature and since the other algorithms that we investigate are also "generational" in nature, the term SSGA is used to avoid unnecessary confusion.

2.3.3.2 mEDEA: Minimal Environment-driven Distributed Evolutionary Adaptation

A 2010 study by Bredeche and Montanier [19] introduced the minimal Environment-driven Distributed Evolutionary Adaptation (mEDEA) algorithm. This is an embodied (or distributed) evolutionary algorithm that aims to provide long-term adaptation to a robot population in the face of unpredictable environment changes. The algorithm operates via genome broadcasting between nearby robots that, at the end of their lifetime, randomly select one of their received genomes, mutate it and use it for the next generation. Pseudocode for this algorithm is presented above (see Algorithm 3).

2.3.4 Optimising Behavioural Diversity

Although the high performance of solutions in isolation is a reasonable and important objective for evolutionary algorithms, it does not fully model the process of natural evolution whereby separate niches are simultaneously optimised and diversified [115]. As such, various evolutionary approaches have recently been adapted and developed that facilitate the evolution of behavioural diversity, in addition to optimising task performance. Novelty search is one such method that was first introduced to solely optimise for solution diversity, without consideration of task performance [83]. After showing promising results, further techniques were developed that combine both performance and diversity optimisation. These have been termed *quality diversity* (QD) algorithms and are often (although not exclusively) based on the popular MAP-Elites algorithm [97, 116]. Applications are

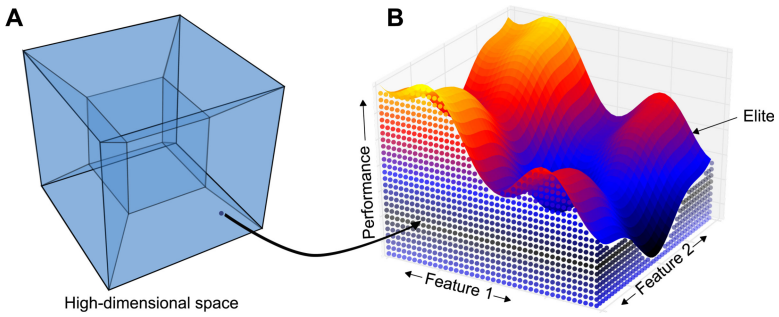

Figure 2.6: MAP-Elites search spaces [97]. The algorithm searches in a high-dimensional space (A) to identify the highest-performing solutions at each point in a low-dimensional feature space (B).

realised for problems where a selection of possible solutions demonstrating different characteristics is desired, rather than only a single effective solution. The following subsections review the MAP-Elites algorithm and some of its recent extensions for evolving behavioural diversity.

2.3.4.1 MAP-Elites: Multi-dimensional Archive of Phenotypic Elites

In 2015, the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) algorithm was first introduced [97]. Unlike most search algorithms which traditionally only return the overall best solution, MAP-Elites is used create an archive of high-performing solutions covering a user-defined, multi-dimensional search space. This type of search algorithm is referred to as an "illumination algorithm", because it illuminates the search space by providing insight into how different features of interest combine to influence performance [132].

The algorithm allows for searching in a high-dimensional space to find the highest-performing solution at each point in a low-dimensional feature space (see Figure 2.6). These highest-performing solutions are referred to as "elites" and are stored in an archive that is progressively explored to illuminate the fitness potential of different areas of the feature space. For example, the high-dimensional search space might be all possible robot designs, whereas the low-dimensional feature space could be height and weight. Pseudo-code for this algorithm is presented on the following page (see Algorithm 4).

MAP-Elites was originally demonstrated for solving optimisation problems within robotics applications such as navigation, exploration and locomotion tasks. For example, there have been studies investigating the evolution of maze navigation behaviours [115, 116], strategies for foraging tasks [16, 61], as well as diverse gaits for legged robots [29, 34, 88]. However, besides robotics, the use of MAP-Elites has been successfully extended to several other domains where solution diversity is desired. One example is in game design where the algorithm has been applied to generate game levels with varying characteristics [4]. Another is in the field of architecture where diverse sets of urban layouts have been produced [54]. Industrial design has also seen applications such as the generation of multiple aero-

Algorithm 4 The MAP-Elites algorithm [97]

```
procedure MAP-Elites
    (P ← ∅, X ← ∅)
    for iter = 1 → I do
        if iter < G then
            x' ← random_solution()
        else
            x ← random_selection(X)
            x' ← random_variation(x)
        b' ← feature_descriptor(x')
        p' ← performance(x')
        if P(b') = ∅ or P(b') < p' then
            P(b') ← p'
            X(b') ← x'
    return feature-performance map (P and X)
```

dynamic airfoil designs [53]. The use of MAP-Elites across these widely different problem spaces provides clear evidence of its versatility and effectiveness as a tool for promoting behavioural diversity and discovering novel solutions.

Despite the benefits of search space exploration, solution optimisation and cross-domain generalisability, there are still some notable limitations of MAP-Elites. Firstly, the algorithm can become computationally expensive, especially when using high-dimensional feature spaces or complex fitness functions [53]. Secondly, since the algorithm is inherently elitist, it can demonstrate a bias towards local optima which is exacerbated by poorly designed fitness functions that assign high fitness to sub-optimal solutions [44]. Thirdly, there is often difficulty in effective parameter tuning due to the consideration of several additional hyperparameters such as the number of niches, mutation rate and choice of behavioural characteristics [106]. As a result, several variants of MAP-Elites have been developed to address some of its general shortcomings or task-specific inadequacies.

MAP-Elites with Evolutionary Strategies was introduced in 2020 to efficiently scale the algorithm for high-dimensional controllers parameterised by large neural networks [32]. In 2019, MAP-Elites with Sliding Boundaries was introduced to avoid the overpopulation of archive cells with conflated behaviours by instead sliding the boundary of cells based on the distribution of solutions [48]. Dynamic mutation rates were tested with MAP-Elites in 2018 to reduce parameter tuning requirements [106]. Multi-task MAP-Elites was introduced in 2020 to simultaneously optimise solutions for multiple tasks, where each niche corresponds with a task [98]. There have been numerous other extensions to the MAP-Elites algorithm, but in the following subsections we review two relevant variants (EDQD and QED) for the purposes of robotic evolution.

Algorithm 5 The EDQD algorithm [61]

```
genome.randomInitialize();
localMap.create();
while (generations < maxGen) do
    for iteration = 0 to lifetime do
        if agent.hasGenome() then
            agent.move();
            broadcast(localMap);
        mapList ← receivedMaps
    localMap ← updateLocalMap(genome, fitness);
    genome.empty();
    if mapList.size() > 0 then
        createSelectMap();
        genome = applyVariation(select_random(selectMap));
        if collectedMapMemory.isForget() then
            mapList.empty();
```

2.3.4.2 EDQD: Embodied Distributed Quality Diversity

The EDQD algorithm was introduced by Hart, Steyven and Paechter in 2018 [61]. This distributed, evolutionary approach is implemented as a hybrid of MAP-Elites and mEDEA. As opposed to broadcasting their current *genome* (like in mEDEA), robots broadcast their *LocalMap* in EDQD. This LocalMap represents an elite archive of previously evaluated genomes (based on MAP-Elites). At the start of each new generation, robots then randomly select a genome from the received maps of the previous generation. Pseudo-code for this algorithm is presented on the following page (see Algorithm 5).

In this study, four variations of the algorithm were tested in a token gathering task environment. The variants are based on different mechanisms by which the active genome is selected from the received archives and how the LocalMap is updated. It was found that all of the EDQD variants outperformed the benchmark mEDEA algorithm in their ability to generate greater behavioural diversity, with one variant also achieving higher precision. This precision (or *opt-in reliability*) metric was introduced in the original MAP-Elites study [97] and measures the reliability of an algorithm to produce the optimal solution for a particular cell in the archive.

These results provide evidence that behavioural diversity can be generated without specific speciation mechanisms or geographical isolation in the task environment. One noted limitation is that useful functional traits for the specified task must be pre-defined, requiring some level of prior knowledge about desired behaviour. However, despite this, it is hypothesised that this approach could be useful in the further development of techniques for evolving swarms that are robust to changing environments.

Algorithm 6 The QED algorithm [15]

$\mathcal{M} \leftarrow \emptyset$
for $i = 1$ to p **do**
    $P[i] \leftarrow$ `random-controller()`
    Perform `add-controller`$(P[i])$
for $i = 1$ to I **do**
    $c \sim \mathcal{M}$
    $c' \leftarrow$ `mutate`(c)
    Perform `add-controller`(c')
procedure ADD-CONTROLLER(controller c)
    Randomly select $A_j \in \boldsymbol{P}_j \ \forall j \in \{1, ..., D\}$
    Generate environment $\tilde{\mathcal{E}}$ parameterised by $\mathbf{A}$
    $\beta \leftarrow$ `environment-descriptor`$(\tilde{\mathcal{E}})$
    if $\mathcal{M}[\beta] = \emptyset$ **or** $f(\tilde{\mathcal{E}}, c) > f(\tilde{\mathcal{E}}, \mathcal{M}[\beta])$ **then**
        $\mathcal{M}[\beta] = c$

2.3.4.3 QED: Quality-Environment Diversity

The QED algorithm was introduced by Bossens and Tarapore in 2020 [15]. As with EDQD, it is also built upon MAP-Elites. In contrast to vanilla MAP-Elites, solutions are located in the archive based on their *environment* descriptor (features of the solution's evaluation environment) rather than their *behavioural* descriptor. Furthermore, unlike the distributed approach for EDQD, QED is implemented as a centralised algorithm. Pseudo-code for this algorithm is presented on the following page (see Algorithm 6).

Rationale behind using environment descriptors for solutions stems from the idea that different evaluation environments are more likely to induce novel behaviours than multiple evaluations in the same environment. As such, QED is implemented with an environment generator that randomly perturbs various attributes of the environment (such as number of obstacles, size of the arena and number of robots) within pre-defined ranges. Robot controllers are then evaluated for a specific task (e.g. aggregation, dispersion, flocking, etc.) in each of these environments and archived with their resulting fitness scores. In this way, the algorithm promotes implicit exploration of a behaviour space through the explicit exploration of an environment space.

In the study, QED was used to train robot controller behaviours for fault recovery scenarios. Results demonstrated successful fault recovery with a high level of behavioural diversity in the generated solutions. It is noted that one of the principal benefits of the QED approach is the ability to simply describe a range of plausible environments without the need for prior knowledge of useful behavioural characteristics. However, a limitation to consider is a trade-off between improved generalisation and reduced performance in the normal operating environment.

2.4 Discussion

Overall, it is evident from the literature that the field of swarm robotics has advanced since its advent in the early 1990s, especially in the area of evolutionary robotics. There has been a notable shift in focus from single-pass evolutionary processes to methods of lifelong learning and adaptation. This has seen interesting applications in the evolution of multi-robot systems robust to environmental changes.

As opposed to the traditional approach of evolutionary algorithms, which exclusively optimise task performance (or fitness) of solutions, there has been increased interest in the generation of behavioural diversity within populations. This has precipitated the rise of a new class of evolutionary algorithms termed quality diversity (QD) algorithms, which aim to search behaviour spaces while concurrently maximising solution fitness. One of the foundational algorithms introduced in this regard is the MAP-Elites algorithm (see Section 2.3.4.1), which underlies many later extensions and other QD algorithms. For these approaches, generated solutions that demonstrate functional uniqueness (based on behavioural characteristics of interest) are usually inserted and stored in an archive to grow and maintain population diversity throughout the evolutionary process.

The evolution of behavioural diversity, with algorithms based on MAP-Elites such as EDQD and QED, presents a promising new approach to the research of population adaptability. With EDQD (see Section 2.3.4.2), embodied evolution has been successfully applied to the development of high-performing, adaptive robot systems. With QED (see Section 2.3.4.3), the evolution of adaptive behaviours (for fault tolerance) has been achieved through the definition and exploration of plausible task environments, without requiring specialised knowledge of desired behaviour patterns.

However, several limitations that affect these QD approaches have also been identified. MAP-Elites has been shown to be computationally expensive when exploring high-dimensional behaviour spaces and has a tendency to bias towards local optima. Additionally, EDQD necessitates that the dimensions of a target behaviour space are defined in advance, possibly constraining the discovery of optimal solutions. Finally, although QED attempts to address this latter issue, it presents a trade-off of reduced performance for improved generalisation.

These limitations present some gaps in the current state of the art. Firstly, there appears to be scope for further work in the investigation of alternative methods for evolving behaviourally diverse, heterogeneous swarms. Secondly, it is apparent that there is a lack of research on the ability for a swarm's memory of alternative behaviours to grow over time. Finally, there is a need to explore tasks that demand greater interactions with the environment.

In addition to the limitations discussed, another area that requires further investigation is the scalability of QD algorithms in the context of large-scale swarm robotics applications. As the number of robots in a swarm increases, the behaviour space that needs to be explored becomes exponentially larger, which could potentially lead to increased computational costs and longer convergence times. Therefore, it is crucial to develop scalable QD algorithms

that can efficiently handle large-scale swarm robotics applications, while still maintaining the diversity and adaptability of the population.

As such, the following chapter describes alternative methods for evolving both homogeneous and heterogeneous robot swarms in a dynamic, collective herding task environment (see Section 2.1.1.4). We will compare a traditional evolutionary algorithm, SSGA (see Section 2.3.3.1), which does not explicitly promote the generation of behavioural diversity, with a more recent quality diversity algorithm, MAP-Elites, which does. Furthermore, a novel approach is introduced for more efficiently generating heterogeneous swarms via the evolution of controller allocations for robot teams of increasing size. By addressing the issue of scalability, we hope to contribute to the development of QD algorithms that can effectively handle large-scale swarm robotics applications, thereby advancing the field towards its potential applications in real-world scenarios.

Chapter 3

Methods

In this chapter, we present the framework and algorithms used for evolving robot swarms in our collective herding task environment. Firstly, the simulation framework and task environment is described. Secondly, the morphology and controller architecture for the simulated agents is elucidated. This includes the robots (or "dogs") being evolved to perform the herding task and the rule-based agents (or "sheep") representing the targets to be herded. Thirdly, the various evolutionary algorithms implemented for evolving both homogeneous and heterogeneous robot swarms are explained, all of which are extensions to the popular SSGA and MAP-Elites algorithms. Finally, we detail the metrics used to assess solution fitness and behavioural diversity.

3.1 Simulation Task Environment

The collective herding task is simulated using an extended version[1] of the Roborobo! (version 4)[2] multi-agent simulation framework [20]. This framework is based on a C++ core engine for efficiency, but interfaced with through Python. For our task, a swarm of N robots, called "dogs", is assigned the objective of capturing a randomly dispersed group of M agents, called "sheep", inside a centrally-located target zone. Sheep move in a flocking pattern and actively avoid entering the target zone, unless pursued by a dog. Once they enter the target zone, they are considered "captured" and removed from the simulation. The 2D environment is bounded on all sides by walls. Figure 3.1 provides a visual snapshot of the environment during a simulation run.

3.1.1 Task Difficulty

Three difficulties of task environment have been defined (easy, medium and difficult) based on the ratio of dogs to sheep and their relative maximum translation speeds. The easy

[1] Extended project source code available at `https://github.com/scotthallauer/sheepdogai`

[2] Original Roborobo! source code available at `https://github.com/nekonaute/roborobo4`

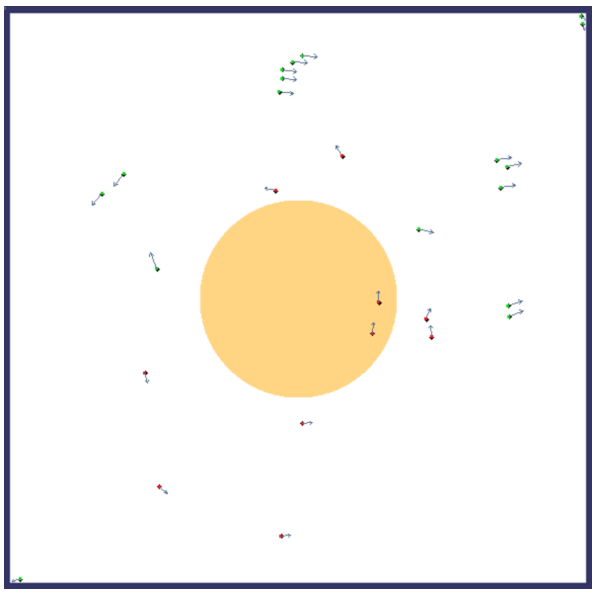

Figure 3.1: Simulation environment for the collective herding task. Red agents are dogs, green agents are sheep and the yellow circle represents the target zone.

task has more dogs that move faster than sheep, while the difficult task has more sheep that move faster than dogs. The medium task has an equal number of dogs and sheep that move with the same maximum speed. Table 4.1 provides parameter values for the different task difficulties.

Under this definition, it is expected that optimal dogs will be effective at herding individually in the easy environment (since the sheep move slower), while greater degrees of collaboration will be necessary in the more difficult environments (since the sheep move faster). Therefore, these task difficulties are expected to provide increasing pressure for the evolution of additional, advanced behaviours demonstrating collaboration.

3.1.2 Framework Extensions

To accommodate the collective herding task environment, several extensions to the Roborobo! framework have been implemented. The most fundamental and crucial change is added support for the concurrent simulation of two different agent types in the same environment. This is achieved through the assignment of a customised controller class (i.e. "dog" controller or "sheep" controller) to each agent in the environment. To implement the ANN controllers for the dog agents (see Section 3.2.1), the PyTorch[3] library [111] is used. To implement the heuristic controllers for the sheep agents (see Section 3.2.2), a variation of the "boids" algorithm first introduced by Reynolds in 1987 [119] is used.

For the implementation of our proposed evolutionary algorithms (see Section 3.3), the popular DEAP[4] library [49] is used. Additionally, to facilitate the MAP-Elites algorithm,

[3]PyTorch library available at `https://pypi.org/project/torch/`
[4]DEAP library available at `https://pypi.org/project/deap/`

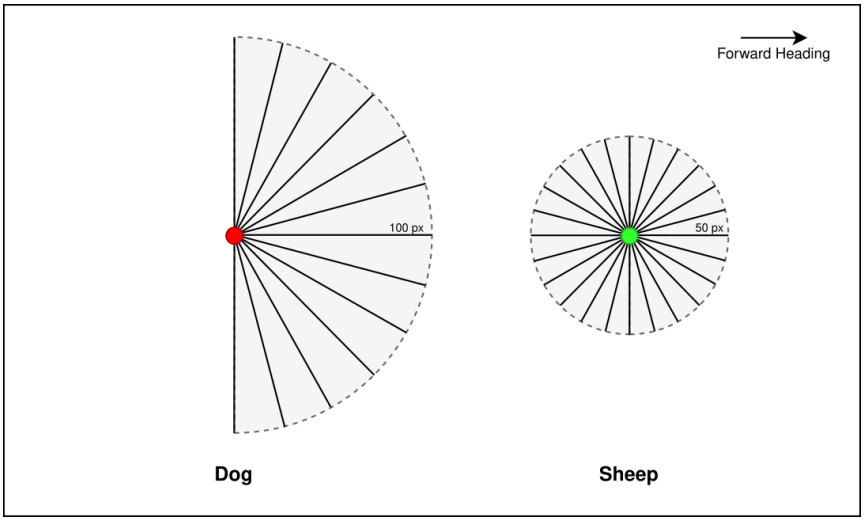

Figure 3.2: Overview of agent sensory configuration. Dogs have a sensory range of 100px and a 180° field of view. Sheep have a sensory range of 50px and a 360° field of view. Sensory detections are made at 15° intervals within these fields of view.

the QDpy[5] library [28] is used. Support for parallelisation has been added to greatly improve the efficiency of independent solution evaluation. Besides these significant extensions, a number of other minor changes were also made to enable the successful analysis of evolved solutions, including behavioural characteristic monitoring, fitness tracking, graphical simulations and generation checkpointing.

3.2 Agent Representation

Two types of agent, dogs and sheep, are simulated in this task environment. Although both incorporate a similar body shape and sensory configuration, there are major differences in the controllers used (and thus behaviours elicited). These agent-specific characteristics are described in the subsections that follow. Figure 3.2 gives a basic overview of the agent morphology.

3.2.1 Dogs

These are the robot agents that undergo neuro-evolution for the herding task. They incorporate a simple, circular morphology similar to Khepera [125] or e-puck robots, as the Roborobo! framework has built-in support for this body plan. In terms of sensory configuration, a radar-type proximity sensor is used which detects the nearest instance of each type of object (dog, sheep and wall) within a specific range (set to 100px) and field of view (set to [-90°, 90°]), where objects are detected at 15-degree intervals.

[5]QDpy library available at `https://pypi.org/project/qdpy/`

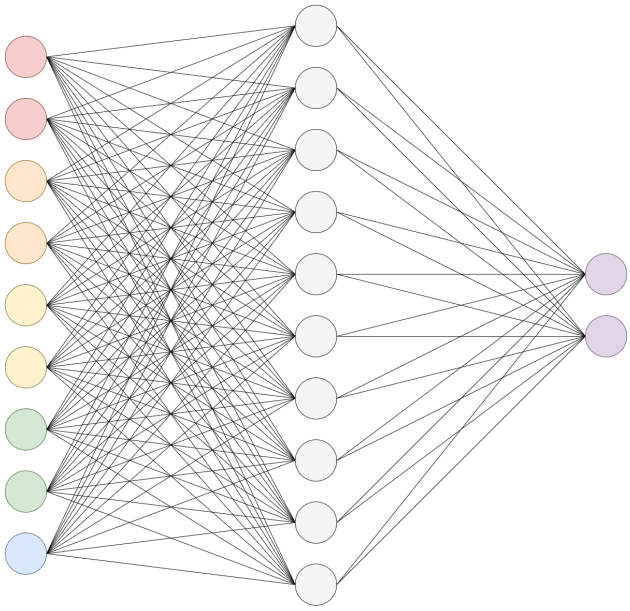

Figure 3.3: Artificial neural network topology for dog agent controllers. It consists of 9 input nodes, 10 hidden nodes and 2 output nodes. Input nodes include distance and angle values from 3 radar sensors (red, orange and yellow), distance and angle values from a target zone sensor (green) and a static bias value (blue). Output nodes include the dog's translation and rotation values (purple).

A fully-connected artificial neural network (ANN) is implemented for each dog's controller (see Figure 3.3). The topology consists of 9 input nodes, 10 hidden nodes and 2 output nodes, resulting in a total of 110 connection weights for the genome. The 9 input nodes include distance and angle values from 3 radar sensors (one for each object type), distance and angle values from a target zone sensor, and a bias input which is set to a constant value of 1. Distance values are normalised in the range [0, 1], where 0 is undetected and 1 is as close as possible. Angle values are normalised in the range [-1, 1], where -1 is -180 degrees and 1 is +180 degrees. The 2 output nodes include the dog's translation value in the range [-1, 1] (where -1 is maximum translation speed backwards and +1 is maximum translation speed forwards) and the dog's rotation value in the range [-1, 1] (where -1 is maximum rotation speed to the left and +1 is maximum rotation speed to the right). The tanh activation function is used between network layers.

3.2.2 Sheep

These are the heuristic agents that wander around the arena and should be herded into the target zone. The same circular morphology employed by the dog agents is also used by the sheep agents. Additionally, the same radar-type proximity sensor is used, although different range (set to 50px) and field of view (set to [-180°, 180°]) values are configured. A 360° field of view enables the sheep to detect dogs from behind them and to, thereby,

continue moving away when chased.

A variation of the "boids" algorithm for flocking behaviour [119] is implemented for each sheep's controller. This controller remains static throughout the evolutionary process and guides the movement of the sheep using simple avoidance and flocking rules. Avoidance rules are based on proximity thresholds for each type of object, ordered by priority (i.e., avoiding dogs is more important than avoiding the target zone). Flocking rules are configured with the *coherence* and *alignment* parameters. Coherence controls the rate at which sheep steer towards each other, while alignment controls the rate at which sheep match the average direction of other surrounding sheep. Unlike the dogs which can vary their translation speed, sheep move at a constant speed throughout their lifetime.

3.3 Evolutionary Algorithms

The evolution of robot swarms for our collective herding task is achieved through two primary approaches that are referred to herein as "behaviour evolution" and "allocation evolution". These are then further subdivided into variations of the SSGA (see Section 2.3.3.1) and MAP-Elites (see Section 2.3.4.1) algorithms which either evolve homogeneous or heterogeneous swarms. Figure 3.4 provides an overview of these evolutionary approaches, with the relevant algorithms explained in the subsections that follow.

3.3.1 Behaviour Evolution

For behaviour evolution, the ANN controllers governing dog behaviour are directly encoded as genomes of floating point weights, each in the range [-1, 1] (see Figure 3.5). These genomes are optimised for high task performance (see Section 3.4.1) using either SSGA or MAP-Elites. The algorithms are applied to the evolution of both homogeneous and heterogeneous swarms as detailed in the following.

3.3.1.1 SHOM: SSGA Homogeneous

In this approach, a population of genomes (set to 100 individuals) is randomly initialised and evaluated for the first generation. Thereafter, for each generation, individuals are selected from the population by tournament selection with a tournament size of 3, keeping the population size constant (as per SSGA). These individuals undergo two-point crossover and Gaussian mutation, each with a specific probability, before being evaluated. Each individual genome is evaluated as a homogeneous team of dogs (i.e., every dog in the simulation task environment has the same ANN weights applied to it). The newly evaluated individual genomes then become the offspring population for the next generation. Evolution continues in this way until the maximum number of generations is completed.

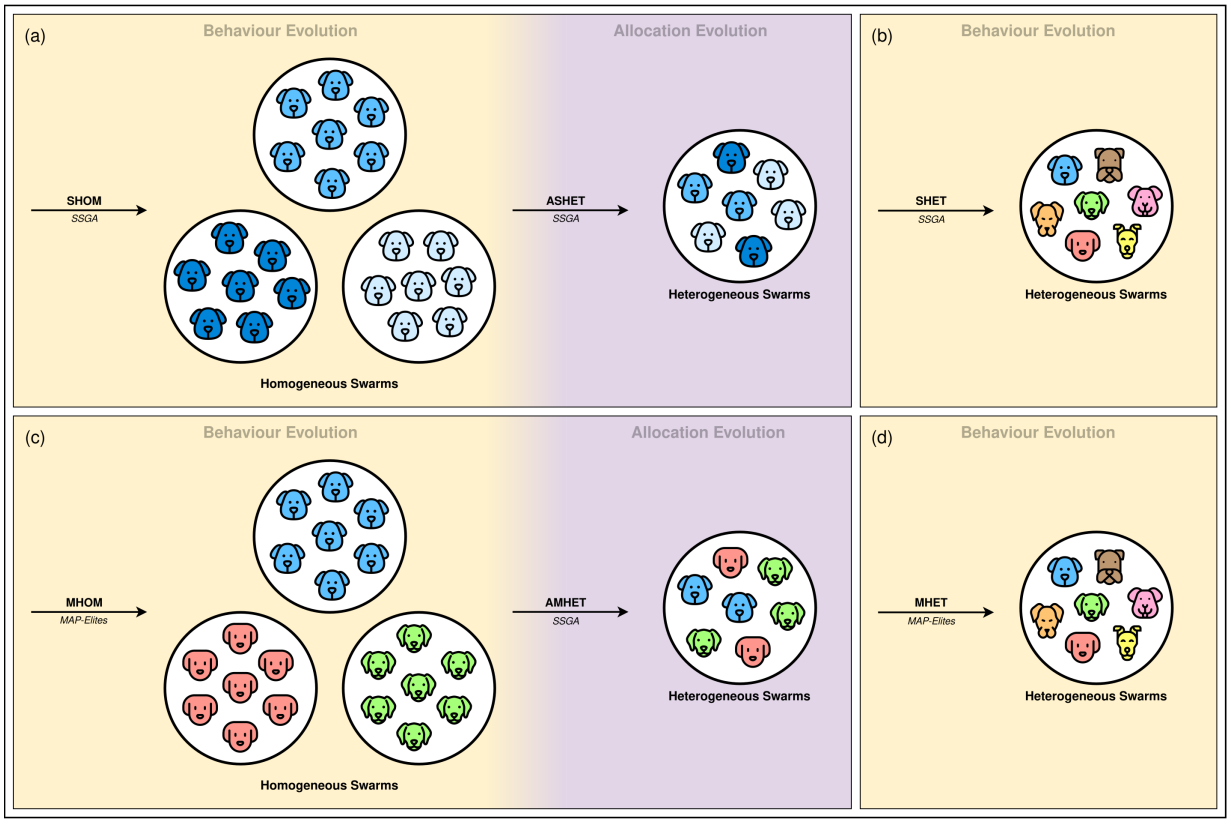

Figure 3.4: Overview of evolutionary algorithms implemented. SHOM (a) and MHOM (c) both evolve homogeneous swarms using SSGA and MAP-Elites, respectively. SSGA converges on similar solutions (represented by different colour shades of the same dog) whereas MAP-Elites ensures behavioural uniqueness between solutions (represented by different dogs). ASHET (a) and AMHET (c) both evolve (using SSGA) heterogeneous swarm allocations of solutions pre-evolved by SHOM and MHOM, respectively. Solutions generated by SHOM are projected into a MAP-Elites archive before running ASHET. SHET (b) and MHET (d) both evolve heterogeneous swarms in a single step using SSGA and MAP-Elites, respectively.

3.3.1.2 SHET: SSGA Heterogeneous

This method is similar to SHOM (see Section 3.3.1.1), except that each genome consists of floating point weights for N dog ANN controllers. Therefore, each individual genome is evaluated as a heterogeneous team of dogs, with each dog using a unique subset of ANN weights from the genome.

3.3.1.3 MHOM: MAP-Elites Homogeneous

Similar to the SSGA-based algorithms (see Section 3.3.1.1 and 3.3.1.2), a population of genomes (set to 100 individuals) is randomly initialised and evaluated for the first generation. However, during evaluation, a set of three behavioural characteristics is also

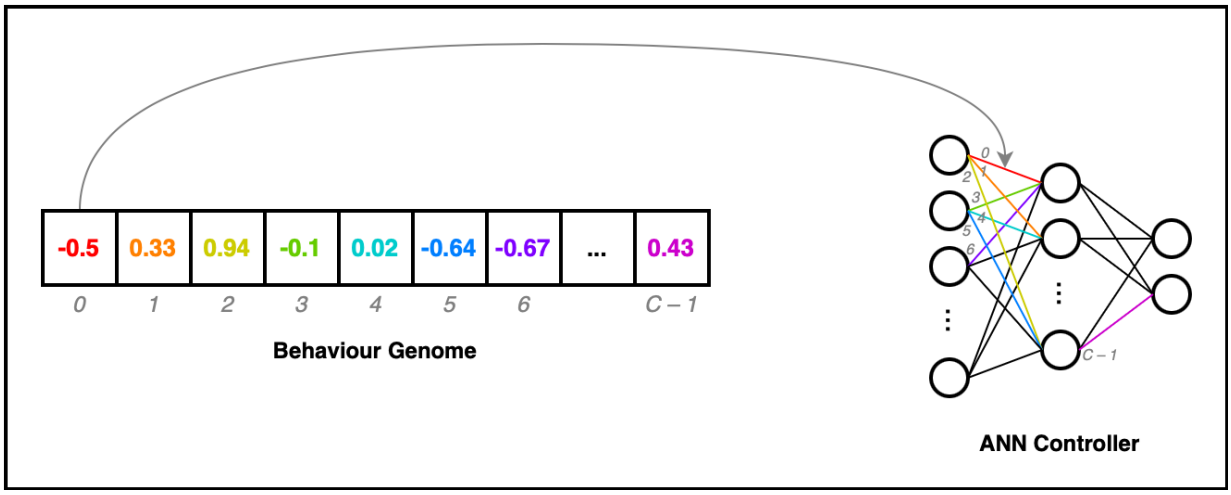

Figure 3.5: Behaviour genome mapping. Each behaviour genome consists of C floating point numbers in the range [-1, 1]. These numbers correspond with the weights for the C connections in an ANN controller which drives dog behaviour.

measured: (1) average distance between each dog and its nearest neighbouring dog, (2) average distance between each dog and its nearest neighbouring sheep, and (3) average distance between each dog and the target zone. These behavioural characteristics are normalised in the range [0, 1], where 0 is an average distance of 0 and 1 is the maximum distance observed for that characteristic in a calibration test run (see Section 4.3.1). Individual genomes are, as with SHOM, evaluated as a homogeneous team of dogs. Evaluated solution genomes are stored in a multi-dimensional archive, positioned in bins based on their behavioural characteristic values (as per the MAP-Elites algorithm). If there already exists another solution genome at the assigned bin, the new solution is only inserted if it has a higher fitness score (otherwise it is discarded and only the elite solution for that bin is retained). For every subsequent generation, individuals are selected from this solution archive by tournament selection with a tournament size of 3. Crossover and mutation is carried out in the same way as for the SSGA-based algorithms, and evolution continues until the maximum number of generations is completed.

3.3.1.4 MHET: MAP-Elites Heterogeneous

This method is similar to MHOM (see Section 3.3.1.3), except genomes consist of floating point weights for N dog ANN controllers. Therefore, each individual genome is evaluated as a heterogeneous team of dogs, with each dog using a unique subset of ANN weights from the genome. In the same way as for MHOM, elite solutions are stored in a multi-dimensional archive, positioned based on their behavioural characteristic values. Evolution continues until the maximum number of generations is completed.

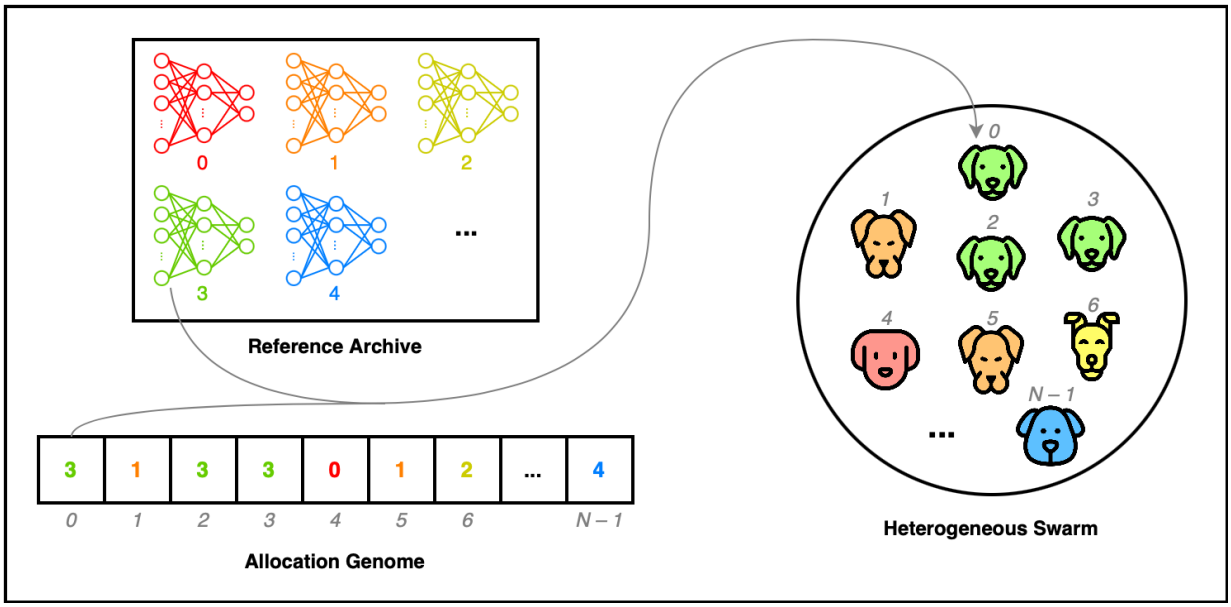

Figure 3.6: Allocation genome mapping. Each allocation genome consists of N integers in the range $[0, M-1]$. These integers represent indices in a reference MAP-Elites archive of M behaviourally diverse ANN controllers (or behaviour genomes). The sequence of integers represents the allocation of these controllers to a heterogeneous swarm of dogs.

3.3.2 Allocation Evolution

For allocation evolution, the heterogeneous swarms are encoded as genomes of integer indices for dog ANN controllers (see Figure 3.6). These ANN controllers are pre-evolved in the behaviour evolution stage and placed in MAP-Elites archives to ensure behavioural uniqueness. Initially, the N indices in a genome (for the N dogs in the swarm) are randomly selected from the range $[0, M-1]$, where M is the number of controllers in the reference archive. Indices are assigned with replacement, allowing for duplicate dogs in a swarm.

3.3.2.1 ASHET: Allocate SSGA Heterogeneous

In this approach, the objective is to optimise a swarm allocation of ANN controllers previously evolved by SHOM (see Section 3.3.1.1). Since SHOM is based on SSGA, its final population contains the same number of individuals as the initial population (set to 100 individuals). These individuals are not guaranteed to be unique and are likely to demonstrate behavioural similarities due to the tendency for SSGA to converge on variations of the same high-performing solution. Therefore, before commencing with allocation evolution, the final SHOM populations (from multiple experimental runs) are projected into MAP-Elites archives based on the tracked behavioural characteristics (see Section 3.3.1.3) for each individual. These archives are aggregated into a single reference archive containing only the elite solutions across all populations. The M individuals in this reference archive are each assigned a unique index.

Using this reference archive of ANN controllers, a population (set to 100 individuals) of random allocation genomes is initialised. Each genome consists of N indices selected, with replacement, from the range $[0,\ M-1]$. The number of dogs in the swarm, N, depends on the task environment difficulty being simulated (see Section 3.1.1). These genomes are evaluated as heterogeneous swarms in which each dog is allocated an ANN controller from the reference archive based on its index. Thereafter, for each generation, individuals (i.e. allocation genomes) are selected from the population by tournament selection with a tournament size of 3, keeping the population size constant (as per SSGA). These individuals undergo two-point crossover and uniform integer mutation, each with a specific probability, before being evaluated. The newly evaluated individual genomes then become the offspring population for the next generation. Evolution continues in this way until the maximum number of generations is completed.

3.3.2.2 AMHET: Allocate MAP-Elites Heterogeneous

As with ASHET (see Section 3.3.2.1), the objective for this approach is to optimise a swarm allocation of pre-evolved ANN controllers. However, in this case, the controllers being allocated are pre-evolved by MHOM (see Section 3.3.1.3). Since MHOM is based on MAP-Elites, its final population is already contained in a MAP-Elites archive of behaviourally unique individuals. To produce a reference archive for AMHET, the final MHOM population archives from multiple experimental runs are aggregated together. The M individuals in this reference archive are each assigned a unique index.

Using this reference archive of ANN controllers, a population (set to 100 individuals) of random allocation genomes is initialised. Thereafter, evolution of these allocation genomes continues in the same manner as for ASHET (using SSGA) until the maximum number of generations is completed.

3.4 Solution Evaluation

There are two perspectives from which the evolved solutions are evaluated and compared. Firstly, individual and population performance (or fitness) in the task environment is assessed and serves as the primary objective to be maximised through evolution. Secondly, individual and population behavioural diversity is measured for correlation with task performance and comparison between methods.

3.4.1 Task Performance Metrics

The quality of generated solutions is measured based on their performance in the collective herding task environment. This fitness scoring represents the objective-based function that drives the evolution of improved swarm behaviour.

3.4.1.1 Individual Fitness

The task performance of solution genomes is evaluated using an *aggregate* fitness function (which measures only high-level ability to accomplish a task) based on the number of sheep captured, c, out of the total number of sheep, t, during a simulation lifetime. Therefore, an evaluation score of 0 corresponds with none of the sheep captured and an evaluation score of 1 corresponds with all of the sheep captured. Due to the stochastic nature of the task environment, final genome fitness is averaged across n evaluation trials (set to 3 trials). Equation 3.1 summarises this fitness calculation for an individual.

$$F = \sum_{i=1}^{n} \left(\frac{c_i}{t_i} \right) \div n \tag{3.1}$$

3.4.1.2 Maximum Fitness

To assess the performance of a over evolutionary time, the fitness score of the best-performing individual in the population is used as a reference value. Unlike *average* fitness, this measure gives an indication of the algorithm's progress in finding an optimal solution. Since multiple experimental runs are completed for each algorithm, these maximum fitness values are averaged across runs for each generation. Equation 3.2 summarises the maximum fitness calculation at a particular generation for a population of P individuals (in a single experimental run).

$$F_{\max} = \max(F_1, F_2, F_3, ..., F_P) \tag{3.2}$$

3.4.2 Behavioural Diversity Metrics

The diversity of generated solutions is measured based on their ability to present distinct behavioural characteristics in the collective herding task environment. These metrics are used drive greater exploration of the search space in quality diversity algorithms (see Section 2.3.4) which aim to produce multiple possible solutions, rather than optimising only one high-performing solution.

3.4.2.1 Archive Size

Archive size refers to the number of solutions (or individuals) in the population displaying unique behavioural characteristics. The MAP-Elites algorithm implicitly tracks this metric by ensuring that only the elite solution for each demonstrated behaviour is stored in the archive.

3.4.2.2 Quality Diversity Score

Quality Diversity (QD) score, as introduced in [116], refers to the sum of fitness scores for all individuals stored in the MAP-Elites archive. This value is maximised by increasing both the fitness (or quality) of solutions and the number (or diversity) of solutions.

3.4.2.3 Unique Behaviours in Swarm

The behavioural diversity of a single heterogeneous swarm produced through allocation evolution (see Section 3.3.2) is measured based on the number of unique behaviours allocated to dogs in the swarm. For example, a swarm of five dogs with an allocation "14 8 14 14 3" has three unique behaviours (i.e. 3, 8 and 14).

3.5 Summary

In this chapter, the implementation of our simulation environment, robotic agents and evolutionary algorithms has been presented. The problem being investigated is a collective herding task (see Section 3.1) in which a swarm of "dog" robots has the objective to capture a dispersed flock of "sheep" agents in a central target zone. The dogs' behaviour is governed by ANN controllers which have their connection weights evolved to maximise task performance (or solution quality). Specifically, it is of interest whether or not behavioural diversity in heterogeneous swarms can provide a performance advantage over homogeneous swarms.

The evolutionary process is conducted by algorithms that are extensions of either SSGA (optimises solutions for quality) or MAP-Elites (optimises solutions for quality *and* diversity). Both homogeneous and heterogeneous swarms are generated through the approaches of either behaviour evolution (see Section 3.3.1) or allocation evolution (see Section 3.3.2). In behaviour evolution, the connection weights of ANN controllers are optimised. The algorithm extensions referred to as SHOM and MHOM are used to evolve homogeneous swarms, whereas SHET and MHET are used to evolve heterogeneous swarms. In allocation evolution, the assignment of pre-evolved ANN controllers to dogs in heterogeneous swarms is optimised. The algorithm extensions called ASHET and AMHET are used to evolve these heterogeneous swarm allocations. To evaluate the evolved solutions, various metrics are used to assess both task performance (including individual fitness and maximum fitness) and behavioural diversity (including archive size, quality diversity score and unique behaviours in swarm).

The following chapter describes the procedure by which these algorithms are systematically evaluated and compared. This includes the specific simulation configurations, experimental design and hyperparameter tuning.

Chapter 4

Experiments

This chapter outlines the structured procedure by which our proposed evolutionary algorithms (introduced in Chapter 3) are simulated and assessed in the collective herding task environment. We start by providing the customised simulator configuration used for our experiments. Thereafter, the overall experiment set-up is described, including the various experimental runs and parameters being compared. Finally, the approach taken for tuning the values of notable hyperparameters is explained.

4.1 Simulator Configuration

As mentioned in Section 3.1, an extended version of the Roborobo! multi-agent simulation framework [20] is used for these experiments. The evolutionary process, environment layout, agent morphology and controller options are all configured via a `.properties` file for each experiment run, which includes both built-in and custom framework parameters. The majority of parameters are assigned the same values across all experiments, besides the group size and translation speed for agents (i.e. dogs and sheep) which are different for each task difficulty. All parameter values are presented in Table 4.1.

With the exception of the algorithm being used (i.e. either SSGA or MAP-Elites), the neuro-evolution parameters are held constant for all experiments. There are 20 runs performed for each experiment to ensure statistical significance for results, with each run consisting of 200 generations for solution convergence. The population is always initialised with 100 random genomes in the first generation and every individual undergoes 3 trial evaluations (i.e. separate task simulations) per generation to determine an average fitness score. This to account for the stochastic nature of the task environment and provide a more reliable performance measure (as suggested for a very similar herding task environment in a 2016 study by Brulé *et al.* [22]). The dimensions for the dog ANN controllers are set to 9 input nodes, 10 hidden nodes and 2 output nodes, as further explained in Section 3.2.1. For the MAP-Elites archives, each of the 3 dimensions (corresponding with the 3 behavioural characteristics) is assigned 9 bins for elite solutions, resulting in a total of 729 bins in every archive. A total of 9 bins was selected along each dimension to provide sufficient capacity

Table 4.1: Neuro-evolution and simulation parameters.

Neuro-Evolution Parameters	
Replications per experiment (runs)	20
Generations per experiment run	200
Trial evaluations per individual	3
Initial population size	100
ANN dimensions (nodes): input / hidden / output	9 / 10 / 2
MAP-Elites archive: dimensions / bins	3 / 729
Crossover probability	0.5
Mutation probability	0.2
Simulation Parameters	
Time steps per trial evaluation	800
Initial agent positions	Random (outside target zone)
Dog team size: easy / medium / difficult	20 / 15 / 10
Sheep flock size: easy / medium / difficult	10 / 15 / 20
Dog translation speed: easy / medium / difficult	1 / 0.75 / 0.5
Sheep translation speed: easy / medium / difficult	0.5 / 0.75 / 1
Arena size (width × height)	600px × 600px
Target zone size (radius)	100px
Dog radar proximity sensor: range / FOV	(0px, 100px] / [-90°, 90°]
Sheep radar proximity sensor: range / FOV	(0px, 50px] / [-180°, 180°]
Sheep object avoidance (radius): wall / dog / sheep	15px / 50px / 5px
Sheep target zone avoidance: radius / strength	50px / 0.25

Note: Presented are the configuration options used for the neuro-evolution process and all experimental simulations of the collective herding task.

to explore a spectrum of different behaviours, rather than potentially over-simplifying the search space. Finally, the evolutionary crossover and mutation probabilities are set to 0.5 and 0.2, respectively, as further explained in Section 4.3.2.

Simulation parameters, which define the environment and agents, are also held constant (except for task difficulty changes, as previously mentioned). The environment is set as a 600px by 600px arena, surrounded by walls and containing a circular gathering pen (or "target zone"), with a radius of 100px, in the centre. All agents are randomly placed outside of the target zone at the start of each simulation which then runs for 800 time steps. Dogs are assigned a 180° field of view (FOV) with a 100px sensory range, while sheep have a 360° field of view with a 50px sensory range. Sheep avoid walls, dogs and other sheep within

Table 4.2: Experiment run configuration.

Evolution Type	Swarm Type	Algorithm	Task Environment		
			Easy	Medium	Difficult
Behaviour	Homogeneous	SSGA	shom-e	shom-m	shom-d
		MAP-Elites	mhom-e	mhom-m	mhom-d
	Heterogeneous	SSGA	shet-e	shet-m	shet-d
		MAP-Elites	mhet-e	mhet-m	mhet-d
Allocation	Heterogeneous	SSGA	ashet-e	ashet-m	ashet-d
			amhet-e	amhet-m	amhet-d

Note: There are 18 different experiments which are differentiated based on evolution type (behaviour or allocation evolution), swarm type (homogeneous or heterogeneous swarms), algorithm (SSGA or MAP-Elites) and task environment (easy, medium and difficult). Both sets of allocation evolution experiments use SSGA to evolve solutions, but differ based on the origin of their reference archive (ASHET uses archives generated by SHOM and AMHET uses archives generated by MHOM).

radii of 15px, 50px and 5px, respectively. They also actively avoid entering the target zone (unless pursued by a dog) within a radius of 50px and with an avoidance strength 25% of the maximum. Avoidance strength specifies the relative speed at which sheep rotate away from the target zone, with the maximum being the fastest possible rotation speed supported by the simulator. Finally, the task difficulty is configured based on the ratio of dogs to sheep (20 : 10 for easy, 15 : 15 for medium and 10 : 20 for difficult) and translation speed of dogs compared to sheep (1 : 0.5 for easy, 0.75 : 0.75 for medium and 0.5 : 1 for difficult).

4.2 Experiment Set-Up

To investigate our research questions (see Section 1.1), several experiments have been designed that evaluate and compare the capabilities of SSGA and MAP-Elites to evolve homogeneous and heterogeneous swarms in three task environment difficulties. These are primarily divided into the *behaviour evolution* and *allocation evolution* experiments. All experiments and the key differences between them are listed in Table 4.2.

4.2.1 Behaviour Evolution

We conducted four sets of *behaviour evolution* (see Section 3.3.1) experiments with the simulation framework, each using a different evolutionary algorithm (SHOM, SHET, MHOM and MHET). SHOM and MHOM each evolved homogeneous swarms, while SHET and MHET each evolved heterogeneous swarms. Three difficulties of task environment (easy, medium and difficult) were tested and results averaged over 20 runs per experiment.

The three behavioural characteristics (see Section 4.3.1) were tracked and recorded for evaluated individuals across all experiments. Although not used in SSGA, these values allow for post-processing the evolved populations and projecting them into three-dimensional solution archives which can be directly compared with those produced by MAP-Elites for behavioural diversity.

4.2.2 Allocation Evolution

We conducted two sets of *allocation evolution* (see Section 3.3.2) experiments, each using a variation of the same evolutionary algorithm (ASHET and AMHET). Both ASHET and AMHET make use of SSGA to evolve and optimise swarm controller allocations, but use different reference archives of pre-evolved controllers. ASHET uses aggregate archives produced by SHOM (using SSGA) whereas AMHET uses aggregate archives produced by MHOM (using MAP-Elites). In the same manner as for the behaviour evolution experiments (see Section 4.2.1), three difficulties of task environment were tested and results averaged over 20 runs per experiment.

In addition to task performance, the number of unique behaviours in evolved swarm allocations (see Section 3.4.2.3) was tracked as measure of behavioural diversity across all allocation evolution experiments.

4.3 Parameter Tuning

Several tuning experiments were performed to determine appropriate values for important simulation hyperparameters. These include both the behavioural characteristics being monitored as well as the crossover and mutation probabilities which configure the evolutionary process.

4.3.1 Behavioural Characteristics

To assess behavioural diversity between solutions, three behavioural characteristics are monitored over the course of each trial evaluation:

1. **Dog-Dog Distance**: Average distance between each dog and its nearest neighbouring dog.
2. **Dog-Sheep Distance**: Average distance between each dog and its nearest neighbouring sheep.
3. **Dog-Pen Distance**: Average distance between each dog and the target zone.

These behavioural characteristics correspond with the dimensions for the solution archives populated by the MAP-Elites algorithm. Since the archives are divided into a finite number of evenly-spaced bins along each dimension, the range of possible values needs to be

Table 4.3: Behavioural characteristic maximum values.

Behavioural Characteristic	Theoretical Maximum (T)	Observed Value (O)	Practical Maximum (P)
Dog-Dog Distance	$\sqrt{2 \times 600^2}$	$(0.06 \pm 0.03) \times T$	$0.1 \times T \approx 85$px
Dog-Sheep Distance	$\sqrt{2 \times 600^2}$	$(0.3 \pm 0.05) \times T$	$0.4 \times T \approx 339$px
Dog-Pen Distance	$\sqrt{2 \times 300^2} - 100$	$(0.4 \pm 0.2) \times T$	$0.8 \times T \approx 259$px

Note: Each behavioural characteristic has a theoretical maximum value (T) determined by the dimensions of the task environment. The observed values (O) are those produced in a calibration test run. The practical maximum values (P) are derived from the observed values and represent realistic upper limits for the behavioural characteristics in practice. The dimensions for the MAP-Elites archives are configured using these P values.

known (or estimated) in advance to ensure the correct placement of solutions. Although each characteristic has an easily determined range of theoretically possible values, the maximum values of these ranges are exceedingly unlikely to occur in practice. Therefore, to more fully utilise the bins available in the solution archives, it makes sense to rather estimate a *practical* maximum value for each characteristic.

The theoretical maximum values for these characteristics can be calculated based on the dimensions of the simulated task environment. For both dog-dog distance and dog-sheep distance, the theoretical maximum is the furthest possible distance between two points in the environment (which is between two diagonally opposite arena corners). This measure serves as the *absolute* maximum distance between two agents, since the actual maximum distance is reduced with more agents in the environment. For dog-pen distance, the theoretical maximum is the furthest possible distance from the perimeter of the target zone (or "pen") to the perimeter of the environment (which is between an arena corner and the target zone).

To estimate practical maximum values, a calibration test run was performed that measured the behavioural characteristics for 100 individuals over 100 generations in the easy task environment using SHOM. The observed mean values and standard deviations were calculated in relation to the theoretical maxima and used to calculate practical maximum values as approximately two standard deviations above the mean. These theoretical, observed and practical values are all presented in Table 4.3.

4.3.2 Variation Operators

Two significant tunable hyperparameters for the evolutionary process are the probabilities for crossover and mutation events. Crossover is the genetic operation whereby two parent genomes in the population are recombined to produce two offspring genomes, each containing components from both parents. Mutation is the genetic operation whereby singular genes in a genome are randomly changed to a different value. Higher probabilities for both

Table 4.4: Variation operator values for tuning experiments.

Hyperparameter	Value 1	Value 2	Value 3
Crossover probability	0.0	**0.5**	1.0
Mutation probability	0.05	0.1	**0.2**

Note: Presented are values tested for both crossover and mutation probabilities in separate runs of an allocation evolution experiment using AMHET in the easy task environment. The bold values are the default values used when not testing that parameter.

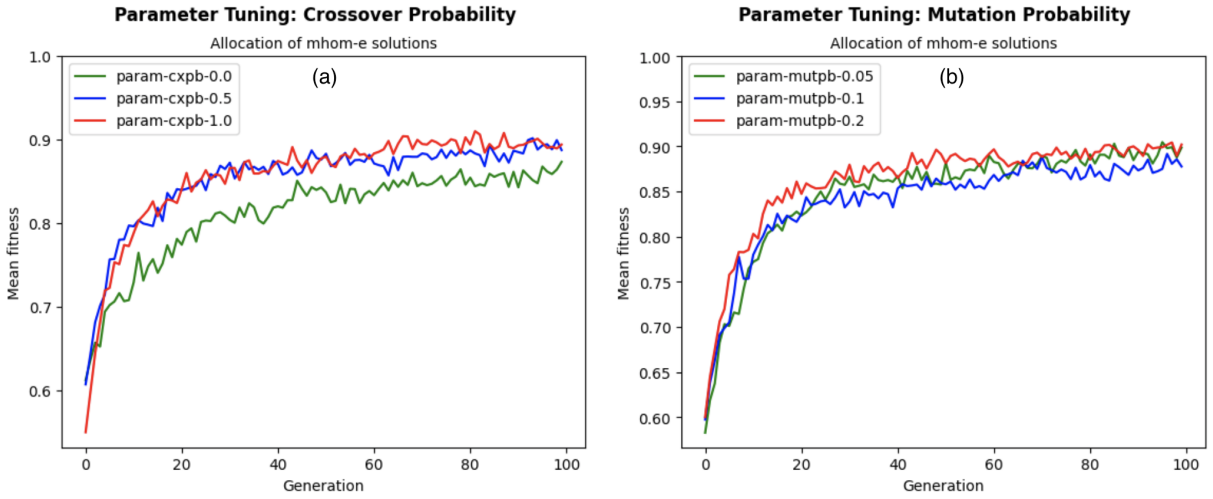

Figure 4.1: Variation operator tuning experiment results. Mean solution fitness score during evolution with different parameter values for crossover probability (a) and mutation probability (b). Crossover probability was held at 0.5 for all mutation probability tuning runs and mutation probability was held at 0.2 for all crossover probability tuning runs. All experiments were performed using AMHET in the easy task environment over 100 generations.

crossover and mutation correspond with faster exploration of the search space, but also with a greater replacement rate of existing solutions.

To optimise these hyperparameters, three different values for each (see Table 4.4) were tested in separate runs of an allocation evolution experiment (i.e. allocation of solutions from MHOM using AMHET in the easy task environment) over 100 generations. A default value was applied when not testing a specific parameter. The mean fitness of solutions was tracked during evolution and compared between parameter configurations to select the best value for each.

As presented in Figure 4.1a, the crossover probability value of 0.0 produced the lowest scoring solutions, while the values 0.5 and 1.0 performed comparably. For the mutation probability tuning experiments (see Figure 4.1b), the value 0.1 produced the lowest scoring solutions, while the values 0.05 and 0.2 produced similar results. Since the default probability values used by the DEAP library [49] for crossover and mutation are 0.5 and

0.2, respectively, and they demonstrate good performance in our tuning experiments, these values were selected for all other experiments.

4.4 Summary

In this chapter, the simulator configuration, experimental set-up and methods for parameter tuning have been described. Regarding simulator configuration, the specific parameter values used for neuro-evolution and simulation environment were outlined. This includes the arena layout, agent morphology, sensory configuration and task difficulty definitions. Then, in terms of experimental set-up, the two categories of behaviour and allocation evolution experiments were explained along with the relevant algorithms being tested. These were broken down into 18 separate experiment runs, differentiated by evolution type, swarm type, underlying algorithm and task environment difficulty. Lastly, the approaches taken for tuning two sets of important hyperparameters, namely behavioural characteristics and variation operators, were detailed as well as their final selected values.

The following chapter presents the results produced from these behaviour and allocation evolution experiments, using the task performance and behavioural diversity metrics (see Section 3.4) as a means of evaluation. Thereafter, a detailed discussion analyses the significance of these results in the context of our research questions and related work.

Chapter 5

Results and Discussion

In this chapter, the final experimental results are presented and interpreted. This includes independent analyses of the behaviour evolution experiments and the allocation evolution experiments, followed by a comparative evaluation of both methods. Lastly, we discuss the significance of these results in line with our research questions and previous work.

5.1 Results

The presentation of results is divided into three subsections, starting with the behaviour and allocation evolution experiments and ending with an overall comparison of evolutionary algorithms. For each experiment, the progressive change in metric values is visualised over evolutionary time. The primary metrics being measured include archive size, maximum fitness and QD score (see Section 3.4). Algorithms are contrasted on the basis of whether they evolve homogeneous or heterogeneous swarms, whether they are based on SSGA or MAP-Elites, and whether they implement behaviour or allocation evolution. Pairwise t-test results are also provided to determine the statistical significance of any differences between experiments.

5.1.1 Behaviour Evolution

As described in the experiment set-up for behaviour evolution (see Section 4.2.1), four alternative evolutionary algorithms (SHOM, MHOM, SHET and MHET) are tested in three difficulties of task environment (easy, medium and difficult). Connection weights for ANN controllers are evolved for *homogeneous* swarms by SHOM and MHOM, and for *heterogeneous* swarms by SHET and MHET.

5.1.1.1 Metric Trends

Figure 5.1 presents the archive size, maximum fitness and QD score over evolutionary time for each behaviour evolution algorithm across task difficulties, averaged over 20 runs.

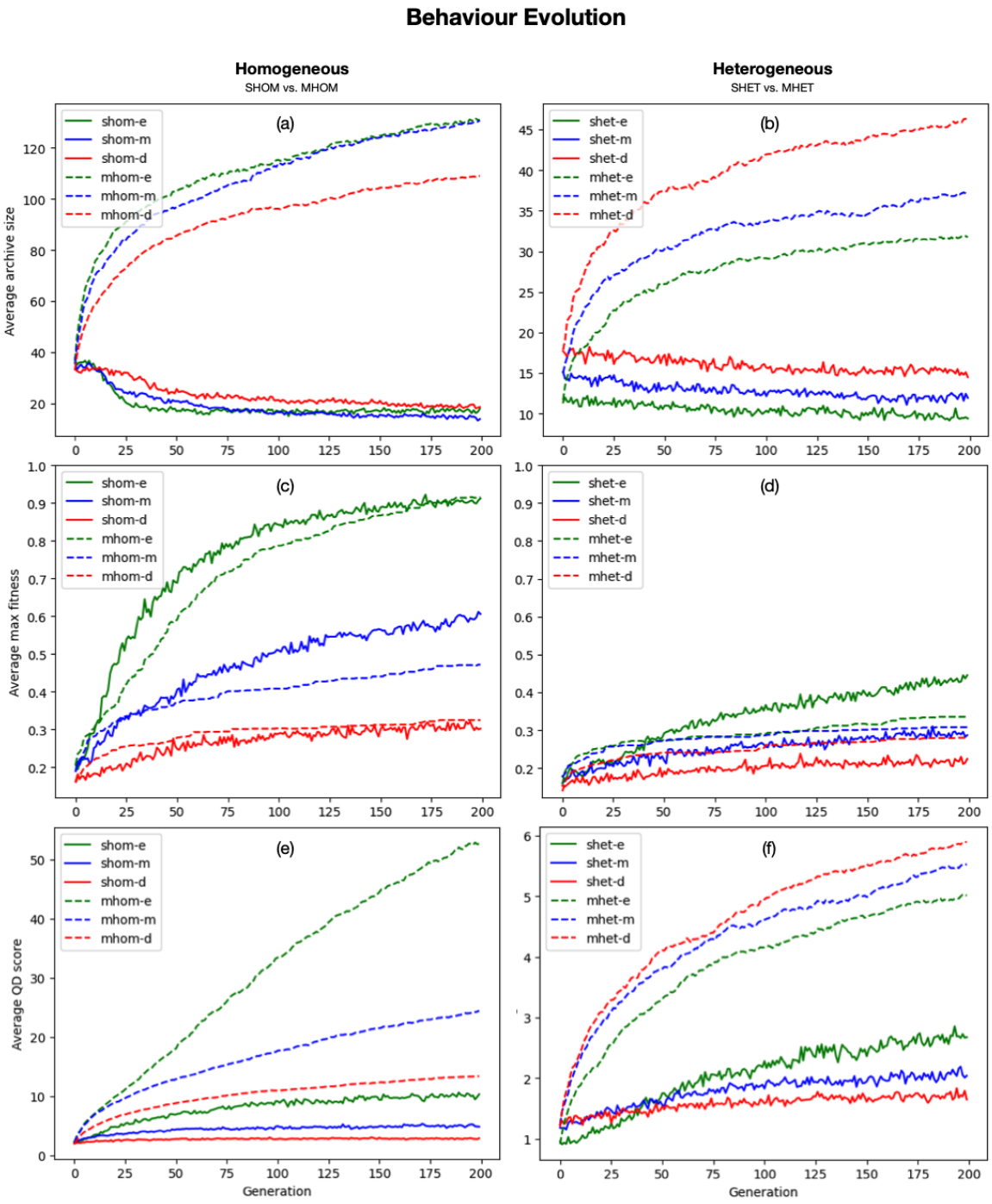

Figure 5.1: Behaviour evolution metric trends. Archive size (a and b), max fitness (c and d) and QD score (e and f) results are presented over 200 generations. The algorithms that evolve homogeneous swarms (SHOM and MHOM) are on the left, while the algorithms that evolve heterogeneous swarms (SHET and MHET) are on the right. Results from the easy (green), medium (blue) and difficult (red) task environments are provided for each algorithm, averaged over 20 runs. Note axis scale differences for archive size and QD score.

Archive size (see Figure 5.1a and 5.1b) is seen to rapidly increase and then taper off over time for MAP-Elites algorithms (MHOM and MHET) whereas it gradually decreases over time for SSGA algorithms (SHOM and SHET). It is also notable that the maximum archive size reached for heterogeneous swarms (SHET and MHET) is substantially lower than that for homogeneous swarms (SHOM and MHOM).

Maximum fitness results (see Figure 5.1c and 5.1d) demonstrate similar trends between SSGA and MAP-Elites algorithms, with both achieving increased fitness over time which begins to plateau in later generations. These results again favour the homogeneous swarms which both produce considerably fitter solutions for the final generation than the heterogeneous swarms.

Finally, the QD score (see Figure 5.1e and 5.1f) quantifies the interplay between solution quality (or fitness) and solution diversity (or archive size) over time. A general increase in QD score over time is witnessed across all algorithms (except for SHOM in the difficult environment), with MAP-Elites algorithms outperforming SSGA algorithms. This is to be expected since SSGA only optimises for solution quality, while MAP-Elites optimises for both quality and diversity. As with previous metrics, the homogeneous swarms achieve far greater QD scores than the heterogeneous swarms.

5.1.1.2 Solution Archives

Since the fundamental objective of these algorithms is to generate a high-performing population of solutions at the end of the evolutionary process, it is natural to assess the effectiveness of each algorithm by examining the quality (and diversity) of solutions in the final generation. To this end, the final populations generated by each algorithm (over 20 replication runs) have been aggregated together to produce a "swarm map" (or solution archive) for every experiment that can be compared. These swarm maps are MAP-Elites archives that hold the behaviourally unique solutions generated by all runs of each experiment. Only the elite solutions are stored in the swarm map, meaning that if multiple runs produce a solution located in the same archive cell then only the one with the highest fitness score is stored. The SSGA algorithms (SHOM and SHET) do not produce MAP-Elites archives and so their final populations are first projected into archives (based on the tracked behavioural characteristics) before being aggregated.

Figure 5.2 presents the flattened swarm maps for each behaviour evolution algorithm across task difficulties. It is evident from the more populated archives that the behaviour search space is more adequately explored for the homogeneous swarms (SHOM and MHOM) than the heterogeneous swarms (SHET and MHET). Additionally, when comparing SSGA algorithms to MAP-Elites algorithms within each type of swarm (i.e. SHOM vs. MHOM and SHET vs. MHET), it seems that the MAP-Elites algorithms populate a greater proportion of the archive cells than SSGA. This is to be expected since MAP-Elites is an "illumination algorithm" (see Section 2.3.4.1) specifically designed to explore the search space more effectively. It is also evident from this figure that certain regions of the behaviour space correspond with higher performing solutions than other regions, which suggests the useful ranges of behavioural characteristics to select solutions from.

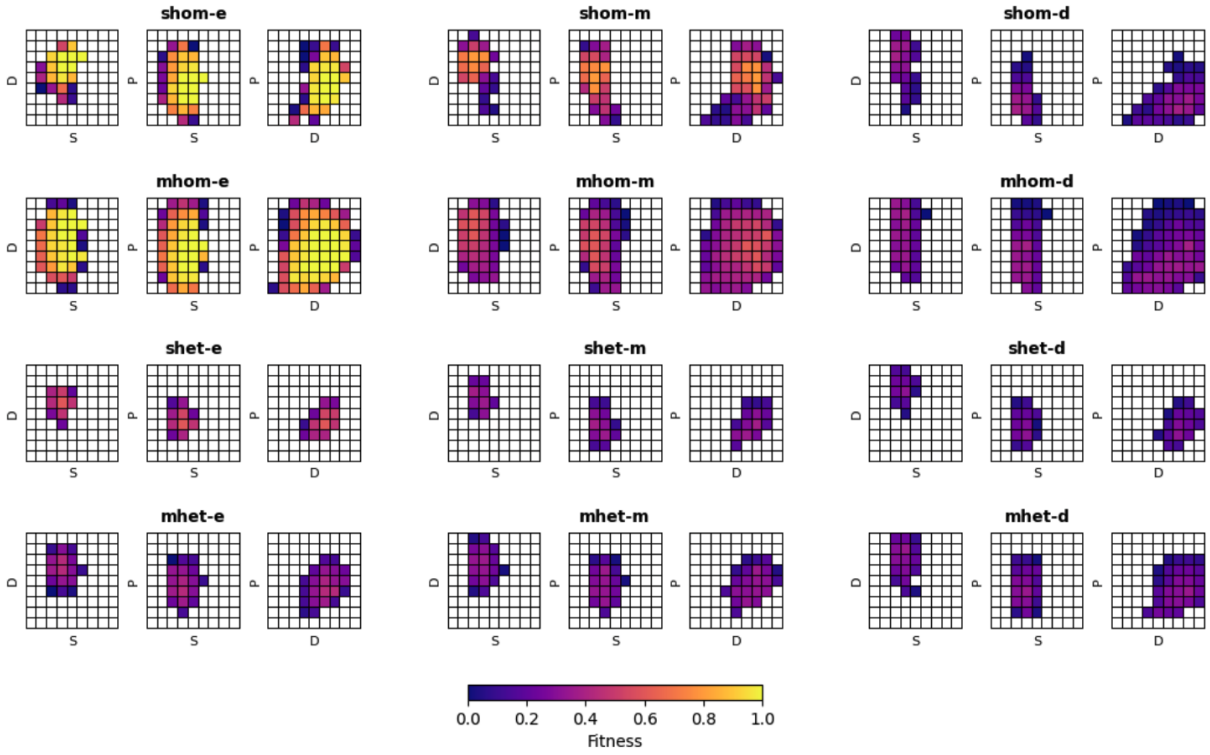

Figure 5.2: Behaviour evolution solution archives. Projected MAP-Elites solution archives for the behaviour evolution algorithms (SHOM, MHOM, SHET and MHET) in the easy (e), medium (m) and difficult (d) task environments at generation 200. Each archive is aggregated over 20 runs where, for each cell, the best solution at that position from any run is selected. Tracked behavioural characteristics are dog-dog distance (D), dog-sheep distance (S) and dog-pen distance (P). The three-dimensional archives are flattened on each plane (taking the best solution along the hidden axis at each position) and presented as two-dimensional grids (D×S, P×S and P×D).

5.1.1.3 Statistical Tests

To properly compare these algorithms, statistical tests are necessary for evaluating significant differences between measured metric values. As such, pairwise t-tests have been performed between algorithms using the final generation from all 20 runs of each experiment. These t-test results are presented in tables that highlight the significant and insignificant differences between algorithms for each metric. Firstly, a comparison between SSGA algorithms and MAP-Elites algorithms for both homogeneous and heterogeneous swarms is presented in Table 5.1. Thereafter, a comparison between homogeneous swarms and heterogeneous swarms for both SSGA and MAP-Elites algorithms is presented in Table 5.2.

When comparing SSGA and MAP-Elites algorithms (see Table 5.1), we find that MAP-Elites (MHOM and MHET) achieves significantly greater ($p < 0.05$) archive sizes and QD scores than SSGA (SHOM and SHET) across all task environment difficulties. Maximum

Table 5.1: Behaviour evolution (SSGA vs. MAP-Elites) statistical test results.

Metric	Homogeneous SHOM vs. MHOM			Heterogeneous SHET vs. MHET		
	Easy	**Medium**	**Difficult**	**Easy**	**Medium**	**Difficult**
Archive Size	MHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓	MHET ↑ ✓	MHET ↑ ✓	MHET ↑ ✓
Max Fitness	MHOM ↑ ×	SHOM ↑ ✓	MHOM ↑ ✓	SHET ↑ ✓	MHET ↑ ×	MHET ↑ ✓
QD Score	MHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓	MHET ↑ ✓	MHET ↑ ✓	MHET ↑ ✓

Note: Pairwise t-test results are presented for independent samples at generation 200 from 20 runs of each algorithm. Swarms evolved via SSGA are compared to those evolved via MAP-Elites, for both homogeneous and heterogeneous swarms (i.e. SHOM vs. MHOM and SHET vs. MHET). Comparisons are made across each task environment difficulty and the algorithm with the greater metric value is indicated. Those with a significant difference between values ($p < 0.05$) are indicated with a tick (green), while those with an insignificant difference ($p \geq 0.05$) are indicated with a cross (red).

Table 5.2: Behaviour evolution (homogeneous vs. heterogeneous) statistical test results.

Metric	SSGA SHOM vs. SHET			MAP-Elites MHOM vs. MHET		
	Easy	**Medium**	**Difficult**	**Easy**	**Medium**	**Difficult**
Archive Size	SHOM ↑ ✓	SHOM ↑ ✓	SHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓
Max Fitness	SHOM ↑ ✓	SHOM ↑ ✓	SHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓
QD Score	SHOM ↑ ✓	SHOM ↑ ✓	SHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓

Note: Pairwise t-test results are presented for independent samples at generation 200 from 20 runs of each algorithm. Homogeneous swarms are compared to heterogeneous swarms, for both those evolved via SSGA and those evolved via MAP-Elites (i.e. SHOM vs. SHET and MHOM vs. MHET). Comparisons are made across each task environment difficulty and the algorithm with the greater metric value is indicated. Those with a significant difference between values ($p < 0.05$) are indicated with a tick (green).

fitness results, however, are not unanimous in favouring MAP-Elites. For homogeneous swarms, MAP-Elites produces fitter solutions in both the easy and difficult environments, but only *significantly* fitter ($p < 0.05$) in the difficult environment. SSGA significantly outperforms MAP-Elites in the medium environment. For heterogeneous swarms, MAP-Elites again produces significantly fitter ($p < 0.05$) solutions in the difficult environment, but only insignificantly fitter ($p > 0.05$) solutions in the medium environment. Here, SSGA

significantly outperforms MAP-Elites in the easy environment.

When comparing homogeneous and heterogeneous swarms (see Table 5.2), it is evident that the evolved homogeneous swarms achieve significantly higher ($p < 0.05$) scores than heterogeneous swarms for all metrics and across all task environment difficulties. This holds true for both SSGA algorithms (SHOM vs. SHET) and MAP-Elites algorithms (MHOM vs. MHET).

5.1.2 Allocation Evolution

As described in the experiment set-up for allocation evolution (see Section 4.2.2), two variations (ASHET and AMHET) of the same evolutionary algorithm are tested in three difficulties of task environment (easy, medium and difficult). For both experiments, swarm allocations of pre-generated ANN controllers are evolved for *heterogeneous* swarms using SSGA. ASHET and AMHET differ based on the method used to produce their reference controller archives, with ASHET using controllers generated by SHOM and AMHET using controllers generated by MHOM.

5.1.2.1 Metric Trends

Figure 5.3 presents the archive size, maximum fitness, QD score and unique behaviours in swarm over evolutionary time for each allocation evolution algorithm across task difficulties, averaged over 20 runs.

Archive size (see Figure 5.3a) appears to decrease over time for all experiments, starting steeply before levelling out. For evolved populations using reference archives generated by MAP-Elites (AMHET), both initial and final archive sizes are greater than those using reference archives generated by SSGA (ASHET). This holds true across all task environment difficulties.

Maximum fitness results (see Figure 5.3b) display different trends between ASHET and AMHET. For all ASHET experiments, maximum fitness only increases slightly from the starting generation and then plateaus for the remainder of evolution with little improvement. On the other hand, for all AMHET experiments, we observe a substantial increase in maximum fitness over the first 25 to 50 generations before flattening out at approximately the same level as for ASHET in each task difficulty.

QD score results (see Figure 5.3c) also follow alternative trends between ASHET and AMHET. Although both seem to plateau quite early in the evolutionary process (around generation 25), QD score for ASHET tends to *decrease* from the first generation whereas QD score for AMHET tends to *increase* from the first generation. This is evident in the medium and difficult task environments, but does not seem to hold in the easy task environment for AMHET.

Finally, in addition to the above metrics which were also monitored for the behaviour evolution experiments, the number of unique behaviours allocated to a swarm (see Figure 5.3d) was tracked for the allocation evolution experiments. This metric is constrained

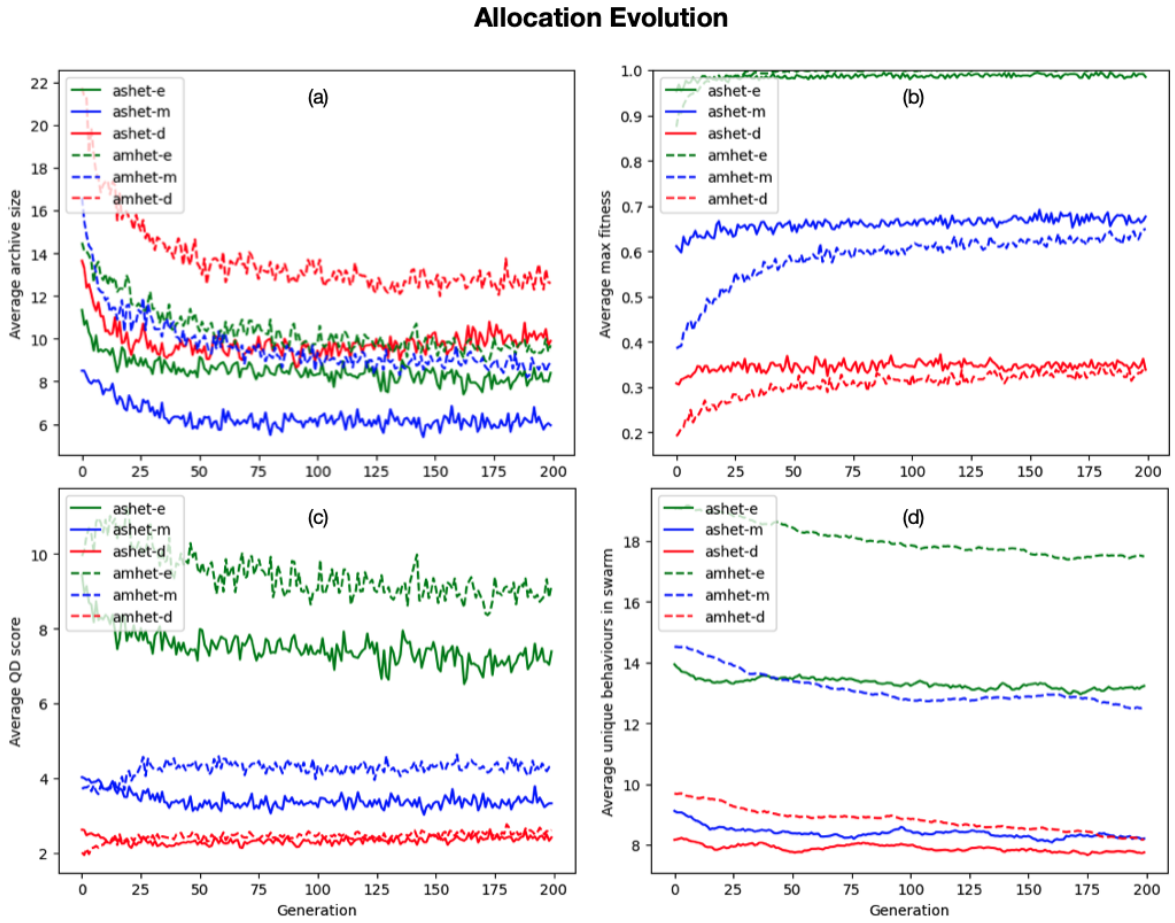

Figure 5.3: Allocation evolution metric trends. Archive size (a), max fitness (b), QD score (c) and unique behaviours in swarm (d) results are presented over 200 generations. Results from the easy (green), medium (blue) and difficult (red) task environments are provided for each algorithm (ASHET and AMHET), averaged over 20 runs.

by the size of the robot swarm being evolved, which differs between task environment difficulties. Teams of 20 dogs are evolved for the easy task environment, teams of 15 dogs for the medium task environment and teams of 10 dogs for the difficult task environment. AMHET appears to allocate more unique behaviours to swarms than ASHET, although there is a trend for this number of behaviours to gradually decrease over the course of evolution.

5.1.2.2 Solution Archives

As with the behaviour evolution experiments, the final populations generated by each algorithm (over 20 replication runs) have been aggregated together to produce solution archives for every allocation evolution experiment. Since both ASHET and AMHET evolve solutions using SSGA as the underlying algorithm, they do not produce MAP-Elites archives.

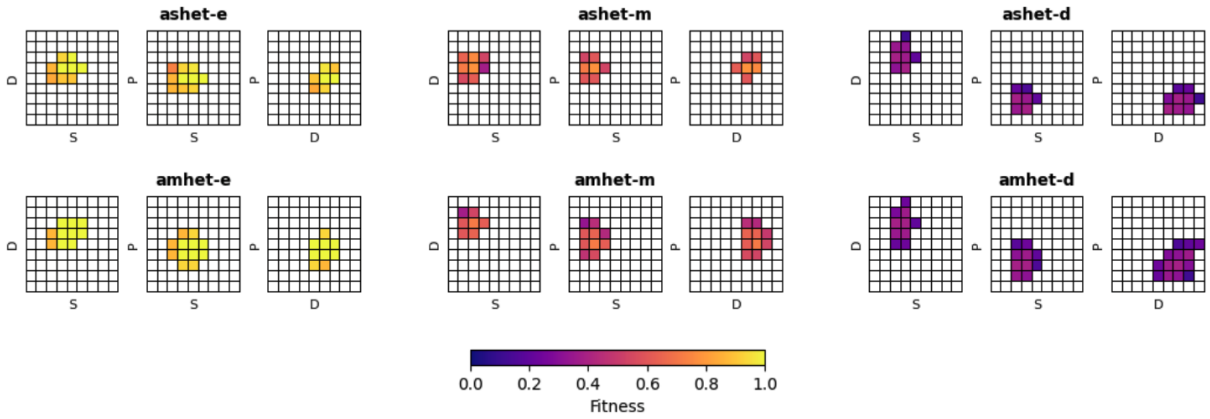

Figure 5.4: Allocation evolution solution archives. Projected MAP-Elites solution archives for the allocation evolution algorithms (ASHET and AMHET) in the easy (e), medium (m) and difficult (d) task environments at generation 200. Each archive is aggregated over 20 runs where, for each cell, the best solution at that position from any run is selected. Tracked behavioural characteristics are dog-dog distance (D), dog-sheep distance (S) and dog-pen distance (P). The three-dimensional archives are flattened on each plane (taking the best solution along the hidden axis at each position) and presented as two-dimensional grids (D×S, P×S and P×D).

Therefore, the final populations are projected into archives using the tracked behavioural characteristics for solutions before being aggregated.

Figure 5.4 presents the flattened solution archives for each allocation evolution algorithm across task difficulties. Despite using different reference archives for evolving swarm-controller allocations, ASHET and AMHET both appear to produce solutions that occupy similar regions of the behaviour search space. A large portion of this behaviour space is left unexplored for both algorithms, although this is to be expected since SSGA only seeks to optimise for solution quality rather than diversity.

5.1.2.3 Statistical Tests

To effectively compare these algorithms, pairwise statistical t-tests have been performed between ASHET and AMHET using the final generation from all 20 runs of each experiment. These t-test results are presented in Table 5.3, which highlights the significant and insignificant differences between algorithms for each metric.

When comparing ASHET and AMHET, we find that AMHET achieves a significantly greater ($p < 0.05$) archive size and number of unique behaviours in evolved swarms than ASHET across all task environment difficulties. AMHET also outperforms ASHET with a greater QD score across all task difficulties, although not significantly ($p > 0.05$) in the difficult task environment. For maximum fitness results, AMHET only produces significantly fitter ($p < 0.05$) solutions in the easy task environment, while ASHET marginally outperforms AMHET in the medium and difficult environments (although not significantly, with

Table 5.3: Allocation evolution (SSGA vs. MAP-Elites) statistical test results.

Metric	Heterogeneous ASHET vs. AMHET		
	Easy	**Medium**	**Difficult**
Archive Size	AMHET ↑ ✓	AMHET ↑ ✓	AMHET ↑ ✓
Max Fitness	AMHET ↑ ✓	ASHET ↑ ×	ASHET ↑ ×
QD Score	AMHET ↑ ✓	AMHET ↑ ✓	AMHET ↑ ×
Unique Behaviours	AMHET ↑ ✓	AMHET ↑ ✓	AMHET ↑ ✓

Note: Pairwise t-test results are presented for independent samples at generation 200 from 20 runs of each algorithm. Heterogeneous swarm allocations evolved from SSGA-generated reference archives (ASHET) are compared to allocations evolved from MAP-Elites-generated reference archives (AMHET). Comparisons are made across each task environment difficulty and the algorithm with the greater metric value is indicated. Those with a significant difference between values ($p < 0.05$) are indicated with a tick (green), while those with an insignificant difference ($p \geq 0.05$) are indicated with a cross (red).

$p > 0.05$). Overall, it is worth noting that in all cases where there are observed *significant* differences between the algorithms, AMHET produces higher metric results than ASHET.

5.1.3 Algorithm Comparison

Having already reviewed the behaviour and allocation evolution experiments separately in the previous sections, this section continues to compare the results between evolutionary methods and between all experiments. We start by evaluating the performance differences between the *behaviour* evolution algorithms and the *allocation* evolution algorithms. Thereafter, we assess the performance of all algorithms together by ranking their relative metric scores.

5.1.3.1 Behaviour vs. Allocation Evolution

There are two meaningful comparisons to be made between the behaviour evolution algorithms and the allocation evolution algorithms. In the first case, it makes sense to assess the differences between the *heterogeneous* behaviour evolution algorithms (SHET and MHET) and the allocation evolution algorithms (ASHET and AMHET). This pairing allows us to determine which evolutionary method (i.e. behaviour evolution or allocation evolution) performs better in generating effective heterogeneous swarms. In the second case, we can evaluate the differences between the *homogeneous* behaviour evolution algorithms (SHOM and MHOM) and the *heterogeneous* allocation evolution algorithms (ASHET and

Table 5.4: Heterogeneous behaviour evolution vs. heterogeneous allocation evolution statistical test results.

Metric	SSGA SHET vs. ASHET			MAP-Elites MHET vs. AMHET		
	Easy	**Medium**	**Difficult**	**Easy**	**Medium**	**Difficult**
Archive Size	SHET ↑ ✓	SHET ↑ ✓	SHET ↑ ✓	MHET ↑ ✓	MHET ↑ ✓	MHET ↑ ✓
Max Fitness	ASHET ↑ ✓	ASHET ↑ ✓	ASHET ↑ ✓	AMHET ↑ ✓	AMHET ↑ ✓	AMHET ↑ ✓
QD Score	ASHET ↑ ✓	ASHET ↑ ✓	ASHET ↑ ✓	AMHET ↑ ✓	MHET ↑ ✓	MHET ↑ ✓

Note: Pairwise t-test results are presented for independent samples at generation 200 from 20 runs of each algorithm. Heterogeneous swarms evolved directly via behaviour evolution are compared to those evolved via allocation evolution (i.e. SHET vs. ASHET and MHET vs. AMHET). Comparisons are made across each task environment difficulty and the algorithm with the greater metric value is indicated. Those with a significant difference between values ($p < 0.05$) are indicated with a tick (green).

AMHET). Since ASHET and AMHET evolve swarm allocations of controllers previously evolved by SHOM and MHOM, respectively, this comparison allows us to deduce whether optimised allocations of different controllers (i.e. heterogeneous swarms) perform better than the same controllers in isolation (i.e. homogeneous swarms).

Table 5.4 presents the pairwise t-test results for the comparison between the heterogeneous behaviour evolution experiments and the allocation evolution experiments. It is evident that the behaviour evolution algorithms (SHET and MHET) produce significantly greater ($p < 0.05$) archive sizes than the allocation evolution algorithms across all task environment difficulties. Conversely, in terms of maximum solution fitness, the allocation evolution algorithms significantly outperform ($p < 0.05$) the behaviour evolution algorithms for all task difficulties. For QD score results, ASHET achieves significantly higher ($p < 0.05$) scores than SHET across all task difficulties. On the other hand, AMHET only achieves significantly higher ($p < 0.05$) QD scores than MHET in the easy task environment, while MHET achieves significantly higher ($p < 0.05$) scores in both the medium and difficult task environments.

Table 5.5 presents the pairwise t-test results for the comparison between the homogeneous behaviour evolution experiments and the allocation evolution experiments. We find that the behaviour evolution algorithms (SHOM and MHOM) achieve significantly greater ($p < 0.05$) archive sizes and QD scores than the allocation evolution algorithms (ASHET and AMHET) across all task environment difficulties. However, for maximum fitness results, the allocation evolution algorithms significantly outperform ($p < 0.05$) the behaviour evolution algorithms in all task difficulties (with the exception of AMHET in the difficult environment, which only marginally surpasses MHOM in that case).

Table 5.5: Homogeneous behaviour evolution vs. heterogeneous allocation evolution statistical test results.

Metric	SSGA SHOM vs. ASHET			MAP-Elites MHOM vs. AMHET		
	Easy	**Medium**	**Difficult**	**Easy**	**Medium**	**Difficult**
Archive Size	SHOM ↑ ✓	SHOM ↑ ✓	SHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓
Max Fitness	ASHET ↑ ✓	ASHET ↑ ✓	ASHET ↑ ✓	AMHET ↑ ✓	AMHET ↑ ✓	AMHET ↑ ×
QD Score	SHOM ↑ ✓	SHOM ↑ ✓	SHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓	MHOM ↑ ✓

Note: Pairwise t-test results are presented for independent samples at generation 200 from 20 runs of each algorithm. Homogeneous swarms evolved via behaviour evolution are compared to heterogeneous swarms evolved via allocation evolution (i.e. SHOM vs. ASHET and MHOM vs. AMHET). Comparisons are made across each task environment difficulty and the algorithm with the greater metric value is indicated. Those with a significant difference between values ($p < 0.05$) are indicated with a tick (green), while those with an insignificant difference ($p \geq 0.05$) are indicated with a cross (red).

5.1.3.2 Overall Algorithm Ranking

To provide a comprehensive comparison between evolutionary algorithms, the metric results for all experiments are summarised and presented in Table 5.6. This table displays the mean metric values for each algorithm in its final generation, ordered from highest to lowest, for every task environment difficulty.

Starting with archive size, it is evident that the two MAP-Elites behaviour evolution algorithms (MHOM and MHET) produce the highest results. Across all task difficulties, MHOM generates solution archives that are two to four times the size of its closest competitor, MHET. In second position, MHET similarly outperforms the remaining SSGA-based algorithms. This is to be expected since MAP-Elites uses an archive during the evolutionary process to explicitly retain solutions that are behaviourally unique, while SSGA does not maintain such an archive and rather tends to converge on similar solutions. It is worth noting that, in general, the allocation evolution algorithms (ASHET and AMHET) produce the smallest archives.

Conversely, for maximum fitness results, the allocation evolution algorithms outperform all of the behaviour evolution algorithms. However, in the difficult task environment, MHOM does still achieve similar maximum fitness to the allocation evolution algorithms. At the other end of the spectrum, the heterogeneous behaviour evolution algorithms (SHET and MHET) generate the worst-performing solutions compared to other methods for all task difficulties.

QD score results follow a similar pattern to the archive size results with the MAP-Elites algorithms (MHOM and MHET) generally outperforming other algorithms. MHOM

Table 5.6: Comparative ranking of evolutionary algorithms.

Rank	Archive Size			Max Fitness			QD Score		
	E	M	D	E	M	D	E	M	D
#1	MHOM (130.90)	MHOM (130.85)	MHOM (109.05)	AMHET (1.00)	ASHET (0.68)	ASHET (0.34)	MHOM (52.59)	MHOM (24.44)	MHOM (13.34)
#2	MHET (31.75)	MHET (36.95)	MHET (46.30)	ASHET (0.99)	AMHET (0.65)	AMHET (0.34)	SHOM (10.31)	MHET (5.53)	MHET (5.90)
#3	SHOM (17.85)	SHOM (13.80)	SHOM (18.60)	MHOM (0.91)	SHOM (0.61)	MHOM (0.32)	AMHET (9.16)	SHOM (4.85)	SHOM (2.86)
#4	AMHET (9.70)	SHET (11.95)	SHET (14.50)	SHOM (0.91)	MHOM (0.47)	SHOM (0.30)	ASHET (7.39)	AMHET (4.32)	AMHET (2.61)
#5	SHET (9.40)	AMHET (8.90)	AMHET (12.60)	SHET (0.45)	MHET (0.31)	MHET (0.28)	MHET (5.01)	ASHET (3.32)	ASHET (2.41)
#6	ASHET (8.40)	ASHET (5.95)	ASHET (9.90)	MHET (0.33)	SHET (0.29)	SHET (0.22)	SHET (2.68)	SHET (2.04)	SHET (1.65)

Note: The mean values of archive size, max fitness and QD score at generation 200 are presented for all six evolutionary algorithms, averaged over 20 runs. Results are sorted from highest to lowest value in the easy (E), medium (M) and difficult (D) task environment difficulties. Homogeneous behaviour evolution algorithms (SHOM and MHOM) are highlighted in red, heterogeneous behaviour evolution algorithms (SHET and MHET) in green, and heterogeneous allocation evolution algorithms (ASHET and AMHET) in purple. Algorithms based on MAP-Elites are highlighted in a darker shade and those based on SSGA in a lighter shade.

achieves the highest QD score across all task difficulties, ranging from two to five times the score of the algorithm in second position. SHOM achieves a relatively high QD score in the easy environment, but this drops in the medium and difficult environments where MHET achieves higher scores. SHET unanimously achieves the lowest QD score for all task difficulties.

Overall, it is observed that the allocation evolution algorithms (ASHET and AMHET) produce the highest quality solutions for the collective herding task out of all the algorithms. The homogeneous behaviour evolution algorithms (SHOM and MHOM), which ASHET and AMHET both use for generating their reference archives, just slightly underperform in comparison. Finally, the heterogeneous behaviour evolution algorithms (SHET and MHET) produce the lowest quality solutions. In terms of behavioural diversity, algorithms based on MAP-Elites (MHOM and MHET) produce significantly more diverse solution populations (i.e. higher archive sizes) than those based on SSGA. QD score results also reflect this pattern, although with a modest improvement in ranking for the allocation evolution algorithms due to their comparatively high fitness scores.

5.2 Discussion

This section provides an analysis of the results from our experiments in relation to the stated research questions, as outlined in Section 1.1. Furthermore, the significance of these results alongside previous work is discussed.

5.2.1 Research Question 1

> *Does the use of MAP-Elites for the evolution of a* homogeneous *swarm result in higher task performance compared with SSGA?*

The SHOM (based on SSGA) and MHOM (based on MAP-Elites) algorithms were evaluated as part of the behaviour evolution experiments for generating homogeneous swarms. In these experiments, easy, medium and difficult environments were simulated to assess algorithm performance under increasing task complexity.

Inspecting the fitness trends during evolution for these algorithms (see Figure 5.1c), we observe a similar pattern across task environment difficulties. For both SHOM and MHOM, the final task performance is highest in the easy environment followed by the medium and difficult environments as would be intuitively expected. There is an initial period of rapid improvement in solution quality (steeper for MHOM in the first few generations) before flattening out over the remainder of evolution for all experiments. This is especially evident in the easy environment, where both algorithms reach approximately 80% maximum fitness in the first 100 generations. It is also worth noting that MHOM produces a steadier, strictly increasing change in maximum fitness over time, whereas SHOM experiences both upward and downward jitters around a similar trend line. The smooth increase for MHOM is due to the MAP-Elites algorithm maintaining an archive of elite solutions which are only replaced with fitter solutions. SSGA, on the other hand, maintains a population of solutions in which the fittest solutions have a chance (albeit low chance) of being mutated and replaced with worse solutions between generations.

The difference in final task performance between algorithms demonstrates inconsistent results across task difficulties (see Table 5.1). It is observed that both SHOM and MHOM perform comparably in the easy environment, while SHOM performs significantly better in the medium environment and MHOM performs significantly better in the difficult environment. From these results, there is inconclusive evidence to suggest that MAP-Elites generates higher-performing solutions than SSGA for a homogeneous swarm. However, this is likely a consequence of the behaviour search space for this collective herding task. Assessing the final solution archives produced by MHOM (see Figure 5.2), it appears that there is only one region of the explored behaviour space where the best solutions are found (as opposed to multiple, discrete regions) across task difficulties. Therefore, although the pressure for increased diversity in MAP-Elites leads to faster initial gains in task performance, SSGA is also able to evolve similarly high-performing solutions over time without the problem of premature convergence on a local (rather than global) maximum. Despite previous research having discussed the challenging complexity of search spaces for

shepherding and swarm control tasks in general [56, 84, 86], our experiments simulate a simplified variation of the task with no obstacles in the environment, a central target zone which can be entered from any direction, and an objective of capturing each sheep only once rather than also retaining them inside the target zone. As such, it is hypothesised that MAP-Elites will provide performance benefits in more elaborate task environments where high-performing solutions can be found in separate regions of the behaviour space.

5.2.2 Research Question 2

> *Does the use of MAP-Elites for the evolution of a* heterogeneous *swarm result in higher task performance compared with SSGA?*

Heterogeneous swarms were generated as part of both the behaviour evolution experiments (via SHET and MHET) and the allocation evolution experiments (via ASHET and AMHET). SSGA is used to evolve robot controllers in the SHET and ASHET algorithms, while MAP-Elites is used in the MHET and AMHET algorithms. Easy, medium and difficult environments were simulated to assess algorithm performance under increasing task complexity.

Fitness trends for the behaviour evolution experiments (see Figure 5.1d) display a slightly different pattern between algorithms. SHET (using SSGA) demonstrates a slow, almost linear, increase in maximum fitness over 200 generations, which is more noticeable in the easy environment. However, despite these results, it is likely that SHET is in fact following a gradual logarithmic growth curve which has not yet converged by the final simulated generation. Conversely, for MHET (using MAP-Elites), the maximum fitness increases quickly in the first few generations before plateauing for the remainder of evolution. The greatest task performance is achieved in the easy environment for both algorithms, followed by the medium and difficult environments as expected. Overall solution fitness amongst these algorithms is underwhelming with none exceeding 50% maximum fitness by the final generation. However, there does seem to be a trend for MHET to perform better than SHET as task difficulty increases, with MHET significantly outperforming SHET in the difficult environment after worse and comparable performance in the easy and medium environments (see Table 5.1). As with the *homogeneous* behaviour evolution algorithms (see Section 5.2.1), MAP-Elites produces a smooth, increasing fitness curve whereas SSGA produces an irregular curve.

The allocation evolution experiments also exhibit contrasting fitness trends between algorithms (see Figure 5.3b). AMHET follows a logarithmic growth curve (as with MHET), whereas ASHET remains almost entirely flat after some very minor fitness gains in the first few generations. Even though ASHET and AMHET both evolve controller allocations using SSGA, the reference controllers themselves are evolved by either SSGA or MAP-Elites, respectively. These results, therefore, suggest that controllers evolved via MAP-Elites are more suitable for heterogeneous allocation than those evolved via SSGA. This makes sense since the MAP-Elites controllers are more likely to be behaviourally varied than those evolved by SSGA. However, maximum fitness starts much lower for AMHET

compared with ASHET, probably due to the presence of poor-performing controllers in the MAP-Elites archives which were never replaced with fitter controllers. ASHET and AMHET appear to achieve comparable maximum fitness in their final generations, with AMHET only significantly outperforming ASHET in the easy environment (see Table 5.3).

Overall, our results indicate task performance differences between SSGA and MAP-Elites for the evolution of heterogeneous swarms. In the behaviour evolution experiments, MAP-Elites is observed to provide a performance benefit in environments of increasing task complexity. However, in the allocation evolution experiments, controllers generated using SSGA and MAP-Elites seem to provide mostly comparable performance across task environment difficulties.

5.2.3 Research Question 3

> *Does a* heterogeneous *swarm elicit any advantage over standard* homogeneous *swarm-behaviour evolution for a collective herding task, given increasing task complexity?*

To produce heterogeneous swarms, the SHET and MHET algorithms were applied in our behaviour evolution experiments and the ASHET and AMHET algorithms were applied in our allocation evolution experiments. Homogeneous swarms were evolved via the SHOM and MHOM algorithms as part of the behaviour evolution experiments. As previously mentioned, algorithm performance was assessed under increasing task complexity in easy, medium and difficult environments.

Starting with the behaviour evolution experiments, we observe considerably different fitness trends between homogeneous swarms (SHOM and MHOM) and heterogeneous swarms (SHET and MHET) over the course of evolution (see Figure 5.1c and 5.1d). Although both approaches follow the same expected pattern of performing best in the easy environment followed by the medium and difficult environments, the homogeneous swarms achieve significantly greater maximum fitness results than the heterogeneous swarms across all difficulties (see Table 5.2). Archive size and QD score results present a similar assessment, with homogeneous swarms obtaining significantly higher values than heterogeneous swarms in all task environments (see Table 5.2). An important distinction between the algorithms evolving homogeneous swarms (SHOM and MHOM) and those evolving heterogeneous swarms (SHET and MHET) is that the heterogeneous swarms have significantly more ANN connection weights (as part of their behaviour genomes) to optimise than the homogeneous swarms. This is due to SHET and MHET simultaneously evolving multiple controllers for a single swarm, while SHOM and MHOM only evolve one controller per swarm. Consequently, there exists a substantially larger search space for heterogeneous swarm optimisation, which explains the much slower increase in fitness observed during evolution when compared with homogeneous swarms (see Figure 5.1d). This is also evidenced by the final solution archives which reflect the limited exploration of the behaviour search space for SHET and MHET (see Figure 5.2).

For the allocation evolution experiments, it is important to note that the algorithms evolving heterogeneous swarms (ASHET and AMHET) are seeded with the final controller solutions produced by the algorithms evolving homogeneous swarms (SHOM and MHOM). In this way, ASHET and AMHET represent extensions of SHOM and MHOM, respectively, which seek to further optimise pre-generated solutions by combining them in novel swarm allocations. It therefore makes sense that when comparing fitness trends from SHOM and MHOM (see Figure 5.1c) with those from ASHET and AMHET (see Figure 5.3b), we see that the heterogeneous swarms start evolution with already high-performing solutions whereas the homogeneous swarms all begin with near zero fitness. Despite ASHET and AMHET not modifying the controllers generated by SHOM and MHOM, both produce heterogeneous swarms with significantly greater task performance than their homogeneous counterparts (see Table 5.5), with the exception of AMHET in the difficult environment (which still performs comparably to MHOM in that case). Final solution archives produced by ASHET and AMHET (see Figure 5.4) demonstrate minimal exploration of the behaviour space in comparison with those produced by SHOM and MHOM (see Figure 5.2). This is reflected in the archive size and QD score results, which are significantly lower across task difficulties for ASHET and AMHET (see Table 5.5). The decreased diversity of solutions is likely due to both allocation algorithms being based on SSGA which tends to converge on similar high-performing individuals (i.e. fitness optimisation rather than diversity optimisation). Furthermore, it is possible for heterogeneous swarms to demonstrate contrasting individual behaviours that might cancel out when averaged across the swarm for the measurement of behavioural characteristics. Altogether, these results suggest that heterogeneous swarm allocations of controllers can indeed achieve better performance than separate homogeneous swarms using the same controllers, despite lower behavioural diversity in the population.

To summarise in answering the research question, there is evidence suggesting that heterogeneous swarms can provide a task performance benefit over homogeneous swarms. However, only heterogeneous swarms produced via allocation evolution demonstrate this advantage, with those produced via behaviour evolution most likely requiring far longer convergence time to achieve high fitness. There also seems to be a sacrifice to population diversity experienced with heterogeneous swarm evolution (i.e. less variation in behavioural characteristics between heterogeneous swarms compared with homogeneous swarms), but this overlooks the fact that *individual* heterogeneous swarms are inherently more functionally diverse (and likely more adaptive to environment changes) than homogeneous swarms.

5.2.4 Research Question 4

> *When evolving a* heterogeneous *swarm for a collective herding task with increasing task complexity, does a multi-step evolutionary process that optimises the allocation of pre-evolved controllers to robots produce higher-performing swarms than the single-step direct evolution of a diverse swarm?*

The ASHET and AMHET algorithms were applied in our allocation evolution experiments to generate heterogeneous swarms by optimising the allocation of pre-evolved controllers to

robots. Heterogeneous swarms were also generated by the SHET and MHET algorithms in our behaviour evolution experiments, but instead through the single-step direct evolution of multiple controllers. Easy, medium and difficult environments were simulated to assess algorithm performance under increasing task complexity.

As discussed for the previous research question, SHET and MHET demonstrate a gradually increasing trend in maximum fitness during evolution starting from near zero (see Figure 5.1d) while ASHET and AMHET also follow a slight increase in maximum fitness (more so for AMHET than ASHET) but from a higher initial value (see Figure 5.3b). As a result, ASHET and AMHET both achieve significantly better task performance in their final generations than SHET and MHET, across all task difficulties (see Table 5.4). Despite their relatively high solution quality, it is evident from the final solution archives that the allocation evolution algorithms (see Figure 5.4) explore a substantially lower proportion of the behaviour space than the behaviour evolution algorithms (see Figure 5.2). This is also exhibited in the archive size results, which are significantly lower for ASHET and AMHET when compared with SHET and MHET in all environments (see Table 5.4). However, even with lower population diversity, the QD score results (which provide a combined measure of both solution fitness and diversity) still favour the allocation evolution algorithms over the behaviour evolution algorithms. This is shown in Table 5.4 where the allocation evolution algorithms achieve significantly higher QD scores across all task difficulties, highlighting their considerably greater solution quality (even outweighing their lower solution diversity) when compared with the behaviour evolution algorithms.

Overall, our results strongly support the hypothesis that multi-step evolution of heterogeneous swarms via the allocation of pre-generated controllers produces significantly greater task performance than single-step evolution of multiple controllers. Even with lower population diversity, the combined QD score for allocation evolution algorithms still exceeds that of behaviour evolution algorithms.

5.2.5 Previous Work

To assess the significance of this research in relation to prior work, we will review the methods and results alongside those presented in the 2018 paper by Hart, Steyven and Paechter [61] which introduced the EDQD algorithm as well as the 2020 paper by Bossens and Tarapore [15] which introduced the QED algorithm. These two studies, inter alia, provided the primary motivation for our research questions.

The EDQD study [61] focused on applying a novel decentralised algorithm to evolving heterogeneous swarms for a static token gathering task. This is in contrast to the centralised approach used for our evolutionary algorithms, and the dynamic environment simulated for our collective herding task. Furthermore, despite evolving much larger individual swarms consisting of 200 robots (compared with our 10 to 20 robots per swarm), EDQD employs a smaller MAP-Elites archive size of 225 bins (compared with our 729 bins). In this way, our method enables a more comprehensive exploration of the solution space to generate a library of possible behaviours for later use in our allocation evolution experiments. This

concept of a "behaviour library" was alluded to as an avenue for future work in the EDQD paper.

A significant difference in the results between EDQD and our allocation evolution algorithm is the observed trend in the number of unique behaviours within the swarm over time. In our experiments, the number of unique behaviours tends to decrease over time during allocation optimisation (see Figure 5.3d), while in EDQD, it shows an increasing trend. This contrasting behaviour indicates potentially divergent mechanisms at play in generating behavioural diversity within a swarm. Additionally, while both studies aim to reduce the computational cost of generating behavioural diversity, there are differences in the evaluation strategies employed. EDQD utilises 200 robots in the population (as well as 200 robots in each simulated swarm) evolved over 1000 generations, whereas our approach achieves convergence on high performance results with even fewer evaluations, using a population of 100 robots (simulated in swarms of 10 to 20 robots) evolved over 200 generations. This distinction showcases our success in optimising computational efficiency while maintaining effectiveness in generating diverse behaviours. Importantly, our research also supports the findings of Hart *et al.*, as both studies provide evidence that behavioural diversity can be generated without specific speciation mechanisms or geographical isolation in the task environment.

The QED study [15] investigated the use of a novel centralised algorithm to evolve homogeneous swarms for multiple collective behaviour tasks, including aggregation, dispersion and flocking. Archives of diverse behaviours based on six *environmental* characteristics (as opposed to the three *behavioural* characteristics used in our approach) were generated to assist in adapting robots for fault recovery. Although similar sized swarms each consisting of 10 robots were simulated (in line with our 10 to 20 robots per swarm), QED employed a significantly larger archive size of 4096 bins and conducted evolution over 30,000 generations (compared with our 729 bins and 200 generations). Given that the attributes of their simulated environments could be freely manipulated, the entire environment search space could be systematically explored for QED, unlike our behaviour search space requiring the nondeterministic discovery of new behaviours through mutation and recombination. It is also worth noting that the use of behaviour archives for later adaptation in QED is similar in nature to our approach of pre-evolving controllers for later allocation.

Results indicate that maximal coverage of the solution archives was achieved for QED within approximately 10,000 generations. This is in contrast to our algorithms which were unable to reach major coverage of the behaviour space. However, not only could the shortened evolutionary period for our experiments (200 generations) be a factor behind this outcome, but also the use of behaviour descriptors for archive dimensions rather than environment descriptors makes it more challenging to effectively explore the search space, as previously mentioned. Furthermore, the use of archived behaviours for robot adaptation in QED demonstrated high robustness to induced faults, which is congruent with the improved task performance achieved for swarm allocations of archived behaviours in our experiments.

In summary, there are some marked similarities and differences between our evolution-

ary approach and those implemented for EDQD and QED. While EDQD evolves heterogeneous swarms via a decentralised algorithm and QED evolves homogeneous swarms via a centralised algorithm, our research makes use of centralised algorithms to generate solution archives for both homogeneous and heterogeneous swarms. Furthermore, our use of behavioural characteristics to define the search space is similar to that of EDQD, but unlike that of QED which instead uses environmental characteristics. Also, although our experiments are all conducted in the same general environment set-up (similar to EDQD), there is some degree of alternative environment exploration (similar to QED) in that we simulate a dynamic environment under three levels of task complexity. Finally, comparing experimental results, we note that our research supports the findings of the EDQD study and extends the evidence presented by the QED study, demonstrating that an archive of diverse behaviours can enhance performance and adaptation in dynamic environments.

Overall, the main contribution of our research is the introduction of a novel evolutionary approach for generating heterogeneous swarm allocations of pre-evolved, behaviourally diverse robot controllers. This method has been demonstrated to achieve significantly improved task performance when compared with the same controllers in isolation (i.e. separate homogeneous swarms), substantiating that there are ways to optimise how a library of behaviours is utilised in practice. We have also showed this allocation evolution strategy to be significantly more effective than the direct evolution of heterogeneous swarms.

Chapter 6

Conclusion

In this thesis, we have investigated the impact of behavioural diversity in the evolution of multi-agent systems robust to dynamic environments. A collective herding task was simulated under varying degrees of complexity for this purpose, requiring evolved robots (or "dogs") to capture a dispersed flock of heuristic agents (or "sheep") in a centrally located target zone. The SSGA (promoting solution quality) and MAP-Elites (promoting solution quality *and* diversity) evolutionary algorithms were alternatively applied to generate both homogeneous and heterogeneous swarms in what were termed our *behaviour* and *allocation* evolution experiments. Connection weights for robot ANN controllers were optimised in the behaviour evolution experiments (using the SHOM, MHOM, SHET and MHET algorithms), while heterogeneous swarm assignments of pre-evolved controllers were optimised in the allocation evolution experiments (using the ASHET and AMHET algorithms).

Our results demonstrate the capability of MAP-Elites to produce a set of solutions with enhanced behavioural diversity compared with a traditional evolutionary approach, namely SSGA. This was achieved in the absence of specific speciation mechanisms or geographical isolation in the task environment, supporting similar findings by Hart, Steyven and Paechter [61]. This alignment strengthens the validity of our research and contributes to the growing body of evidence in support of this notion. Furthermore, we have exhibited significantly improved task performance for heterogeneous swarms generated by our novel allocation evolution approach. This supports the hypothesis that there are benefits to be gained from using behavioural allocation as opposed to the direct evolution of heterogeneous swarms. Finally, we provide some evidence suggesting that increased behavioural diversity emerges in response to increased task difficulty, although further study is still necessary.

The investigation of our research questions has provided several notable insights, as described above. However, specifically the third and fourth research questions have yielded the most significant contribution to the larger research field. Along these lines of enquiry, our results demonstrate the capacity of heterogeneous swarms to outperform separate homogeneous swarms, using identical controllers, when generated by a novel allocation evolutionary method. As such, the introduction of our multi-step approach for evolving swarm-controller allocations (via the ASHET and AMHET algorithms) represents the main

contribution of this work.

In light of this research, there are several recommendations for the direction of future work. Firstly, having already demonstrated the potential for behavioural allocation approaches, it would be valuable to conduct more comparative experiments with similar algorithms evolving heterogeneous swarms (such as EDQD) in different task environments. There should also be a deeper focus on analysing the differences in computational cost and scalability between these algorithms. Secondly, the capabilities of pre-evolved heterogeneous swarms to adapt and perform under changing environmental conditions should also be tested, possibly implementing similar environment generation strategies to those employed by Bossens and Tarapore [15] in their QED study. Lastly, an interesting extension to our allocation evolution approach could involve using QED to generate solution archives based on environmental characteristics rather than using explicitly-defined behavioural characteristics.

References

[1] A. Agapitos and S. Lucas. Learning recursive functions with object oriented genetic programming. In *European Conference on Genetic Programming*, pages 166–177. Springer, 2006.

[2] J. Aitken, S. Veres, A. Shaukat, Y. Gao, E. Cucco, L. Dennis, M. Fisher, J. Kuo, T. Robinson, and P. Mort. Autonomous nuclear waste management. *IEEE Intelligent Systems*, 33(6):47–55, 2018.

[3] Y. Alsultanny and M. Aqel. Pattern recognition using multilayer neural-genetic algorithm. *Neurocomputing*, 51:237–247, 2003.

[4] A. Alvarez, S. Dahlskog, J. Font, and J. Togelius. Empowering quality diversity in dungeon design with interactive constrained map-elites. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

[5] S. Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[6] T. Bäck and H. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[7] G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behaviors. *Artificial Life*, 9(3):255–267, 2003.

[8] W. Banzhaf, P. Nordin, and M. Olmer. Generating adaptive behavior using function regression within genetic programming and a real robot. In *2nd International Conference on Genetic Programming*, pages 35–43. Citeseer, 1997.

[9] H. Barlow. Unsupervised learning. *Neural Computation*, 1(3):295–311, 1989.

[10] L. Bayındır. A probabilistic geometric model of self-organized aggregation in swarm robotic systems. 2012.

[11] L. Bayındır. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016.

[12] L. Bayındır and E. Şahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering and Computer Sciences*, 15(2):115–147, 2007.

[13] R. Beckers, O. Holland, and J. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. *Artificial Life*, 4:181–189, 1994.

[14] J. Bellingham and K. Rajan. Robotics in remote and hostile environments. *Science*, 318(5853):1098–1102, 2007.

[15] D. Bossens and D. Tarapore. Qed: Using quality-environment-diversity to evolve resilient robot swarms. *IEEE Transactions on Evolutionary Computation*, 25(2):346–357, 2020.

[16] D. Bossens and D. Tarapore. Rapidly adapting robot swarms with swarm map-based bayesian optimisation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9848–9854. IEEE, 2021.

[17] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[18] N. Bredeche, E. Haasdijk, and A. Prieto. Embodied evolution in collective robotics: A review. *Frontiers in Robotics and AI*, 5:12, 2018.

[19] N. Bredeche and J. Montanier. Environment-driven embodied evolution in a population of autonomous agents. In *International Conference on Parallel Problem Solving from Nature*, pages 290–299. Springer, 2010.

[20] N. Bredeche, J. Montanier, B. Weel, and E. Haasdijk. Roborobo! a fast robot simulator for swarm and collective robotics. *arXiv preprint arXiv:1304.2888*, 2013.

[21] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

[22] J. Brulé, K. Engel, N. Fung, and I. Julien. Evolving shepherding behavior with genetic programming algorithms. *arXiv preprint arXiv:1603.06141*, 2016.

[23] G. Buason, N. Bergfeldt, and T. Ziemke. Brains, bodies, and beyond: Competitive co-evolution of robot controllers, morphologies and environments. *Genetic Programming and Evolvable Machines*, 6(1):25–51, 2005.

[24] R. Bürger and M. Lynch. Adaptation and extinction in changing environments. *Environmental Stress, Adaptation and Evolution*, pages 209–239, 1997.

[25] G. Burgin. On playing two-person zero-sum games against nonminimax players. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):369–370, 1969.

[26] M. Cadotte, K. Carscadden, and N. Mirotchnick. Beyond species: Functional diversity and the maintenance of ecological processes and services. *Journal of Applied Ecology*, 48(5):1079–1087, 2011.

[27] R. Cardoso and A. Ferrando. A review of agent-based programming for multi-agent systems. *Computers*, 10(2):16, 2021.

[28] L. Cazenille. Qdpy: A python framework for quality-diversity. 2018.

[29] K. Chatzilygeroudis, V. Vassiliades, and J. Mouret. Reset-free trial-and-error learning for robot damage recovery. *Robotics and Autonomous Systems*, 100:236–250, 2018.

[30] A. Christensen, R. O'Grady, and M. Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766, 2009.

[31] C. Coello. An introduction to evolutionary algorithms and their applications. In *International Symposium and School on Advancex Distributed Systems*, pages 425–442. Springer, 2005.

[32] C. Colas, V. Madhavan, J. Huizinga, and J. Clune. Scaling map-elites to deep neuroevolution. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 67–75, 2020.

[33] N. Correll and A. Martinoli. Modeling self-organized aggregation in a swarm of miniature robots. In *IEEE International Conference on Robotics and Automation*, 2007.

[34] A. Cully, J. Clune, D. Tarapore, and J. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[35] K. DeMarco, E. Squires, M. Day, and C. Pippin. Simulating collaborative robots in a massive multi-agent game environment (scrimmage). In *Distributed Autonomous Robotic Systems*, pages 283–297. Springer, 2019.

[36] J. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the 1st International Conference on Simulation of Adaptive Behaviour*, pages 356–365, 1991.

[37] C. Dima, M. Hebert, and A. Stentz. Enabling learning from large datasets: Applying active learning to mobile robotics. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 108–114. IEEE, 2004.

[38] B. Donald, J. Jennings, and D. Rus. Information invariants for distributed manipulation. *International Journal of Robotics Research*, 16(5):673–702, 1997.

[39] M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. Oliveira, and A. Christensen. Evolution of collective behaviors for a real swarm of aquatic surface robots. *PloS ONE*, 11(3):e0151834, 2016.

[40] F. Ducatelle, G. Di Caro, C. Pinciroli, F. Mondada, and L. Gambardella. Communication assisted navigation in robotic swarms: self-organization and cooperation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4981–4988. IEEE, 2011.

[41] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*, volume 53. Springer, 2003.

[42] A. Eiben and J. Smith. From evolutionary computation to the evolution of things. *Nature*, 521(7553):476–482, 2015.

[43] S. Ficici, R. Watson, and J. Pollack. Embodied evolution: A response to challenges in evolutionary robotics. In *Proceedings of the Eighth European Workshop on Learning Robots*, pages 14–22. Citeseer, 1999.

[44] M. Flageat and A. Cully. Fast and stable map-elites in noisy domains using deep grids. *arXiv preprint arXiv:2006.14253*, 2020.

[45] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: From architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[46] L. Fogel. Artificial intelligence through a simulation of evolution. In *Proceedings of the 2nd Cybernetics Science Symposium*, 1965.

[47] L. Fogel, A. Owens, and M. Walsh. Adaption of evolutionary programming to the prediction of solar flares. *National Aeronautics and Space Administration*, 1966.

[48] M. Fontaine, S. Lee, L. Soros, F. de Mesentier Silva, J. Togelius, and A. Hoover. Mapping hearthstone deck spaces through map-elites with sliding boundaries. In *Proceedings of the 2019 Genetic and Evolutionary Computation Conference*, pages 161–169, 2019.

[49] F. Fortin, F. De Rainville, M. Gardner, M. Parizeau, and C. Gagné. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13(1):2171–2175, 2012.

[50] G. Francesca, M. Brambilla, V. Trianni, M. Dorigo, and M. Birattari. Analysing an evolved robotic behaviour using a biological model of collegial decision making. In *International Conference on Simulation of Adaptive Behavior*, pages 381–390. Springer, 2012.

[51] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira. Virtual robot experimentation platform v-rep: A versatile 3d robot simulator. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 51–62. Springer, 2010.

[52] P. Funes and J. Pollack. Evolutionary body building: Adaptive physical designs for robots. *Artificial Life*, 4(4):337–357, 1998.

[53] A. Gaier, A. Asteroth, and J. Mouret. Aerodynamic design exploration through surrogate-assisted illumination. In *Proceedings of the 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 3330, 2017.

[54] T. Galanos, A. Liapis, G. Yannakakis, and R. Koenig. Arch-elites: Quality-diversity for urban design. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 313–314, 2021.

[55] S. Garnier, J. Gautrais, M. Asadpour, C. Jost, and G. Theraulaz. Self-organized aggregation triggers collective decision making in a group of cockroach-like robots. *Adaptive Behavior*, 17(2):109–133, 2009.

[56] A. Gee and H. Abbass. Transparent machine education of neural networks for swarm shepherding using curriculum design. In *Proceedings of the 2019 International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2019.

[57] Y. Gong, W. Chen, Z. Zhan, J. Zhang, Y. Li, Q. Zhang, and J. Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286–300, 2015.

[58] J. Grefenstette. Genetic algorithms and machine learning. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 3–4, 1993.

[59] A. Gutiérrez, A. Campo, F. Monasterio-Huelin, L. Magdalena, and M. Dorigo. Collective decision-making based on social odometry. *Neural Computing and Applications*, 19(6):807–823, 2010.

[60] L. Haldurai, T. Madhubala, and R. Rajalakshmi. A study on genetic algorithm and its applications. *International Journal of Computer Sciences and Engineering*, 4(10):139, 2016.

[61] E. Hart, A. Steyven, and B. Paechter. Evolution of a functionally diverse swarm via a novel decentralised quality-diversity algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 101–108, 2018.

[62] I. Harvey. The microbial genetic algorithm. In *Proceedings of the European Conference on Artificial Life*, pages 126–133. Springer, 2009.

[63] P. Hoehn, T. Tscharntke, J. Tylianakis, and I. Steffan-Dewenter. Functional group diversity of bee pollinators increases crop yield. In *Proceedings of the Royal Society B: Biological Sciences*, volume 275, pages 2283–2291. The Royal Society London, 2008.

[64] S. Höfer, K. Bekris, A. Handa, J.C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, et al. Sim2real in robotics and automation: Applications and challenges. *IEEE transactions on automation science and engineering*, 18(2):398–400, 2021.

[65] J. Holland. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, 9(3):297–314, 1962.

[66] L. Hong and S. Page. Groups of diverse problem solvers can outperform groups of high-ability problem solvers. In *Proceedings of the National Academy of Sciences*, volume 101, pages 16385–16389. National Academy of Sciences, 2004.

[67] G. Hornby, S. Takamura, O. Hanagata, M. Fujita, and J. Pollack. Evolution of controllers from a high-level simulator to a high dof robot. In *International Conference on Evolvable Systems*, pages 80–89. Springer, 2000.

[68] L. Hugues and N. Bredeche. Simbad: An autonomous robot simulation package for education and research. In *International Conference on Simulation of Adaptive Behavior*, pages 831–842. Springer, 2006.

[69] N. Jacobstein, J. Bellingham, and G. Yang. Robotics for space and marine sciences. *Science Robotics*, 2(7):5594, 2017.

[70] A. Jain, J. Mao, and K. Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[71] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer, 1995.

[72] F. Jansson, M. Hartley, M. Hinsch, I. Slavkov, N. Carranza, T. Olsson, R. Dries, J. Grönqvist, A. Marée, and J. Sharpe. Kilombo: A kilobot simulator to enable effective research in swarm robotics. *arXiv preprint arXiv:1511.04285*, 2015.

[73] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[74] Y. Kassahun, M. Edgington, J. Metzen, G. Sommer, and F. Kirchner. A common genetic encoding for both direct and indirect encodings of networks. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 1029–1036, 2007.

[75] Y. Kim, Y. Yoon, and Z. Geem. A comparison study of harmony search and genetic algorithm for the max-cut problem. *Swarm and Evolutionary Computation*, 44:130–135, 2019.

[76] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2149–2154. IEEE, 2004.

[77] J. Koza. *Genetic programming: On the programming of computers by means of natural selection*, volume 1. MIT Press, 1992.

[78] M. Krieger and J. Billeter. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30(1-2):65–84, 2000.

[79] C. Kube and H. Zhang. Collective robotic intelligence. In *Second International Conference on Simulation of Adaptive Behavior*, pages 460–468, 1992.

[80] C. Kube and H. Zhang. Collective robotics: From social insects to robots. *Adaptive Behavior*, 2(2):189–218, 1993.

[81] V. Lavric, P. Iancu, and V. Pleşu. Genetic algorithm optimisation of water consumption and wastewater network topology. *Journal of Cleaner Production*, 13(15):1405–1415, 2005.

[82] W. Lee, J. Hallam, and H. Lund. Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, pages 501–506. IEEE, 1997.

[83] J. Lehman and K.O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

[84] J. Lien and E. Pratt. Interactive planning for shepherd motion. In *AAAI Spring Symposium: Agents that Learn from Human Teachers*, pages 95–102, 2009.

[85] H. Lipson and J. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978, 2000.

[86] N. Long, K. Sammut, D. Sgarioto, M. Garratt, and H. Abbass. A comprehensive review of shepherding as a bio-inspired swarm-robotics guidance approach. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(4):523–537, 2020.

[87] U. Mahlab, J. Shamir, and H. Caulfield. Genetic algorithm for optical pattern recognition. *Optics Letters*, 16(9):648–650, 1991.

[88] C. Mailer, G. Nitschke, and L. Raw. Evolving gaits for damage control in a hexapod robot. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 146–153, 2021.

[89] G. Martin. The effects of cultural diversity in the workplace. *Journal of Diversity Management*, 9(2):89–92, 2014.

[90] A. Martinoli and F. Mondada. Collective and cooperative group behaviours: Biologically inspired experiments in robotics. In *Experimental Robotics*, volume 4, pages 1–10. Springer, 1997.

[91] P. Maxim, W. Spears, and D. Spears. Robotic chain formations. *IFAC Proceedings Volumes*, 42(22):19–24, 2009.

[92] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[93] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt. Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *AAAI Spring Symposium*, pages 72–75. Palo Alto, CA, 2006.

[94] C. Melhuish, O. Holland, and S. Hoddell. Convoying: Using chorusing to form travelling groups of minimal agents. *Robotics and Autonomous Systems*, 28(2-3):207–216, 1999.

[95] O. Michel. Webots: Symbiosis between virtual and real mobile robots. In *Virtual Worlds: First International Conference*, volume 1434, pages 254–263. Paris, France, 1998.

[96] J. Miller. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1135–1142, 1999.

[97] J. Mouret and J. Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[98] J. Mouret and G. Maguire. Quality diversity for multi-task optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 121–129, 2020.

[99] D. Muni, N. Pal, and J. Das. A novel approach to design classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation*, 8(2):183–196, 2004.

[100] H. Nakamura, A. Ishiguro, and Y. Uchilkawa. Evolutionary construction of behavior arbitration mechanisms based on dynamically-rearranging neural networks. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 158–165. IEEE, 2000.

[101] N. Nedjah and L. Junior. Review of methodologies and tasks in swarm robotics towards standardization. *Swarm and Evolutionary Computation*, 50:100565, 2019.

[102] A. Nelson, G. Barlow, and L. Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370, 2009.

[103] G. Nitschke and D. Howard. Autofac: The perpetual robot machine. *IEEE Transactions on Artificial Intelligence*, 3(1):2–10, 2021.

[104] G. Nitschke, M. Schut, and A. Eiben. Evolving behavioral specialization in robot teams to solve a collective construction task. *Swarm and Evolutionary Computation*, 2:25–38, 2012.

[105] S. Nolfi and D. Floreano. Synthesis of autonomous robots through evolution. *Trends in Cognitive Sciences*, 6(1):31–37, 2002.

[106] J. Nordmoen, E. Samuelsen, K. Ellefsen, and K. Glette. Dynamic mutation in map-elites for robotic repertoire generation. In *Artificial Life Conference Proceedings*, pages 598–605. MIT Press, 2018.

[107] S. Nouyan, A. Campo, and M. Dorigo. Path formation in a robot swarm. *Swarm Intelligence*, 2(1):1–23, 2008.

[108] V. Oduguwa and R. Roy. Bi-level optimisation using genetic algorithm. In *Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems*, pages 322–327. IEEE, 2002.

[109] R. O'Grady, A. Christensen, and M. Dorigo. Swarmorph: Multirobot morphogenesis using directional self-assembly. *IEEE Transactions on Robotics*, 25(3):738–743, 2009.

[110] R. O'Grady, R. Groß, A. Christensen, and M. Dorigo. Self-assembly strategies in a group of autonomous mobile robots. *Autonomous Robots*, 28(4):439–455, 2010.

[111] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[112] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee. Pheromone robotics. *Autonomous Robots*, 11(3):319–324, 2001.

[113] K. Petersen, N. Napp, R. Stuart-Smith, D. Rus, and M. Kovac. A review of collective robotic construction. *Science Robotics*, 4(28):8479, 2019.

[114] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, and F. Ducatelle. Argos: A modular, multi-engine simulator for heterogeneous swarm robotics. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5027–5034. IEEE, 2011.

[115] J. Pugh, L. Soros, and K. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, page 40, 2016.

[116] J. Pugh, L. Soros, P. Szerlip, and K. Stanley. Confronting the challenge of quality diversity. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 967–974, 2015.

[117] M. Quinn, L. Smith, G. Mayley, and P. Husbands. Evolving team behaviour for real robots. In *EPSRC/BBSRC International Workshop on Biologically-Inspired Robotics*, volume 2, pages 14–16. Citeseer, 2002.

[118] I. Rechenberg. *Evolutionary strategies: Optimizing technical systems with principles of biological evolution*. PhD thesis, Technical University of Berlin, 1973.

[119] C. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 25–34, 1987.

[120] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3293–3298. IEEE, 2012.

[121] E. Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International Workshop on Swarm Robotics*, pages 10–20. Springer, 2005.

[122] B. Santos-Diez, F. Bellas, A. Faiña, and R. Duro. Lifelong learning by evolution in robotics: Bridging the gap from theory to reality. In *Proceedings of the IEEE International Conference on Evolving and Adaptive Intelligent Systems*, pages 48–53. Citeseer, 2010.

[123] T. Schmickl, C. Möslinger, and K. Crailsheim. Collective perception in a robot swarm. In *International Workshop on Swarm Robotics*, pages 144–157. Springer, 2006.

[124] G. Smith, N. Steele, R. Albrecht, K. Dahal, and J. McDonald. Generational and steady-state genetic algorithms for generator maintenance scheduling problems. In *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, pages 259–263. Springer, 1998.

[125] J. Soares, I. Navarro, and A. Martinoli. The khepera iv mobile robot: Performance evaluation, sensory data and software toolbox. In *Proceedings of the Robot 2015: Second Iberian Robotics Conference*, volume 1, pages 767–781. Springer, 2016.

[126] X. Song, Y. Jiang, S. Tu, Y. Du, and B. Neyshabur. Observational overfitting in reinforcement learning. *arXiv preprint arXiv:1912.02975*, 2019.

[127] W. Spears, D. Spears, J. Hamann, and R. Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2):137–162, 2004.

[128] V. Sperati, V. Trianni, and S. Nolfi. Self-organised path formation in a swarm of robots. *Swarm Intelligence*, 5(2):97–119, 2011.

[129] G. Syswerda. A study of reproduction in generational and steady-state genetic algorithms. In *Foundations of Genetic Algorithms*, volume 1, pages 94–101. Elsevier, 1991.

[130] V. Trianni, R. Groß, T. Labella, E. Şahin, and M. Dorigo. Evolving aggregation behaviors in a swarm of robots. In *European Conference on Artificial Life*, pages 865–874. Springer, 2003.

[131] S. Tzafestas. Neural networks in robot control. In *Artificial Intelligence in Industrial Decision Making, Control and Automation*, pages 327–387. Springer, 1995.

[132] V. Vassiliades, K. Chatzilygeroudis, and J. Mouret. A comparison of illumination algorithms in unbounded spaces. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1578–1581, 2017.

[133] P. Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication*, pages 261–265. IEEE, 2016.

[134] S. Villéger, N. Mason, and D. Mouillot. New multidimensional functional diversity indices for a multifaceted framework in functional ecology. *Ecology*, 89(8):2290–2301, 2008.

[135] R. Watson, S. Ficici, and J. Pollack. Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18, 2002.

[136] S. Whiteson, B. Tanner, M. Taylor, and P. Stone. Protecting against evaluation overfitting in empirical reinforcement learning. In *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 120–127. IEEE, 2011.

[137] D. Whitley. An overview of evolutionary algorithms: Practical issues and common pitfalls. *Information and Software Technology*, 43(14):817–831, 2001.

[138] D. Whitley and J. Kauth. Genitor: A different genetic algorithm. In *Proceedings of the 4th Rocky Mountain Conference on Artificial Intelligence*, 1988.

[139] S. Winter. Natural selection and evolution. In *Allocation, Information and Markets*, pages 214–222. Springer, 1989.

[140] X. Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 8(4):539–567, 1993.

[141] X. Yao. Evolving artificial neural networks. *Proceedings of the Institute of Electrical and Electronics Engineers*, 87(9):1423–1447, 1999.