ECOLE
POLYTECHNIQUE
DE BRUXELLES

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

Automatic modular design of robot swarms
with Map-Elites

Thesis submitted for the award of the degree of
Master of science in Computer Science Engineering

Laurent Colpaert

Director
Professor Mauro Birattari

Supervisor
Garzon Ramos David

Service
Iridia

Academic year
2022 - 2023

ULB UNIVERSITÉ LIBRE DE BRUXELLES

ULB UNIVERSITÉ LIBRE DE BRUXELLES

Exemplaire à apposer sur le mémoire ou travail de fin d'études,
au verso de la première page de couverture.

Fait en deux exemplaires, **Bruxelles, le 26/05/2023**
Signature

Réservé au secrétariat :	**Mémoire réussi***	**OUI**
		NON

CONSULTATION DU MEMOIRE/TRAVAIL DE FIN D'ETUDES

Je soussigné

NOM :
……Colpaert……
……

PRENOM :
……Laurent……
……..

TITRE du travail :
…Automatic modular design of robot swarms with MapElites……

……
……

AUTORISE*

REFUSE*

la consultation du présent mémoire/travail de fin d'études par les utilisateurs des bibliothèques de l'Université libre de Bruxelles.

Si la consultation est autorisée, le soussigné concède par la présente à l'Université libre de Bruxelles, pour toute la durée légale de protection de l'œuvre, une licence gratuite et non exclusive de reproduction et de communication au public de son œuvre précisée ci-dessus, sur supports graphiques ou électroniques, afin d'en permettre la consultation par les utilisateurs des bibliothèques de l'ULB et d'autres institutions dans les limites du prêt inter-bibliothèques.

Formulaire version 04/2015

** Biffer la mention inutile*

** Biffer la mention inutile*

Acknowledgement

I would like to offer my sincere gratitude to my supervisor, David Garzón Ramos, for his precious help throughout this thesis. I would like to thank him for his support and time he has taken for me, as well as for allowing me to discover the world of a researcher.

I would also like to thank my thesis director, Mauro Birattari, for his guidance towards my work and to push this thesis further. Furthermore, I would also like to thanks all the researchers of IRIDIA, especially, Jonas Kuckling and Miquel Kegeleirs, for the countless times they helped me.

Finally, i would like to express my deepest thanks to my family and Adela for their unconditional love, support and encouragement.

Abstract

This thesis explores the application of the Map-Elites algorithm, an illumination algorithm, for generating control software in the field of robotics. The aim is to investigate whether Map-Elites can produce effective control software as part of an automatic modular design method.

Hence, the research involves defining an encoding method and mutation strategy to transform probabilistic finite state machines into binary strings. Furthermore, two search spaces, focusing on robot perception and robot positioning, are utilized to define the exploration space. A comparative analysis is conducted between Map-Elites, the two search spaces, and AutoMoDe-Chocolate, a state-of-the-art automatic modular design method. The results show that Map-Elites can outperform AutoMoDe-Chocolate in terms of control software performance. However, Map-Elites exhibits reduced robustness when transitioning to a pseudo-reality model. Additionally, variations in performance are observed between the two search spaces, with the sensor and actuator-based feature set yielding superior results compared to the positional feature set.

These findings contribute to our understanding of the strengths and limitations of the Map-Elites algorithm and shed light on the impact of different search spaces on control software generation in robotics.

Keywords : swarm robotics, illumination algorithm, Map-Elites, AutoMoDe, E-Puck, pseudo-reality

Abstract

french Cette thèse explore l'application de l'algorithme Map-Elites, un algorithme d'illumination, pour générer des logiciels de contrôle dans le domaine de la robotique. L'objectif est de déterminer si Map-Elites peut produire un logiciel de contrôle efficace dans le cadre d'une méthode de conception modulaire automatique.

La recherche consiste donc à définir une méthode d'encodage et une stratégie de mutation pour transformer les machines probabilistes à états finis en chaînes binaires. De plus, deux espaces de recherche, axés sur la perception et le positionnement du robot, sont utilisés pour définir l'espace d'exploration. Une analyse comparative est menée entre Map-Elites avec les deux espaces de recherche et AutoMoDe-Chocolate, une méthode de conception modulaire automatique de pointe. Les résultats montrent que Map-Elites peut surpasser AutoMoDe-Chocolate en termes de performance du logiciel de contrôle. Cependant, Map-Elites présente une robustesse réduite lors de la transition vers un modèle de pseudo-réalité. En outre, des variations de performance sont observées entre les deux espaces de recherche, l'ensemble de caractéristiques basé sur les capteurs et les actionneurs donnant des résultats supérieurs à l'ensemble de caractéristiques positionnelles.
Ces résultats contribuent à notre compréhension des forces et des limites de l'algorithme Map-Elites et mettent en lumière l'impact des différents espaces de recherche sur la génération de logiciels de contrôle en robotique.
Mot-clés : swarm robotics, algorithm d'illumination, Map-Elites, AutoMoDe, E-Puck, réalité virtuelle

Contents

Chapter 1

Introduction

Robotics [1] is a multidisciplinary field that involves the design, construction, and application of mechanical robots. These robots can range from technical marvels such as the Mars rover Curiosity to more common applications like robotic arms used in manufacturing. However, designing the mechanical structure is only part of the equation. Equipping robots with the ability to perform their intended tasks is just as important.

Some tasks are too complex for a single robot to accomplish alone. Such tasks require parallel computation or numerous actuators that are better handled by a team of robots. This is where swarm robotics [2], a subfield of robotics, comes into play. It involves a decentralized group of robots working locally together to perform a task that no individual robot could accomplish alone. These characteristics lead to interesting properties such as flexibility and resilience, making swarm robotics an area of great interest in robotics research.

One problem that arises with the approach of swarm robotics is the challenge of describing a global behavior for the group by only modifying the local behavior of the robots, which can be seen as a micro-macro problem.

Currently, there is no general methodology for the design of such groups of robots. However, two distinct design approaches can be identified: manual and automatic. In the manual approach, a human expert in swarm robotics is involved in creating the control software through a trial-and-error approach. In the automatic design approach, once the specifications of the mission have been defined, no human expert is needed [3, 4].

Design methods in swarm robotics typically rely on fitness functions to evaluate the performance of a solution for a given task. However, these methods often neglect the behavior of the robots in favor of only the most performing solutions. A more interesting approach would be to observe the behavior of groups of robots either from an external perspective, such as measuring the distance between objects and robots, or from an internal perspective, by examining the sensors of the robots, rather than solely evaluating performance.

Humans can often come up with multiple solutions to a problem, and the same approach could be applied to robots. Illumination algorithms are particularly adept at exploring diverse solutions by mapping the behavior space [5]. Apply-

ing such algorithms to swarm robotics could enable experts to better understand swarm robotics and identify various ways to solve a particular problem such as foraging food to a nest.

However, a critical question arises: Can these illumination algorithms enable the creation of control software that is as efficient and effective as the most widely used and high-performing algorithms in the current state of the art? The key advantage offered by the illumination algorithm is the ability to select and observe various behaviors. The crucial question is whether these behaviors have an impact on the development of high-performing control software for swarm robotics.

1.0.1 Objective of this master thesis

This master's thesis aims to investigate the applicability of the Map-Elites algorithm in creating control software for swarm robots. The primary objective is to evaluate the performance of the solutions generated by Map-Elites and compare them to the state-of-the-art approaches. The second objective is to see if a different behaviour set influence the creation of high-performing control software.

1.0.2 Contributions of the master thesis

During this master thesis, I delivered the following contributions to meet the previous goals:

1. The encoding of finite state machine following the architectures of AutoMoDe-Chocolate.
2. An automatic modular design methods that uses an illumination algorithm to create control software.

Both goals will be described and explained in the following chapters.

Chapter 2

Related work

The following chapter provides an overview of the state of the art in swarm robotics literature and the illumination repertoire-based algorithm literature.

Swarm robotics is an exciting and rapidly evolving field of study that draws inspiration from the collective behaviour of social animals such as ants, bees, and termites [6]. In recent years, researchers have increasingly recognized the potential of swarm robotics to address complex and challenging problems that are difficult for traditional robotics approaches to solve.

The fundamental challenge in robotics is the design problem. Traditional robotics systems rely on a centralized control system, where a single computer or operator is responsible for controlling all the robots. However, this approach is limited in its scalability, flexibility, and adaptability to changing environments. Swarm robotics offers a novel solution to this problem by using decentralized control, where the individual robots in the swarm interact with each other and their environment to achieve a collective goal.

2.1 Swarm Robotics

Swarm robotics is the study of swarm intelligence applied to groups of robots, as well as how to operate and design them [7]. It is characterized by the emergence of complex behaviour, which arises from the interactions that occur between the individual robots within the swarm, as well as between the swarm and its surrounding environment.

The robot swarm have as main properties [7, 8] :

- **Scalability**: the behaviour of each robot in a swarm is primarily influenced by its immediate surroundings, meaning that each robot is only able to perceive and interact with its neighboring robots and local environment. As the size of a swarm increases, the neighborhood of each individual robot within the swarm remains relatively constant. This advantage of swarm robotics systems is what allows them to be scalable, since the behaviour of the swarm can adapt to larger or smaller scales without requiring significant changes to the behaviour of individual robots or the system as a whole.

- **Fault tolerance** : the loss of an individual robot or part of the swarm should not cause the whole system to fail. This is achieved through the use of decentralized control and redundancies in the system. The decentralized nature of swarm robotics means that there is no single point of failure that can bring down the entire system, since each individual robot within the swarm is responsible for its own behaviour and interactions with the environment. Additionally, redundancies in the system provide backup mechanisms to ensure that the swarm can continue to operate even in the event of a failure or malfunction of one or more individual robots.

- **Flexibility**: the ability of swarm robotics systems to operate effectively across different environments and under varying working conditions is largely due to their decentralized nature and local sensing capabilities. Since each individual robot within the swarm is responsible for its own behaviour and interactions with the environment, the swarm as a whole is able to adapt to different conditions.

Swarm robotics has many potential applications in areas such as search and rescue, environmental monitoring, and agriculture. For example, swarms of robots could be used to explore and map disaster zones or to monitor the health of crops in a field [9, 10].

2.2 Automatic Design

The main difficulty of swarm robotics is to create the control software of individual robots to obtain the global behaviour desired.

While manual design method can be used to create control software for swarm robotics systems, it typically requires an expert to design the behaviour of the robots. This approach often involves a trial-and-error process to achieve the desired results, and the resulting behaviour of the swarm may only be effective for a specific mission or set of conditions [4, 11]. Despite these limitations, manual design can still be a valuable approach in certain contexts, particularly when the requirements of the mission are well-defined and the behaviour of the swarm can be accurately predicted. However, as the complexity and variability of the environment increase, manual design becomes less practical and scalable, which has led to the development of more automated and adaptive approaches to swarm robotics control.

Automatic design is a promising approach to swarm robotics in which the design problem is framed as an optimization problem. Algorithms search through the space of control software in order to find the most effective solutions, based on a specified fitness metric to evaluate the performance of the control software. This approach can be highly efficient, as it allows for the exploration of large and complex design spaces that would be impractical to search manually.

2.2.1 Online and Offline Design

Automatic design is divided into two categories :

- **Online design** is a dynamic approach to swarm robotics in which optimization of the swarm's behaviour is carried out in real time. This allows for a high degree of adaptability to changes in the environment, as the swarm can update its behaviour rapidly in response to new information. However, one potential drawback of this approach is that the swarm requires time to optimize its behaviour, and during the initial deployment phase, there may be some instances of damaging behaviour as the system adjusts to its new environment [12].

  Pugh and Martinoli [13] designed an online method to create control software using particle swarm optimization for an obstacle avoidance mission.

- **Offline design** involves creating control software for a swarm before deploying it in a real environment. This can be done either by optimizing the software with real robots or through simulation. Simulation is often preferred as it is less costly. However, the offline design method is susceptible to the reality gap, which occurs when the optimization process fits the simulation too closely and leads to poor performance in the real world [8].

  Francesca *et al* [14] designed a control software for an aggregation and a foraging mission using modular design method. The control software, AutoMoDe-Vanilla, is the first method of the AutoMoDe family. The idea is to reduce the reality gap by injecting bias into the design process by using pre-existing modules.

In this thesis, we will focus on an offline design method.

2.2.2 Evolutionary Swarm Robotics

Evolutionary robotics is a type of automatic design method that relies on the principles of natural selection first proposed by Darwin [15, 16]. This approach uses a genetic algorithm to identify the optimal set of parameters for a candidate robot controller, often represented by a neural network. When this algorithm is used in conjunction with neural networks, it is referred to as neuro-evolutionary [17].

The use of evolutionary robotics for swarm design has been shown to be effective in optimizing control software. However, a major challenge in this approach is the difficulty of transferring the solutions obtained in simulation to the real world, due to the high variance in the parameters optimized by the neural networks. This high variance can lead to poor robustness of neuro-evolution when the simulated solution is transferred to the real world, which is a common challenge in evolutionary swarm robotics [18].

Francesca *et al* [14] introduce EvoStick, an automatic design algorithm with evolutionary robotics.

2.2.3 Automatic Modular Design

To mitigate the reality gap associated with evolutionary methods, one approach is to use pre-designed modules in the optimization algorithm [19]. These modules can be combined into a high-level architecture, such as a probabilistic finite

state machine or other methods, to form the control software. Automatic modular design is an offline method that is fully automatic.

The first challenge of this approach is to generate the modules themselves, as the number and properties of the modules directly affect the swarm's capabilities. The second challenge is to determine the best way to combine the modules, which has a significant impact on the overall performance of the swarm.

AutoMoDe

AutoMoDe is the family of automatic modular design methods proposed by Francesca *et al* [14]. The version of AutoMoDe that will be mentioned and used in this thesis are the following :

- **AutoMoDe-Vanilla** : Francesca *et al* [14] designed the first flavour of AutoMoDe where it designed a control software for an aggregation and a foraging mission as said previously. The module are manually predefined and divided in two types : the behaviour and transition modules. The behaviour modules represent actions, they are described in Tables 2.1. And the transition modules allows to jump from one behaviour to another when a certain condition is met, described in Tables 2.2. Vanilla selects, combines and fine-tunes the pre-defined modules into control software in the form of a probabilistic finite-state machine (pfsm). The states of the pfsm are the behaviour modules and the edges of the pfsm are the transition modules.

  F-Race is the algorithm used by AutoMoDe-Vanilla to optimise the pfsm, it works by evaluating solutions in parallel on several instances and at each instance, the statistically worse solutions are discarded. F-Race will converge to a set of elite solutions [20].

  In Francesca *et al*, the authors compared Vanilla to human experts. They had to create a control software for a swarm of e-puck robots. The human outperformed Vanilla when constrained to using the same modules, but lost when they were free to code the control software by them self. The only difference between Vanilla and the human is the way the solution are searched so the authors conjectured that using a better optimization algorithm could improve Vanilla, this lead to AutoMoDe-Chocolate.

- **AutoMoDe-Chocolate** : Francesca *et al* [21] proposed an upgraded version of the Vanilla version of AutoMoDe. The F-Race algorithm is replaced by iterated F-Race, also called irace [22]. In all the other aspect, Chocolate is identical to Vanilla.

  The result obtained by Chocolate outperformed the human expert when they had to use the same module as Chocolate.

- **AutoMoDe-Demonstration-Cho** : Gharbi *et al* [23] proposed an automatic design method that combines inverse reinforcement learning with automatic modular design of control software for robot swarms. The algorithm is provided with a demonstration of how to perform. For example, if the mission is to aggregate on a white spot. We provide an example where

all the robots are on the white spot. The algorithm will then use inverse reinforcement learning to infer the fitness function to optimize the control software.

The result obtained are similar with AutoMoDe-Chocolate but it's important to note that no fitness function was given prior to the experience for the AutoMoDe-Demonstration-Cho version.

For an in-depth introduction to the different flavour of AutoMoDe refer to: for AutoMoDe-Gianduja [24], for AutoMoDe-Waffle [25], for AutoMoDe-Maple [26], for AutoMoDe-Mate [27], for AutoMoDe-TuttiFrutti [28], for AutoMoDe-Coconut [29], for AutoMoDe-Icepop [30], for AutoMoDe-Cedrata [31] and for AutoMoDe-Arlequin [32, 33] and finally AutoMoDe-Nata [33]

Behaviour	Parameter(s)	Description
Exploration	$\tau \in \{1, 2, ..., 100\}$	The robot moves straight. If $\text{prox}_i \geq 0.1$ for $i \in \{1, 2, 7, 8\}$, the robot turns on iself for random number of cycles chosen in $\{0, 1, ..., \tau\}$.
Stop	None	The robot stays still.
Phototaxis	k fixed to 5	The robot moves straight to light source if perceived; otherwise, moves straight. Obstacle avoidance is embedded and depends on k.
Anti-phototaxis	k fixed to 5	The robot moves away from light source if perceived; otherwise, moves straight. Obstacle avoidance is embedded and depends on k.
Attraction	$\alpha \in$ [1,5] and k fixed to 5	The robot moves straight to the neighbour robot thanks to the rang-and-bearing device; otherwise, move straight. Obstacle avoidance is embedded and depends on k and α.
Repulsion	k fixed to 5	The robot moves away from neighbour robots; otherwise, moves straight. Obstacle avoidance is embedded and depends on k.

Table 2.1: This table represents the set of behaviour modules used in AutoMoDe-Vanilla and AutoMoDe-Chocolate [14, 21]. These behaviour modules are the state in the probabilistic finite state machine representing the behaviour of each robot in the swarm. The behaviour modules were conceived for the E-puck's reference model RM1.1.

Condition	Parameter(s)	Description
Black-floor	$\beta \in [0,1]$	If $\text{gnd}_i = 0$ for $i \in \{1,2,3\}$, the transition is enable with probability β.
Grey-floor	$\beta \in [0,1]$	If $\text{gnd}_i = 0.5$ for $i \in \{1,2,3\}$, the transition is enable with probability β.
White-floor	$\beta \in [0,1]$	If $\text{gnd}_i = 1$ for $i \in \{1,2,3\}$, the transition is enable with probability β.
Neighbor-count	$\eta \in [0,20]$ and $\zeta \in \{0,1,...,10\}$	The transition is enable with probability z(n) = $\frac{1}{1+e^{\eta(\zeta-\eta)}}$, where n is the number of neighbouring robots.
Inverted-Neighbor-count	$\eta \in [0,20]$ and $\zeta \in \{0,1,...,10\}$	The transition is enable with probability $1 - z(n)$.
Fixed-probability	$\beta \in [0,1]$	The transition is enable with a fixed probability of β.

Table 2.2: This table represents the set of transition condition modules used in AutoMoDe-Vanilla and AutoMoDe-Chocolate [14, 21]. These transition condition modules are the transition conditions in the finite state machine representing the condition of transition of the behaviour of each robot in the swarm. The transition condition modules were conceived for the e-puck's reference model RM1.1

2.3 AutoMoDe-Demonstration-Cho

As explain earlier, AutoMoDe-Demonstration-Cho proposed by Gharbi *et al* [23] is an automatic design method that combines inverse reinforcement learning with automatic modular design of control software for swarm robotics.

To infer the fitness function from the demonstration, the inverse reinforcement learning algorithm, called apprenticeship learning is used [34].

In inverse reinforcement learning, the reward function R is not provided as opposed to reinforcement learning algorithm which learns policies π that maximize the reward function. But it is assumed that a 'true' reward function R^* exists and it is such that the policy π^* that maximizes the value function based on R^* would generate the given demonstrations. Furthermore, in apprenticeship learning, it is assumed that a mapping between the state of the system to a k-dimensional vector of features. The 'true' reward function R^* is assumed to be a linear combination of the features.

In order to represent the state of the entire swarm, the feature vector μ must include information related to the positions of the individual robots in relation to specific "landmarks." For example, in the experiments, these landmarks took the form of circular and rectangular floor patches placed within the arena where the

swarm was operating. In addition to this positional information, Gharbi *et al* also included data about the density of the swarm, such as the distance to the nearest neighboring robot.

All the distances are normalized with the help of an exponential. It has two impact on the features. It penalize more the robot that are the farthest as well as limit the distance between [0,1]. It is define as follows :

$$\nu = e^{-\frac{2\ln 10}{d}x} \tag{2.1}$$

where ν is the normalized distance, x is the distance and d is the diameter of the inner circle of the arena were the swarm perform. The function can be seen in the figure 2.1.

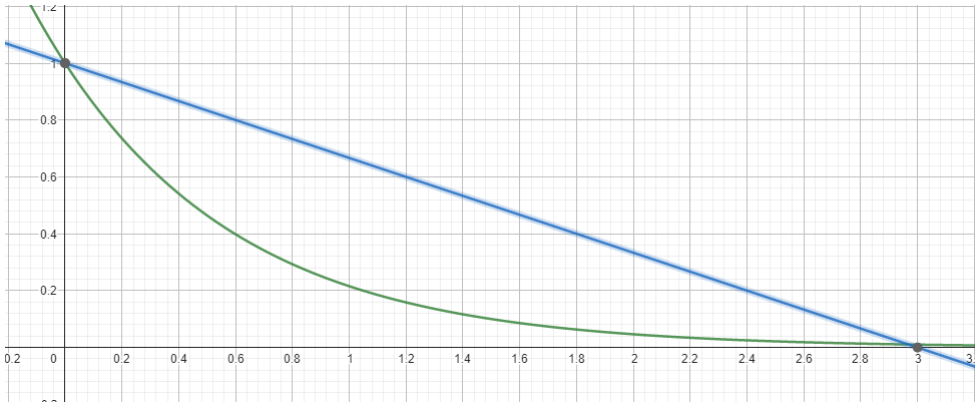


Figure 2.1: From the paper of Gharbi *et al* [23], The x-axis is the distance taken into account. The y-axis is the normalized distance ν. The green curve is the exponential function described in Equation 2.1, and its decreasing speed is such that when the distance is more significant than half of the arena's inner radius (equal to 3 meters in this graph), the value of ν is already below 0.1.

The features vector can be defined as follows :

$$\mu = (\mu_{\text{patch}_{11}}, ..., \mu_{\text{patch}_{1n}}, ..., \mu_{\text{patch}_{kn}}) \tag{2.2}$$

$$\text{s.t.} \quad \mu_{\text{patch}_{ij}} = \begin{cases} 1, & \text{if robot i inside patch j;} \\ 0, & \text{if an obstacle is between robot i and patch j;} \\ e^{-\frac{2\ln 10}{d}x_{\text{patch}_{ij}}} & \text{otherwise;} \forall i = 1, ..., n \text{ and } \forall j = 1, ..., k; \end{cases} \tag{2.3}$$

AutoMoDe-Demonstration-Cho is a promising algorithm but still has a problem which is that it doesn't diversify its strategy which may leads to problem with the reality gap.

2.4 Diversity algorithm

Many local optimization algorithms encounter a common issue, which is getting stuck in local optima. Local optima occur when the algorithm finds a solution that is better than its immediate neighbors but not the best solution in the entire search space. This can lead to premature convergence, where the algorithm

stops making progress towards finding the global optimum. To mitigate this issue, many optimization algorithms employ strategies to encourage diversity in the search process. Despite these efforts, search algorithms may still converge to a limited number of solutions, which can hinder the discovery of novel and more performing solutions [35, 36].

An alternative approach to addressing this issue is to abandon the idea of optimization altogether and instead focus solely on exploring the search space in the hope of finding new and improved solutions. This approach prioritizes exploration over exploitation, with the aim of discovering previously unknown regions of the search space that may contain better solutions.

2.4.1 Optimization vs. Illumination

As said previously, to class of algorithm appear when looking for good and new solutions in a search space:

- **Optimization algorithms** are designed to find the highest-performing solution(s) within a given search space. These algorithms typically aim to identify one or a set of solutions that optimize a specific objective function or a combination of multiple objectives. In multi-objective optimization, the goal is often to find a set of solutions that represent the Pareto front, which is the trade-off curve of optimal solutions that cannot be improved in one objective without sacrificing performance in another [37]. Optimization algorithms use various techniques, such as mutation and crossover operations, to explore the search space and converge towards optimal or near-optimal solutions. The focus of these algorithms is on finding the best solutions according to the defined optimization criteria.

- Illumination algorithms, in contrast to optimization algorithms, aim to identify the best solutions for each point in the feature space, also known as the behavior space. These algorithms generate phenotype-fitness maps that highlight regions of the map with high fitness values, providing a comprehensive view of the fitness landscape [38]. By illuminating areas with high fitness values, illumination algorithms can reveal diverse solutions that may not be found by traditional optimization algorithms.

2.4.2 Evolutionary Algorithm

From the many optimization algorithm, evolutionary algorithms are one of the most successful families of search algorithms.

Definitions

This thesis will use the vocabulary used from evolutionary algorithm. A solution is a *phenotype*. These solutions are described by a *genotype*, and the action performed by the phenotype are the solution's *behaviour*. The performance of a solution can be found by a *fitness function* which compute the *fitness* of the solution.
To produce new solutions, evolutionary algorithm will *mutate* the genome by altering it randomly and to keep the influence of previous solutions, new solutions

can be sampled from two parents solution, this process is called *crossover*.

Evolutionary algorithms are inspired by Darwin's principle of natural selection [35]. They start with a population of solutions and mimic the passage of time by iteratively selecting the most performing solutions, which act as parents to generate a new population through mutation and crossover operations. The intuition is that, over time, only the fittest solutions will survive and propagate in the population. To mitigate the issue of local optima, some approaches propose increasing the mutation rates when the rate of performance improvement stagnates, allowing for more exploration of the search space [39].

Doncieux *et al* [40] demonstrate that the use of evolutionary algorithms is continuously growing and evolving, with ongoing improvements in the field.
Francesca *et al* [41] proposed EvoStick, a neuroevolutionary automatic design method, based on an evolutionary algorithm that have high performance in simulation but has some difficulties to keep the performance when transferring the control software to reality.

2.4.3 Novelty Search

Novelty search is a type of illumination algorithm that aims to explore the feature space comprehensively. Instead of optimizing for a specific fitness objective, it focuses on discovering as many different behaviors as possible according to a distance metric. The distance metric is used to measure the dissimilarity between behaviors. However, one limitation of novelty search is that it may face difficulties when dealing with very large behavioral spaces, as it may require significant computational resources to efficiently explore and evaluate a large number of behaviors [42]. Nonetheless, novelty search has gained attention as an effective approach for promoting diversity and exploration in evolutionary algorithms, and has been applied in various domains to generate interesting and diverse solutions [43].

2.4.4 Multi-dimensional Archive of Phenotypic Elites

In traditional search algorithms, such as hill climbing, simulated annealing or evolutionary algorithms, the algorithm return a set of solutions that represent the best tradeoffs between objectives [5].

In this thesis, the Map-Elites algorithm has been chosen as the diversity-quality algorithm. This algorithm maintains a Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) that maps the performance of individuals onto a set of behavioral descriptors, or dimensions. By doing so, Map-Elites promotes both diversity and quality in the search space, and has been shown to be effective in a range of optimization problems. This algorithm will thus maintain a repertoire of Elites solutions.

The algorithm begins with the user specifying a fitness function, denoted as $f(x)$, which evaluates the performance of a given solution x. The user also selects N

Algorithm 1 MAP-Elites

```
1: procedure MAP_ELITE
2:     P ← Ø                                  ▷ N-dimensional performances array
3:     X ← Ø                                  ▷ N-dimensional solutions array
4:     for iter = 1 → I do
5:         if iter < G then
6:             x′ = random_selection()
7:         else
8:             x = random_selection(X)
9:             x′ = random_variation(x)
10:        end if
11:        b′ = feature_descriptor(x′)
12:        p′ = performance(x′)
13:        if P(b′) = Ø or P(b′) < p′ then
14:            P(b′) = p′
15:            X(b′) = x′
16:        end if
17:    end for
18: end procedure
```

dimensions of interest that define the feature space. Each of these N dimensions is discretized based on user-defined parameters or computational resources.
The algorithm then searches within the search space, which comprises all possible values of x, and evaluates the performance of each solution x using the fitness function $f(x)$. In addition to the fitness function, the user also defines a feature function denoted as $b(x)$, which determines the value of x in each of the N feature dimensions. The output of the feature function $b(x)$ is an array of N dimensions, representing the features of the solution x in the feature space.
The algorithm will initialize an empty N dimensional array P for the performance of the solutions and an empty N dimensional array X for solutions. The algorithm consists of two main steps:

1. **Initialization Phase**: Map-Elites generates G random solutions and determines their performance and features using the fitness function $f(x)$ and the feature function $b(x)$. Each solution is then placed in the corresponding cell in the feature space. It's important to note that multiple solutions may be mapped to the same cell, and only the highest-performing solution is kept in that case.

2. **Illumination Phase**: The algorithm randomly selects a solution from the feature space and applies various variation operations, such as mutation and crossover, to generate a new solution. The performance and features of the new solution are computed using the fitness function $f(x)$ and the feature function $b(x)$, and the new solution is placed in the feature space. This step is repeated until a certain condition is met, such as a specified number of iterations or when a certain portion of the feature space is explored.

The illumination phase is iteratively performed to explore the feature space, generate and evaluate new solutions, and update the cells in the feature space with the best-performing solutions. This allows Map-Elites to efficiently explore and

illuminate regions of the feature space that contain high-performing solutions, leading to the discovery of diverse and optimal solutions.

Furthermore, it's important to note that not all cells in the feature space will be explored due to the inherent constraints of the problem. For instance, in the case of robots where two features are computed based on the distance to landmarks, it's physically impossible for a robot to be in two places at the same time. Therefore, certain regions of the feature space may remain unexplored, and Map-Elites focuses on illuminating regions that are feasible and meaningful within the problem domain.

Mouret *et al* [5] conducted a study where they applied the Map-Elites algorithm to simulated and real soft robots with the objective of discovering diverse and viable designs. They compared Map-Elites with novelty search and random sampling algorithms, and found that Map-Elites outperformed them in terms of exploration in simulation. Map-Elites was able to efficiently explore the feature space and generate a wide range of diverse solutions. Even in the case of limited real-world experiments, Map-Elites demonstrated effectiveness in exploring the feature space and discovering viable designs. This highlights the potential of Map-Elites as a powerful algorithm for promoting diversity and exploration in evolutionary algorithms, and its applicability in various domains including robotics and design optimization.

Measuring Performance

Map-Elite can be evaluated through performance measures.

- **Coverage**: This performance measure calculates the ratio of the number of cells in the feature space that have been explored by Map-Elites to the total number of cells in the feature space. It provides an indication of the algorithm's ability to explore and cover the feature space. However, it's important to note that there may be areas in the feature space that are impossible to explore due to constraints or limitations.

- **Global performance**: This measure is commonly used in optimization algorithms and is computed by dividing the highest-performance solution found by the algorithm so far by the highest performance possible in the entire feature space. It gives an indication of how close the algorithm is to achieving the best possible performance in the feature space.

- **Global reliability**: This measure is calculated as the average performance of all the cells in the feature space, divided by the average of the highest performance obtained by any run of the algorithm across all the cells in the feature space. If a cell has not been explored, its value is replaced by zero. It provides an indication of the algorithm's reliability in consistently achieving good performance across the entire feature space, accounting for unexplored areas as well.

2.5 Repertoires in Swarm Robotics

2.5.1 AutoMoDe-Nata

Hasselmann *et al* [**empty citation**] introduce Nata, a novel modular automatic design method that uses repertoires of behaviors and principled design to automatically generate the modules, with the aim to reduce expert knowledge required in the implementation of automatic design methods for robot swarms.

To automatically generate a vast amount of modules without the need of an expert in swarm robotics, AutoMoDe-Nata create a repertoire of modules with a quality-diversity algorithm. The algorithm chosen is novelty search with local with local competition that uses behavioral novelty of candidate behaviour as the objective function of an optimization algorithm [44, 36].

The behaviour are represented by neural networks that can be used as control software in a robot. The algorithm will generate an empty repertoire which will hold the set of best candidate neural networks. Afterwards, all the neural networks will be evaluated in randomly generated environments by computing their median behaviour and mean quality score. The randomly generated environments all have the same size and shape but may vary in having a light source, an obstacle in the center of the arena or having multiple circular floor patch of random color (white, grey, or black). The quality score represents the number of collisions that occur during the simulation o fa neural network. The behaviour is characterized by a feature vector composed of 14 real values: the mean and standard deviation of the linear and angular speed; the distance to walls/obstacles, to other robots, and to the closest robot; the ambient light and the ground color perceived.

The family of AutoMoDe works with control software in the form of probabilistic finite state machine. The behavioral modules, which are the states, have been generated with the archive, but to make a working pfsm, transition module are needed. Nata automatically generate the transition module via a set of rules that operates on the reference model of the robots 3.3.

AutoMoDe-Nata uses the same optimization algorithm as AutoMoDe-Chocolate, iterated f-race [22], to generate the control software by assembling behaviour and transition modules into probabilistic finite state machine.

Hasselmann has tested his automatic modular design method to create control software of swarm robot on the same set of mission that Francesca *et al* [21] used to test AutoMoDe-Chocolate. AutoMoDe-Nata did not reach the performance level of Chocolate. In contrast to Chocolate, Nata is fully automatic and did not need the help of expert in swarm robotics to create the behavioral and transitional modules. The performance of Nata exceeded the performance of the evolutionary algorithm Evostick [41].

2.5.2 Other Repertoires in Swarm Robotics

Novelty Search in Swarm Robotics

Gomes *et al* [45] proved that quality-diversity algorithm with the use of task-agnostic behaviour and quality metric can be used to create repertoire of control software for swarm robots. They demonstrated the use of such an algorithm for 8 different missions such as aggregation, coverage, ... And for each of those mission, a single repertoire was able to provide good results.

Map-Elites for multi-objective optimization

Engebraaten *et al* [46] proposed to use Map-Elites to generate a repertoire of solution candidates, with the idea of finding the Pareto front for a multi-objective optimization. They tried to combine two behaviours into one control software for robot swarm, the tasks are perimeter surveillance and communication network creation.

Engebraaten demonstrated that it is possible to automatically synthesize swarm controllers for a multi-function swarm system.

Chapter 3

Methodology

This chapter presents various methods utilized in this thesis, including the adapted version of the Map-Elites algorithm, the encoding method employed to convert the finite state machine into binary strings compatible with the Map-Elites algorithm, the description of the feature set utilized, the technical assets employed, and an explanation of the pseudo-reality model.

3.1 Map-Elites

The main algorithm employed in the thesis is the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites). MAP-Elites was chosen for its unique property of ”illumination,” which involves exploring phenotypes and mapping their behaviors in a discretized space known as the behavior space. The algorithm retains only the elite solutions for each point in this search space, thus maintaining a repertoire of elite solutions. This approach allows for a comprehensive exploration of the phenotypic space and the identification of high-performing solutions across multiple dimensions or behaviors, leading to a diverse and robust set of elite solutions.

To relate to the vocabulary of optimization algorithm, a phenotype/solution is a control software for robot swarm. These control software are described by probabilistic finite state machine (pfsm) which are thus called genotype. The fitness function will be the objective function of each missions, they will be described in the section 3.3.3.
The algorithm of MAP-Elites described in the algorithm 1 is a simple version of the algorithm which may not be efficient time-wise. The algorithm is modified to be able to handle batch of solutions that can be parallelized to efficiency. The initialization of the algorithm has also been removed from the main-loop. The modified version of the algorithm can be found in the algorithm 2.

3.1.1 Encoding of the Probabilistic Finite State Machines in a Grammar

AutoMoDe-Chocolate uses pfsm that are encoded as chains of characters written in a language that consists in the succession of variable-value pairs [47].

Algorithm 2 Parallelized MAP-Elites

procedure MAP_ELITE
  $P \leftarrow \emptyset$ ▷ N-dimensional performances array
  $X \leftarrow \emptyset$ ▷ N-dimensional solutions array
  P,X = initialization(G)
  batch = select_population(p)
  x′ = random_variation(batch)
  for iter = 1 → I **do**
    for i = 1 → p **do** ▷ Parallelized loop
      b′ = feature_descriptor(x'_i)
      p′ = performance(x'_i)
      if P(b′) = $\emptyset$ or P(b′) < p′ **then**
        P(b′) = p′
        X(b′) = x'_i
      end if
    end for
    batch = select_population(p)
    x′ = random_variation(batch)
  end for
end procedure

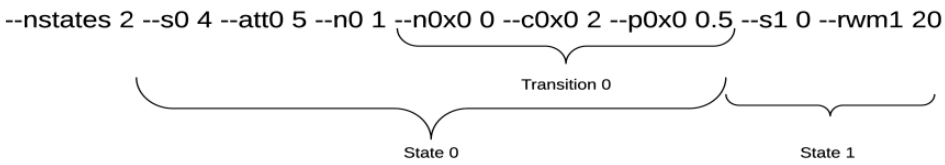
A pfsm start by indicating the number of states S (−−nstates) then it is followed by the description of the behaviour modules. The description of the behaviour module is composed of the encoding of the behaviour itself as well as the encoding of its outgoing transition modules. Each variable of the module consists of a concatenation of a variable identifier with the index b ($b \in$[0,S)) of the module in the pfsm. The encoding of a behaviour module is composed of the behavior identifier (−−sb), its potential parameter (see Table 3.1), and the number of outgoing transitions T_b (−−nb). The encoding of the transitions module of a behaviour module are written as the concatenation of the variable identifier, the index of the behaviour module and the index of the outgoing transition t_b ($t_b \in$ [0,T_b)) withing the behavior module. The index of the behaviour and transition are separated by the character x. For the potential parameters of the outgoing transition, see Table 3.2. An example of such grammar used to create a pfsm can be seen in the figure 3.1a, the grammar has also been transform into a schema to allow a better understanding of the pfsm, see figure 3.1b.

Identifier	Behaviour	Parameter
0	Exploration	–rwm$b$
1	Stop	none
2	Phototaxis	none
3	Anti-Phototaxis	none
4	Attraction	–att$b$
5	Repulsion	–rep$b$

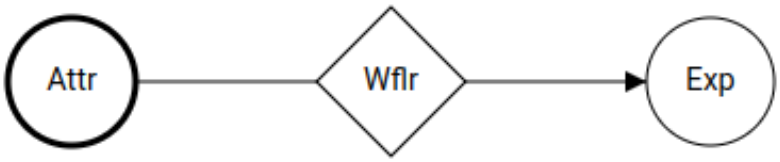
Table 3.1: Identifier and parameters of the behavior modules with b ∈ [0, S), where S is the number of state

Identifier	Condition	Parameter
0	Black floor	–pbxt_b
1	Gray floor	–pbxt_b
2	White floor	–pbxt_b
3	Neighbors-count	–pbxt_b and –pbxt_b
4	Inverted-neighbors-count	–pbxt_b and –pbxt_b
5	Fixed-probability	–pbxt_b

Table 3.2: : Identifier and parameters of the transition modules with b $\in$ [0, S) and tb $\in$ [0, T_b), where S is the number of state and T is the number of transition for the state b.



(a)



(b)

Figure 3.1: (a) The grammar of a pfsm : it has two states and one transition from the state 0 to 1. (b) The pfsm in schema, by reading it, we can see that they are two states (the circles) and one transition that has a conditions. The robot starts with the behaviour *Attr*, Attraction, and once the condition *Wflr* is met, when the robot walks on a white floor, it switches behaviour to *Exp*, an exploration phase where the robot will make random walk.

3.1.2 Encoding of the Probabilistic Finite State Machines in a Binary String

The important step to allow the illumination of the behaviour space in the Map-Elites algorithm is the operation of mutation and cross-over. Mutating a grammar is not a simple operation. To avoid any complication, the grammar will be transformed into a binary string to allow easy mutation and crossover.

To create the binary string, we always try to map the actual pfsm to a pfsm that has the maximum number of states, maximum number of transitions and for each variable, the maximum value possible. By doing so, it is possible to map any pfsm that has the grammar of AutoMoDe-Chocolate to a binary string of length 622. The pfsm represented by a grammar in the figure 3.1a is transformed into the following binary string :

011000000000000000011111010000000000000000000100010000000
000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000100110000000
000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000

In the context of binary strings, mutation refers to the random swapping of '0' and '1' bits, either by changing '0' to '1' or vice versa. .

Once the probabilistic finite state machine (pfsm) representations have been modified through mutation and crossover operations, they can be reconverted into grammar strings. This allows them to be read by the AutoMoDe-Chocolate control software.

3.2 Features

The primary objective of this thesis is to investigate the capability of the Map-Elites algorithm in generating control software. To accomplish this, a feature set is required to define the search space utilized during the illumination and optimization processes. The underlying concept of the feature set is to describe and quantify the behavior of a swarm of robots.

In the literature, two sets of features have been employed and proven effective in their respective domains. These features can be categorized as either external or internal. When speaking about external, it's features that can be obtained by observing the whole swarm, they are features focused on distances [23]. For internal features, they are features that can be obtained by the sensors of the robots [33]. The two sets of features are defined as follows :

- **Sensory features**: Hasselmann *et al* [33] defined a set of 14 features to explain the behaviour of robot swarm. They were used in AutoMoDe-Nata, which used novelty search with local competition to generate behaviour modules, described by neural networks, that are later one assembled with transition modules with the help of the algorithm irace used in AutoMoDe-Chocolate. With this set of features, they were able to create a vast amount of different behaviour module which could then be used to perform various

missions.

This set of feature will be referenced as sensory features because each feature of the set are taken directly from what the robot is able to sense. The robot is the e-puck model RM1.1, and as described in the Table 3.3, and has multiple sensors: a proximity sensor used to sense objects that are closed, a light sensor which detect the amount of light, a ground sensor that detect color and a range-and-bearing sensor to detect the neighboring robots. Hasselmann defined 7 features and for each features, the mean and standard deviation is computed and used in the set of sensory features. Those features are defined as follows:

1. the linear speed,
2. the angular speed,
3. the distance to wall and obstacle,
4. the distance to robots,
5. the distance to the closest robot,
6. the amount of light,
7. the color of the ground.

- **Positional features**: Gharbi *et al* [23] defined a set of features that are related to the position of the robot to each other as well as their positions in the arena. This set is composed of 40 features. These features were used to extract the behaviour of the swarm to reverse engineer the objective function of a mission with the help of the apprenticeship learning algorithm. This algorithm is part of AutoMoDe-Demonstration-Cho which showed that this set of positional features it is possible to reverse engineer the objective function and that can be used to create an automatic modular design of control software for swarm robotics.

  These 40 features are composed of two main features computed for each robot. The main features are defined as follows:

  1. The normalized distance to landmarks, where a landmark is the center of a floor patch which can either be circular or rectangular.
  2. The normalized distance to the closest robot which, in this case, is not computed by using the range-and-bearing sensor of the e-puck but externally.

  These set of features is only influenced by the position of the robot in the arena. Therefore, even if another type of robot would be used, this set of features could still be applied because there is no need to access the sensor values.

  The positional features are modified in this thesis to keep a consistency with the sensory features. For each feature, the mean and standard deviation will be kept. It means that the positional feature set has a size of 80.

Due to the broader range of missions employed in this thesis compared to the study by Gharbi et al., the feature "distance to a landmark" is duplicated for each patch floor present on the arena. The number of patch floors can range from 1 to 3, resulting in a variation in the size of the positional feature set from 80 to 160.

3.3 Technical assets

3.3.1 E-puck

Francesco *et al* [48] of the EPFL developed the e-puck. A small differential-drive robots created with the idea of becoming the ideal tool for research and education. It is equipped with several sensors and actuators. It has three ground sensors to read the gray-scale color of the ground. It has eight infrared transceivers positioned to be able to detect the presence of surrounding obstacles and/or measure the intensity of the ambient light.

The model used in this thesis has been enhanced by a range-and-bearing which allows the robot the sense the neighboring robots, an omnidirectional camera as well as an overo gumstix board, the e-puck can be seen in figure 3.2 [49].
AutoMoDe use a reference model that formalises what the robot can do and sense, it is the reference model RM1.1 given in Table 3.3. The range-and-bearing device provides the attraction vector defined as follows:

$$V = \begin{cases} \sum_{m=1}^{n} (\frac{1}{1+r_m}, \angle b_m), & \text{if robots are perceived} \\ (1, \angle 0), & \text{otherwise} \end{cases} \tag{3.1}$$

where range $r_m \in$ [0,0.7]m and the bearing $\angle b \in$ [0,2π]rad are to the distance and the angle to the neighbour m.

The output of the reference model are the velocity of the two wheels actuators, v_l and v_r are for the left and right wheels.

3.3.2 Simulator

The simulator used in this thesis is ARGoS [50], which is a simulator conceived specifically for robot swarms. To be able to use the reference model RM1.1 of the e-puck robot describe in the previous section, ARGoS is used along with the ARGoS-Epuck library [51].

ARGoS use XML file with the extension *.argos* to launch an experience, it is structured with tags as follows:

- <**experiment**> allows to determine the duration of the experience as well as the frequency of the update cycle of ARGoS with the option ticks_per_second.
- <**loop_functions**> section in the experiment refers to a script that contains details of the loop functions that will be used during the simulation. These loop functions are responsible for various aspects of the simulation environment, such as setting the floor color, initializing the positions of the robots,

Input	Value	Description
$prox_{i\in1,...,8}$	[0,1]	Reading of proximity sensor i
$light_{i\in1,...,8}$	[0,1]	Reading of light sensor i
$ground_{i\in1,...,8}$	black, gray, white	Reading of ground sensor i
n	[0,20]	number of neighboring robots perceived
V	([0.5,20],[0,2π])	attraction vector

Output	Value	Description
$v_{l,r}$	[-0.12,0.12]ms^{-1}	target linear wheel velocity

Table 3.3: Reference model RM1.1 for the modified version of the e-puck

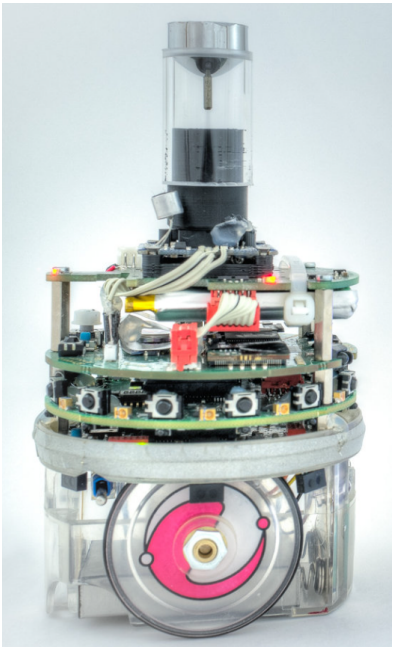


Figure 3.2: The modified e-puck used for the reference model

and defining landmarks on the ground of the arena. The landmarks can be in the form of rectangles or circles, and they are included in the script to facilitate parsing of this information in the Python script for further processing or analysis.

- <**controller**> allows to tell what library to use to read the control software as well as all the parameters for the sensors and actuators of the e-puck robots.
- <**arena**> is the tag that will handle the disposition of the arena such as wall

and light. It also tells how to distribute the e-puck in the arena.

- <**physics_engines**> defines which physic engine ARGoS will use for the simulation.
- <**media**> defines the information about the LED used during the experiment as well as the parameter of the range-and-bearing of the e-puck.
- <**visualisation**> allows to enable the visualisation of ARGoS. If it's not defined, the simulation will run in the background.

3.3.3 Missions

Iridia lab has created a comprehensive list of missions to explore the behaviour of swarm robotics. The list is composed of the following missions:

- **Stop**: The swarm must stop whenever a signal is emitted. This signal can be initiated when a robot steps on a certain floor patch [24, 28, 31].
- **Shelter**: The swarm must find and aggregate inside of a shelter [52]. Their may be ambient cues or constrained access to the shelter [53][21].
- **Foraging**: The swarm must collect a maximum number of objects from two sources and drop them in the nest. In this mission, an object is considered as retrieved when an individual robot visit a source, and an object is dropped when a robot visits the nest. A light source is placed outside the arena to give a cue for the robots [25, 26, 28, 30, 31, 32, 52].
- **Aggregation**: The robots must aggregate in a specific locations such as, for example, a white spot on the ground [24, 25, 26, 28, 31, 32, 52]. A variant of the aggregation is the aggregation with ambient cues where a light source is put outside the arena to give a cue to robots [30, 53, 21].
- **Directional-Gate**: The robots must pass through a gate in the correct sense [52].
- **Coverage**: The swarm has to spread as much as possible in the arena to cover the biggest area possible. There exists variants of this mission such as coverage with forbidden areas, coverage with the largest covering network or coverage of specific region[53, 21].
- **Homing**: The robots start on one side of the arena and must aggregate on a floor patch on the other side, there is also no ambient cue to help the robot [52].
- **Decision**: The swarm must relocate them-self in one-half of the arena depending on the color of a patch in the center of the arena [24].
- **Anytime-Selection**: The robot must aggregate on one of the two floor patches disposed in the arena. Once a robot enter a patch, it cannot leave the patches because the performance is measured over time [25].

3.4 Pseudo-reality

One of the main concern that was raised in the section 2.2.1 is the reality gap that appears when transferring a control software generated on a simulation to the reality. The difference of protocol that appears between the simulation and the reality often leads to a drop of real-world performance.

Ligot [54] introduced the concept of pseudo-reality which is a secondary simulation model, different from the one used to create the control software, that is used for the evaluation of control software and which plays the role of reality. The pseudo-reality model differ from the original simulation model by the amount of noise injected to the sensors and actuators of the robots.

The model used to create the secondary simulation model is called M_B while the original model used to create the control software is called M_A. Both of these models can be seen in the table 3.4.

Actuator/Sensor	Parameter	M_A	M_B	Range R
Wheels	p_g	0.05	0.15	[0.00,0.20]
Proximity	p_u	0.05	0.05	[0.00,0.10]
Light	p_u	0.05	0.90	[0.00,1.50]
Ground	p_u	0.5	0.05	[0.00,0.10]
Range-and-bearing	p_{fail}	0.85	0.90	[0.70,1.00]

Table 3.4: This table represent the models M_A, M_B, and the ranges of possible values for models within the range R used by Ligot [54]. The values correspond to the parameters of ARGoS3 controlling the noise applied to the actuator values and sensor readings.

A good practice is to inject noise in the sensors and actuators of the simulated robots [55]. The simulator ARGoS [50] use a uniform white noise for the readings of the proximity, light, and ground sensors of the e-puck robot. The parameter p_u controls the level of noise, a value is sampled uniformly in the range [-p_u,p_u] which is added to the reading. A Gaussian white noise is applied to the velocities of each wheel, it has a mean of 0 and a standard deviation of p_g. Furthermore, for the range-and-bearing module, a robot fails to estimate the relative position of a neighboring robot with a probability of p_{fail}.

It was observed that in AutoMoDe-Chocolate a rank inversion occurs when passing from simulation to reality. Ligot was able to reproduce the effect of the reality-gap with the help of the pseudo-reality model.

The concept of pseudo-reality can be applied to grasp the change of performance between reality and simulation without the need of running experiment in the real-world.

Chapter 4

Experimental Protocol

This chapter provides an overview of the experimental protocol used in the conducted experiments for each mission.

In this thesis, the main goal is to compare the performance of Map-Elites and the state of the art AutoMoDe-Chocolate as well as to see if a different feature set can influence the performance of the control software generated by Map-Elites. The two behavioural sets used are the following:

1. The positional set of feature defined by Gharbi *et al* [23],
2. The sensory set of feature defined by Hasselmann [33].

The experimental protocol is the following for all the missions :

1. Run Map-Elites for the two different feature sets to generate control software.
2. Retrieve the 10 best control software from each missions obtained from the two different feature set and run these control software on a pseudo-reality experiment.
3. Take the control software obtained for AutoMoDe-Chocolate in the literature with the same budget as the one used for the first step and run these control software on the missions as well as on the pseudo-reality experiment.
4. Construct notched box-and-whiskers plots to analyse the previous results as well as a Friedman test to aggregate the result.

To avoid introducing any bias, robot experiments were randomized and no experimental run performed was discarded.

4.1 Missions

As part of this thesis goal, which is to validate two different feature set that have been used by AutoMoDe-Demonstration-Cho [23] and AutoMoDe-Nata [33]. This thesis will use the same mission used by both version of AutoMoDe. Those missions are aggregation with ambient cues, homing, sheltering with ambient cues and coverage with forbidden areas and foraging.

The following mission all have the same disposition for the arena, they are composed of twelves walls of length equal to 66cm forming a regular dodecagon.

Aggregation with Ambient Cues

There are two circular patches disposed inside the arena, one white and one black. The radius of those circle is of 30cm. A light source is placed outside of the arena on the side of the black circle and aligned with the two circular patches.

The goal is for the swarm to aggregate on the black spot as quickly as possible. The light source is there as a cue for the robots while the white spot is there to disturb the swarm. The robots are randomly distributed in the arena. The ideal result can be seen in the figure 4.1.

The objective function of this mission is defined as follows [21]:

$$F_{AAC} = \sum_{t=1}^{T} N(t) \tag{4.1}$$

where N(t) is the number of robots on the black spot at time t and T = 120 seconds.

This parameters of Map-Elite for this missions are launched with 3125 iterations and a population size of 32 to make a budget of 100 000 simulations to be able to compare the result with the one's from the literature.

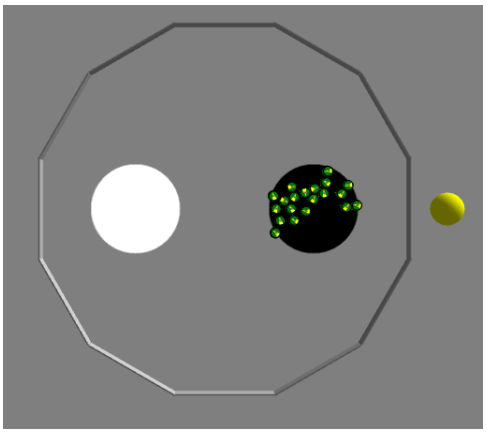


Figure 4.1: Final and best position a swarm can achieve for the aggregation with ambient cues mission.

Homing

In this mission, there is a black circular patch of radius of 60cm placed on one side of the arena.

The goal is for the swarm to gather on the black spot as quickly as possible on the black spot. There is no ambient cue to help the swarm. The initial position of the robot is on the other side of the arena. The ideal result can be seen in the figure 4.2.

The objective function of this mission is defined as follows [52]:

$$F_{\text{Homing}} = \frac{1}{N} \sum_{i=1}^{N} I_i(T) \tag{4.2}$$

$$I_i(T) = \begin{cases} 1, & \text{if the robot i is in the black area at time T} \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where the number of robots N = 20 and T = 120 seconds.

This parameters of Map-Elite for this missions are launched with 6250 iterations and a population size of 32 to make a budget of 200 000 simulations.

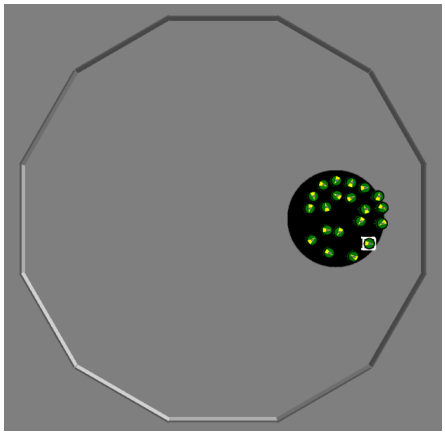


Figure 4.2: Final and best position a swarm can achieve for the homing mission.

Sheltering with Ambient Cues

One half of the arena is completely covered by a black patch. At the border of the black patch and in the middle of the arena, a white rectangle floor patch of 25cm and 15cm in width and height respectively. Two walls of 35 cm in length and one of 50 in length are placed around the white floor patches, leaving an opening far away from the black part of the arena. A light is placed outside of the arena on the side of the opening of the shelter in the center of the arena.

The goal is for the swarm to aggregate inside of the shelter as fast as possible. The outside light source is there to guide the swarm. The robots are randomly distributed in the arena. The ideal result can be seen in the figure 4.3.

The objective function of this mission is defined as follows [21]:

$$F_{SAC} = \sum_{t=1}^{T} N(t) \tag{4.4}$$

where N(t) is the number of robots in the shelter at time t and T = 120s.

This parameters of Map-Elite for this missions are launched with 6250 iterations and a population size of 32 to make a budget of 200 000 simulations.

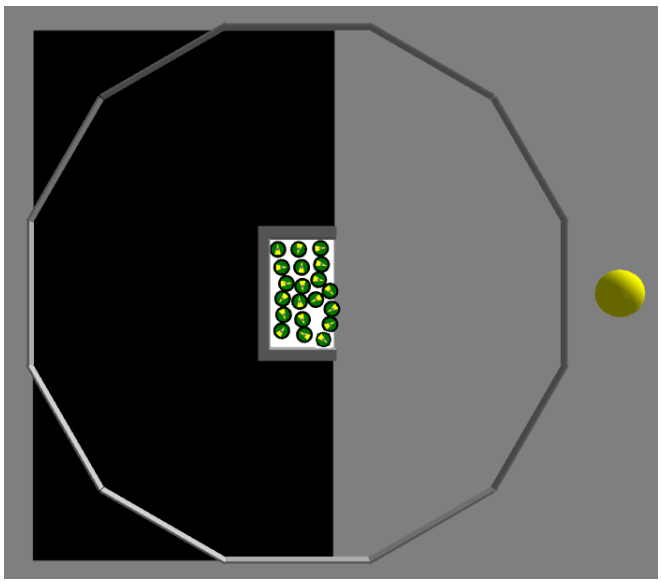


Figure 4.3: Final and best position a swarm can achieve for the sheltering with ambient cues.

Coverage with Forbidden Areas

There are three black circle floor patches with a radius of 30cm within the arena. They are placed in a triangular formation around the center of the arena's center.

The goal of the swarm is to spread as much as possible across the arena while avoiding the black spots. There are no ambient cues to help guide the robot. The robots are randomly distributed in the arena. The ideal result can be seen in the figure 4.4.

The objective function of this mission is defined as follows [21]:

$$F_{CFA} = 25000 - E[d(T)] \tag{4.5}$$

where $E[d(T)]$ is the expected distance between a generic point in the arena and the closest robot not on a black spot, at the end of T, and T = 120s. The objective function has been modified so that it can be maximized like the other missions.

This parameters of Map-Elite for this missions are launched with 6250 iterations and a population size of 32 to make a budget of 200 000 simulations.

Foraging

There are two black circle floor patches with a radius of 15cm within the arena. There disposed in on the top and bottom of the arena. The right side of the arena is covered in a white zone that represent the nest of the swarm. A light is placed outside of the arena on the side of the nest.

The goal of the swarm is to find the black spot which represent the food sources, and to go back to the white zone which is the nest. The outside light source is there to guide the swarm. The robots are randomly distributed in the arena. The overall configuration of this mission can be sen in the figure 4.5.

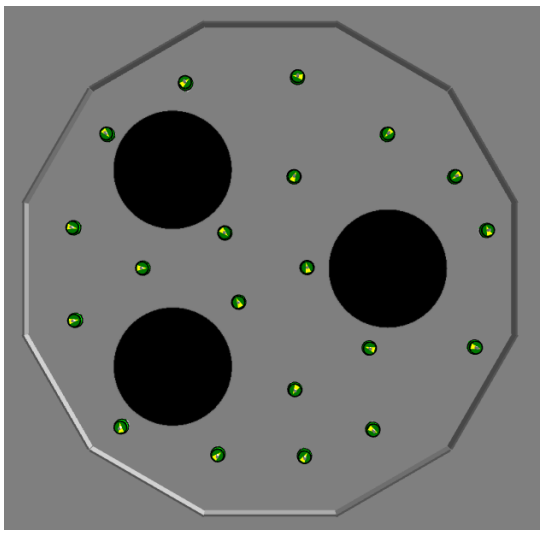


Figure 4.4: Final and best position a swarm can achieve for the coverage with forbidden areas.

The objective function of this mission is defined as follows [21]:

$$F_{\text{Foraging}} = K \tag{4.6}$$

where K is the total number of round trips performed.

This parameters of Map-Elite for this missions are launched with 6250 iterations and a population size of 32 to make a budget of 200 000 simulations.

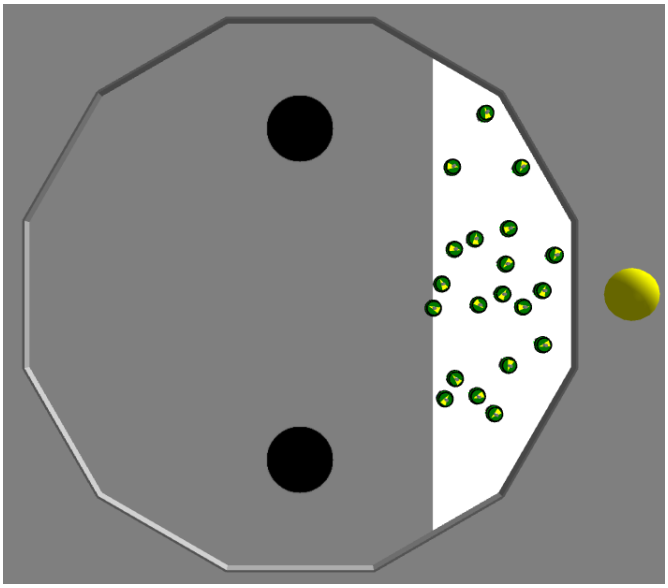


Figure 4.5: The view of the foraging mission.

T = 180s for this mission to be consistent with the mission used in AutoMoDe-Nata [33].

Chapter 5

Result

In the upcoming chapter, he results obtained from conducting the experimental protocol outlined in Section 4 will be presented and analyzed. The experiments were carried out for various missions, including aggregation with ambient cues, homing, sheltering with ambient cues, coverage with forbidden areas, and foraging. The budget allocated for the Map-Elites and Chocolate algorithms during these missions was 200,000, except for the aggregation with ambient cues, which followed the budget of 100,000 as per a previous study in the state-of-the-art literature [28, 52].

Box plots are utilized to summarize the performance of the generated control software for each mission across the three compared algorithms. The box plots represent the top 10 performance results obtained from simulations, indicated by thinner boxes, as well as results from a pseudo-reality environment, represented by thicker boxes. These box plots are constructed as notched box plots, which enable statistical comparisons between the three automated design processes. The notches in the box plots represent the 95% confidence interval of the median value. Therefore, if the notches of two box plots do not overlap, it signifies a statistically significant difference between the performance of the two methods.

In order to combine the results of all missions into a single ranking test, the Friedman test [56] was utilized, with the task serving as a blocking factor. The Friedman test is a rank-based non-parametric test that does not necessitate scaling the performance measures or making assumptions about the underlying distribution of the data. To conduct the test, the objective functions of all tasks were transformed into equivalent minimization problems by applying a rank order inversion function. This transformation is straightforward due to the rank-based nature of the Friedman test. For maximization problems, the inverse of the original objective function was used as the new objective function for minimization [21].

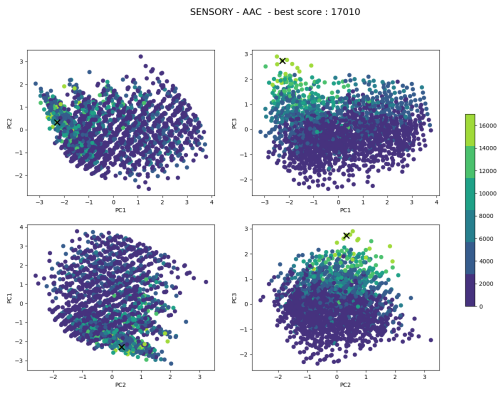
The results of the Friedman test are graphically represented by a plot that illustrates the expected rank achieved by each design method, accompanied by a 95% confidence interval. When the confidence intervals of two methods do not overlap, it indicates a statistically significant difference in the expected ranks between the two methods. This graphical representation facilitates an intuitive comparison of the design methods based on their expected performance ranks.

To visualize the explored space, the Principal Component Analysis (PCA) algorithm is employed to reduce the dimensionality [57]. The sensory feature set consists of 14 features, whereas the positional feature set varies in size from 80 to 160 features. The first three components, which capture the majority of the variance in the search space, are utilized to generate plots of the obtained control software. The color variation in the plots represents the corresponding fitness levels achieved by each control software.

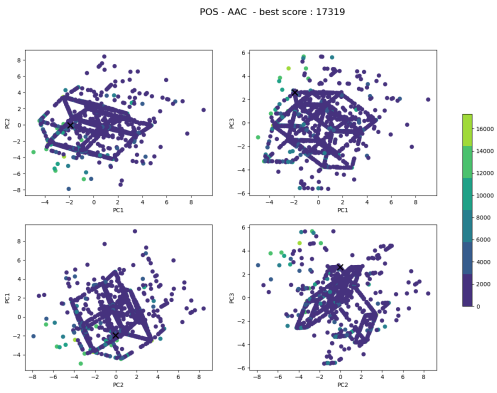
5.1 Aggregation with Ambient Cues

Figure 5.1a illustrates the obtained search space for the sensory feature set. Notably, a distinct cluster of control software representing the aggregation with ambient cues mission is concentrated in one corner of the search space. The visualization of the search space for the positional feature set is depicted in Figure 5.1b. The shape of the search space appears more rectangular and linear compared to the sensory feature set, which exhibits a more compact distribution, this particularity comes from the fact that the positional set has a high dimensionality (120 dimensions). However, similar to the sensory set, it is evident that a significant concentration of high-performing control software for the aggregation with ambient cues mission can be found in one corner of the search space.

An hypothesis can be obtained from the previous result. This mission require a specific set of sensory features values or positional features values to achieve optimal performance. This hypothesis will be checked for the following missions as well.



(a) The sensory feature set



(b) The positional feature set

Figure 5.1: The visual representation of the Map-Elites algorithm for the aggregation with ambient cues mission obtained with the PCA algorithm to reduce the dimensionality.

The comparison between AutoMoDe-Chocolate and the two versions of Map-Elites is illustrated in the figure 5.2. It can be observed that both versions of Map-Elites exhibit higher performance in simulations compared to AutoMoDe-Chocolate. However, when transitioning to the pseudo-reality environment, Map-Elites experiences a larger reality-gap, resulting in a significant drop in perfor-

mance. In contrast, AutoMoDe-Chocolate demonstrates greater resilience to the reality-gap, as it maintains relatively stable performance despite one control software performing significantly worse.

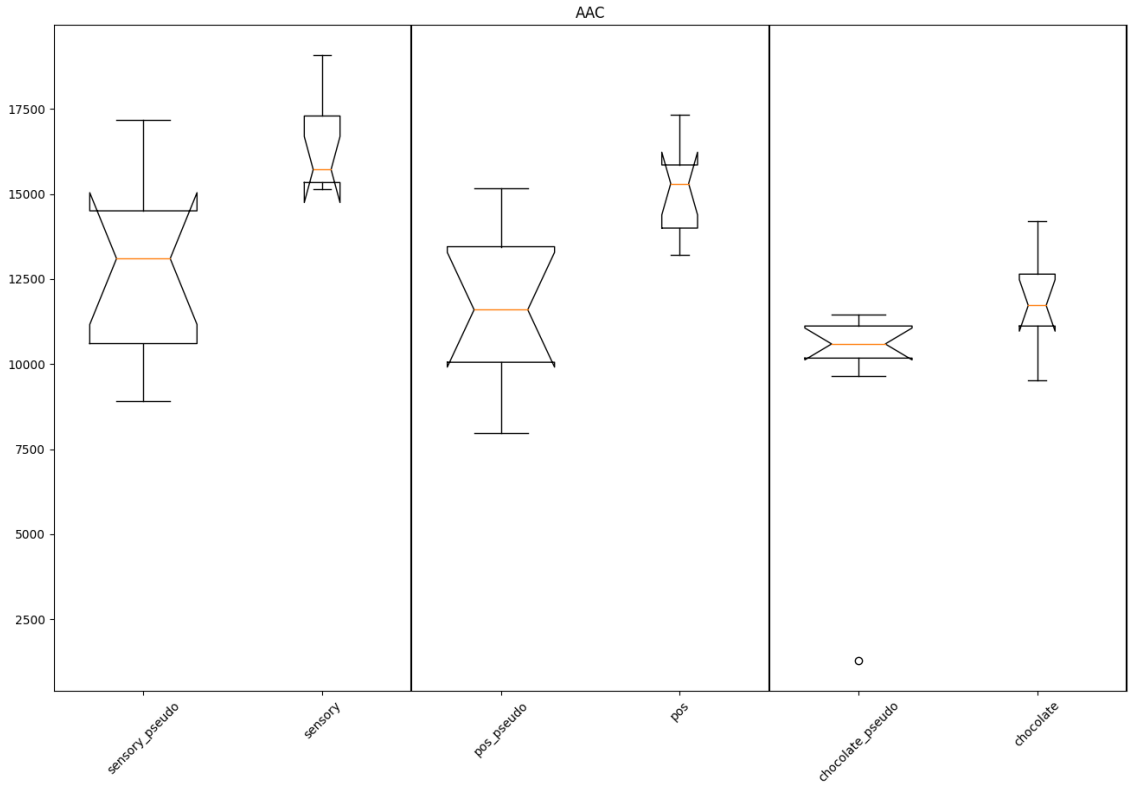
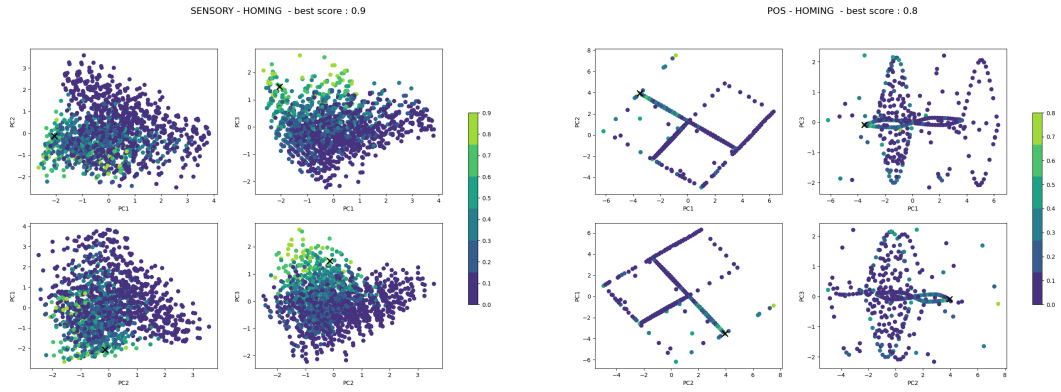


Figure 5.2: Comparison of the performance of the three design methods in simulation and pseudo-reality.

5.2 Homing

The search space of the sensory feature set depicted in the figure 5.3a is compact and has one corner that contain all the performing control software. The figure 5.3b shows the search space for the positional feature set. The space is more sparsely distributed due to the high dimensionality (80 features) but it can still be seen that one of the side of the space contain all the good control software, even though it is not as clustered as for the sensory set.

The analysis of the notched box-plots representing the performance of each design method , as seen in figure 5.4, reveals interesting insights. In the case of the sensory feature set, the performance of Map-Elites is statistically comparable to AutoMoDe-Chocolate, as indicated by the overlapping notched parts of the box-plots. However, for the positional feature set, the Map-Elites algorithm exhibits lower performance in simulation. Moreover, the Map-Elites algorithm is more affected by the reality-gap when transitioning from simulation to a pseudo-reality model. Although it achieves good results, such as in the case of the sensory feature set, the variance of the outcomes remains higher compared to the



(a) The sensory feature set

(b) The positional feature set

Figure 5.3: The visual representation of the Map-Elites algorithm for the homing mission obtained with the PCA algorithm to reduce the dimensionality.

AutoMoDe-Chocolate version.

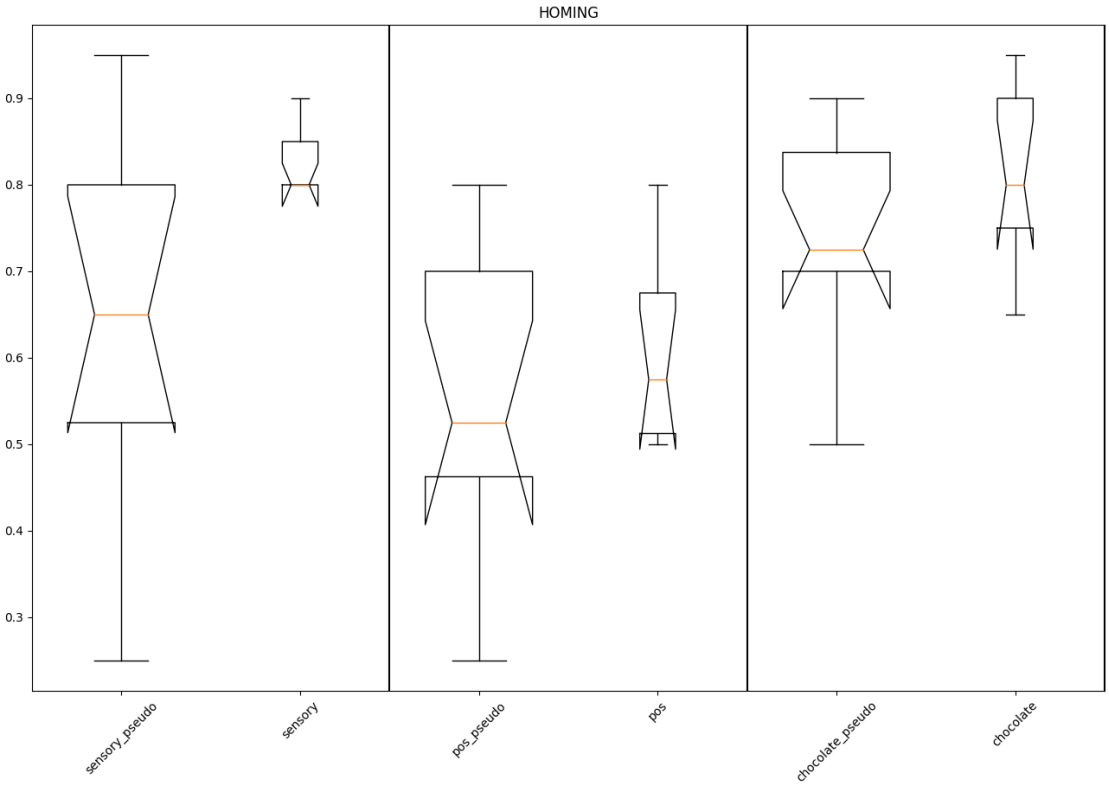
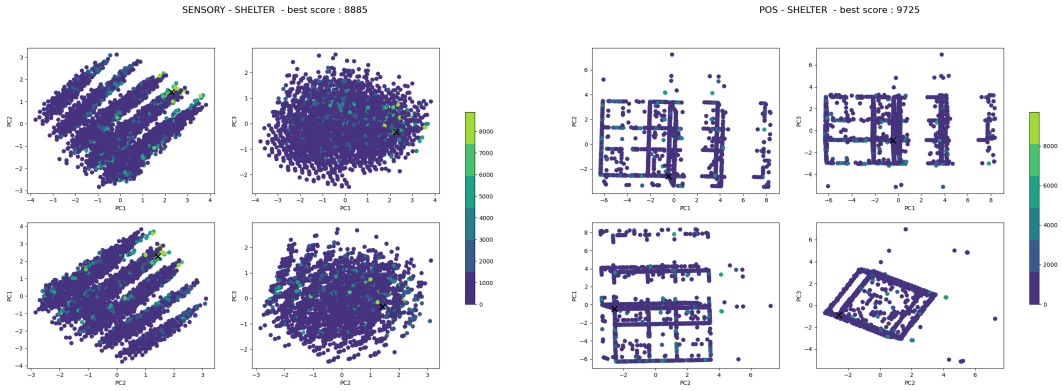


Figure 5.4: Comparison of the performance of the three design methods in simulation and pseudo-reality.

5.3 Shelter with Ambient Cues

The behavioral space of the sensory feature set, as depicted in the figure 5.5a, exhibits a pattern of control software arranged in rows. The distribution of high-

performing solutions shows some dispersion across the entire space, although there is a higher concentration observed at the end of two of the rows. In contrast, the search space of the positional feature set for the shelter with ambient cues mission, shown in the figure 5.5b, takes on a more square-shaped form. In this space, there is no apparent cluster of high-performing solutions; instead, they are scattered throughout the entire space. This suggests that the algorithm has not yet discovered a specific combination of features that leads to highly performing control software for this mission.



(a) The sensory feature set

(b) The positional feature set

Figure 5.5: The visual representation of the Map-Elites algorithm for the sheltering with ambient cues mission obtained with the PCA algorithm to reduce the dimensionality.

Additionally, the figure 5.6 provides a comparison of the performance of the top 10 control software obtained by the three design methods. In simulation, the performance of the two versions of Map-Elites is statistically comparable, while AutoMoDe-Chocolate exhibits lower average performance and higher variance. However, when transitioning to the pseudo-reality model, a reality-gap is observed across all three design methods. Interestingly, the positional feature set for Map-Elites appears to demonstrate greater resilience to the reality-gap compared to the other design methods.

5.4 Foraging

The search space for the sensory feature set, as depicted in the figure 5.7a, exhibits a high density of control software configurations. Notably, one side of the space is dominated by a majority of high-performing solutions. On the other hand, the search space for the positional feature set, shown in the figure 5.7b, takes on a biological hexagonal shape. In this case, the performing control software configurations are not concentrated in a single specific location. This outcome was anticipated due to the nature of the foraging mission. Unlike the missions considered by Gharbi et al. [23], the foraging mission, studied by Hasselmann [33], does not lend itself to a demonstration based on a single general position. As a result, there is no specific set of positional feature values that correspond to

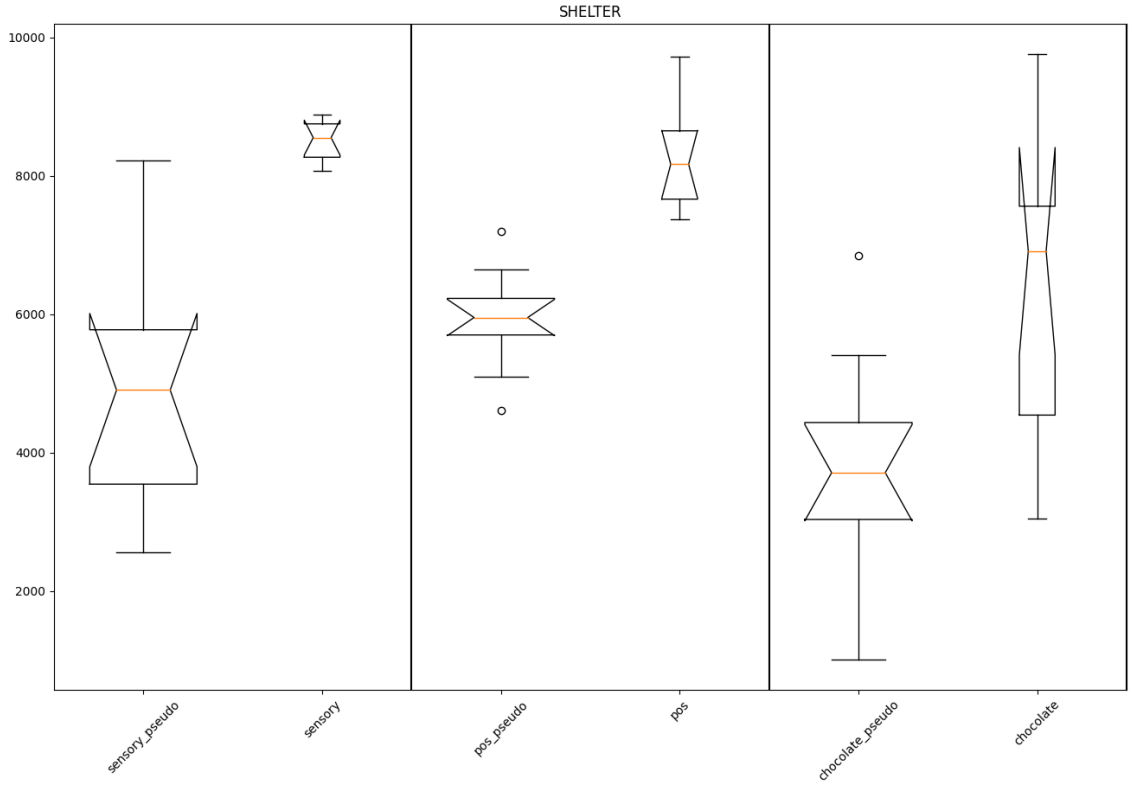
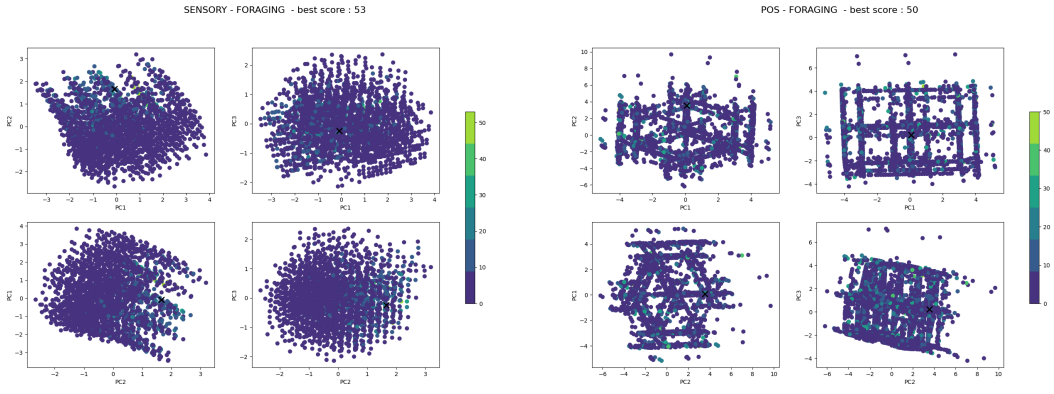


Figure 5.6: Comparison of the performance of the three design methods in simulation and pseudo-reality.

highly performing control software configurations.



(a) The sensory feature set

(b) The positional feature set

Figure 5.7: The visual representation of the Map-Elites algorithm for the foraging mission obtained with the PCA algorithm to reduce the dimensionality.

By analyzing the results obtained from the three design methods, as depicted in the figure 5.8, it can be observed that the sensory and positional feature sets for the Map-Elites algorithm yield statistically similar results in both simulation and pseudo-reality evaluations. In contrast, AutoMoDe-Chocolate exhibits comparable average performance but with higher performance variance. Additionally, all

three design methods experience a small decline in performance when transitioning from simulation to the pseudo-reality model.

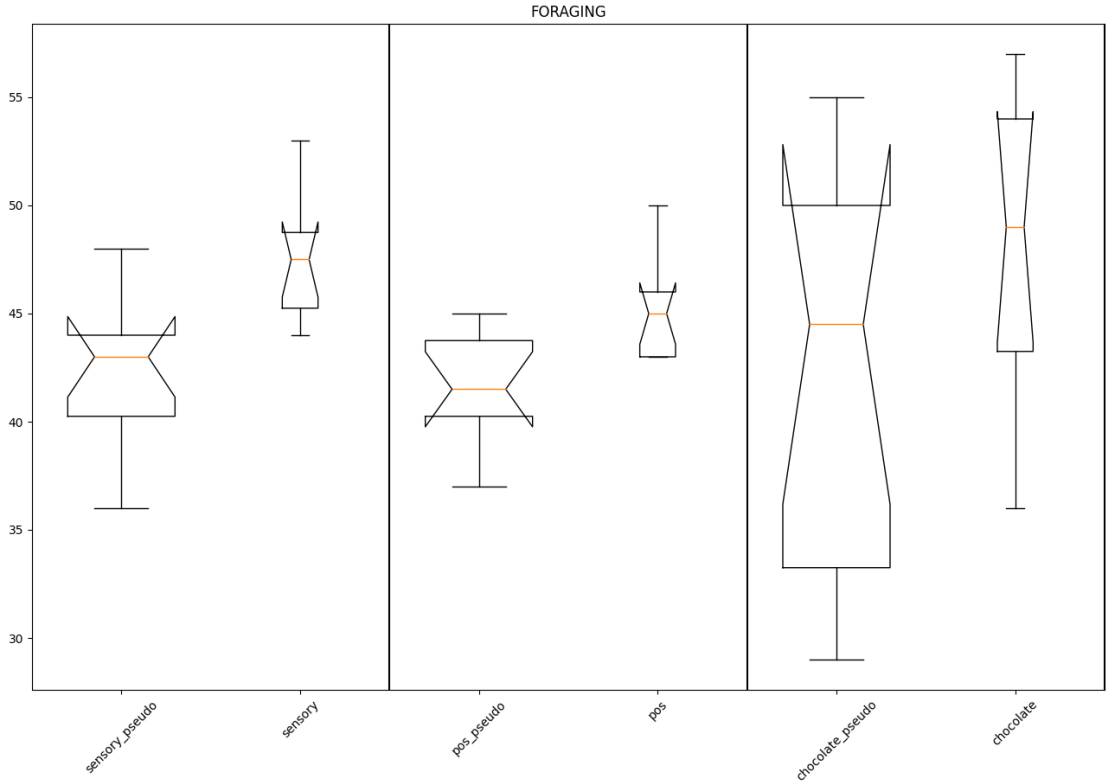
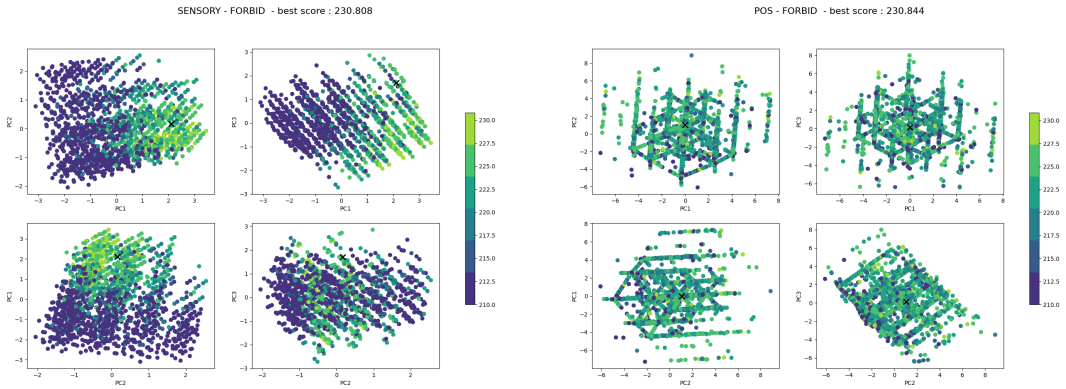


Figure 5.8: Comparison of the performance of the three design methods in simulation and pseudo-reality.

5.5 Coverage with Forbidden Areas

In the figure 5.9a, the search space for the sensory feature set is depicted, revealing a densely compact shape. Notably, one side of the space contains the majority of high-performing control software. In contrast, the figure 5.9b displays the search space of the positional feature set, which exhibits a more geometric shape resulting from the higher dimensionality of the feature set (160 features). The high-performing control software is scattered throughout the entire space without any specific clusters emerging. This suggests that there is no particular combination of values within the positional feature set that leads to high-performance control software. Moreover, when comparing the different design methods, both versions of Map-Elites demonstrate higher results with lower variance in simulation compared to AutoMoDe-Chocolate. However, AutoMoDe-Chocolate appears to be less affected by the reality gap when transitioning to a pseudo-reality model. In terms of performance, the sensory feature set consistently yields better solutions in both simulation and pseudo-reality scenarios.



(a) The sensory feature set

(b) The positional feature set

Figure 5.9: The visual representation of the Map-Elites algorithm for the coverage with forbidden area mission obtained with the PCA algorithm to reduce the dimensionality.

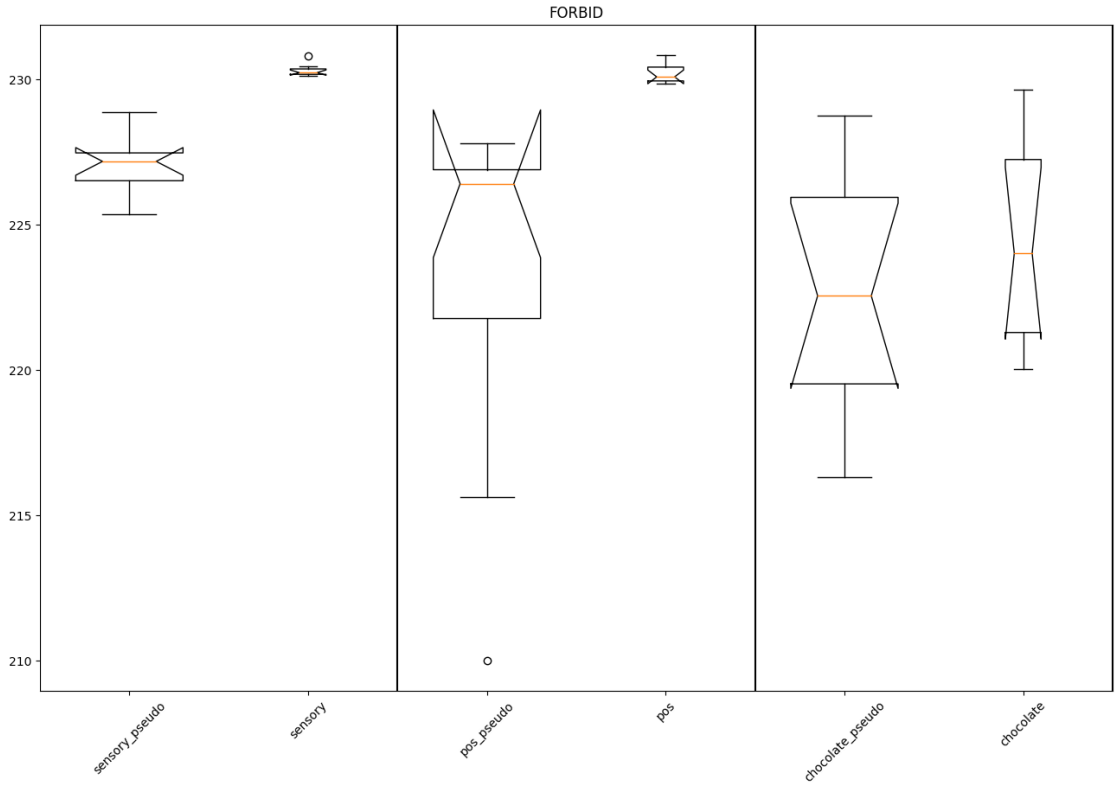


Figure 5.10: Comparison of the performance of the three design methods in simulation and pseudo-reality.

5.6 Ranking

As previously mentioned, a Friedman test is employed to rank the three design methods: Map-Elites with the sensory feature set, Map-Elites with the positional feature set, and AutoMoDe-Chocolate [56]. This test combines the results across missions and scenarios to generate a comprehensive ranking of the methods. This test can be seen in the figure 5.11. The Map-Elites algorithm with the sensory

feature set is considerably better than the two other design method. Furthermore, the Map-Elites with positional feature set has a better average ranking than the AutoMoDe-Chocolate method but it's not statistically better than the latter.

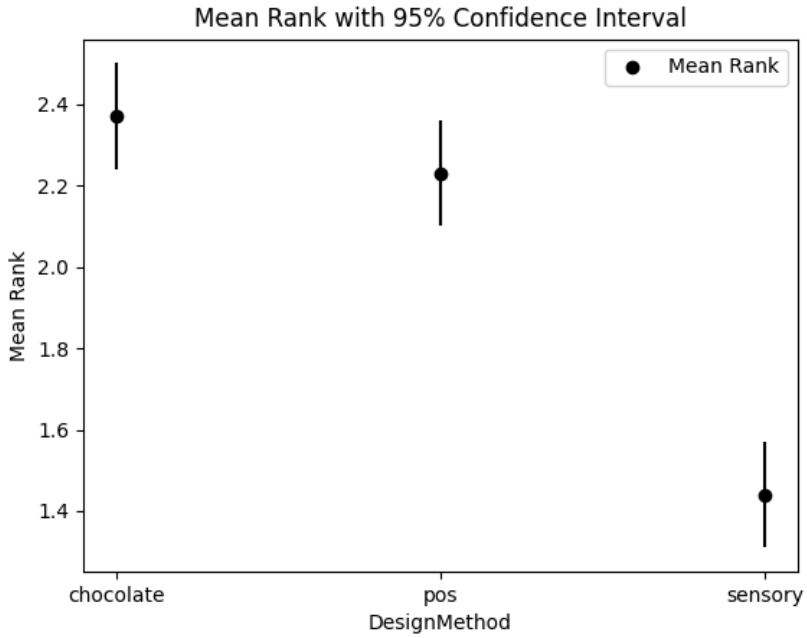


Figure 5.11: Friedman test on aggregate data of the five missions.

Code, results, and data is available as supplementary material at https://github.com/LaurentColpaert/thesis_supp.git.

Chapter 6

Discussion and Further Work

The goal of this master thesis is to produce a new method to create control software. This method is based on an illumination algorithm which has for priority to prioritize the exploration of the space before the optimization of the control software. In this chapter, the success of this method is discussed.

As discussed in the previous chapter, the Map-Elites algorithm demonstrates its capability to produce control software that is comparable or even superior to the AutoMoDe-Chocolate algorithm. Among the two feature sets utilized, the positional feature set yields results similar to AutoMoDe-Chocolate, whereas the sensory feature set yields overall improved results.

The primary factor that can potentially explain the variation in rankings, as depicted in the figure 5.11, is the difference in dimensionality between the two feature sets. The sensory feature set comprises 14 features, whereas the positional feature set ranges from 80 to 160 features. The Map-Elites algorithm optimizes the control software by selecting elite solutions within each cell. However, when considering the extent of the explored space, a higher number of features results in a larger space being explored. Consequently, the probability of multiple control software falling within the same cell decreases, thereby limiting the promotion of exclusively high-performing solutions.

Nevertheless, reducing the dimensionality of the positional feature set requires further investigation. Since this feature set employs distance measurements, omitting certain robots from the calculations could potentially lead to the loss of critical information about the swarm. Thus, a comprehensive study is necessary to explore potential strategies for addressing this issue.

Moreover, when examining the explored space for the two feature sets across various missions, noticeable differences can be observed. The sensory feature set exhibits a distinct cluster of high-performing solutions, indicating a concentrated region of favorable outcomes. On the other hand, the positional feature set showcases smaller and more dispersed clusters, suggesting a less prominent grouping of successful solutions. An ad hoc study exploring the meaning of the feature set can be found in the Appendix A.

Ultimately, another notable distinction between the Map-Elites algorithm and AutoMoDe-Chocolate emerges when considering their resilience to the reality

gap during the transition from simulation to a pseudo-reality model. AutoMoDe-Chocolate demonstrates greater resistance to this phenomenon compared to Map-Elites. This observation suggests that the Map-Elites algorithm may have a tendency to overfit the specific missions at hand. One potential explanation for this behavior is that by exclusively retaining elites, the algorithm disregards the opportunity to generalize control software that perform well on average but may not excel in specific scenarios. An investigation was conducted to analyze the convergence behavior of the Map-Elites algorithm in order to identify potential average solutions that exhibit similar performance and resilience to the reality gap as AutoMoDe-Chocolate. However, the study yielded inconclusive results. Further details of this investigation can be found in Appendix B.

Chapter 7

Conclusion

In the present era, robots play an increasingly significant role in our daily lives. It is crucial to be able to specify the behavior of these robots, whether they operate individually or in a swarm. Ultimately, being able to automate the process of generating the desired behaviour of robots is the ideal goal. This thesis primarily concentrates on investigating the efficacy of an illumination algorithm known as Map-Elites in generating control software within an automated modular design framework.

In order to utilize the aforementioned algorithm, it was necessary to establish an encoding method for the control software, as well as a mechanism for their mutation. This was accomplished by converting probabilistic finite state machines into binary strings. Additionally, a search space had to be defined, with a particular emphasis on capturing the collective behavior of robot swarms. Two previously utilized search spaces from the literature were employed: one focused on the sensory perception of individual robots, while the other focused on their spatial positioning in relation to their environment and neighboring robots.

A comparison was conducted to assess the performance of the Map-Elites algorithm in two different search spaces, as well as a third design method known as AutoMoDe-Chocolate. AutoMoDe-Chocolate is an advanced automatic modular design method that employs the iterated race algorithm as an optimizer. Through this comparison, it was discovered that Map-Elites has the potential to outperform AutoMoDe-Chocolate. However, Map-Elites exhibited a lower robustness when transitioning from simulation to a pseudo-reality model, resulting in a loss of performance. Moreover, notable differences in performance were observed between the two search spaces. The search space utilizing sensors and actuators as its feature set outperformed the search space based on positional features.

Appendix A

Ad hoc

This annexe is an ad hoc study with the goal of trying to understand the difference that might arise from different feature set and understand each feature set better.

Hence, the PCA algorithm will be used to find the features that allows to explain the best the variance of the search space while a random forest regression algorithm will be used to see which features has a higher impact on the performance of each control software. This study will be made on the homing mission (it is an arbitrary choice).

A.1 Feature to explore

The PCA algorithm uses as underlying principles a combination of features to explain as much variance as possible [57]. Thus, it is possible to classify which combination of features has a higher impact on exploring the space.

A.1.1 The positional feature set

In the figure A.1, the amount of explained variance per principal component. The first component explains 45% of the variance while the second explains 30% of the variance. By taking into account the first four component, almost all the search space is explained.

Furthermore, in the figure A.2, the first four component are depicted. Due to the principle of the positional feature set, each feature is repeated for all the robots, which normally imply that they all have the same impact. It can directly be seen in that figure. The feature which is the most impact is the distance to the closest robot, firstly the mean and afterward the standard deviation.

A.1.2 The sensory feature set

In the figure A.3, the explained variance per principal component is shown. Compared to the positional feature set, more component are needed to explain the

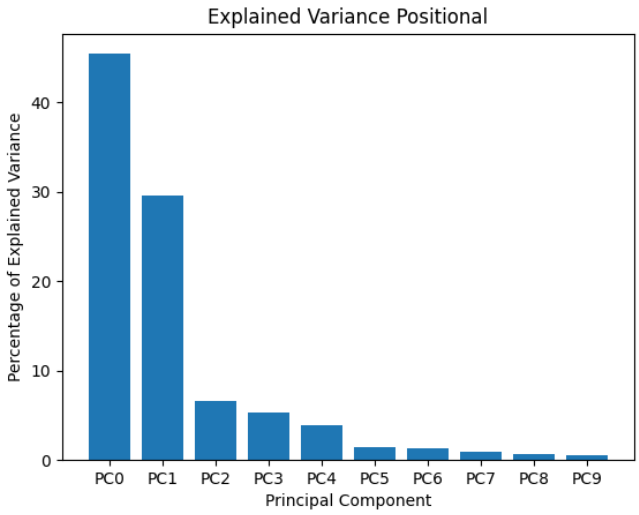
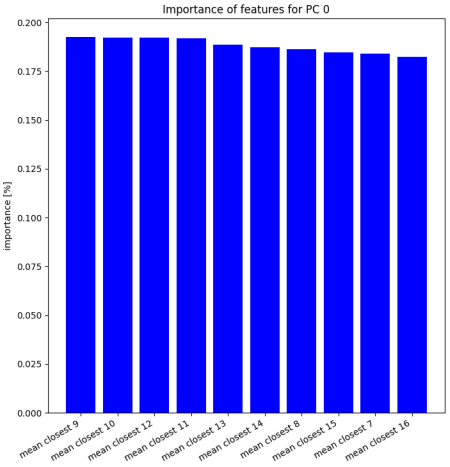
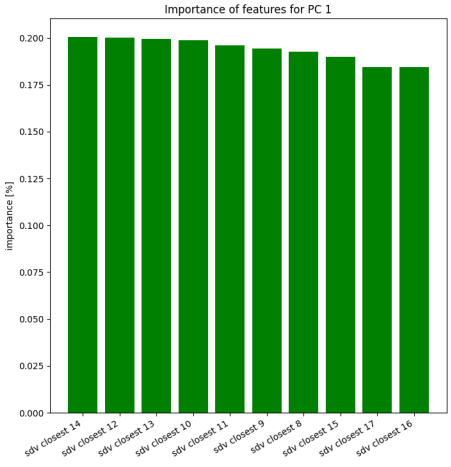


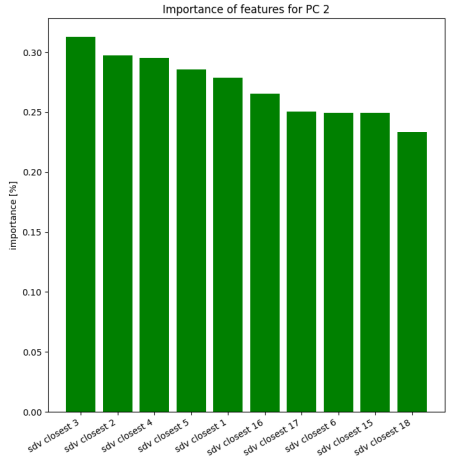
Figure A.1: The amount of explained variance per principal component



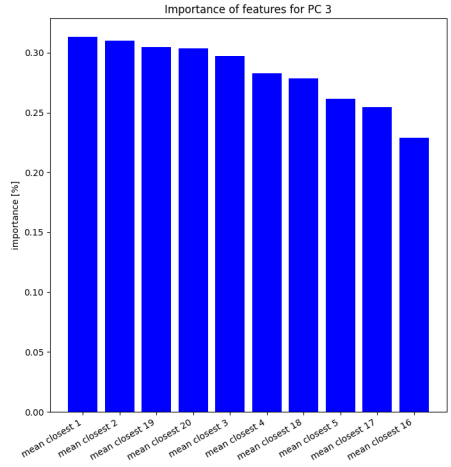
(a) The first component



(b) The second component



(c) The third component



(d) The fourth component

Figure A.2: The importance of features for each component. In blue, the mean of the feature. In green, the standard deviation of the feature

whole behaviour space. The first component takes into account 35% of the variance while the second component explains 20%.

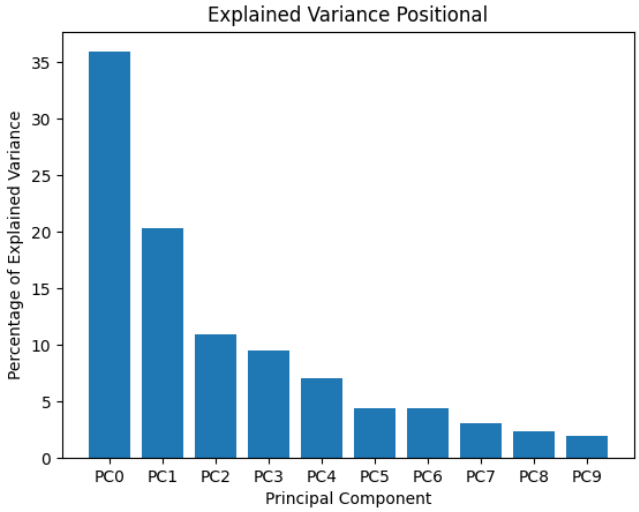


Figure A.3: The amount of explained variance per principal component

Furthermore, in the figure A.4, for the first component, the features using distances are the more impactful, the distance to the closest robot and the distance to other robots. It can also be observed that the color and light have an impact on the exploration. For the distance feature, the mean of these feature has more impact while for the feature about the color and light, the standard deviation has more impact. Finally some feature seems to have no impact on the exploration such as the linear velocity, the angular velocity and the distance to walls.

A.2 Feature to perform

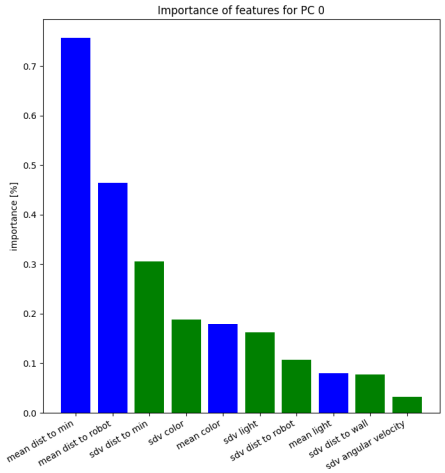
The random forest regression algorithm has the property to explain which feature was the more impactful for the regression **??**. It means that it's possible to see which feature has a direct impact on the performance of the control software.

A.2.1 The positional feature set

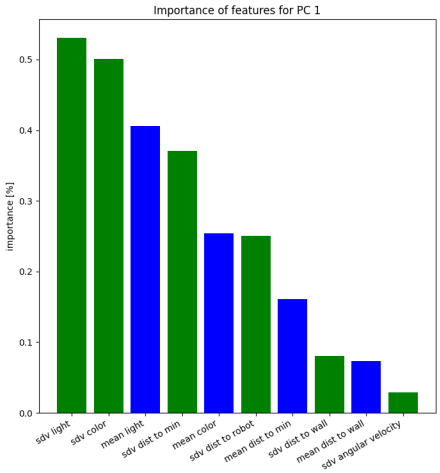
In the figure A.5, the importance of the feature to distinguish between performing and non performing control software. The 10 most important feature are composed of only the mean distance to the closest robot.

A.2.2 The sensory feature set

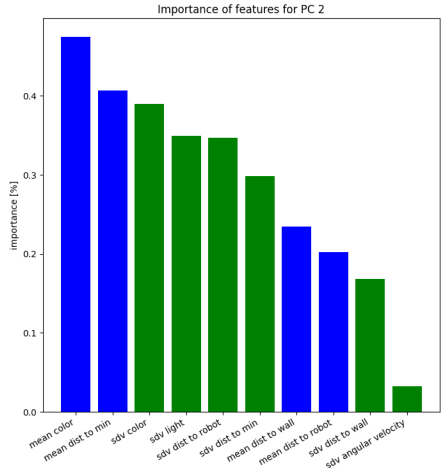
In the figure A.6, the feature based on the sensor are ranked by importance to classify the performance of control software. The two most important features are the mean color and the standard deviation of the color. It seems normal because



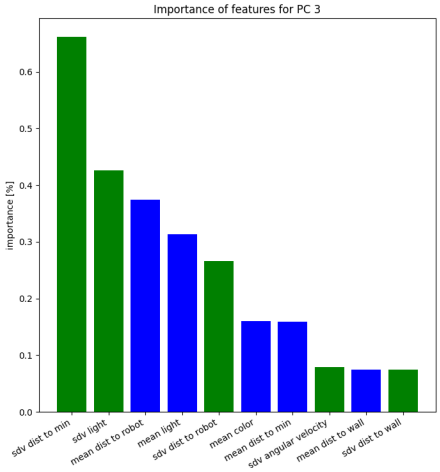
(a) The first component



(b) The second component



(c) The third component



(d) The fourth component

Figure A.4: The importance of features for each component. In blue, the mean of the feature. In green, the standard deviation of the feature

the main goal of the homing mission is to aggregate on a black spot. Furthermore, the following features are the distance to other robot and to the closest robot, which again make sense because to aggregate on a spot, the robot must be close to each other, if that is not the case, it's impossible to have a high performing control software. Finally it's important to notice that the feature mean distance to walls has an impact in classifying the control software but none to explore the space.

A.3 Conclusion

By looking at the two features set and the mission homing, it seems that some features could be removed because they have no impact and the exploration and distinction of performing solution. However no conclusion can be drawn because this conclusion might not be the same for different missions. Further study is needed to understand the underlying concept behind those feature set. How-

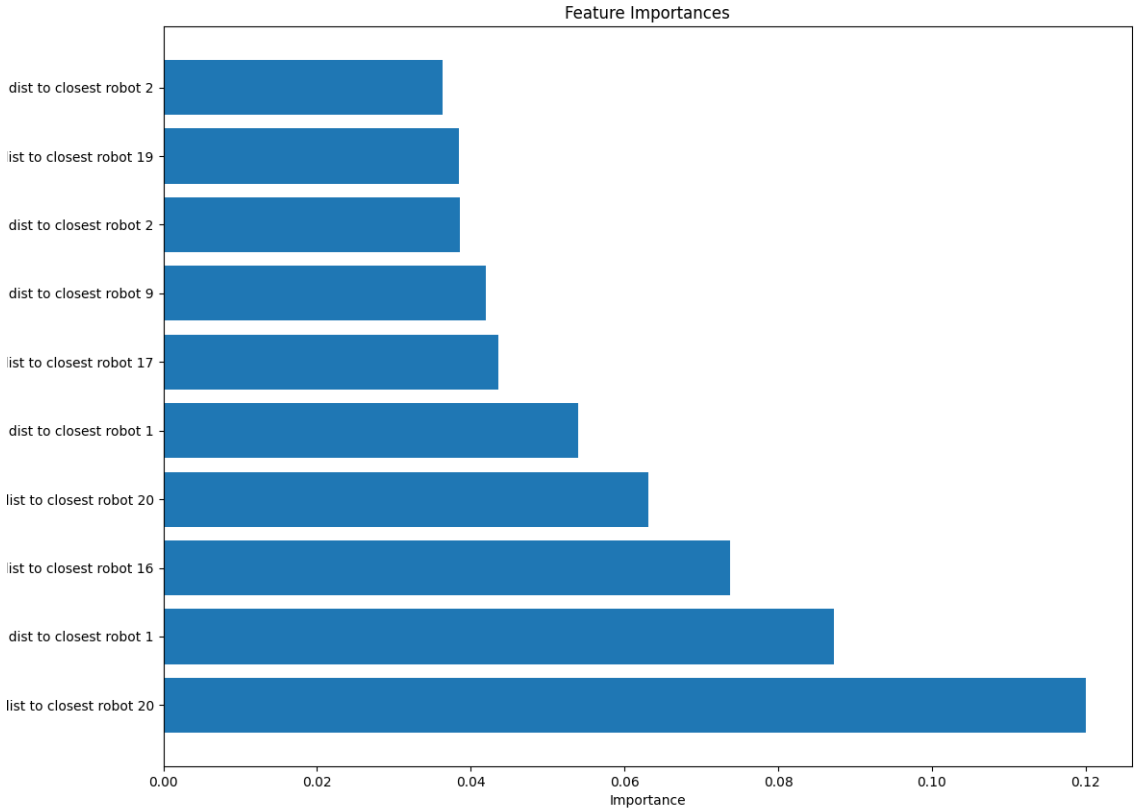


Figure A.5: The 10 most impactful features on the fitness of the control software

ever this ad hoc study demonstrate that the feature set could be reduce to a more impactful feature set.

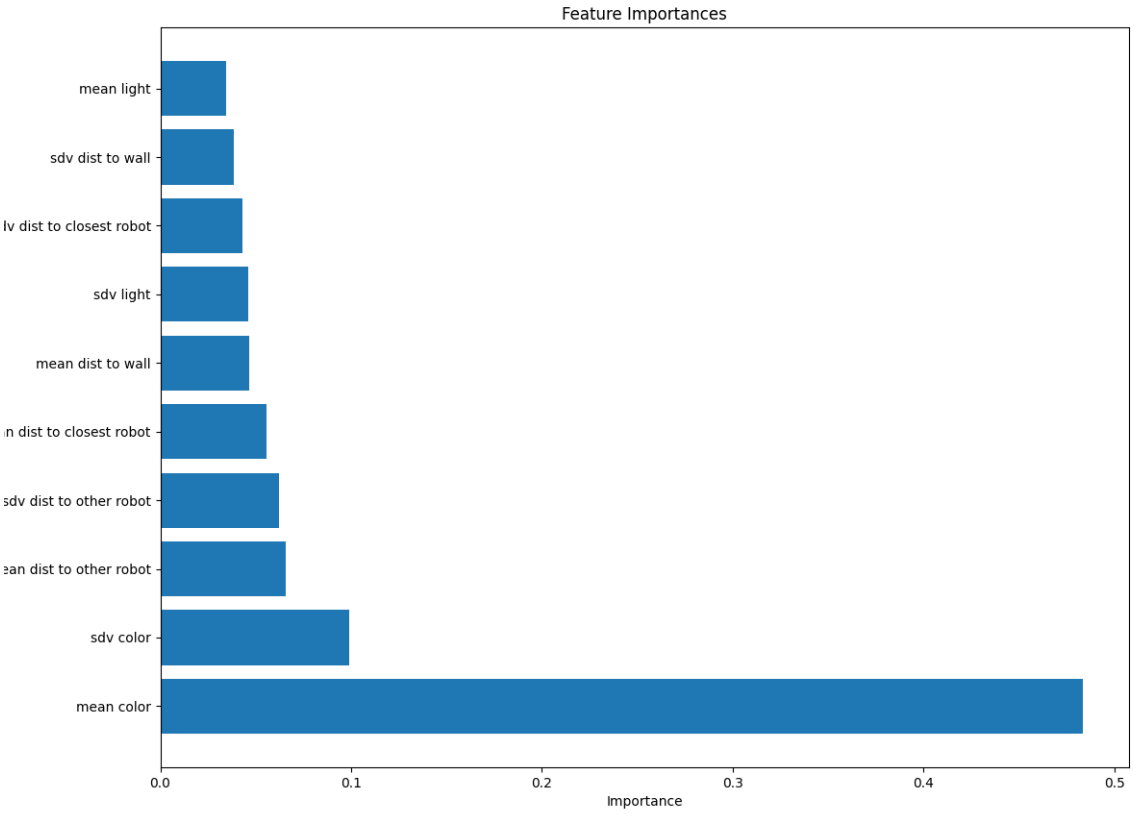
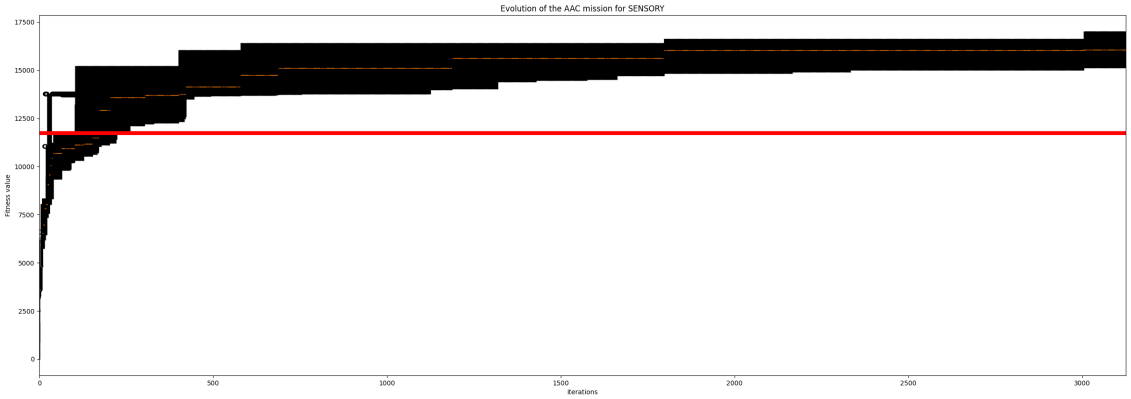


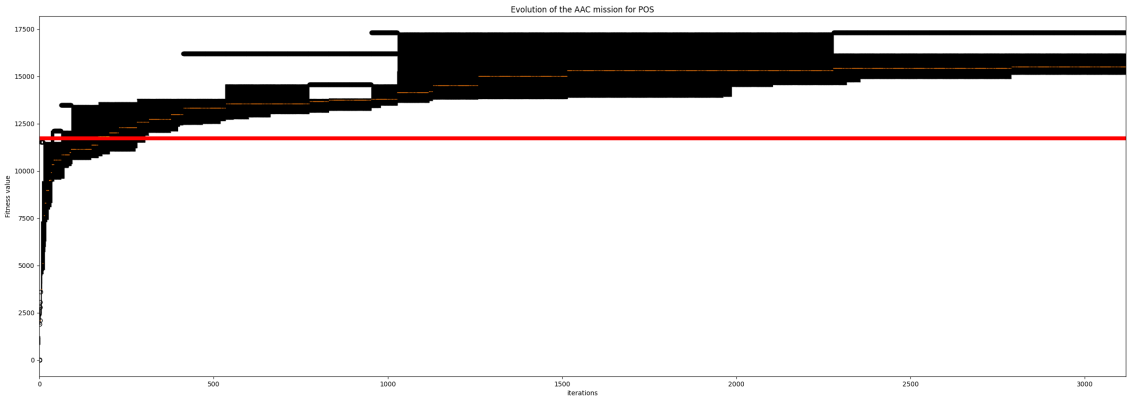
Figure A.6: The 10 most impactful features on the fitness of the control software

Appendix B

Evolution

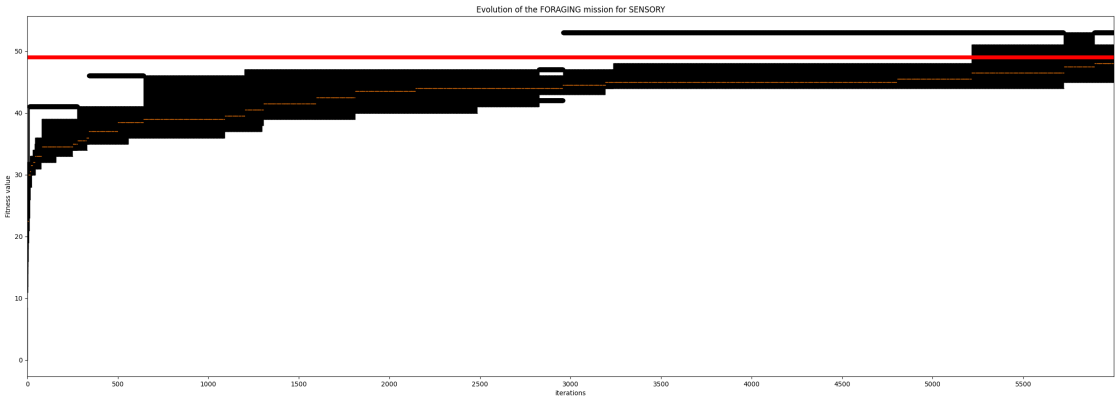


(a) The sensory feature set

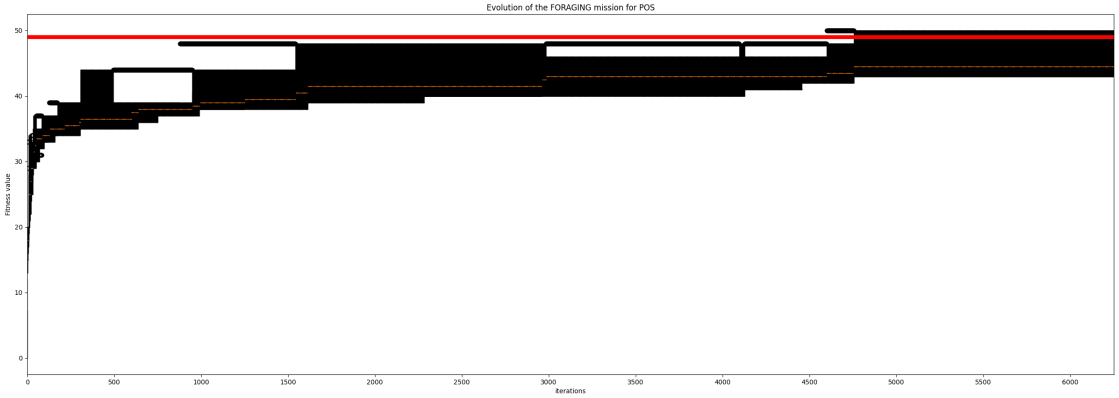


(b) The positional feature set

Figure B.1: The evolution of the 10 best control software throughout the iteration of the algorithm for the aggregation with ambient cues mission.

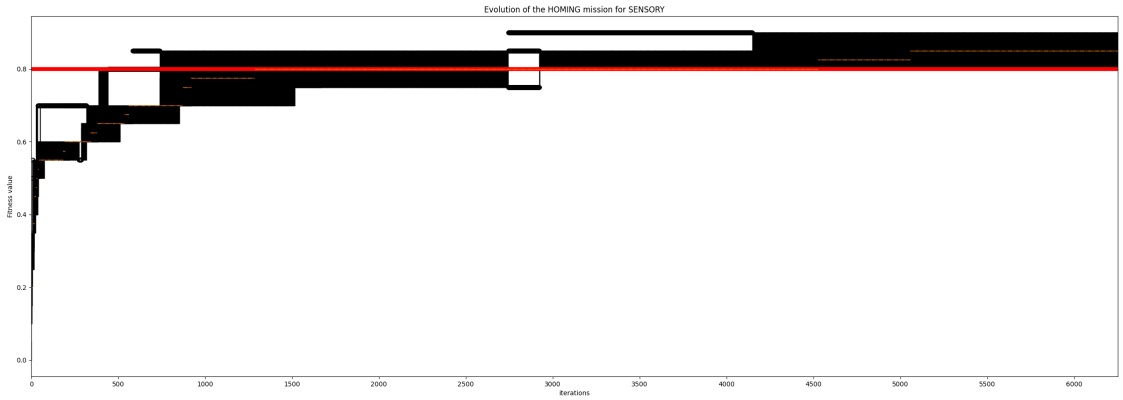


(a) The sensory feature set

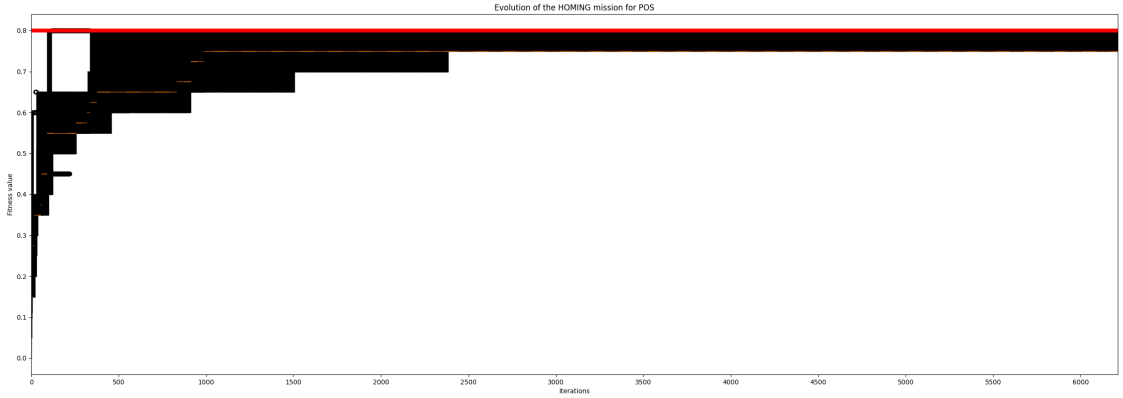


(b) The positional feature set

Figure B.2: The evolution of the 10 best control software throughout the iteration of the algorithm for the foraging mission.

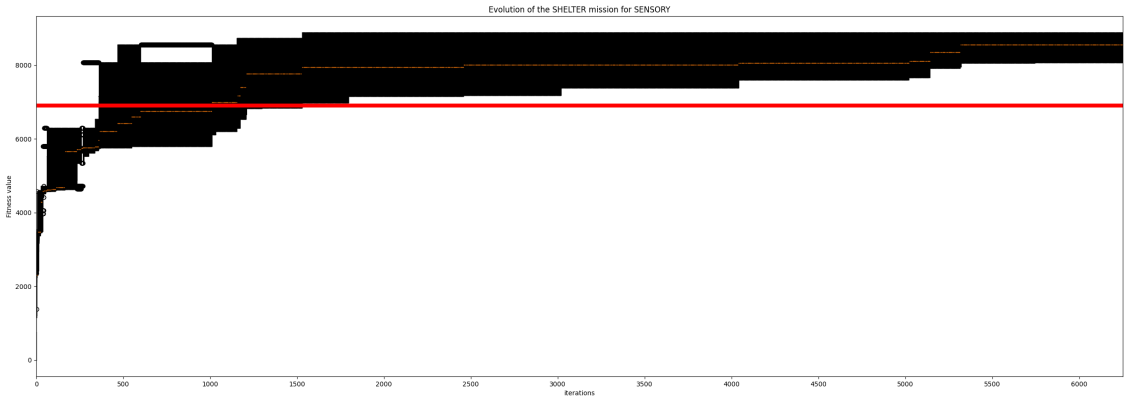


(a) The sensory feature set

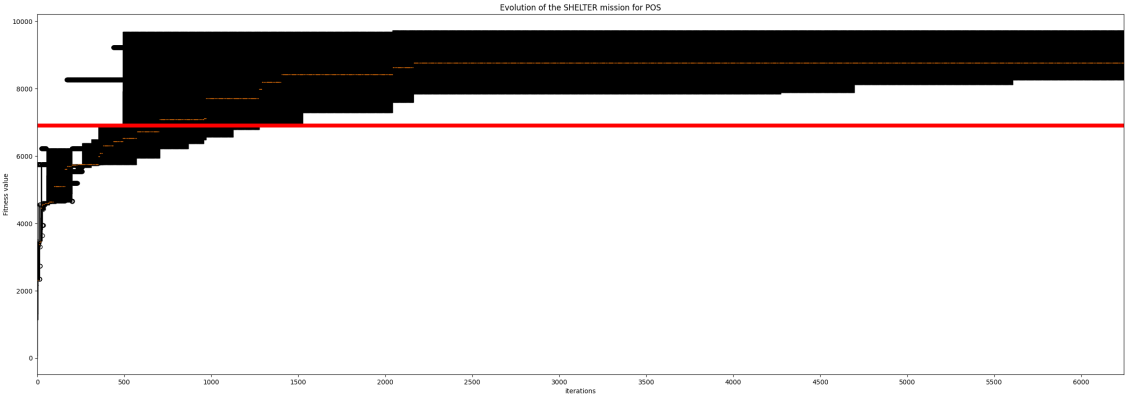


(b) The positional feature set

Figure B.3: The evolution of the 10 best control software throughout the iteration of the algorithm for the homing mission.

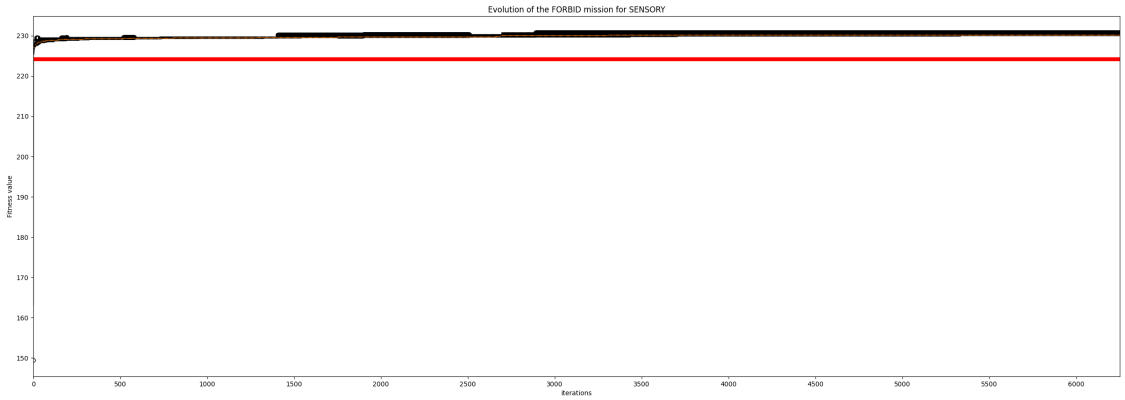


(a) The sensory feature set

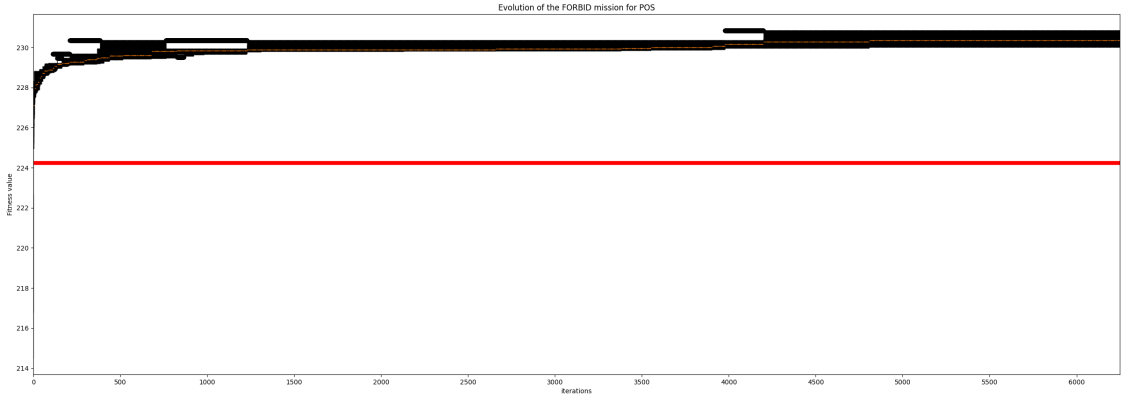


(b) The positional feature set

Figure B.4: The evolution of the 10 best control software throughout the iteration of the algorithm for the shelter mission.



(a) The sensory feature set



(b) The positional feature set

Figure B.5: The evolution of the 10 best control software throughout the iteration of the algorithm for the coverage with forbidden area mission.

Bibliography

[1] *Robotics: What Are Robots? Robotics Definition & Uses. | Built In*. URL: https://builtin.com/robotics (visited on 05/08/2023).

[2] M. Birattari M. Brambilla E. Ferrante and M. Dorigo. "Swarm robotics: a review from the swarm engineering perspective." In: *Scholarpedia* (2013).

[3] Mauro Birattari, Antoine Ligot, and Ken Hasselmann. "Disentangling automatic and semi-automatic approaches to the optimization-based design of control software for robot swarms". en. In: *Nature Machine Intelligence* 2.9 (Aug. 2020), pp. 494–499. ISSN: 2522-5839. DOI: 10.1038/s42256-020-0215-0. URL: https://www.nature.com/articles/s42256-020-0215-0 (visited on 05/08/2023).

[4] Yuri K. Lopes, André B. Leal, Tony J. Dodd, and Roderich Groß. "Application of Supervisory Control Theory to Swarms of e-puck and Kilobot Robots". In: *Swarm Intelligence*. Ed. by Marco Dorigo, Mauro Birattari, Simon Garnier, Heiko Hamann, Marco Montes De Oca, Christine Solnon, and Thomas Stützle. Vol. 8667. Cham: Springer International Publishing, 2014, pp. 62–73. ISBN: 9783319099514 9783319099521. DOI: 10.1007/978-3-319-09952-1_6. URL: http://link.springer.com/10.1007/978-3-319-09952-1_6 (visited on 04/15/2023).

[5] Jean-Baptiste Mouret and Jeff Clune. "Illuminating search spaces by mapping elites". In: (2015). DOI: 10.48550/ARXIV.1504.04909. URL: https://arxiv.org/abs/1504.04909 (visited on 04/16/2023).

[6] Gerardo Beni. "From Swarm Intelligence to Swarm Robotics". In: *Swarm Robotics*. Ed. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Erol Şahin, and William M. Spears. Vol. 3342. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–9. ISBN: 9783540242963 9783540305521. DOI: 10.1007/978-3-540-30552-1_1. URL: http://link.springer.com/10.1007/978-3-540-30552-1_1 (visited on 04/15/2023).

[7] Marco Dorigo, Mauro Birattari, and Manuele Brambilla. "Swarm robotics". en. In: *Scholarpedia* 9.1 (2014), p. 1463. ISSN: 1941-6016. DOI: 10.4249/scholarpedia.1463. URL: http://www.scholarpedia.org/article/Swarm_robotics (visited on 04/15/2023).

[8] Mauro Birattari, Marco Dorigo, and Andreagiovanni Reina. "INFO-H414 : Swarm Intelligence". In: (2022). URL: https://www.ulb.be/en/programme/info-h414 (visited on 04/15/2023).

[9] Marco Dorigo, Guy Theraulaz, and Vito Trianni. "Reflections on the future of swarm robotics". en. In: *Science Robotics* 5.49 (Dec. 2020), eabe4385. ISSN: 2470-9476. DOI: 10.1126/scirobotics.abe4385. URL: https://www.science.org/doi/10.1126/scirobotics.abe4385 (visited on 04/15/2023).

[10] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. "Swarm Robotic Behaviors and Current Applications". In: *Frontiers in Robotics and AI* 7 (Apr. 2020), p. 36. ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00036. URL: https://www.frontiersin.org/article/10.3389/frobt.2020.00036/full (visited on 04/15/2023).

[11] Heiko Hamann. *Swarm robotics: a formal approach*. New York, NY: Springer Science+Business Media, 2018. ISBN: 9783319745268.

[12] Gianpiero Francesca and Mauro Birattari. "Automatic Design of Robot Swarms: Achievements and Challenges". In: *Frontiers in Robotics and AI* 3 (2016). ISSN: 2296-9144. URL: https://www.frontiersin.org/articles/10.3389/frobt.2016.00029 (visited on 04/15/2023).

[13] Jim Pugh and Alcherio Martinoli. "Distributed scalable multi-robot learning using particle swarm optimization". en. In: *Swarm Intelligence* 3.3 (Sept. 2009), pp. 203–222. ISSN: 1935-3812, 1935-3820. DOI: 10.1007/s11721-009-0030-z. URL: http://link.springer.com/10.1007/s11721-009-0030-z (visited on 04/15/2023).

[14] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari. "AutoMoDe: A novel approach to the automatic design of control software for robot swarms". en. In: *Swarm Intelligence* 8.2 (June 2014), pp. 89–112. ISSN: 1935-3812, 1935-3820. DOI: 10.1007/s11721-014-0092-4. URL: http://link.springer.com/10.1007/s11721-014-0092-4 (visited on 04/15/2023).

[15] Vito Trianni. "Evolutionary Robotics: Model or Design?" In: *Frontiers in Robotics and AI* 1 (Dec. 2014). ISSN: 2296-9144. DOI: 10.3389/frobt.2014.00013. URL: http://journal.frontiersin.org/article/10.3389/frobt.2014.00013/abstract (visited on 04/15/2023).

[16] Dario Floreano, Phil Husbands, and Stefano Nolfi. "Evolutionary Robotics". en. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1423–1451. ISBN: 9783540239574 9783540303015. DOI: 10.1007/978-3-540-30301-5_62. URL: http://link.springer.com/10.1007/978-3-540-30301-5_62 (visited on 04/15/2023).

[17] Tian Yu, Toshiyuki Yasuda, Kazuhiro Ohkura, Yoshiyuki Matsumura, and Masanori Goka. "Apply incremental evolution with CMA-NeuroES controller for a robust swarm robotics system". In: *2014 Proceedings of the SICE Annual Conference (SICE)*. Sapporo, Japan: IEEE, Sept. 2014, pp. 295–300. ISBN: 9784907764463 9784907764456. DOI: 10.1109/SICE.2014.6935195. URL: http://ieeexplore.ieee.org/document/6935195/ (visited on 04/15/2023).

[18] Stuart Geman, Elie Bienenstock, and René Doursat. "Neural Networks and the Bias/Variance Dilemma". en. In: *Neural Computation* 4.1 (Jan. 1992), pp. 1–58. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1992.4.1.1. URL: https://direct.mit.edu/neco/article/4/1/1-58/5624 (visited on 04/15/2023).

[19] Ken Hasselmann and Mauro Birattari. "Modular automatic design of collective behaviors for robots endowed with local communication capabilities". en. In: *PeerJ Computer Science* 6 (Aug. 2020), e291. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.291. URL: https://peerj.com/articles/cs-291 (visited on 04/15/2023).

[20] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. "A Racing Algorithm for Configuring Metaheuristics". In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. GECCO'02. New York City, New York: Morgan Kaufmann Publishers Inc., 2002, 11–18. ISBN: 1558608788.

[21] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Lorenzo Garattoni, Roman Miletitch, Gaëtan Podevijn, Andreagiovanni Reina, Touraj Soleymani, Mattia Salvaro, Carlo Pinciroli, Franco Mascia, Vito Trianni, and Mauro Birattari. "AutoMoDe-Chocolate: automatic design of control software for robot swarms". en. In: *Swarm Intelligence* 9.2-3 (Sept. 2015), pp. 125–152. ISSN: 1935-3812, 1935-3820. DOI: 10.1007/s11721-015-0107-9. URL: http://link.springer.com/10.1007/s11721-015-0107-9 (visited on 04/15/2023).

[22] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. "The irace package: Iterated racing for automatic algorithm configuration". en. In: *Operations Research Perspectives* 3 (2016), pp. 43–58. ISSN: 22147160. DOI: 10.1016/j.orp.2016.09.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S2214716015300270 (visited on 04/15/2023).

[23] I. Gharbi, J. Kuckling, D. Garzón Ramos, and M. Birattari. "Show me what you want: inverse reinforcement learning to automatically design robot swarms by demonstration". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. Piscataway, NJ, USA: IEEE, 2023. URL: https://dx.doi.org/.

[24] Ken Hasselmann, Frédéric Robert, and Mauro Birattari. "Automatic Design of Communication-Based Behaviors for Robot Swarms". In: *Swarm Intelligence*. Ed. by Marco Dorigo, Mauro Birattari, Christian Blum, Anders L. Christensen, Andreagiovanni Reina, and Vito Trianni. Vol. 11172. Cham: Springer International Publishing, 2018, pp. 16–29. ISBN: 9783030005320 9783030005337. DOI: 10.1007/978-3-030-00533-7_2. URL: http://link.springer.com/10.1007/978-3-030-00533-7_2 (visited on 04/15/2023).

[25] Muhammad Salman, Antoine Ligot, and Mauro Birattari. "Concurrent design of control software and configuration of hardware for robot swarms under economic constraints". en. In: *PeerJ Computer Science* 5 (Sept. 2019), e221. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.221. URL: https://peerj.com/articles/cs-221 (visited on 04/15/2023).

[26] Jonas Kuckling, Antoine Ligot, Darko Bozhinoski, and Mauro Birattari. "Behavior Trees as a Control Architecture in the Automatic Modular Design of Robot Swarms". In: *Swarm Intelligence*. Ed. by Marco Dorigo, Mauro Birattari, Christian Blum, Anders L. Christensen, Andreagiovanni Reina, and Vito Trianni. Vol. 11172. Cham: Springer International Publishing, 2018, pp. 30–43. ISBN: 9783030005320 9783030005337. DOI: 10.1007/978-3-030-

00533-7_3. URL: http://link.springer.com/10.1007/978-3-030-00533-7_3 (visited on 04/15/2023).

[27] Fernando J. Mendiburu, David Garzón Ramos, Marcos R.A. Morais, Antonio M.N. Lima, and Mauro Birattari. "AutoMoDe-Mate: Automatic off-line design of spatially-organizing behaviors for robot swarms". en. In: *Swarm and Evolutionary Computation* 74 (Oct. 2022), p. 101118. ISSN: 22106502. DOI: 10.1016/j.swevo.2022.101118. URL: https://linkinghub.elsevier.com/retrieve/pii/S2210650222000888 (visited on 04/15/2023).

[28] David Garzón Ramos and Mauro Birattari. "Automatic Design of Collective Behaviors for Robots that Can Display and Perceive Colors". en. In: *Applied Sciences* 10.13 (July 2020), p. 4654. ISSN: 2076-3417. DOI: 10.3390/app10134654. URL: https://www.mdpi.com/2076-3417/10/13/4654 (visited on 04/15/2023).

[29] Gaëtan Spaey, Miquel Kegeleirs, David Garzón Ramos, and Mauro Birattari. "Evaluation of Alternative Exploration Schemes in the Automatic Modular Design of Robot Swarms". en. In: *Artificial Intelligence and Machine Learning*. Ed. by Bart Bogaerts, Gianluca Bontempi, Pierre Geurts, Nick Harley, Bertrand Lebichot, Tom Lenaerts, and Gilles Louppe. Vol. 1196. Cham: Springer International Publishing, 2020, pp. 18–33. ISBN: 9783030651534 9783030651541. DOI: 10.1007/978-3-030-65154-1_2. URL: http://link.springer.com/10.1007/978-3-030-65154-1_2 (visited on 04/15/2023).

[30] Jonas Kuckling, Keneth Ubeda Arriaza, and Mauro Birattari. "AutoMoDe-IcePop: Automatic Modular Design of Control Software for Robot Swarms Using Simulated Annealing". en. In: *Artificial Intelligence and Machine Learning*. Ed. by Bart Bogaerts, Gianluca Bontempi, Pierre Geurts, Nick Harley, Bertrand Lebichot, Tom Lenaerts, and Gilles Louppe. Vol. 1196. Cham: Springer International Publishing, 2020, pp. 3–17. ISBN: 9783030651534 9783030651541. DOI: 10.1007/978-3-030-65154-1_1. URL: http://link.springer.com/10.1007/978-3-030-65154-1_1 (visited on 04/15/2023).

[31] *AutoMoDe-Cedrata*. URL: https://iridia.ulb.ac.be/supp/IridiaSupp2021-004/ (visited on 04/15/2023).

[32] *AutoMoDe-Arlequin*. URL: https://iridia.ulb.ac.be/supp/IridiaSupp2020-005/index.html (visited on 04/15/2023).

[33] K. Hasselmann. "Advances in the automatic modular design of control software for robot swarms. Using neuroevolution to generate modules." PhD thesis. Brussels, Belgium: Université libre de Bruxelles, 2023.

[34] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". en. In: *Twenty-first international conference on Machine learning - ICML '04*. Banff, Alberta, Canada: ACM Press, 2004, p. 1. DOI: 10.1145/1015330.1015430. URL: http://portal.acm.org/citation.cfm?doid=1015330.1015430 (visited on 04/15/2023).

[35] Dario Floreano and Claudio Mattiussi. *Bio-inspired artificial intelligence: theories, methods, and technologies*. Intelligent robotics and autonomous agents. OCLC: ocn212627224. Cambridge, Mass: MIT Press, 2008. ISBN: 9780262062718.

[36] Joel Lehman and Kenneth O. Stanley. "Abandoning Objectives: Evolution Through the Search for Novelty Alone". en. In: *Evolutionary Computation* 19.2 (June 2011), pp. 189–223. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/EVCO_a_00025. URL: https://direct.mit.edu/evco/article/19/2/189-223/1365 (visited on 04/17/2023).

[37] David A. Van Veldhuizen and Gary B. Lamont. "Evolutionary Computation and Convergence to a Pareto Front". In: *Late Breaking Papers at the Genetic Programming 1998 Conference*. Ed. by John R. Koza. University of Wisconsin, Madison, Wisconsin, USA: Stanford University Bookstore, 1998, pp. 221–228. URL: http://www.lania.mx/~ccoello/EMOO/vanvel2.ps.gz.

[38] James J. Bull, Richard H. Heineman, and Claus O. Wilke. "The Phenotype-Fitness Map in Experimental Evolution of Phages". en. In: *PLoS ONE* 6.11 (Nov. 2011). Ed. by Barry L. Williams, e27796. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0027796. URL: https://dx.plos.org/10.1371/journal.pone.0027796 (visited on 04/17/2023).

[39] Jeff Clune, Dusan Misevic, Charles Ofria, Richard E. Lenski, Santiago F. Elena, and Rafael Sanjuán. "Natural Selection Fails to Optimize Mutation Rates for Long-Term Adaptation on Rugged Fitness Landscapes". en. In: *PLoS Computational Biology* 4.9 (Sept. 2008). Ed. by Laurence D. Hurst, e1000187. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000187. URL: https://dx.plos.org/10.1371/journal.pcbi.1000187 (visited on 04/17/2023).

[40] Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E. (Gusz) Eiben. "Evolutionary Robotics: What, Why, and Where to". In: *Frontiers in Robotics and AI* 2 (Mar. 2015). ISSN: 2296-9144. DOI: 10.3389/frobt.2015.00004. URL: http://www.frontiersin.org/Evolutionary_Robotics/10.3389/frobt.2015.00004/abstract (visited on 04/17/2023).

[41] Gianpiero Francesca, Manuele Brambilla, Vito Trianni, Marco Dorigo, and Mauro Birattari. "Analysing an Evolved Robotic Behaviour Using a Biological Model of Collegial Decision Making". In: Jan. 2012, pp. 381–390. ISBN: 9783642330926.

[42] Giuseppe Cuccu and Faustino Gomez. "When Novelty Is Not Enough". In: *Applications of Evolutionary Computation*. Ed. by Cecilia Di Chio, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Anikó Ekárt, Anna I. Esparcia-Alcázar, Juan J. Merelo, Ferrante Neri, Mike Preuss, Hendrik Richter, Julian Togelius, and Georgios N. Yannakakis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 234–243. ISBN: 978-3-642-20525-5.

[43] Joel Lehman and Kenneth O. Stanley. "Novelty Search and the Problem with Objectives". In: *Genetic Programming Theory and Practice IX*. Ed. by Rick Riolo, Ekaterina Vladislavleva, and Jason H. Moore. New York, NY: Springer New York, 2011, pp. 37–56. ISBN: 978-1-4614-1770-5. DOI: 10.1007/978-1-4614-1770-5_3. URL: https://doi.org/10.1007/978-1-4614-1770-5_3.

[44] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. "Quality Diversity: A New Frontier for Evolutionary Computation". In: *Frontiers in Robotics and AI* 3 (July 2016). ISSN: 2296-9144. DOI: 10.3389/frobt.2016.00040. URL: http://journal.frontiersin.org/Article/10.3389/frobt.2016.00040/abstract (visited on 04/22/2023).

[45] Jorge Gomes and Anders Lyhne Christensen. "Task-Agnostic Evolution of Diverse Repertoires of Swarm Behaviours". In: *Swarm Intelligence*. Ed. by Marco Dorigo, Mauro Birattari, Christian Blum, Anders L. Christensen, Andreagiovanni Reina, and Vito Trianni. Vol. 11172. Cham: Springer International Publishing, 2018, pp. 225–238. ISBN: 9783030005320 9783030005337. DOI: 10.1007/978-3-030-00533-7_18. URL: http://link.springer.com/10.1007/978-3-030-00533-7_18 (visited on 04/23/2023).

[46] Sondre A. Engebråten, Jonas Moen, Oleg Yakimenko, and Kyrre Glette. "Evolving a Repertoire of Controllers for a Multi-function Swarm". In: *Applications of Evolutionary Computation - 21st International Conference, EvoApplications 2018*. Vol. 10784. Lecture Notes in Computer Science. Springer, 2018, pp. 734–749. ISBN: 978-3-319-77537-1.

[47] A. Ligo, K. Hasselmann, B. Delhaisse, L. Garattoni, G. Francesca, and M. Birattari. "AutoMoDe, NEAT, and Evostick: Implementatoins for the E-puck robot in ARGoS3". In: *IRIDIA* (Jan. 2017). ISSN: 1781-3794. DOI: TR/IRIDIA/2017-002. URL: https://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2017-002.pdf (visited on 04/23/2023).

[48] Francesco Mondada, Paulo Gonçalves, Paulo Torres, Carlos Alves, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and A. Martinoli. "The e-puck, a Robot Designed for Education in Engineering". In: *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions* 1 (Jan. 2009).

[49] Alvaro Gutierrez, Alexandre Campo, Marco Dorigo, Jesus Donate, Felix Monasterio-Huelin, and Luis Magdalena. "Open E-puck Range Bearing miniaturized board for local communication in swarm robotics". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 3111–3116. DOI: 10.1109/ROBOT.2009.5152456.

[50] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems". en. In: *Swarm Intelligence* 6.4 (Dec. 2012), pp. 271–295. ISSN: 1935-3812, 1935-3820. DOI: 10.1007/s11721-012-0072-5. URL: http://link.springer.com/10.1007/s11721-012-0072-5 (visited on 04/18/2023).

[51] Garattoni L., Francesca G., Brutschy A., Pinciroli C., and Birattari M. "Software Infrastructure for E-puck (and TAM)". In: vol. Tech. rep. TR/IRIDIA/2015-004. 2015.

[52] Ken Hasselmann, Antoine Ligot, Julian Ruddick, and Mauro Birattari. "Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms". en. In: *Nature Communications* 12.1 (July 2021), p. 4345. ISSN: 2041-1723. DOI: 10.1038/s41467-021-24642-3. URL: https://www.nature.com/articles/s41467-021-24642-3 (visited on 04/18/2023).

[53] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Lorenzo Garattoni, and Roman Miletitch. "An Experiment in Automatic Design of Robot Swarms". In: *Swarm Intelligence*. Cham: Springer International Publishing, 2014, pp. 25–37. ISBN: 978-3-319-09952-1.

[54] A. Ligot. "Assessing and forecasting the performance ofoptimization-based design methods for robot swarms. Experimental protocol pseudo-reality predictors." PhD thesis. Brussels, Belgium: Université libre de Bruxelles, 2023.

[55] Federico Morán, Alvaro Moreno, Juan Julián Merelo, Pablo Chacón, G. Goos, J. Hartmanis, J. Van Leeuwen, Jaime G. Carbonell, and Jörg Siekmann, eds. *Advances in Artificial Life: Third European Conference on Artificial Life Granada, Spain, June 4–6, 1995 Proceedings*. en. Vol. 929. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. ISBN: 9783540594963 9783540492863. DOI: 10.1007/3-540-59496-5. URL: http://link.springer.com/10.1007/3-540-59496-5 (visited on 05/23/2023).

[56] W. J. Conover. *Practical nonparametric statistics*. 3rd ed. Wiley series in probability and statistics. Applied probability and statistics section. New York: Wiley, 1999. ISBN: 9780471160687.

[57] Andrzej Maćkiewicz and Waldemar Ratajczak. "Principal components analysis (PCA)". In: *Computers Geosciences* 19.3 (1993), pp. 303–342. ISSN: 0098-3004. DOI: https://doi.org/10.1016/0098-3004(93)90090-R. URL: https://www.sciencedirect.com/science/article/pii/009830049390090R.