ECOLE
POLYTECHNIQUE
DE BRUXELLES

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

The influence of random walks on automatic design of robot swarms

An experiment with AutoMoDe

Mémoire présenté en vue de l'obtention du diplôme
d'Ingénieur Civil en Informatique à finalité spécialisée

Gaëtan Spaey

Directeur
Professeur Mauro Birattari

Superviseurs
David Garzón Ramos, Miquel Kegeleirs

Service
IRIDIA

Année académique
2018 - 2019

Acknowledgements

First, I would like to thank my thesis director, Mauro Birattari, for his support and for introducing me to the field of swarm robotics.

I would like to acknowledge David Garzón Ramos for helping me define the outline of my master thesis, for assisting me with the set-up of the experiments on the cluster and for giving me precious advice regarding the interpretation of the results.

I would also like to thank Miquel Kegeleirs for providing me day-to-day support in all aspects of my master thesis. He patiently accompanied me during the set-up of ARGoS and AutoMoDe and helped me fix technical problems on multiple occasions. Furthermore, he read this master thesis several times and greatly helped me in improving it.

Finally, I would like to thank my fellow students and friends. Julian Ruddick for helping me on numerous occasions in fixing technical problems, as well as David Boyker, Samy Badreddine and Stéphane Sercu for providing me with continuous feedback and support through the writing of this master thesis.

Résumé

Des études sur la robotique en essaim ont montré que des tâches complexes peuvent être résolues par de grands groupes de robots simples interagissant entre eux et avec leur environnement. La plupart de ces tâches exigent que les robots explorent leur environnement, ce qui fait de l'exploration une composante de base du comportement des essaims de robots. Les comportements d'exploration n'ont cependant pas fait l'objet d'évaluations approfondies sur ces derniers, en particulier dans la conception automatique de logiciels de contrôle. C'est notamment le cas de la méthode de conception modulaire automatique AutoMoDe, pour laquelle les comportements d'exploration au sein des modules ont été arbitrairement sélectionnés. La question suivante est donc à se poser : Quelle est l'influence de différents comportements d'exploration sur la conception automatique d'essaims de robots ? Pour étudier cette problématique, nous introduisons AutoMoDe Coconut, une version alternative d'AutoMoDe Chocolate comportant plusieurs comportements d'exploration configurables intégrés dans ses modules. Nous mesurons les performances et l'utilisation des modules issues des deux versions d'AutoMoDe et les comparons afin de comprendre l'impact de ces nouveaux comportements d'exploration. Les résultats montrent que AutoMoDe Coconut n'est pas plus performant que AutoMoDe Chocolate, même dans les situations où les seuls comportements d'exploration disponibles pour AutoMoDe Chocolate sont désavantagés. Cela peut s'expliquer par la nature de la conception automatique, qui tend à générer des comportements d'exploration particuliers à la tâche donnée plutôt qu'à utiliser des comportements spéficiques prédéterminés.

Abstract

Studies on swarm robotics have shown that complex tasks can be solved by large groups of simple robots interacting with each other and their environment. Most of these tasks require the robots to explore their environment, making exploration a basic building block of the behaviors of robot swarms. Exploration behaviors were however not extensively evaluated on robot swarms, especially in automatic design of controllers. This is notably the case with the automatic modular design approach AutoMoDe, for which the exploration behaviors within the modules were arbitrarily selected. The following question is therefore studied: What is the influence of different exploration behaviors, such as random walks, on automatic design of robot swarms ? To tackle this problematic, we introduce AutoMoDe Coconut, an alternate version of AutoMoDe Chocolate with multiple configurable exploration behaviors embedded within its modules. We measure both the performances and the use of modules of the two versions of AutoMoDe and compare them in order to understand the impact of those new exploration behaviors. The results show that AutoMoDe Coconut does not perform better than AutoMoDe Chocolate, even in situations where the only exploration behaviors available to AutoMoDe Chocolate are at an apparent disadvantage. This can be explained by the nature of automatic design, which tends to generate custom exploration behaviors rather than use specific predetermined ones.

Contents

Chapter 1

Introduction

Swarms can be found under multiple forms in nature, be it in large crowds of flocking birds, complex ant colonies or fish schools. These different swarms of animals share the same core idea: the local interactions between a large group of simple agents and with their environment can lead to intelligent global behaviors [47]. Following the same idea, a swarm of robots is a large group of robots whose collective behavior results from local interactions of the robots between themselves and with their environment. Those robots operate without relying on any external structure or any form of centralized control [16]. Those characteristics make swarms of robots scalable, robust and flexible, allowing them to reach applications unattainable by single robots. Such applications often require the robots to explore their environment, be it to find specific targets or simply gather information about it.

The design of control software for swarm robotics is an inherently complex task. Indeed, there is no reliable way to anticipate the global behavior of a swarm of robots based on the behavior of a single robot of the swarm [22]. It is therefore common to resort to automatic design, for which multiple methods have been developed. One of those approaches is the automatic modular design, tackled with AutoMoDe [24]. In this method, the control software of the robots is a probabilistic finite state machine. The possible states are predetermined and consist in atomic behaviors, called modules, such as the exploration of the environment, the attraction to light or the repulsion from other robots.

However, the exploration behaviors used within the modules of AutoMoDe were chosen arbitrarily for the sake of simplicity. Indeed, the use of exploration behaviors is rarely discussed in swarm robotics, and was never studied in automatic design. One could therefore wonder whether or not injecting new exploration behaviors within the modules of AutoMoDe would increase its performances. In this context, we introduce AutoMoDe Coconut, a new version of AutoMoDe with modified modules allowing the use of new exploration behaviors already evaluated with swarm robotics in the literature: random walks.

We evaluate AutoMoDe Coconut on multiple missions and compare it with the original version of AutoMoDe both in terms of performances and behaviors used. The goal of this experiment with AutoMoDe is, beyond verifying if it increases its performances, to study the influence of random walks on automatic design of robot swarms.

This master thesis is structured as follows.

- The related work regarding swarm robotics, automatic design and exploration behaviors is mentioned in Chapter 2.
- The automatic modular design approach AutoMoDe is then detailed, along with exploration behaviors seen in the literature that could be used in automatic modular design, in Chapter 3
- AutoMoDe Coconut is introduced and detailed in Chapter 4.
- The experiments used to assess the performances of and behaviors generated by AutoMoDe Coconut are described in Chapter 5.
- The results of these experiments are shown and interpreted in Chapter 6.
- Finally, a brief summary of the findings of this master thesis is given along with some concluding remarks in Chapter 7.

Chapter 2

Related work

Recent discoveries in artificial intelligence (AI) have changed the way some complex problems have been tackled, from image recognition do decision models design. Those new technologies have greatly benefited robotics. In particular, AI has applications in the design of controllers in swarm robotics. Swarm robotics presents different properties than classical robotics, encouraging local interactions in a large group of simple robots rather than a complex unified system. Usually, those simple robots need to explore their environment in order to find key information necessary to solve a task. Therefore, the question of which exploration behaviors should be used in this context is relevant.

While the design of the controller of the robot is usually handled by humans, the complexity of manual design for swarm robotics has encouraged the development of automatic design methods. However, those methods encountered overfitting problems which led to the development to a new automatic modular design approach, AutoMoDe.

This chapter covers the following topics. First, the concepts of swarm intelligence and swarm robotics are defined and some of their applications are described (2.1). Different exploration behaviors used in swarm robotics are then presented (2.2). Finally, an overview of the applications and characteristics of automatic design in robot swarms is given (2.3) and the automatic modular design method AutoMoDe is presented (2.4).

2.1 Swarm robotics

Swarm intelligence, as its name implies, is based on the assumption that a large group (a swarm) of simple agents can show an intelligent behavior through the interactions of the agents with each other and with their environment [16]. This core idea was, for instance, applied in the Ant Colony Optimization algorithm [17] inspired from real ants, where agents search the solution space of complex

optimization problems for the best solution. While doing so, those agents indirectly interact with each other for a biased exploration of the solution space. Those algorithms are used, among other applications, for adaptive best-effort routing in IP networks with AntNet [12].

Swarm robotics is the application of the core idea of swarm intelligence to robotics, where the goal is not about exploring a solution space but working on a physical task. Indeed, swarm robotics studies the use of a swarm of simple physical agents (robots) interacting locally to show an intelligent global behavior. In order to be considered a swarm, a multi-robots system also needs to present some specific properties. The most recurring of these properties in the literature are fault tolerance, scalability, local interactions, decentralized system and cooperation [10][45][47].

2.1.1 Properties

A swarm of robots should be both fault tolerant and scalable. The fault tolerance is necessary to allow the whole system to keep working correctly even if one of more robots fail, which is more likely to happen with multiple robots. This property is helped by the swarm being redundant, with multiple robots assigned to the same function. The scalability allows the swarm to show no drastic change in performance with the addition or removal of robots to the system. This property is enabled thanks to the local communication and local sensing abilities of the robots. Indeed, as long as the density of the swarm doesn't change, the size of the swarm has no impact on the local behavior.

A swarm of robots should be completely decentralized, with no main controller for the whole system. Furthermore, it should not have access to any kind of global, centralized information. Otherwise, the system would not exploit the advantages of local interactions altogether and lose its scalability and fault tolerance properties. For example, a GPS or an internet access constitute a centralized single point of failure that should be avoided in a swarm of robots. Since there is no such centralized control or information to organize the behavior of the swarm, the individual behavior of the robots should aim for their cooperation to achieve the global goal.

Examples of applications of swarm robotics include a multi-agent construction system inspired by mound-building termites, encompassing three independent robots building a user-defined structure (Werfel et al. [53]). Another example is Swarmonoid (Dorigo et al. [15]), a complex heterogeneous system of cooperating robots including aerial eye-bots used for scouting, hand-bots used for grabbing objects and foot-bots used to move the hand-bots.

2.2 Exploration in swarm robotics

Exploration is an integral part of problem solving in swarm intelligence, whether it is in swarm optimization for the search of a solution in the solution space or in swarm robotics for the search of key elements to the task at hand. For a single robot, spatial information regarding its mission can simply be gathered by more powerful sensors and then be processed immediately. The limited range of the sensors of the robots in a swarm prevents this convenient behavior. This forces the robots to move around the experiment space in order to find this spatial information, relying on some kind of exploration behavior.

Therefore, different exploration behaviors have been studied in swarm robotics. The first exploration behaviors for robots were inspired by physics phenomenons, like the movement of particles in a fluid for the Brownian motion [20] or the movement of bacteria depending on their chemical environment for the chemotaxis [19]. An exploration behavior widely used in robotics is the Lévy flight or Lévy walk [55], a random walk with a heavy-tailed step length distribution. The Lévy walk is also used as part of more complex random walk methods, such as the Yuragi-based adaptive searching behavior [38], which also makes use of chemotaxis.

Most of these behaviors are designed for single robot exploration. While the literature regarding exploration behaviors used in the context of swarm robotics is sparse, some of the aforementioned methods were tested with a swarm of robots. For instance, bacteria chemotaxis has been used for target search with 30 robots in simulation by Yang et al. [54]. A more complete work on the subject has been conducted by Dimidov et al. [14] where different exploration behaviors were evaluated with a hundred Kilobots.

More recently, common exploration behaviors were also tested for mapping in a closed space with a swarm of ten E-Puck robots by Kegeleirs et al. [31]. There are also instances of those methods being specifically modified to work more efficiently when used for exploration by a swarm of robots. For instance, Pang et al. [39] presented a modified version of Lévy flight that increases the step length when the swarm density is high, avoiding repeated exploration of the same area in the process.

When exploring environments enclosed by walls or containing obstacles, exploration behaviors need to be used along with some obstacle avoidance method. Although they are often not considered in random walk experiments, many obstacle avoidance methods have been mentioned in the literature. Those methods are often pretty simple, like the vector field obstacle avoidance method [7][8] that modifies the trajectory of the exploring robots with repulsive forces representing the obstacles. Another example is the curvature-velocity method [46] that handles obstacle avoidance as an optimization problem.

2.3 Automatic design

In classical single-robot systems, the design process of the controller is often left to the human developer, the behavior of the robot being easy to derive from its task. In swarm robotics however, the link between the local behavior of the robots and the global behavior of the swarm is not that obvious. Indeed, it is difficult to anticipate the behavior of a swarm of robots solely based on the behavior of a single robot of the swarm [22]. This forces the manual design of controllers in swarm robotics into a trial and error process [10], which is often time consuming, prone to bias and errors and difficult to replicate [9].

It is therefore common in swarm robotics to resort mainly to two design approaches originally used for single-robot systems: the behavior-based approach and evolutionary robotics. In the behavior-based approach, the controller of the robots is created manually by a human developer. This process is based on trial and error and requires evaluating the performances of the swarm as a whole. With a control software based on a modular architecture, this method has been easily applied to swarm robotics [10]. This modular architecture of controlled design is also coupled with reinforcement learning methods by Parker et al. [40] and Mataric et al. [35]. While the application of reinforcement learning on this approach is successful to some extent, it encounters challenges such as the difficulty to reward individual contributions based on the global behavior or the large size of the state space.

2.3.1 Evolutionary robotics

Evolutionary robotics is an automatic design approach using artificial evolution to generate the control software of the robots [37]. The said control software takes the form of a neural network that takes as inputs the readings of the sensors of the robots and as outputs the values to be fed to the actuators of the robots. The neural network parameters and structure are optimized off-line in simulation for a specific mission by the evolutionary algorithm [48][22]. This approach was successfully used for the design of controllers in different types of missions. It was first used for coordinated movement in simulation by Quinn et al. [44], then with physical robots by Baldassarre et al. [2]. It was also used for the avoidance of holes with the help of phototaxis capabilities (Christensen et al. [13]), for the creation of communication networks with a swarm of aerial robots (Hauert et al. [28]) and for a synchronization behavior (Trianni et al. [49]).

Although the evolutionary robotics approach was used efficiently in various applications, the protocols used to apply it vary widely depending on the missions themselves. Indeed, the focus for most of these studies was to highlight the performances of evolutionary robotics in a particular situation, with a design method configured specifically for this situation. As a result,

those studies fail to assess the repeatability and robustness of the approach, as well as the impact of the reality gap [4].

2.3.2 Reality gap

The reality gap is the noticeable drop in performances of a robot controller designed in simulation when it is evaluated with actual robots. This difference is due to the fact that, unlike in simulation, there is a discrepancy between reality and how the robot perceives (through its sensors) and interacts (though its actuators) with it. The reality gap is particularly problematic for the off-line automatic design of controllers meant for applications with real robots. Indeed, the robot controllers designed to be efficient in simulation can perform poorly in reality. Therefore, some research has been conducted in order to decrease this performance drop and two approaches were tackled.

On one hand, some studies have tried to modify the simulated environment to have it match as much as possible with reality. For instance, Jakobi et al. [30] as well as Miglino et al. [36] fine-tuned the noise values to create a realistic simulated environment. Bongard et al. [6] made the simulator and the robot control software adapt to each other in a co-evolution approach. A similar method is used by Zagal et al. [56] as well. On the other hand, other studies have instead focused on making the controllers more resistant to the small changes in inputs and outputs that the transition from simulation to reality entails. The automatic modular design approach introduced by Francesca et al. [24][23] is a good example of this method. Besides, Jakobi et al. [29] attempt to modify the simulation to force the generated controllers to be more robust. Finally, Urzelai et al. [50] show that evolutionary adaptive controllers can adapt on-line to environmental changes without additional training.

According to Ligot and Birattari [33], the reality gap is not caused by the reality being too complex or the simulation being too simple. Indeed, the study manages to simulate the reality gap by changing the noise on environmental values in a new simulated environment. This simulated reality, called pseudo-reality, is not necessarily more complex than the simulation on which the controller is trained. The pseudo-reality becomes a tool to evaluate the robustness of a controller to the reality gap in simulation without relying on expensive and time consuming robots experiments.

The reality gap is actually a manifestation of a broader problem present in most fields of artificial intelligence, the bias-variance tradeoff. Indeed, the controller tends to overfit the particular conditions encountered during the design process in simulation. As long as the control software is refined in subsets of possible behaviors with higher representational power, the automatic design process will match it with the specifics of its simulated environment [21] and hence make the control software vulnerable to the reality gap.

2.4 Automatic modular design

In order to tackle the reality gap (and by extension overfitting) problem met in swarm robotics, Francesca et al. proposed the use of a new automatic modular design process: AutoMoDe [24]. The point of AutoMoDe is to inject a bias in the automatic design process by increasing the granularity of the architecture of the control software. This modification is meant to reduce overfitting of the solutions during the design process and reduce the reality gap for the generated controllers.

The approach of AutoMoDe generates modular robot controllers in the form of probabilistic finite state machines, composed of states and transitions. Those states are selected among a set of preexisting atomic behaviors called modules, while the transitions are also chosen among another set of preexisting conditional state transitions. This new approach yields good results comparing to a standard evolutionary approach called EvoStick and successfully overcomes the reality gap.

The modular design architecture of AutoMoDe for the controllers was arbitrarily restricted to finite state machines [24]. Indeed, other modular architectures exist and could just as well be used for modular design in swarm robotics. This possibility was explored by Kuckling et al. [32], who developed Maple, an alternate version of AutoMoDe that uses behavior trees instead of finite state machines with the same modules as the ones used in AutoMoDe. Maple yielded results similar to AutoMoDe and was able to cross the reality gap as well.

AutoMoDe was also extended with AutoMoDe Gianduja, a modification of the modules of AutoMoDe providing communication capabilities between the robots [27]. Those communication capabilities do not have fixed semantics, the conditions for sending a message and the effects of receiving one being determined during the design process. The results of the evaluations of AutoMoDe Gianduja showed that meaningful semantics were automatically associated to messages.

Chapter 3

AutoMoDe and exploration behaviors

AutoMoDe is an automatic modular design approach to the automatic design of robot swarm controllers. This new approach is successful in crossing the reality gap by avoiding overfitting the robot controller to the simulated environment. To do so, it restricts the architecture of the generated control software to probabilistic finite state machines using a preexisting set of states and transitions.

The preexisting states, or atomic behaviors, used by AutoMoDe embed some exploration behaviors necessary for the robots to work on their missions. Those behaviors were however chosen arbitrarily. Therefore, one could wonder if exploration behaviors existing in the literature could be used in the context of automatic modular design.

This chapter covers the following topics. At first, it describes more thoroughly the algorithm of AutoMoDe and its modules (3.1). Afterwards, it details exploration behaviors seen in the literature that could be used in automatic modular design (3.2).

3.1 AutoMoDe

AutoMoDe can be described by two important characteristics. First, by its modular design restricting the controller software of the robots to probabilistic finite state machines. Then, by its optimization algorithm, based on the racing approach and used to determine the best finite state machine during the design process.

The probabilistic finite state machines of AutoMoDe are composed of six preexisting atomic behaviors called modules and six preexisting conditional

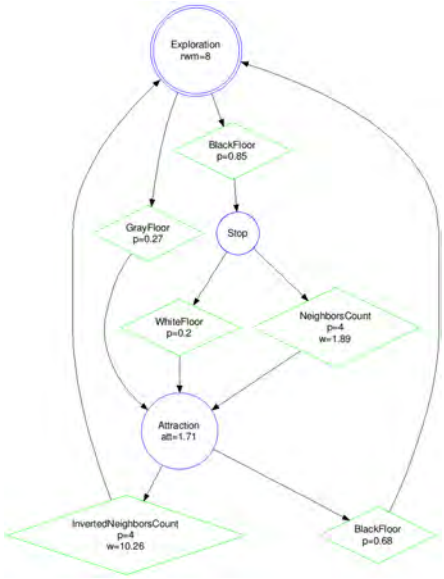


Figure 3.1: Finite state machine generated by AutoMoDe for an aggregation mission. The robot controlled by this finite state machine explores the environment and stops when it finds a black spot. It also tends to be attracted to its neighbors when it counts multiple surrounding robots.

state transitions. Each module is characterized by a set of parameters that can be fine tuned when selected during the design process. Multiple modules of the same type can appear in the same finite state machine, but not necessarily with the same parameter values. Just like the modules, the conditional state transitions have tunable parameters and can be used more than once in the same finite state machine. The solution space, and therefore the search space for the optimization algorithm, is composed of all the possible configurations of states and transitions with all the possible parameters. A finite state machine generated by AutoMoDe is shown in Figure (3.1) as an example.

3.1.1 The racing approach

The optimization algorithm used by AutoMoDe to determine the best finite state machine configuration is based on the racing approach. This approach consists in the evaluation of multiple candidate configurations using a systematic way to allocate the computational resources among them [5]. The evaluation of the candidates is performed step by step. At each step, or instance, all the candidates are evaluated in parallel, the inefficient ones being removed as soon as sufficient statistical evidence is gathered against them. The remaining candidates are re-evaluated during the next step. A visual representation

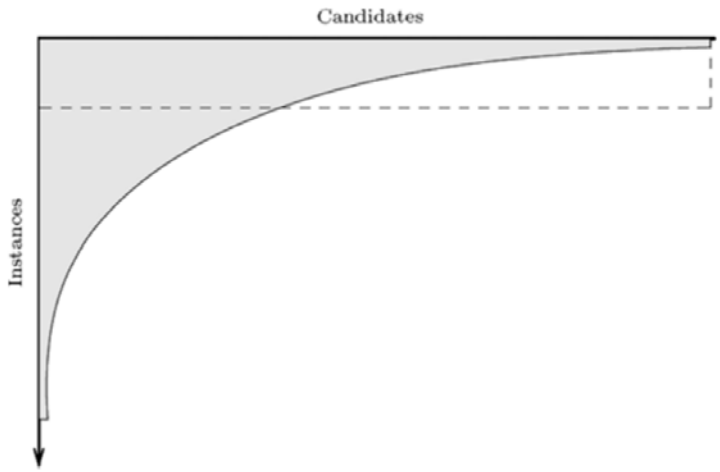


Figure 3.2: Representation of the number of candidates evaluated for each instance during the racing approach [5]. The dashed rectangle represents the same values for a brute-force approach. The surfaces of those two shapes represent the computational budget required to run the algorithm.

of the number of evaluated candidates for each instance is shown in Figure (3.2).

AutoMoDe Vanilla [24], the proof of concept of AutoMoDe, uses a racing algorithm called F-Race [3]. The particularity of F-Race is that it uses a ranking method to evaluate the candidates at each step of the evaluation, the Friedman test. While the F-Race algorithm was efficient enough to show the advantages of automatic modular design with AutoMoDe Vanilla, it failed to exploit the full potential of the modules. Indeed, human experts were evaluated as performing better than AutoMoDe with the same restrictions regarding the architecture of the controllers.

Therefore, a new version of AutoMoDe called AutoMoDe Chocolate [23] was developed with a new racing algorithm, I-Race [1][5]. I-Race, or iterated F-Race, is as its name suggests an algorithm consisting in a series of iterations, each of which is an execution of F-Race. In the first iteration, the candidate configurations are generated randomly and uniformly based on the set of all possible configurations. The candidates then undergo a first execution of the F-Race algorithm. At the end of this iteration, the surviving candidates are used as the seed to generate the new set of candidates used for the next generation.

Thanks to this new racing algorithm, AutoMoDe Chocolate outperformed both EvoStick and human designers in reality making it the state-of-the-art

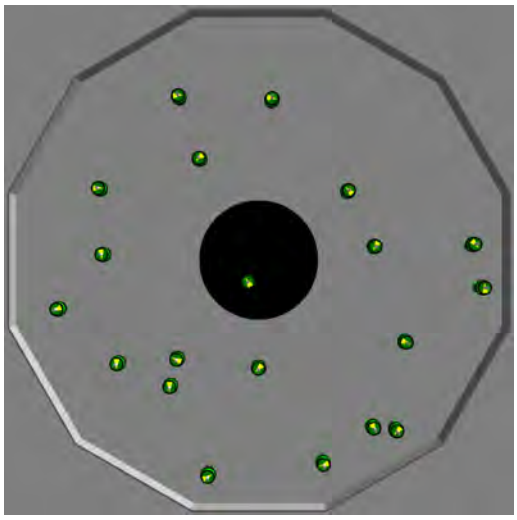


Figure 3.3: Example of aggregation mission. This mission takes place in a dodecagonal arena with 20 E-Puck robots.

AutoMoDe version. Furthermore, it successfully overcame the reality gap as well.

3.1.2 Design process

In summary, the design process of AutoMoDe Chocolate is described below. First, an experiment has to be manually designed by a human user. This experiment is the mission on which the controllers are going to be evaluated. AutoMoDe will therefore design a controller that is meant to perform efficiently on this particular experiment. Parameters of the experiment include but are not limited to: The spatial environment in which the experiment occurs or arena (walls, light sources, floor color), the number of robots in the experiment, the nature of the task to be accomplished by the robots (represented by an objective function). An example of mission is shown in Figure (3.3).

This experiment is then fed to AutoMoDe, which starts the I-Race algorithm. The I-Race algorithm works as explained above, evaluating the performances of the candidate finite state machines on the experiment. Once the computational power budget is depleted, AutoMoDe outputs the finite state machine that performed the best on the given experiment.

In order to evaluate the controllers during the design process, AutoMoDe needs to simulate the behavior of the robots resulting from the controller on the given experiment. To do so, it uses ARGoS, a popular multi-physics simulator

for swarm robotics experiments developed by Pinciroli et al. [43][42]. It can support multiple types of robots and handle large scale swarm simulations. AutoMoDe uses it along with the plugin "E-Puck for ARGoS" developed by Garattoni et al. [25] in order to model the E-Puck robot.

3.1.3 Modules of AutoMoDe

The six modules of AutoMoDe are the building blocks of the controllers it designs. Indeed, those modules are used as the states of the probabilistic finite state machines which constitute the controllers. Some of these behaviors have tunable parameters optimized by AutoMoDe. The available modules are the following [24]:

- *Exploration* : The robot moves in a straight line. When it encounters an obstacle, it turns on itself for a random number of control cycles uniformly chosen between $[0, \tau]$, where τ is a parameter of the module. The parameter τ is an integer in the range $[0, 100]$. The robot turns away from the direction of the obstacle. Note that a control cycle corresponds to a tenth of a second.
- *Stop* : The robot stops moving.
- *Phototaxis* : The robot moves towards the light source(s), or moves in a straight line if no light is perceived. This movement embeds a vector field obstacle avoidance. The robot follows the two-dimensional vector $w = w_l - k * w_o$, where w_l represents the perceived light vector, w_o the perceived obstacle vector and k a fixed value empirically set to 5. The light vector w_l is calculated with equation (3.1).

$$w_l = \sum_{i=1}^{8} (r_i, \angle b_i) \tag{3.1}$$

  r_i represents the reading of i, one of the eight light sensors of the robot and b_i the angle between this sensor and the front of the robot. The vector w_l therefore represents the average position of the sensed light(s) as the sum of the eight vectors corresponding to light readings. Similarly, the vector w_o represents the average position of the sensed obstacle(s) as the sum of the eight vectors corresponding to proximity readings.

- *Anti-phototaxis* : The robot moves away from the light source(s), or moves in a straight line if no light is perceived. This movement embeds a vector field obstacle avoidance. The robot follows the two-dimensional vector $w = -w_l - k * w_o$, where w_l, w_o and k are defined as in Phototaxis.
- *Attraction* : The robot moves towards the center of mass of the other robots in communication range, or moves in a straight line if no other robot

is detected. This movement embeds a vector field obstacle avoidance. The robot follows the two-dimensional vector $w = \alpha * w_{r\&b} - k * w_o$ where $w_{r\&b}$ represents the range and bearing vector, α is a parameter of the module and w_o and k are defined as in Phototaxis. The parameter α is an integer in the range $[1, 5]$. The range and bearing vector $w_{r\&b}$ is calculated with equation (3.2).

$$w_{r\&b} = \sum_{m \in M} (r_m, \angle b_m) \tag{3.2}$$

M is the set of messages received by the range & bearing module, r_m and b_m the range and bearing of the robot corresponding to the message m.

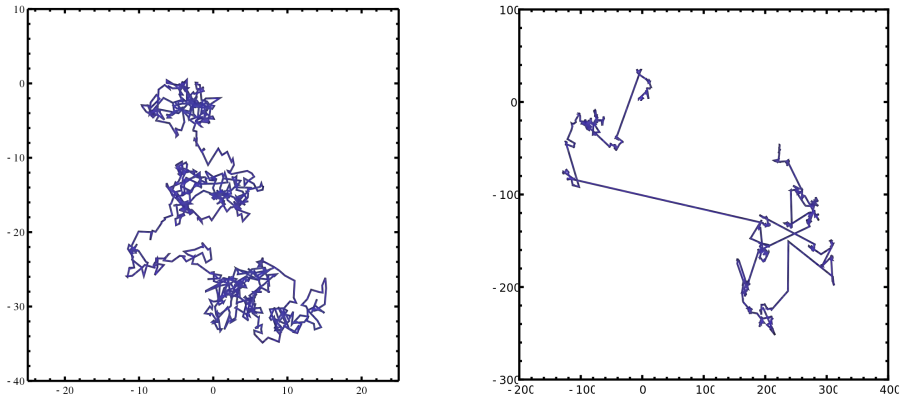
- *Repulsion* : The robot moves away from the center of mass of the other robots in communication range, or moves in a straight line if no other robot is detected. This movement embeds a vector field obstacle avoidance. The robot follows the two-dimensional vector $w = -\alpha * w_{r\&b} - k * w_o$ where $w_{r\&b}$, α, w_o and k are defined as in Attraction.

Within those modules, two different exploration behaviors stand out. The first and most obvious one is the Exploration module, consisting in a ballistic motion with random rotation obstacle avoidance. Ballistic motion corresponds to a straight-line movement. The second one is actually embedded withing four modules: Phototaxis, Anti-phototaxis, Attraction and Repulsion. Indeed, if no light source/no other robot is detected by a robot controlled by one of those behaviors, it moves in a straight line while avoiding obstacles. This behavior corresponds to a ballistic motion with vector field obstacle avoidance.

Those exploration behaviors within the modules of AutoMoDe were chosen arbitrarily. One could wonder if other exploration behaviors could have been selected instead of the ones currently used. Indeed, as shown in Section 2.2, multiple exploration behaviors have been studied in the context of swarm robotics, some of which might increase the efficiency of the exploration behaviors of AutoMoDe.

3.2 Exploration behaviors

While exploration behaviors are widely studied in classical robotics, only a few were evaluated in the context of swarm robotics. The exploration behaviors that are most fitted for swarm robotics are simple ones, considering the limited abilities of the robots in the swarm. For that reason, the studied behaviors were ballistic motion and different variations of random walk. Indeed, correlated random walk and Lévy walk were compared with a swarm on a hundred Kilobots [14]. More recently, more random walk variations and ballistic motion were tested for mapping with ten E-Pucks. [31]. Those variations were Brownian motion, correlated random walk, Lévy walk and Lévy taxis.



(a) Browmian motion [52].

(b) Lévy walk [51].

Figure 3.4: Comparison of Brownian motion and Lévy walk with two visual examples. While the former constitutes a completely random movement, the latter makes occasional jumps, covering further areas. The motion starts at the coordinates $[0, 0]$ and lasts 1000 steps.

- *Brownian motion* : This motion is inspired by the movement of a particle in a fluid [20]. A particle in a fluid moves in a straight line but constantly bumps into other particles, changing its trajectory. This motion can be classified as an uncorrelated random walk, meaning that it has no bias and is completely unpredictable, unlike the other three behaviors.
- *Correlated random walk* : This motion is a random movement where the next direction of the trajectory is biased towards the previous one [11]. This results in a motion that encourages thoroughly exploring a local area rather than moving further to discover new areas.
- *Lévy walk* : This motion corresponds to a random movement similar to the Brownian motion [55]. However, it includes occasional large movements in a straight line, or jumps. This results in an efficient exploration of large environments along with some local deeper exploration.
- *Lévy taxis* : This motion, proposed by Pasternak et al. [41], is a balance between a correlated random walk and a Lévy walk. Therefore, it offers the advantages of the two methods, being efficient both in local and global exploration of the environment.

3.2.1 Algorithms

Those four random walk variants are based on the same algorithm. When started, the robot walks for a random number of steps M_l (move length) and with a random rotation angle T_a, both calculated from a fixed probability

distribution. Once the robot has crossed the distance corresponding to the move length, it calculates a new move length and turning angle based on the same distributions. The differences between Brownian motion, correlated random walk, Lévy walk and Lévy taxis lie in the choice of those probability distributions, determined by key parameters μ and ρ.

Therefore, despite their significant differences, those four methods share the same mathematical model with different parameters. The move length M_l is calculated with equation (3.3), with L_{min} representing the minimum move length, r a random variable chosen in the range $[0, 1]$ and μ the key parameter governing the distribution.

$$M_l = L_{min} * r^{\frac{1}{1-\mu}} \tag{3.3}$$

The turning angle T_a is calculated with equation (3.4), with r representing a random variable chosen in the range $[0, 1]$, *bias* the directional bias added to the turning angle and ρ the key parameter governing the distribution.

$$T_a = \left[2 * arctan\left(\frac{1-\rho}{1+\rho} * tan\big(\pi * (r - 0.5)\big)\right)\right] + bias \tag{3.4}$$

The two key parameters μ and ρ are either fixed or within the same value range depending on the random walk variant. For instance, correlated random walk has the parameter μ fixed to 0, making the move length distribution asymptotically Gaussian-like and the parameter ρ contained within the range $[0, 1]$, making the turning angle distribution a wrapped Cauchy. All the ranges of the key parameters for the different random walk variants are detailed in Table (3.1).

	Move length	μ	Turning angle	ρ
Brownian motion	Asymptotically Gaussian-like	3	Uniform	0
Correlated random walk	Asymptotically Gaussian-like	3	Wrapped Cauchy	$\in [0, 1]$
Lévy walk	Power law	$\in]1, 3]$	Uniform	0
Lévy taxis	Power law	$\in]1, 3]$	Wrapped Cauchy	$\in [0, 1]$

Table 3.1: Different value ranges of the key parameters of the random walks, along with the corresponding distributions [18].

It is interesting to note that the Lévy taxis method is a generalization of the three other random walks. Indeed, it can be reduced to any other random walk method given the right values of the key parameters μ and ρ. Lévy taxis is therefore the most complete of the four since it encompasses all of them.

Ballistic motion is what could be considered as the most simple movement method. The robot controlled by ballistic motion simply moves in a straight line without ever stopping. In order to be actually useful, this motion needs to be coupled with an obstacle avoidance method. It was evaluated with two. First with the vector field obstacle avoidance method that modifies the trajectory of the exploring robot with repulsive forces representing the obstacles. Then with the random rotation method that makes the exploring robot turn on itself for a random amount of time upon encountering an obstacle.

Chapter 4

AutoMoDe Coconut

As explained in Section 2.2, exploration is a primary behavior in swarm robotics. Indeed it plays a role in the completion of most tasks requiring interactions with elements of the environment, acting as building blocks of the individual behaviors of the robots. Therefore, exploration has a significant role to play within the modules of AutoMoDe.

Indeed, as shown in Section 3.1 AutoMoDe embeds two types of exploration behaviors within its modules. However, those exploration behaviors were chosen arbitrarily, for the sake of simplicity. Moreover, most relevant research regarding random walks in swarm robotics were published after AutoMoDe. In this context, we introduce AutoMoDe Coconut, an alternate version of AutoMoDe Chocolate with multiple configurable exploration behaviors embedded within its modules.

This chapter covers the problematic regarding the exploration behaviors in AutoMoDe and the options considered to answer it (4.1). This discussion is followed by the description of AutoMoDe Coconut and its modified modules (4.2).

4.1 Problematic

Two exploration behaviors stand out within the six modules of AutoMoDe: ballistic motion with random rotations obstacle avoidance and ballistic motion with vector field obstacle avoidance. Those exploration behaviors within the modules of AutoMoDe were arbitrarily chosen for the sake of simplicity. Since then, multiple exploration behaviors have been evaluated with swarm robotics as shown in Section 3.2. Therefore, the following question becomes relevant to investigate: How would new exploration behaviors embedded within the modules of AutoMoDe impact its performances ? In particular, it would be

interesting to know how efficient the ballistic motions and random walks are in AutoMoDe.

In order to answer this question, a modification of the modules of AutoMoDe is necessary. There is however several ways to implement new exploration behaviors within the modules of AutoMoDe. For that reason, multiple implementation options were considered.

4.1.1 Implementation choice

The first attempt was simply to add a new module for each new exploration behavior. However, this option does not change anything about the ballistic motion embedded within the Phototaxis, Anti-phototaxis, Attraction and Repulsion modules. Therefore, another idea was examined: It involves the replacement of all the exploration behaviors in AutoMoDe with a specific exploration behavior, either a ballistic motion or a random walk. While this choice allows to directly measure the performance of an exploration behavior in AutoMoDe, manually selecting the said exploration behavior injects a bias in the automatic design process. Furthermore, it doesn't allow the combination of multiple different exploration behaviors within the same controller.

A last option was then considered. It consists in directly embedding all the new exploration behaviors within all the modules, through a tunable module parameter governing the type of exploration behavior used by the robot. This option allows controllers to use any of the new exploration behaviors either as a pure atomic behavior through the Exploration module, or as part of another atomic behavior through the Phototaxis, Anti-phototaxis, Attraction or Repulsion modules. It was therefore chosen to be implemented as an alternate version of AutoMoDe Chocolate, AutoMoDe Coconut.

4.1.2 Exploration behaviors

The new exploration behaviors chosen to be implemented in AutoMoDe Coconut were the ballistic motion with random rotation obstacle avoidance, the ballistic motion with vector field obstacle avoidance and the Lévy taxis random walk. The two ballistic motion methods are already used as exploration behaviors in AutoMoDe Chocolate, the one with random rotation obstacle avoidance in the Exploration module and the one with vector field obstacle avoidance within the Phototaxis, Anti-Phototaxis, Attraction and Repulsion modules. Making them available for all modules (except the Stop module) allows AutoMoDe Coconut to have the exact same modules as AutoMoDe Chocolate given the right parameters. Furthermore, it helps in assessing the use of those exploration behaviors both as stand-alone modules and as part of other modules.

The Lévy taxis random walk was chosen to be implemented in AutoMoDe Coconut since it encompasses the four random walk methods (presented in Section 3.2) given the right set of parameters. It is therefore the most general type of random walk among the ones mentioned in this work. Further mentions of random walk will refer to Lévy taxis.

4.2 AutoMoDe Coconut

AutoMoDe Coconut is a new version of AutoMoDe based on AutoMoDe Chocolate. Besides its modified modules, it is the same automatic modular design process as AutoMoDe Chocolate. The point of this new version is to assess the performances of AutoMoDe with new exploration behaviors embedded within its modules.

In AutoMoDe Coconut, each module (except the Stop module) has a new tunable parameter ϵ governing the type of exploration behavior to be used. The parameter ϵ is categorical and is represented by an integer in the range $[0, 2]$. If $\epsilon = 0$, the exploration used is the ballistic motion with vector field obstacle avoidance. If $\epsilon = 1$, the exploration used is the ballistic motion with random rotation obstacle avoidance. In this configuration, an additional parameter to be tuned is added, the τ parameter governing the maximal number of rotations done by the robot. If $\epsilon = 2$, the exploration used is the random walk. In this configuration, two additional parameters to be tuned are added, the μ and the ρ parameters. They govern the distributions of the move length and turning angle for Lévy taxis. For more details about those parameters, please refer to Section 3.2.1.

The resulting new modules for AutoMoDe Coconut are the following (please refer to Section 3.1.3 for more details about the original modules):

- *Exploration* : The robot explores the environment with an exploration behavior depending on ϵ, a parameter of the module. The parameter ϵ is an integer in the range $[0, 2]$.

  If $\epsilon = 0$, the exploration behavior is a ballistic motion with vector field obstacle avoidance. The robot follows the two-dimensional vector $w = w_b - w_o$, where w_b represents the ballistic vector and w_o the perceived obstacle vector. While the ballistic vector is trivial, defined as $w_b = (1, \angle 0)$, the obstacle vector w_o is calculated with equation (4.1).

$$w_o = \sum_{i=1}^{8} (r_i, \angle b_i) \tag{4.1}$$

Where r_i represents the reading of i, one of the eight proximity sensors of the robot and b_i the angle between this sensor and the front of the robot. The vector w_o therefore represents the average position of the sensed obstacle(s) as the sum of the eight vectors corresponding to proximity readings.

If $\epsilon = 1$, the exploration behavior is a ballistic motion with random rotation obstacle avoidance. The robot moves in a straight line. When it encounters an obstacle, it turns on itself in the opposite direction for a random number of control cycles uniformly chosen between $[0, \tau]$, where τ is a parameter of the module. The parameter τ is an integer in the range $[0, 100]$.

If $\epsilon = 2$, the exploration behavior is the random walk. The robot follows the two-dimensional vector $w = w_{Lt} - w_o$, where w_{Lt} represents the Lévy taxis vector and w_o is defined as in the ballistic motion with vector field obstacle avoidance. The Lévy taxis vector is calculated as $w_{Lt} = (1, \angle T_a)$ where T_a is the turning angle defined by equation (3.4). This Turning angle changes after a number of control cycles governed by the movement length M_l defined by equation (3.3). These equations depend on the parameters μ and ρ, both are parameters of the module. The parameter μ is real-valued and chosen in the range $]1, 3]$. The parameter μ is real-valued as well and chosen in the range $[0, 1]$. Please refer to Section 3.2.1 for more details about the Lévy taxis algorithm.

- *Stop* : The robot stops moving.
- *Phototaxis* : The robot moves towards the light source(s) or, if no light is perceived, adopts an exploration behavior depending on ϵ, a parameter of the module. If some light is perceived, the behavior is exactly the same as in the original Phototaxis module. If no light is perceived, the behavior corresponds to the Exploration module.
- *Anti-phototaxis* : The robot moves away from the light source(s) or, if no light is perceived, adopts an exploration behavior depending on ϵ, a parameter of the module. If some light is perceived, the behavior is exactly the same as in the original Anti-phototaxis module. If no light is perceived, the behavior corresponds to the Exploration module.
- *Attraction* : The robot moves towards the center of mass of the other robots in communication range or, if no other robot is detected, adopts an exploration behavior depending on ϵ, a parameter of the module. If other robots are detected, the behavior is exactly the same as in the original Attraction module. If no other robot is detected, the behavior corresponds to the Exploration module.

- *Repulsion* : The robot moves away from the center of mass of the other robots in communication range or, if no other robot is detected, adopts an exploration behavior depending on ϵ, a parameter of the module. If other robots are detected, the behavior is exactly the same as in the original Repulsion module. If no other robot is detected, the behavior corresponds to the Exploration module.

Chapter 5

Experiments

In order to assess the performance impact of the new exploration behaviors allowed by the new modules of AutoMoDe Coconut, this version of AutoMoDe should be compared to AutoMoDe Chocolate on which it is based. To do so, controllers are automatically designed by the two versions of AutoMoDe for a set of specific missions. The performances of the controllers are then assessed on those same missions. Missions are challenges to be handled by a swarm of simple robots, such as aggregating on a colored spot or exploring the environment.

This chapter covers the following topics. First, it describes the evaluation protocol used to compare AutoMoDe Chocolate and AutoMoDe Coconut (5.1). Then, it presents the fours missions used as the basis of the experiments (5.2). Finally, it details the experimental setup of the experiments (5.3).

5.1 Evaluation protocol

The choice of the missions is motivated by the need to challenge both the general problem-solving performances of the two versions and their exploration capabilities. Therefore, the missions consist in missions already used to test other versions of AutoMoDe as well as new missions specifically targeting the exploration capabilities offered by the new exploration behaviors. The chosen missions are AGGREGATION, FORAGING, FAST POINT FINDING and GRID EXPLORATION.

The evaluation protocol is the following. AutoMoDe Coconut and AutoMoDe Chocolate are both executed ten times on each of the four missions with a budget of 100.000 evaluations. This design process produces ten controllers per mission and version of AutoMoDe. Each of these controllers is then evaluated once on his respective mission. The results of these evaluations are then

aggregated for each mission. The performance of the two version of AutoMoDe can then be compared based on the performances of the controllers on each mission.

The performance of the controllers is assessed not only in simulation, but also in pseudo-reality. Pseudo-reality is another simulated environment with modified noise on environmental values. With this modification, the pseudo-reality helps in evaluating the robustness of the controller to the reality gap, as explained in Section 2.3.2.

The evaluation of the performances of the controllers on each mission is represented by notched box plots. For each mission, the score obtained in simulation and in pseudo reality for AutoMoDe Chocolate and AutoMoDe Coconut is reported. Statements about the relative performances of the two versions of AutoMoDe on a specific mission are supported by the confidence intervals of those box plots. The evaluation of the aggregated performances of the controllers over all of the missions is represented by a Friedman test comparing AutoMoDe Chocolate and AutoMoDe Coconut. Once again, statements about the relative performances of the two versions of AutoMoDe are supported by the confidence intervals of this test. Any statement like "A performs *significantly* better/worse than B" means that the confidence intervals of the box plots of the scores obtained or the Friedman test for A and B do not overlap.

Notched box plots show relevant information regarding the data distribution. The minimum and maximum values are represented by horizontal bars. The interquartile is defined by the upper and lower limits of the box. The "notch" where the box becomes thinner represents the 95% confidence interval of the median. The middle of the box, where it is the thinnest, is the median. Furthermore, possible outliers are represented by small circles.

Besides the performances dictated by the objective function of the mission, other information can prove useful. In order to interpret those performances, one needs to have some insight into the modules used by the two versions of AutoMoDe for the different missions. Two ways to measure the use of the different modules during a mission are used.

The first one consists in counting, for each module, the proportion of controllers using it in their finite state machine. While this measurement gives some information about the finite state machines and the behavior of the controller, it also shows modules that might not actually be used at run-time. Indeed, some states of the finite state machines can be bypassed completely by high-probability transitions, making them useless in the controller.

Therefore, the second measurement is the average (across all of the robots of the swarm and all controllers of the mission) of the proportion of time each robot uses the behavior of each module. While this measure gives a

much better idea of the actual use of the different modules at run-time, it fails to differentiate important modules used for a short time and useless modules used as transitions. For that reason, the two measurements are needed.

5.2 Missions

The missions were chosen as to allow AutoMoDe Coconut to exploit the exploration behaviors offered by its new modules. The four missions are AGGREGATION, FORAGING, FAST POINT FINDING and GRID EXPLORATION. Each mission takes place in a dodecagonal arena of 4.91 m^2 surrounded by walls. Those walls are tall enough to prevent the robots from seeing anything beyond them. The floor is gray with the exception of black or white areas specific to each mission. All missions are performed by a swarm of $N = 20$ E-Puck robots for a duration $t = 120$ seconds. In the following descriptions, the coordinates are in meters with the origin of the axes at the center of the arena. The x axis is parallel to a wall of the arena and the y axis perpendicular to it. As a reference for Figure (5.1), the x axis points right and the y axis point up. The four missions are detailed below.

- AGGREGATION : In this mission, the robots need to aggregate as fast as possible on a black spot at the center of the arena. The floor of the arena is completely gray except for a black circular area of diameter 0.60 m at its center. At the beginning of the experiment, the 20 robots are randomly placed in the whole arena. Figure (5.1a) shows the arena used for this mission. The performance of the swarm is measured by the sum of the time spent, in seconds, by each robot in the black area during the whole duration of the mission. The corresponding objective function is defined by equation (5.1).

$$F_{aggregation} = \sum_{i=1}^{N} T_i \tag{5.1}$$

  Where $N = 20$ is the number of robots and T_i is the aggregated time spent in the black area by the robot i during the whole duration of the mission.

- FORAGING : In this mission, the robots need to retrieve as many objects as possible from two sources and drop them in a specific area, the nest. The sources and nest are represented respectively by two black spots and a white area. The two black spots are black circular areas of diameter 0.30 m located at the coordinates $(0, 0.75)$ and $(0, -0.75)$. The white area covers the whole area of the arena where $x > 0.60$. Moreover, a light source is placed behind the nest at the coordinates $(1.25, 0)$ at 0.75 m from the ground. Figure (5.1b) shows the arena used for this mission. Since the E-Puck robot doesn't have grasping capabilities, the transportation of objects is abstracted. Therefore, it is supposed that a robot grabs an

object (if he isn't already holding one) when it enters a source and drops the object (if he has one) when it enters the nest. At the beginning of the experiment, the 20 robots are randomly placed in the whole arena. The performance of the swarm is measured by the sum of the number of objects retrieved by each robot, abstracted as source-nest trips, during the whole duration of the mission. The corresponding objective function is defined by equation (5.2).

$$F_{foraging} = \sum_{i=1}^{N} No_i \tag{5.2}$$

Where $N = 20$ is the number of robots and No_i is the number of objects retrieved by the robot i.

- FAST POINT FINDING : In this mission, the robots are supposed to find an objective point in the top left of the arena as fast as possible. The floor of the arena is completely gray except for the objective point, a white circular area of diameter 0.20 m at the coordinates $(-0.8, 0.6)$. At the beginning of the experiment, the 20 robots are randomly placed in the right-hand half of the arena so that no robot starts on the objective point. Figure (5.1c) shows the arena used for this mission. The performance of the swarm is measured by the time left after the objective point has been found by a robot. The corresponding objective function is defined by equation (5.3).

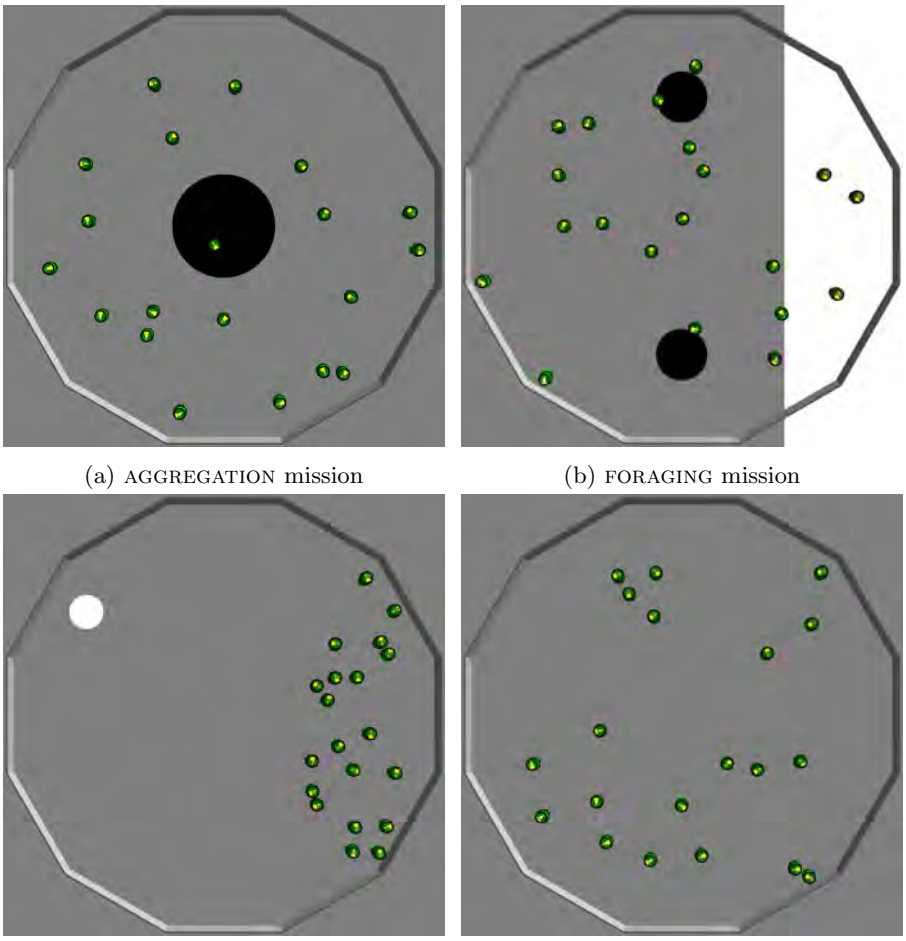
$$F_{fastpointfinding} = t_{tot} - t_{found} \tag{5.3}$$

Where t_{tot} is the duration of the experiment and t_{found} is the time taken by the swarm of robots to find the objective point.

- GRID EXPLORATION : In this mission, the robots are supposed to explore and cover as much space as possible in the arena. The floor of the arena is completely gray. At the beginning of the experiment, the 20 robots are randomly placed in the whole arena. Figure (5.1d) shows the arena used for this mission. In order to measure the performance of the swarm, the arena is divided in a grid of 10 tiles by 10 tiles. For each tile is retained the time tc elapsed since the last time it was crossed by a robot. Each time the tile is crossed by a robot, this time is reset to 0. The performances of the swarm are measured by the sum over all control cycles of the opposite of the average time tc over all the tiles. The corresponding objective function is defined by equation (5.4).

$$F_{gridexploration} = \sum_{i=1}^{N_{cc}} \left(\frac{1}{N_t} \sum_{j=1}^{N_t} -tc_{i,j} \right) \tag{5.4}$$

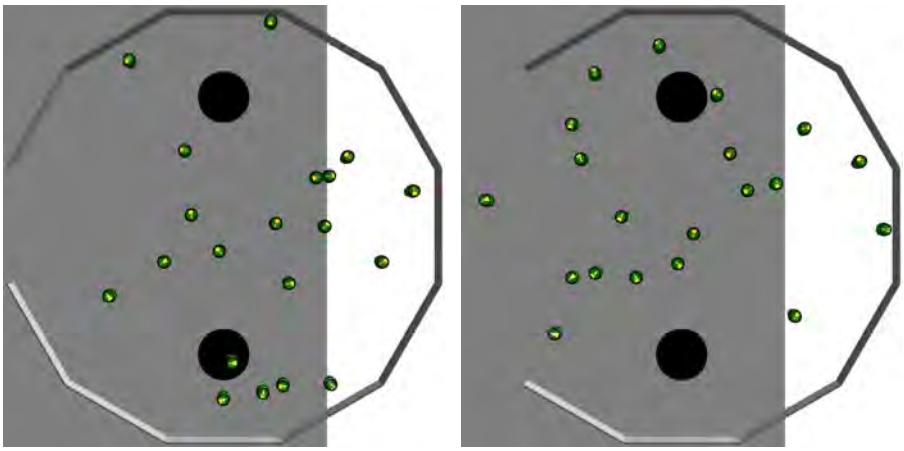
Where N_{cc} is the number of control cycles for the whole experiment, N_t is the number of tiles and $tc_{i,j}$ is the time, at the control cycle i, since the tile j was crossed by a robot.



(a) AGGREGATION mission (b) FORAGING mission

(c) FAST POINT FINDING mission (d) GRID EXPLORATION mission

Figure 5.1: Visualization of the arenas used for the four missions, along with an example of initial positions for the robots.



(a) FORAGING with one missing wall. (b) FORAGING with three missing walls.

Figure 5.2: Visualization of the arenas used for the FORAGING mission with one and three missing walls.

These missions were selected since, a priori, exploration should have a different role in each of them. Indeed, it is reasonable to assume that AGGREGATION and FORAGING require the use of exploration as part of a more complex behavior, and that better exploration methods should lead to an increase in performance. Likewise, it is reasonable to expect better performances in FAST POINT FINDING and GRID EXPLORATION since these missions should rely almost exclusively on exploration.

Additionally, each of these missions was modified into two additional versions, one in which one wall is missing from the arena and the other one in which three walls are missing. The class of the missions, so far defined within closed walls, is therefore changed. It is considered here that any robot exiting the arena is lost, and therefore removed from the experiment. The point of those alternate versions of the missions is to study the use and performances of the different exploration behaviors when exploration becomes a dangerous endeavor for the robots.

Specifically, the walls removed are the same for all the experiments. When one wall is removed, it is the leftmost wall of the arena located at the coordinates $(-1.25, 0)$. When three walls are removed, they are the leftmost wall of the arena located at the coordinates $(-1.25, 0)$ and the two walls adjacent to it. The FORAGING missions with one and two missing walls are shown in Figure (5.2).

5.3 Experimental setup

This section details the specific values of the parameters used for the evaluation process and the specific tools used to represent the results of this evaluation. The formalization of the capabilities of the E-Puck used to perform the experiments is shown by the reference model in Table (5.1). The noise applied to the sensor readings and actuator values for simulation and pseudo-reality is detailed in Table (5.2).

sensor / actuator	variables
proximity	$prox_i \in [0,1]$, with $i \in \{1,2,...,8\}$
light	$light_i \in [0,1]$, with $i \in \{1,2,...,8\}$
ground	$ground_i \in \{white, gray, black\}$, with $i \in \{1,2,3\}$
range-and-bearing	$n \in [0,20]$ $r_m \in [0,0.70]$, with $m \in \{1,2,...,20\}$ $b_m \in [0,2\pi]$ rad, with $m \in \{1,2,...,20\}$
wheels	$v_l, v_r \in [-0.12, 0.12]$ m/s

Table 5.1: Reference model RM1 of the E-Puck robot [26]. RM1 abstracts sensors and actuators by defining the input and the output variables that are made available to the control software at each control step. Sensors are defined as input variables: the control software can only read them. Actuators are defined as output variables: the control software can only write them. Input and output variables are updated with a period of 100 ms.

sensor / actuator	$Simulation$	$Pseudo-reality$
proximity	0.05	0.05
light	0.05	0.90
ground	0.05	0.05
range-and-bearing	0.85	0.90
wheels	0.05	0.15

Table 5.2: Noise applied to the sensor readings and actuator values for both simulation and pseudo-reality [33]. The values for the proximity, light, and ground sensors determine the upper bounds of symmetric ranges for uniform white noises. The value for the range-and-bearing sensor is the probability of failing to detect a neighboring peer. The value for the wheels actuator is the standard deviation of a Gaussian white noise with mean 0.

Chapter 6

Results and discussion

AutoMoDe Coconut was evaluated and compared with AutoMoDe Chocolate according to the missions described in Section 5.2. The experiments were executed as explained in Section 5.1 with the parameters detailed in Section 5.3. This section presents the results obtained from those experiments.

The results presented consist in the performances of the two versions of AutoMoDe for each mission both in simulation and in pseudo-reality. Furthermore, the average use of the different modules across the controllers for each mission is shown for both AutoMoDe Chocolate and AutoMoDe Coconut. For the latter, the exploration behavior used by each module is also detailed. The use of the different modules is measured by both their presence within the finite state machines of the controllers and their relative time of use during the missions.

This chapter is structured as follows. First, the results of the experiments corresponding to the missions in a closed arena are presented and discussed (6.1). Then, the impact of the removal of one or multiple walls of the arena is shown for each mission (6.2).

6.1 Experiments in closed environment

6.1.1 Performances comparison

The performances of AutoMoDe Chocolate and AutoMoDe Coconut are presented in Figure (6.1). For the AGGREGATION mission, the performances are generally the same between the two versions of AutoMoDe. AutoMoDe Coconut seems to perform slightly better for the FAST POINT FINDING mission but since the 95% confidence intervals overlay, it cannot be said to be significantly better than AutoMoDe Chocolate. Those intervals do not overlay for the GRID EXPLORATION mission, where AutoMoDe Coconut performs significantly worse

than AutoMoDe Chocolate. For the FORAGING mission however, AutoMoDe Coconut performs significantly better than AutoMoDe Chocolate.

It should be noted that the scores measured in simulation and in pseudo-reality for the same version of AutoMoDe are very similar for all missions. Indeed, the scores cannot be said to be significantly worse in pseudo-reality compared to simulation. This can be interpreted as a small reality gap, meaning that the controllers designed by AutoMoDe Chocolate and AutoMoDe Coconut should perform similarly in reality.

The aggregated performances of AutoMoDe Chocolate and AutoMoDe Coconut across all missions are compared using the Friedman test. The results of this tests are represented in Figure (6.2). They show that while AutoMoDe Coconut has a slight edge thanks to its good performances in the FORAGING mission, the global performances between the two versions of AutoMoDe are not significantly different.

6.1.2 Modules use comparison

To interpret those results, one needs to have some insight into the modules used by the two versions of AutoMoDe for the different missions. Besides, in order to study the use of the different exploration behaviors within the modules of AutoMoDe, the particular exploration behavior used along with each module of AutoMoDe Coconut needs to be known. Therefore, the average use of each module of AutoMoDe Chocolate is measured and shown below. Likewise, the average use of each module of AutoMoDe Coconut is detailed along with the actual exploration behaviors used. The exploration behaviors used by AutoMoDe Chocolate are not detailed since they are not parameters of the modules. As a reminder, the exploration behaviors used in AutoMoDe Chocolate are a ballistic motion with rotation obstacle avoidance in the Exploration module and a ballistic motion with vector field obstacle avoidance in all the other modules (except the Stop module).

For instance, the modules use for the AGGREGATION mission is shown for the two versions of AutoMoDe in Figure (6.3). The graph (6.3a) shows that all controllers of AutoMoDe Chocolate use the Exploration, Stop and Attraction modules. Ten percent of the controllers used the Repulsion module, but the graph (6.3b) reveals that the module is never actually used. This second graph also shows that the robots spend 57% of their time using the Stop module, probably stopped on the black spot of the arena. They spend 31% of their time looking for the black spot using the Exploration module and the remaining 12% of their time using the Attraction module.

The graphs regarding AutoMoDe Coconut show that its finite state machines are very similar to the ones of AutoMoDe Chocolate. Indeed, while it uses

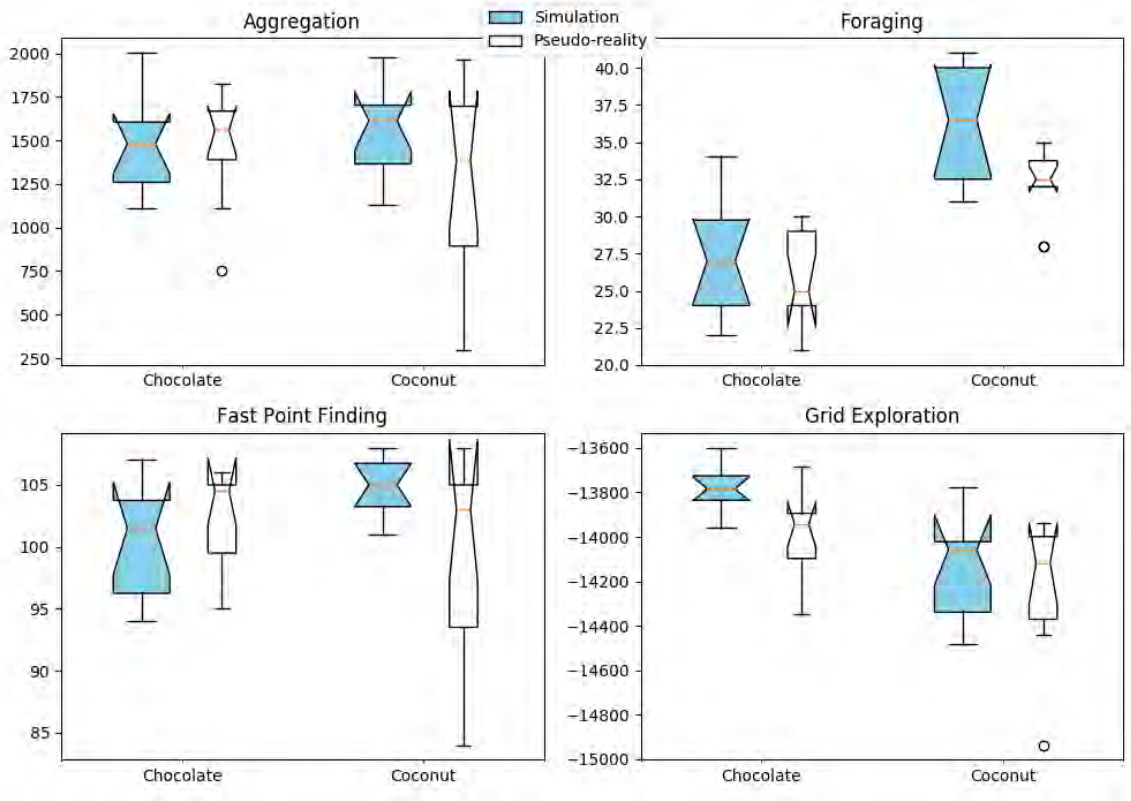


Figure 6.1: Performances across all missions for the experiments in closed arena. Each graph represents the score obtained by all controllers of AutoMoDe Chocolate and AutoMoDe Coconut in simulation and pseudo-reality on a particular mission. Higher is better.

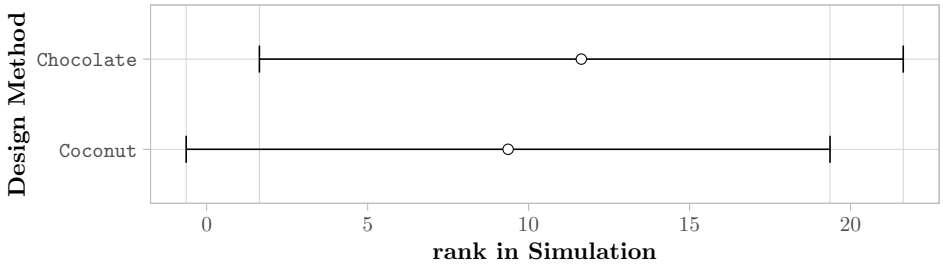
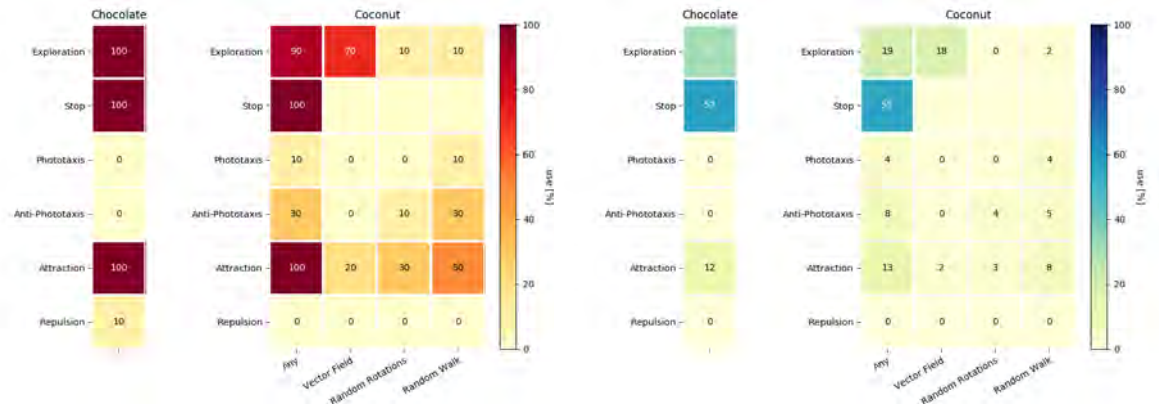


Figure 6.2: Friedman test for the enclosed experiments. Lower is better.



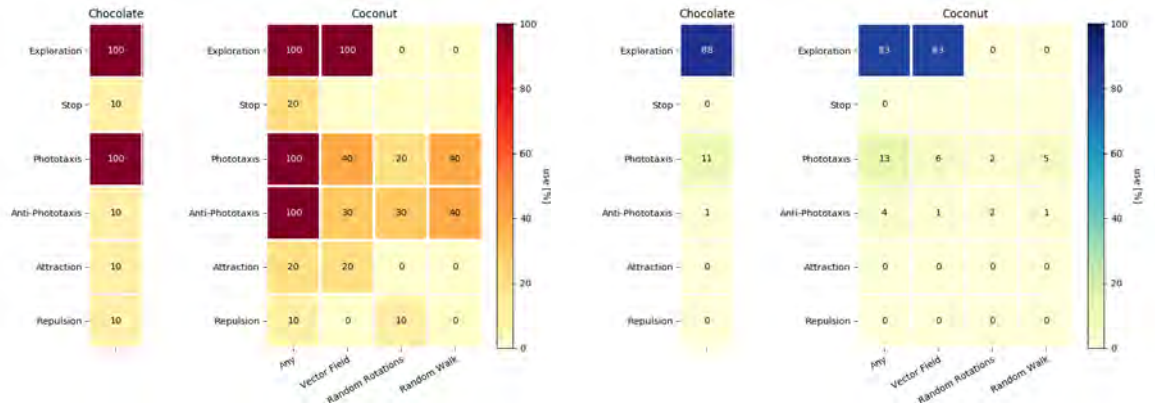
(a) Modules use in the finite state machines (b) Modules use during the mission

Figure 6.3: Use of the modules for AGGREGATION in a closed arena.

the Exploration behavior a little less, it compensates with the Phototaxis and Anti-phototaxis ones which, without light (as is the case in the AGGREGATION mission), act as the Exploration module. The exploration behaviors used are mostly ballistic motion with vector field obstacle avoidance, along with some ballistic motion with random rotation obstacle avoidance. This small change in the exploration behaviors compared to AutoMoDe Chocolate is not significant enough to have an impact on the performances. This explains why AutoMoDe Chocolate and AutoMoDe Coconut perform similarly on the AGGREGATION mission.

The case of the FORAGING mission is interesting since it presents very different behaviors between AutoMoDe Chocolate and AutoMoDe Coconut. While the graph (6.4b) seems to indicate that the two versions of AutoMoDe present the same behavior since the use percentages of the different modules are very similar, the graph (6.4a) shows that there is a big difference between the two versions of AutoMoDe in terms of finite state machines. Indeed, while the use of Anti-phototaxis seems equally negligible for AutoMoDe Chocolate and AutoMoDe Coconut (with 1% and 4% of use respectively), the former only has 10% of its controllers including Anti-phototaxis and the latter has all of its controllers including this module.

Even though it is not used for long periods of time, the Anti-phototaxis module is extremely useful in the FORAGING mission since it can allow the robots to exit the nest immediately after dropping their objects. The observations described above hint towards the fact that the controllers of AutoMoDe Coconut use this feature, allowing them to perform better than the controllers of AutoMoDe Chocolate for this particular mission. Figure (6.5) shows examples of those controllers.



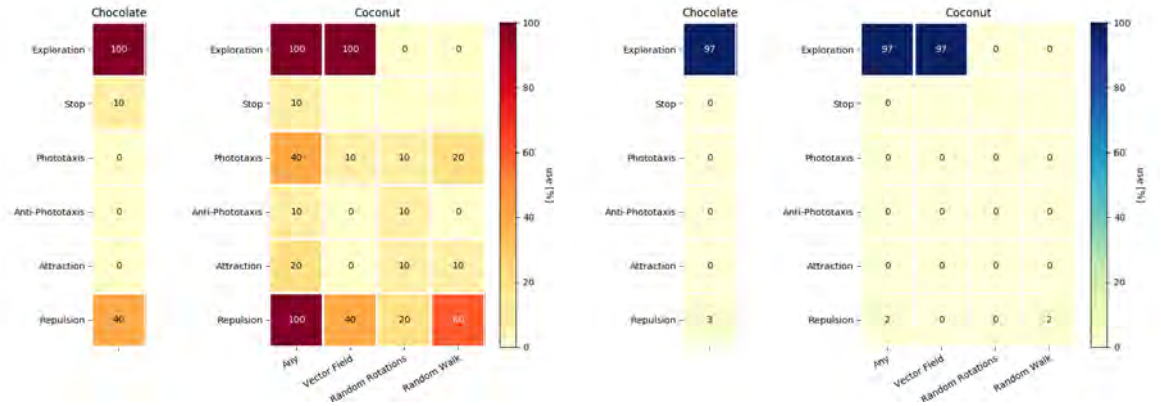
(a) Modules use in the finite state machines (b) Modules use during the mission

Figure 6.4: Use of the modules for FORAGING in a closed arena.

(a) Finite state machine generated by AutoMoDe Chocolate.

(b) Finite state machine generated by AutoMoDe Coconut.

Figure 6.5: Examples of finite state machines generated by the two versions of AutoMoDe for the FORAGING mission.



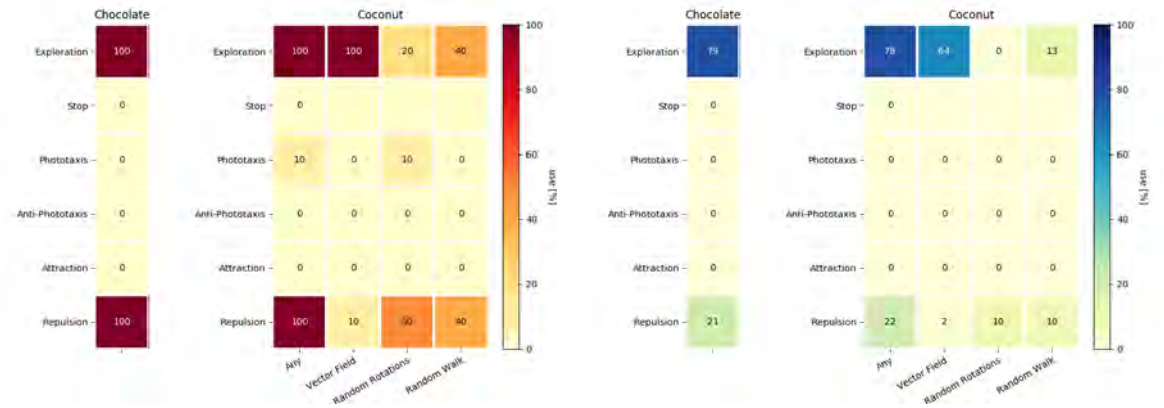
(a) Modules use in the finite state machines (b) Modules use during the mission

Figure 6.6: Use of the modules for FAST POINT FINDING in a closed arena.

The modules use of AutoMoDe Coconut for the FAST POINT FINDING mission is a good example of very simple controllers with complex finite state machines. Indeed, while the actual modules used during the mission almost exclusively (with a use of 97%) consist in Exploration with ballistic motion and vector field obstacle avoidance and Repulsion, the finite state machines are way more diverse. They show a presence of all the modules, in particular Phototaxis included in 40% of them, while none of these modules are used at all during the mission (Except of course for Exploration and Repulsion). Those modules are therefore useless in the behaviors of the controllers but have a strong presence in the corresponding finite state machines.

Besides, the behaviors of the controllers generated by AutoMoDe Chocolate and AutoMoDe Coconut are very similar, with almost the same use of the Exploration and Repulsion behaviors. Even though only half of the controllers of AutoMoDe Chocolate use the Repulsion behavior compared to all of them for AutoMoDe Coconut, this small difference doesn't have much impact on their performances. The same can be said about the fact that AutoMoDe Coconut uses Exploration with ballistic motion and vector field obstacle avoidance. Indeed, the two versions of AutoMoDe are not significantly different in terms of performances for this mission.

As shown in Figure (6.7), the modules used by the controllers for the GRID EXPLORATION mission are very similar between AutoMoDe Chocolate and AutoMoDe Coconut. Since this similarity in the modules use is found both in the finite state machines and during the mission, it is safe to assume that the controllers generated by the two versions of AutoMoDe follow the same behavior. The difference lies within the exploration behaviors used by the controllers



(a) Modules use in the finite state machines (b) Modules use during the mission

Figure 6.7: Use of the modules for GRID EXPLORATION in a closed arena.

of AutoMoDe Coconut. Indeed, they mostly use ballistic motion with vector field obstacle avoidance and some random walk for the Exploration module, and those two explorations behaviors equally for the Repulsion module. This choice of exploration behaviors might cause the worse performances observed with AutoMoDe Coconut on this particular mission.

A tendency that can be observed across the modules use of all the finite state machines is that AutoMoDe Coconut is more likely to generate configurations with useless modules than AutoMoDe Chocolate. This tendency can be explained by the larger search space of the finite state machines of AutoMoDe Coconut. This larger search space is caused by the additional parameters to tune governing the choice of exploration behaviors to be used by the modules. Indeed, the I-Race optimization algorithm used by the two versions of AutoMoDe performs an elitist strategy, which often results in a faster convergence to good parameter values. This has the disadvantage of reducing the exploration of new alternative configurations [34]. Therefore, the larger search space of AutoMoDe Coconut might cause a slower convergence and a broader exploration of the possible configurations.

Besides generating less-specialized configurations with more useless modules, this effect might also explain the better configurations found by AutoMoDe Coconut for the FORAGING mission and the worse configurations for the GRID EXPLORATION MISSION. For the former, the broader exploration might allowed AutoMoDe Coconut to evaluate configurations using a better strategy with the Anti-phototaxis module, while AutoMoDe Chocolate converged faster to configurations without this module. For the latter, the broader exploration did not allow AutoMoDe Coconut to find a better strategy since the mission is simpler. Its slower convergence made the final configuration less optimized than

the one found by AutoMoDe Chocolate, hence the difference in performances.

It should be noted that this interpretation of the results has not been verified in this work. No further experiments were conducted to verify those assumptions.

6.2 Experiments in open environment

The removal of one or three walls of the arena constitutes a hazard for the robots, since any robot leaving the arena is discarded. Therefore, the removal of walls results in an increase of difficulty for the missions since any discarded robot can no longer contribute to the objective function. Figure (6.8) shows that the removal of walls has a very different impact on the score depending on the mission.

For the AGGREGATION and FAST POINT FINDING missions, there is no significant difference between the missions with open and closed arenas. For the case of the AGGREGATION mission, this lack of difference can be explained by the use of a strategy based on Attraction, preventing the robots to go to the edges of the arena. For the FAST POINT FINDING mission, the loss of multiple robots doesn't have much impact on the score since the point is found quickly after the beginning of the experiment anyway.

The impact of the removal of the walls is much clearer for the FORAGING and GRID EXPLORATION missions with some significant differences compared to the missions in closed arena. This is expected, since these missions both require extensive exploration and as many robots as possible until the end of the mission. The loss of robots is therefore both likely to happen and severely punished in terms of performances.

Surprisingly, the performances of AutoMoDe Chocolate and AutoMoDe Coconut are very similar for the GRID EXPLORATION missions even with three missing walls. Indeed, one could expect AutoMoDe Chocolate to completely fail at this mission since its only exploration behaviors are based on ballistic motion which makes it easy to send robots off the arena.

However, as shown in Figure (6.9), the configurations generated by AutoMoDe Chocolate tackle this problem by using an exploration behavior based on the Exploration, Attraction and Repulsion modules. The configurations of AutoMoDe Coconut do not rely at all on the Attraction module since they use different exploration behaviors, in particular the random walk. The configurations of AutoMoDe Chocolate compared to the ones of AutoMoDe Coconut are particularly interesting since they show that even without exploration behaviors fitting a particular problem, the automatic modular design can mix multiple

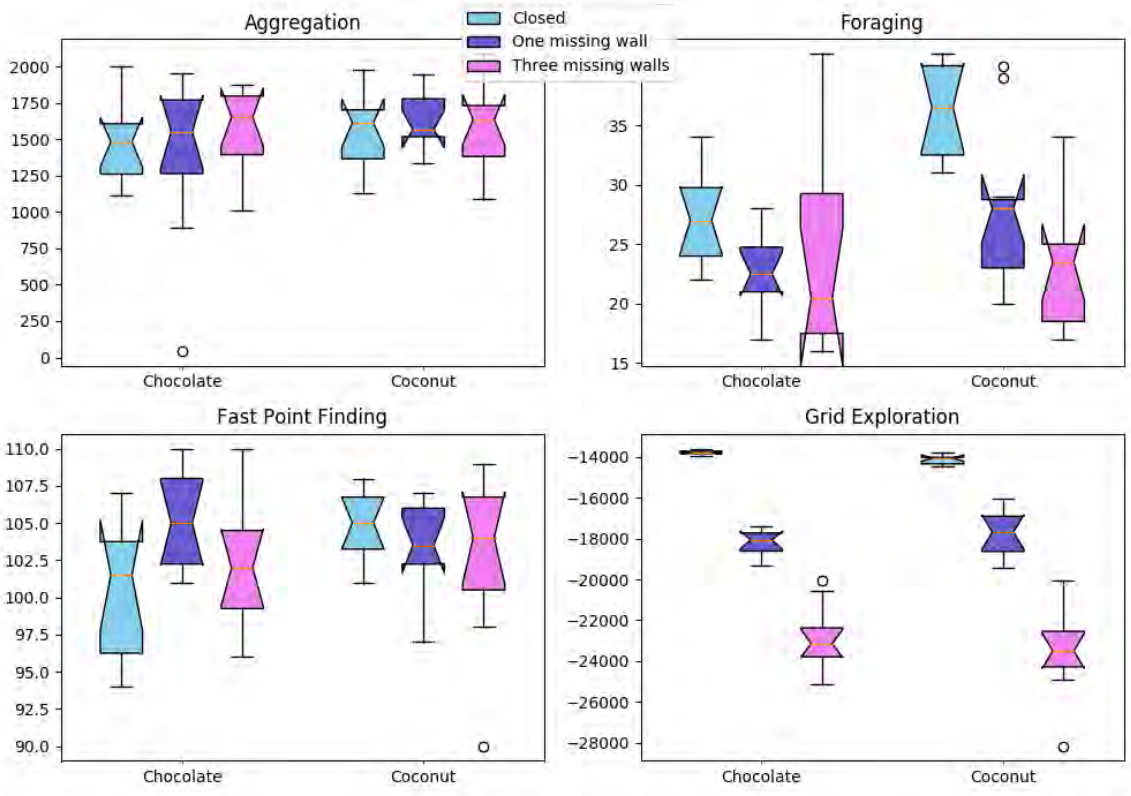
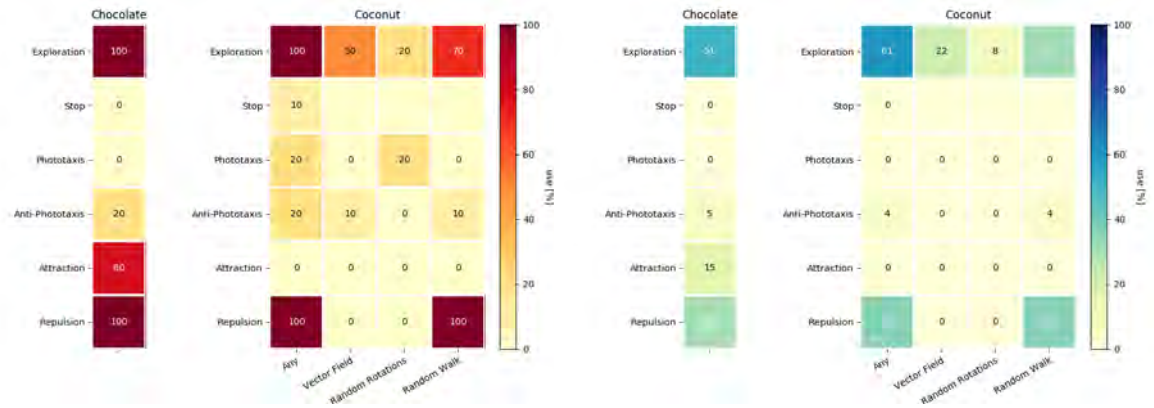


Figure 6.8: Performances across all missions for the experiments in closed and open environments. Each graph represents the score obtained by all controllers of AutoMoDe Chocolate and AutoMoDe Coconut in simulation on a particular mission. Higher is better.



(a) Modules use in the finite state machines (b) Modules use during the mission

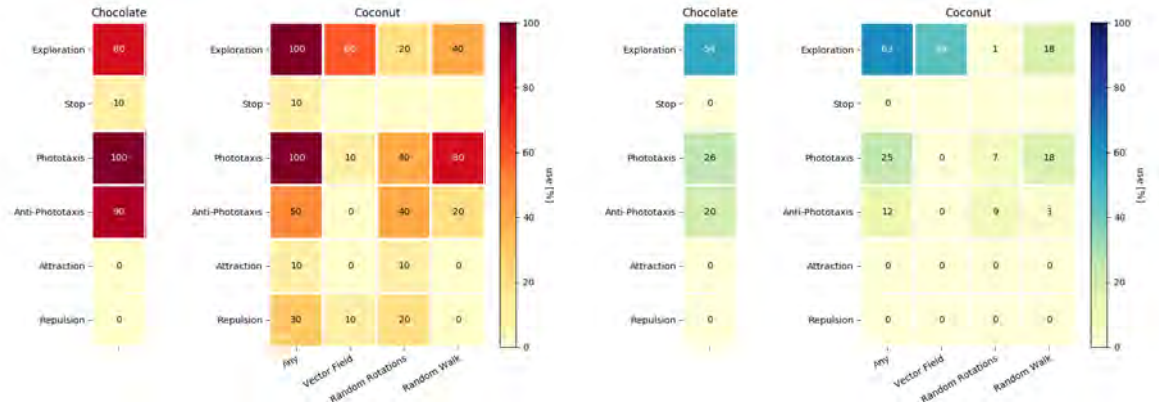
Figure 6.9: Use of the modules for GRID EXPLORATION with three missing walls.

modules to make a new one.

It is interesting to note that the use of the Phototaxis and Anti-phototaxis modules in the FORAGING mission greatly increases for both versions of AutoMoDe in an open environment. This is effect is particularly noticeable with AutoMoDe Chocolate. Indeed, while Figure (6.4b) shows a total use of Phototaxis and Anti-phototaxis of 12% with a behavior mostly based on the Exploration module, Figure (6.10b) shows that this use increases to 46%. This drastic change indicates that a new strategy is performed by the robots. It is reasonable to suppose that in an open environment, the light is used a lot more by the robots. Thanks to the information given by the location of the light source, they avoid going too far from the nest and are less likely to exit the arena.

It is remarkable that both versions of AutoMoDe are able to handle, to some extent, this new class of missions. Furthermore, the modules use of the GRID EXPLORATION and FORAGING missions shows that the behaviors evolve with their environment, focusing more on preserving the robots from exiting the arena. On a side note, the reality gap is also successfully crossed by AutoMoDe Chocolate and AutoMoDe Coconut for the missions in open environment. The performances in simulation and pseudo reality as well as the modules use of the two versions of AutoMoDe for each mission in open environment are shown in appendix A for informational purposes only.

As shown by the Friedman tests in Figure (6.11) and Figure (6.12), the performances of AutoMoDe Chocolate and AutoMoDe Coconut are not significantly different for the experiments in open environment. This means that the new exploration behaviors brought by AutoMoDe Coconut do not allow it to outperform AutoMoDe Chocolate in this new class of missions. However,



(a) Modules use in the finite state machines (b) Modules use during the mission

Figure 6.10: Use of the modules for FORAGING with three missing walls.

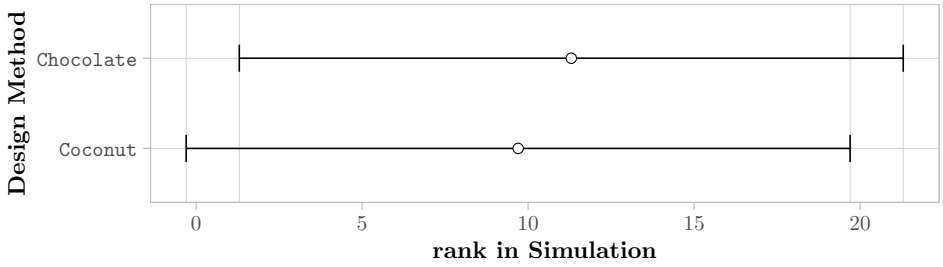


Figure 6.11: Friedman test for the experiments with one missing wall.

some exploration behaviors might be more efficient for some missions in open environment without having a noticeable impact of the score.

To answer this interrogation, the evolution of the use of all the exploration behaviors aggregated across all modules of AutoMoDe Coconut in closed and open environments is presented in Figure (6.13). The trends of the curves is approximately the same for the four missions. Indeed, it shows that as walls are removed and the missions become more open, the use of ballistic motion with vector field obstacle avoidance decreases while the use of the random walk increases. This can be explained by the fact that ballistic motion is dangerous when walls are missing, since the only way for the robot to turn is to collide with a wall or another robot. The random walk is safer since it allows the robot to exploit a smaller area and has less chance to send it to the other side of the arena.

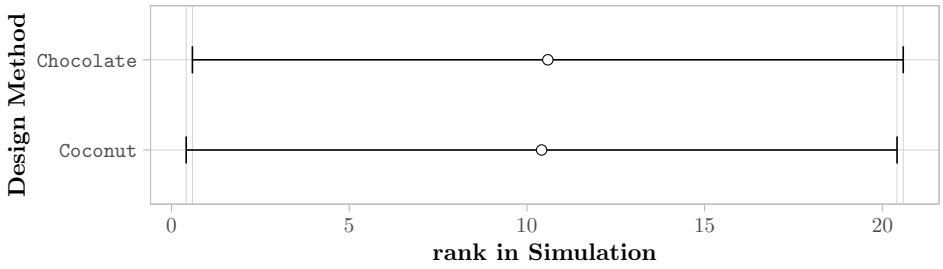


Figure 6.12: Friedman test for the experiments with three missing walls.

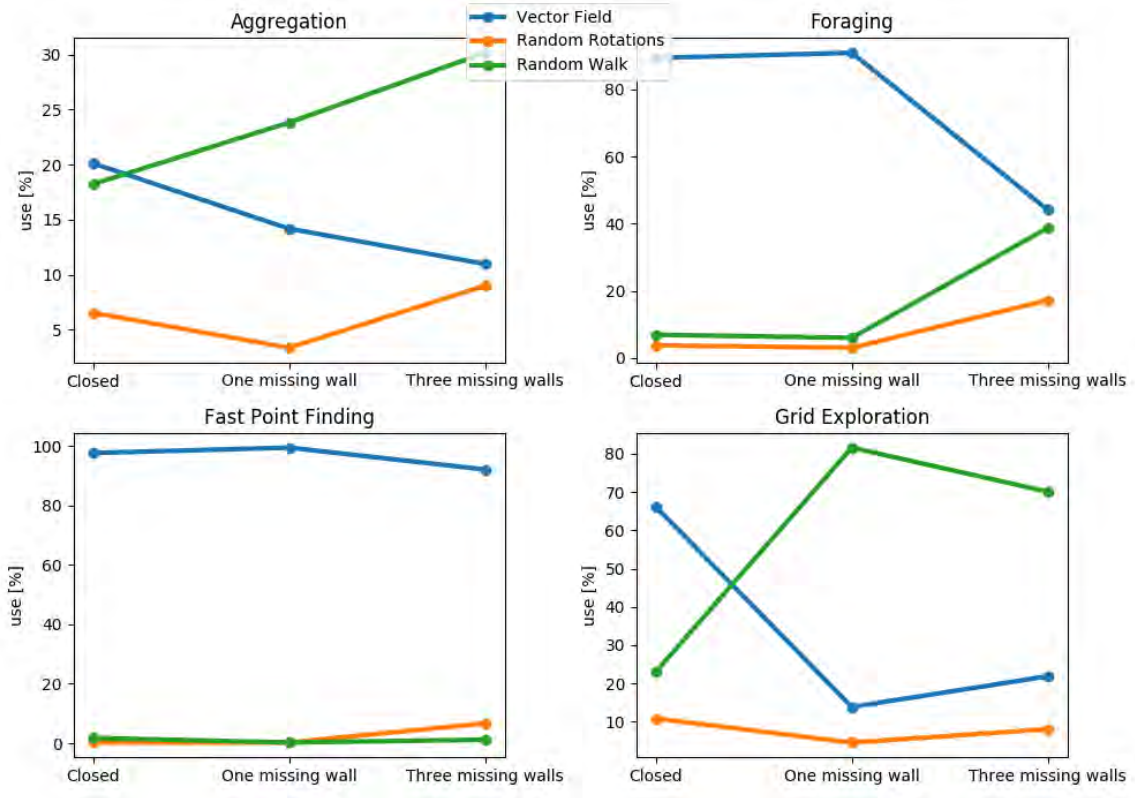


Figure 6.13: Evolution of the use of all the exploration behaviors aggregated across all modules of AutoMoDe Coconut in closed and open environments. Each graph represents the use percentage of each exploration behavior during a particular mission.

Chapter 7

Conclusion

In this work, we introduced a new version of AutoMoDe, AutoMoDe Coconut. AutoMoDe Coconut extends AutoMoDe Chocolate with new exploration behaviors. Indeed, it adds parameters to every module of AutoMoDe (except the Stop module) to govern the exploration behavior used in the module. The available exploration behaviors consist in the two types of ballistic motion already present in AutoMoDe Chocolate and a random walk, Lévy taxis. The point of AutoMoDe Coconut is to measure the impact of those new exploration behaviors in automatic modular design.

We evaluated both AutoMoDe Chocolate and AutoMoDe Coconut on four different missions in closed environment in order to compare their performances and use of exploration behaviors. Those missions consist in AGGREGATION, FORAGING, FAST POINT FINDING and GRID EXPLORATION. They allowed us to assess the exploration and general problem solving capabilities of the controllers generated by the two versions of AutoMoDe. Additionally, those missions were modified to make open environment versions with walls removed from the arena. Extending the experiments to a new class of mission was meant to reveal changes in the behaviors of the controllers and the choices of exploration behaviors. The evaluation consisted in a measurement, for each controller and each mission, of the score obtained and the modules used by all of the robots of the swarm.

The results showed that AutoMoDe Chocolate and AutoMoDe Coconut perform similarly across all missions in closed and open environments. With a broader search space, AutoMoDe Coconut was at times unable to converge to an efficient configuration as well as AutoMoDe Chocolate. However, this same characteristic allowed AutoMoDe Coconut to explore more varied strategies, making it more performant than its counterpart in other situations. The behaviors of the two versions of AutoMoDe changed in an open environment, focusing more on keeping the robots inside the arena. This evolution of the behavior indicates that AutoMoDe might be able to handle this new class

of missions. Besides, random walk was chosen more consistently by AutoMoDe Coconut in open environments, avoiding the dangerous ballistic motion. AutoMoDe Chocolate was however able to overcome its limited exploration behavior by using it along with other modules. This indicates that additional exploration behaviors are not necessary for AutoMoDe since it can generate custom exploration behaviors when necessary.

While this master thesis studies exploration behaviors in AutoMoDe, further research would be needed to tackle this subject in the broader context of automatic design of robot swarms. Besides, while we can presume that AutoMoDe can handle missions in open environments, additional studies would be necessary to assess its performances for this new class of missions. On a side note, it would be interesting to further investigate the impact of the size of the search space on the configurations found by AutoMoDe.

Bibliography

[1] P. Balaprakash, M. Birattari, and T. Stützle. "Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement". In: *Hybrid Metaheuristics, 4th International Workshop, HM 2007*. Vol. 4771. LNCS. Berlin, Germany: Springer, 2007, pp. 108–122.

[2] G. Baldassarre et al. "Self-Organized Coordinated Motion in Groups of Physically Connected Robots". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37.1 (Feb. 2007), pp. 224–239. ISSN: 1083-4419. DOI: 10.1109/TSMCB.2006.881299.

[3] M. Birattari et al. "A Racing Algorithm for Configuring Metaheuristics". In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO*. Ed. by W.B. Langdon and et al. San Francisco CA: Morgan Kaufmann, 2002, pp. 11–18.

[4] Mauro Birattari et al. "Automatic Off-Line Design of Robot Swarms: A Manifesto". In: (2019), p. 11.

[5] Mauro Birattari et al. "F-Race and Iterated F-Race: An Overview". In: *Experimental Methods for the Analysis of Optimization Algorithms*. Ed. by Thomas Bartz-Beielstein et al. Berlin, Germany: Springer, 2010, pp. 311–336. DOI: 10.1007/978-3-642-02538-9_13.

[6] J. Bongard and H. Lipson. "Once More unto the Breach: Co-Evolving a Robot and Its Simulator". In: *Artificial Life IX: Proceedings of the Conference on the Simulation and Synthesis of Living Systems*. Ed. by Jordan Pollack and et al. 2004, pp. 57–62.

[7] J. Borenstein and Y. Koren. "Real-Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments". In: *, IEEE International Conference on Robotics and Automation Proceedings.* , IEEE International Conference on Robotics and Automation Proceedings. May 1990, 572–577 vol.1. DOI: 10.1109/ROBOT.1990.126042.

[8] J. Borenstein and Y. Koren. "The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots". In: *IEEE Transactions on Robotics and Automation* 7.3 (June 1991), pp. 278–288. ISSN: 1042-296X. DOI: 10.1109/70.88137.

[9] Darko Bozhinoski and Mauro Birattari. "Designing Control Software for Robot Swarms: Software Engineering for the Development of Automatic Design Methods". In: *ACM/IEEE 1st International Workshop on Robotics Software Engineering, RoSE*. New York: ACM, 2018, pp. 33–35.

[10] M. Brambilla et al. "Swarm Robotics: A Review from the Swarm Engineering Perspective". In: *Swarm Intelligence* 7.1 (2013), pp. 1–41.

[11] Meng-Li Cao et al. "Experimental Comparison of Random Search Strategies for Multi-Robot Based Odour Finding without Wind Information". In: (Jan. 1, 2015). DOI: 10.5281/zenodo.33822.

[12] G. Di Caro and M. Dorigo. "AntNet: Distributed Stigmergetic Control for Communications Networks". In: *Journal of Artificial Intelligence Research* 9 (Dec. 1, 1998), pp. 317–365. ISSN: 1076-9757. DOI: 10.1613/jair.530. URL: https://jair.org/index.php/jair/article/view/10217 (visited on 08/05/2019).

[13] Anders Lyhne Christensen and Marco Dorigo. "Evolving an Integrated Phototaxis and Hole-Avoidance Behavior for a Swarm-Bot". In: *Proceedings of the 10th International Conference on the Simulation and Synthesis of Living Systems (Alife X)*. Citeseer, 2006, pp. 248–254.

[14] Cristina Dimidov, Giuseppe Oriolo, and Vito Trianni. "Random Walks in Swarm Robotics: An Experiment with Kilobots". In: *Swarm Intelligence*. Ed. by Marco Dorigo et al. Lecture Notes in Computer Science. Springer International Publishing, 2016, pp. 185–196. ISBN: 978-3-319-44427-7.

[15] M. Dorigo et al. "Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms". In: *IEEE Robotics Automation Magazine* 20.4 (Dec. 2013), pp. 60–71. ISSN: 1070-9932. DOI: 10.1109/MRA.2013.2252996.

[16] Marco Dorigo and Mauro Birattari. "Swarm Intelligence". In: *Scholarpedia* 2.9 (Sept. 29, 2007), p. 1462. ISSN: 1941-6016. DOI: 10.4249/scholarpedia.1462. URL: http://www.scholarpedia.org/article/Swarm_intelligence (visited on 08/06/2019).

[17] Marco Dorigo, Mauro Birattari, and Thomas Stützle. "Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique". In: 2006. DOI: 10.1109/CI-M.2006.248054.

[18] R. Emery et al. "Adaptive Lévy Taxis for Odor Source Localization in Realistic Environmental Conditions". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017 IEEE International Conference on Robotics and Automation (ICRA). May 2017, pp. 3552–3559. DOI: 10.1109/ICRA.2017.7989407.

[19] J. J. Falke et al. "The Two-Component Signaling Pathway of Bacterial Chemotaxis: A Molecular View of Signal Transduction by Receptors, Kinases, and Adaptation Enzymes". In: *Annual Review of Cell and Developmental Biology* 13 (1997), pp. 457–512. ISSN: 1081-0706. DOI: 10.1146/annurev.cellbio.13.1.457. pmid: 9442881.

[20] Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics, Vol. 1: Mainly Mechanics, Radiation, and Heat*. 1 edition. Reading/Mass.: Addison Wesley, Feb. 11, 1977. 560 pp. ISBN: 978-0-201-02116-5.

[21] Dario Floreano, P. Husbands, and S. Nolfi. "Evolutionary Robotics". In: *Handbook of Robotics* (2008), pp. 1423–1451.

[22] G. Francesca and M. Birattari. "Automatic Design of Robot Swarms: Achievements and Challenges". In: *Frontiers in Robotics and AI* 3.29 (2016), pp. 1–9.

[23] Gianpiero Francesca et al. "AutoMoDe-Chocolate: Automatic Design of Control Software for Robot Swarms". 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 07/25/2019).

[24] Gianpiero Francesca et al. "AutoMoDe: A Novel Approach to the Automatic Design of Control Software for Robot Swarms". 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 07/25/2019).

[25] L. Garattoni et al. *Software Infrastructure for E-Puck (and TAM)*. TR/IRIDIA/2015-004. Belgium: IRIDIA, Université libre de Bruxelles, 2015.

[26] K. Hasselmann et al. *Reference Models for AutoMoDe*. TR/IRIDIA/2018-002. Belgium: IRIDIA, Université libre de Bruxelles, 2018.

[27] Ken Hasselmann and Mauro Birattari. *Modular Automatic Design of Collective Behaviors for Robots Endowed with Local Communication Capabilities: Supplementary Material*. Published: \par http://iridia.ulb.ac.be/supp/IridiaSupp2019-005/. 2019.

[28] Sabine Hauert, Jean-Christophe Zufferey, and Dario Floreano. "Evolved Swarming without Positioning Information: An Application in Aerial Communication Relay". In: *Autonomous Robots* 26.1 (2009), pp. 21–32.

[29] N. Jakobi. "Evolutionary Robotics and the Radical Envelope-of-Noise Hypothesis". In: *Adaptive Behavior* 6.2 (1997), pp. 325–368.

[30] N. Jakobi, P. Husbands, and I. Harvey. "Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics". In: *Advances in Artificial Life*. Ed. by Federico Morán and et al. Vol. 929. LNCS. London, UK: Springer, 1995, pp. 704–720.

[31] Miquel Kegeleirs, David Garzon, and Mauro Birattari. "Random Walk Exploration for Swarm Mapping". In: (2019), p. 13.

[32] Jonas Kuckling et al. “Behavior Trees as a Control Architecture in the Automatic Modular Design of Robot Swarms”. In: *Swarm Intelligence*. Ed. by Marco Dorigo et al. Vol. 11172. Cham: Springer International Publishing, 2018, pp. 30–43. ISBN: 978-3-030-00532-0 978-3-030-00533-7. DOI: 10.1007/978-3-030-00533-7_3. URL: http://link.springer.com/10.1007/978-3-030-00533-7_3 (visited on 07/25/2019).

[33] Antoine Ligot and Mauro Birattari. “Simulation-Only Experiments to Mimic the Effects of the Reality Gap in the Automatic Design of Robot Swarms”. In: (2019), p. 27.

[34] M. López-Ibáñez et al. “The Irace Package: Iterated Racing for Automatic Algorithm Configuration”. In: *Operations Research Perspectives* 3 (2016), pp. 43–58.

[35] M. J. Matarić. “Learning in Behavior-Based Multi-Robot Systems: Policies, Models, and Other Agents”. In: *Cognitive Systems Research* 2.1 (2001), pp. 81–93.

[36] O. Miglino, HH. Lund, and S. Nolfi. “Evolving Mobile Robots in Simulated and Real Environments”. In: *Artificial life* 2.4 (1995), pp. 417–434.

[37] S. Nolfi and D. Floreano. *Evolutionary Robotics*. Cambridge MA: MIT Press, 2000.

[38] S. G. Nurzaman et al. “Yuragi-Based Adaptive Searching Behavior in Mobile Robot: From Bacterial Chemotaxis to Levy Walk”. In: *2008 IEEE International Conference on Robotics and Biomimetics* (2009). URL: https://www.academia.edu/11042187/Yuragi-based_adaptive_searching_behavior_in_mobile_robot_From_bacterial_chemotaxis_to_Levy_walk (visited on 08/05/2019).

[39] Bao Pang et al. “A Swarm Robotic Exploration Strategy Based on an Improved Random Walk Method”. In: *Journal of Robotics* 2019 (Mar. 13, 2019), pp. 1–9. ISSN: 1687-9600, 1687-9619. DOI: 10.1155/2019/6914212. URL: https://www.hindawi.com/journals/jr/2019/6914212/ (visited on 08/04/2019).

[40] L. E. Parker. “Task-Oriented Multi-Robot Learning in Behavior-Based Systems”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. Piscataway NJ: IEEE Press, 1996, pp. 1478–1487.

[41] Zohar Pasternak, Frederic Bartumeus, and Frank Grasso. “Lévy-Taxis: A Novel Search Strategy for Finding Odor Plumes in Turbulent Flow-Dominated Environments”. In: *Journal of Physics A: Mathematical and Theoretical* 42 (Oct. 13, 2009), p. 434010. DOI: 10.1088/1751-8113/42/43/434010.

[42] C. Pinciroli et al. “ARGoS: A Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems”. In: *Swarm Intelligence* 6.4 (2012), pp. 271–295.

[43] Carlo Pinciroli. *The ARGoS Website*. URL: https://www.argos-sim.info/ (visited on 08/07/2019).

[44] Matt Quinn et al. "Evolving Controllers for a Homogeneous System of Physical Robots: Structured Cooperation with Minimal Sensors". In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 361.1811 (2003), pp. 2321–2343.

[45] Erol Şahin. "Swarm Robotics: From Sources of Inspiration to Domains of Application". In: *Swarm Robotics, SAB*. Ed. by Erol Şahin and William M. Spears. Vol. 3342. LNCS. Berlin Heidelberg, Germany: Springer, 2004, pp. 10–20.

[46] R. Simmons. "The Curvature-Velocity Method for Local Obstacle Avoidance". In: *Proceedings of IEEE International Conference on Robotics and Automation*. Proceedings of IEEE International Conference on Robotics and Automation. Vol. 4. Apr. 1996, 3375–3382 vol.4. DOI: 10.1109/ROBOT.1996.511023.

[47] Ying Tan and Zhong-yang Zheng. "Research Advance in Swarm Robotics". In: *Defence Technology* 239 (Mar. 31, 2013). DOI: 10.1016/j.dt.2013.03.001.

[48] V. Trianni. *Evolutionary Swarm Robotics*. Berlin, Germany: Springer, 2008.

[49] Vito Trianni and Stefano Nolfi. "Self-Organizing Sync in a Robotic Swarm: A Dynamical System View". In: *IEEE Transactions on Evolutionary Computation* 13.4 (2009), pp. 722–741.

[50] J. Urzelai and D. Floreano. "Evolutionary Robotics: Coping with Environmental Change". In: *Proceedings of Conference on the Genetic and Evolutionary Computation Conference, GECCO*. Ed. by L. Darrell Whitney and et al. San Francisco CA: Morgan Kaufmann, 2000, pp. 941–948.

[51] User:PAR. *English: A Plot of a Thousand Steps in a Levy Flight with α=1 and β=0 (A Cauchy Distribution). The Angular Directions Are Uniformly Distributed, and the Step Size Is Cauchy Distributed.* Feb. 25, 2010. URL: https://commons.wikimedia.org/wiki/File:LevyFlight.svg (visited on 08/17/2019).

[52] User:PAR. *English: An Example of 1000 Steps of an Approximation to a Brownian Motion Type of Lévy Flight in Two Dimensions. The Origin of the Motion Is at [0, 0], the Angular Direction Is Uniformly Distributed and the Step Size Is Distributed According to a Lévy (i.e. Stable) Distribution with α=2 and &beta=0; (i.e. a (Normal Distribution).* Feb. 25, 2010. URL: https://commons.wikimedia.org/wiki/File:BrownianMotion.svg (visited on 08/17/2019).

[53] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. "Designing Collective Behavior in a Termite-Inspired Robot Construction Team". In: *Science* 343.6172 (Feb. 14, 2014), pp. 754–758. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1245842. pmid: 24531967. URL: https://science.sciencemag.org/content/343/6172/754 (visited on 08/08/2019).

[54] Bin Yang et al. “Self-Organized Swarm Robot for Target Search and Trapping Inspired by Bacterial Chemotaxis”. In: *Robot. Auton. Syst.* 72.C (Oct. 2015), pp. 83–92. ISSN: 0921-8890. DOI: 10.1016/j.robot.2015.05.001. URL: http://dx.doi.org/10.1016/j.robot.2015.05.001 (visited on 08/05/2019).

[55] V. Zaburdaev, S. Denisov, and J. Klafter. “L\’evy Walks”. In: *Reviews of Modern Physics* 87.2 (June 9, 2015), pp. 483–530. DOI: 10.1103/RevModPhys.87.483. URL: https://link.aps.org/doi/10.1103/RevModPhys.87.483 (visited on 08/05/2019).

[56] J. C. Zagal and J. Ruiz-Del-Solar. “Combining Simulation and Reality in Evolutionary Robotics”. In: *Journal of Intelligent & Robotic Systems* 50.1 (2007), pp. 19–39.

Appendix A

Evaluations in open environment

This appendix contains the performances and modules use measured during the experiments in open environment. Those results are the ones that were not shown in Section 6.2.

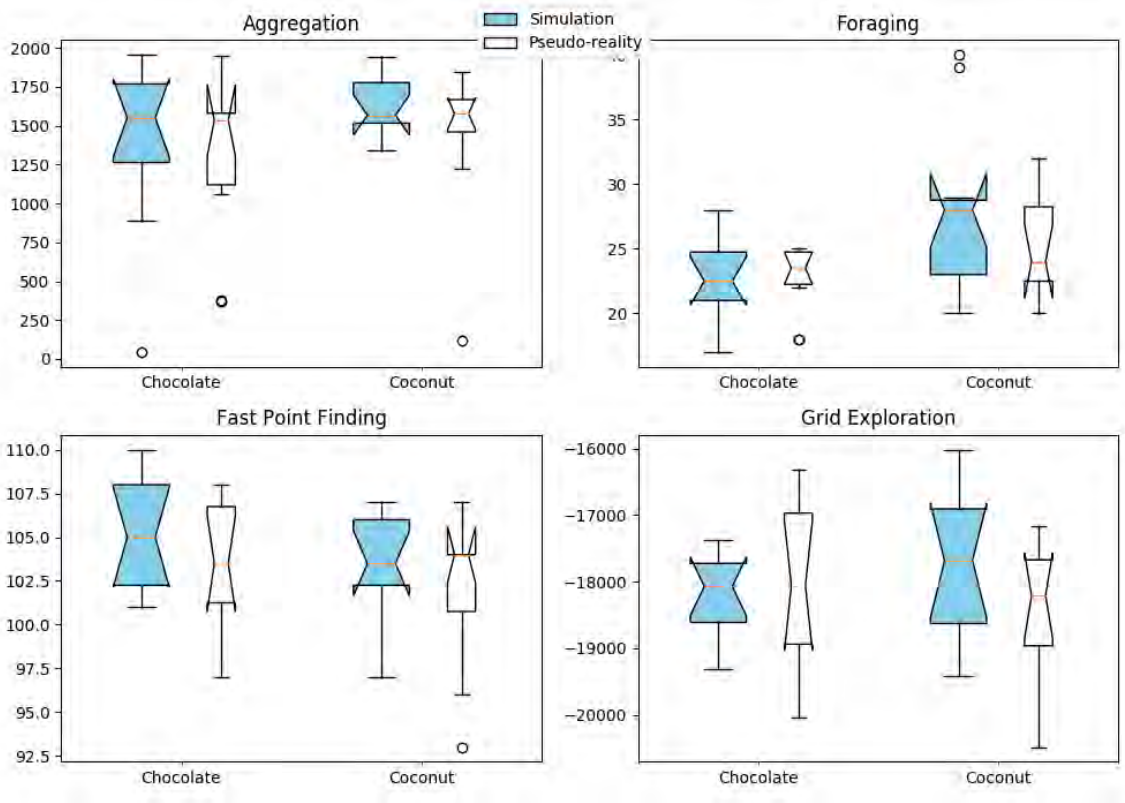


Figure A.1: Performances across all missions for the experiments with one missing wall

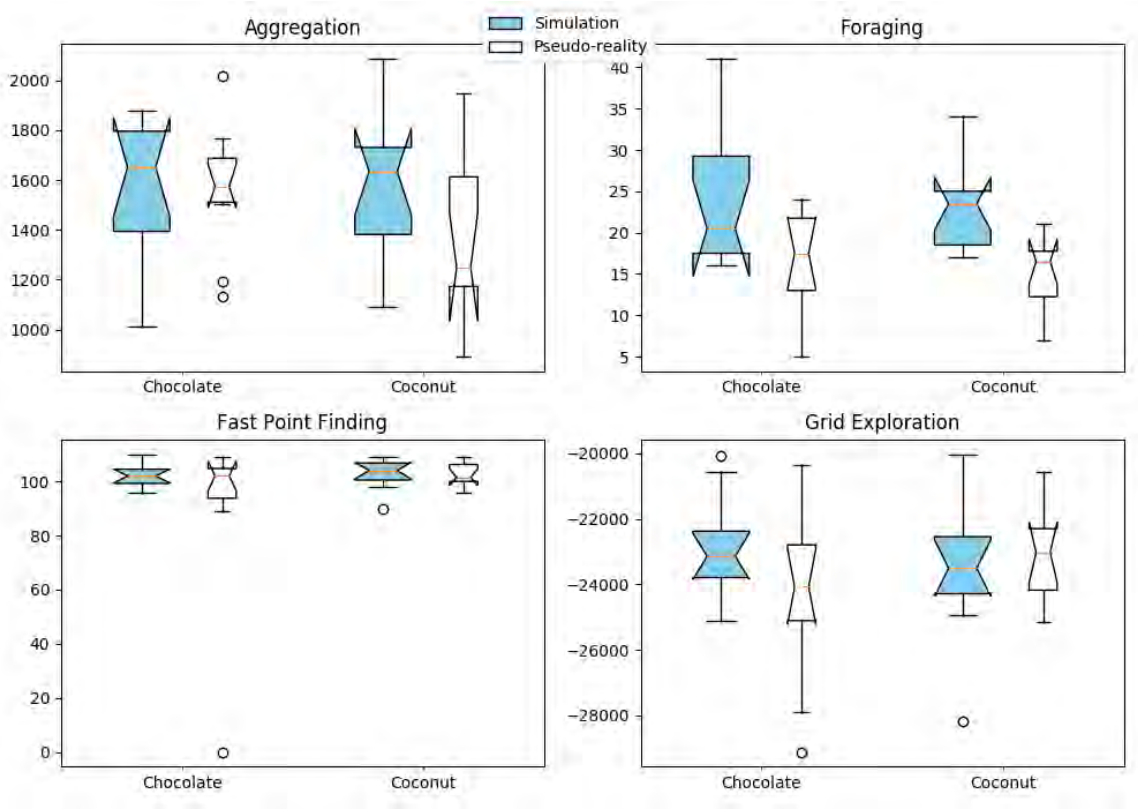
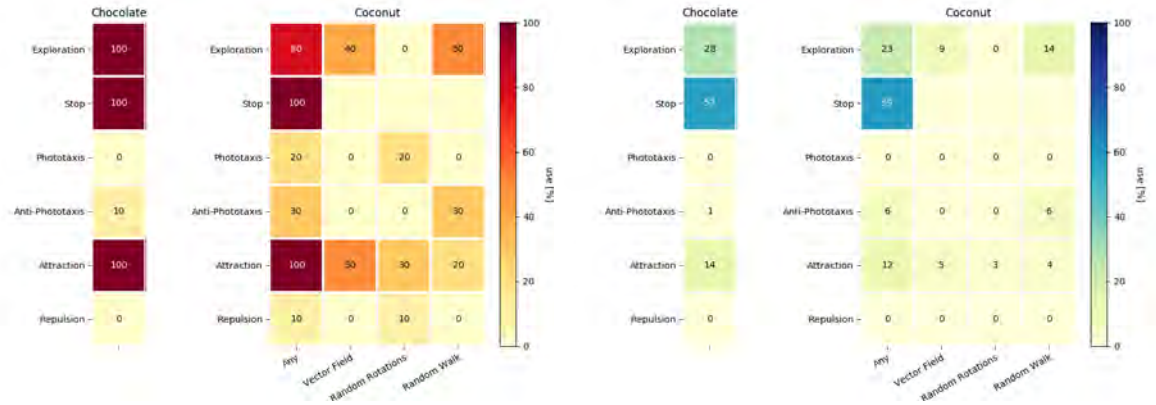
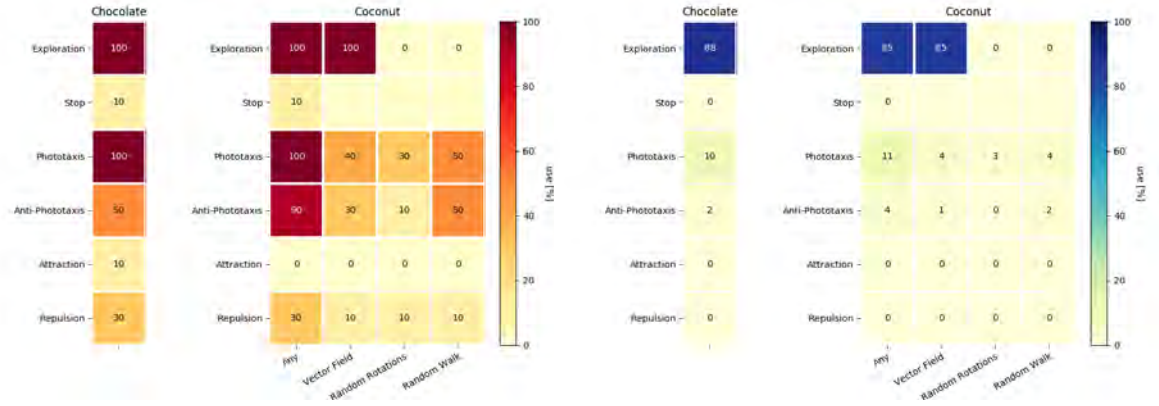


Figure A.2: Performances across all missions for the experiments with three missing walls



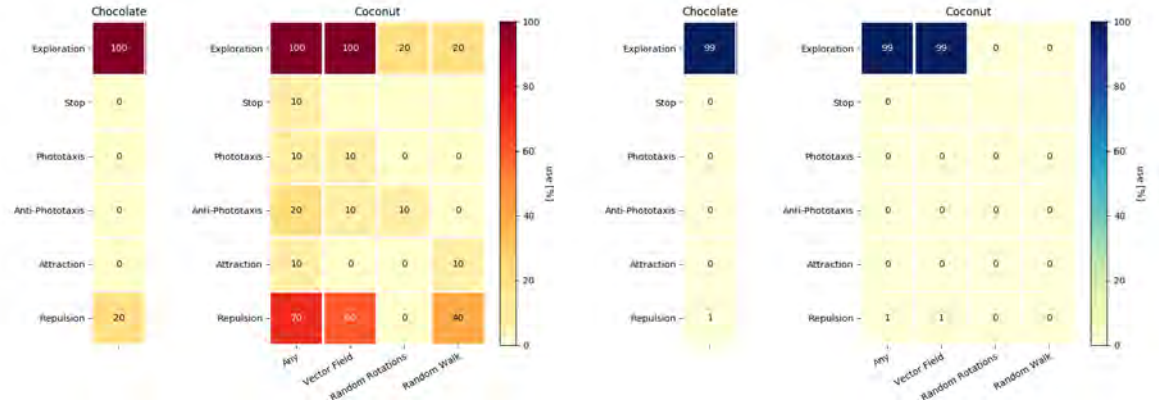
(a) Modules use in the finite state machines (b) Modules use during the mission

Figure A.3: Use of the modules for AGGREGATION with one missing wall.



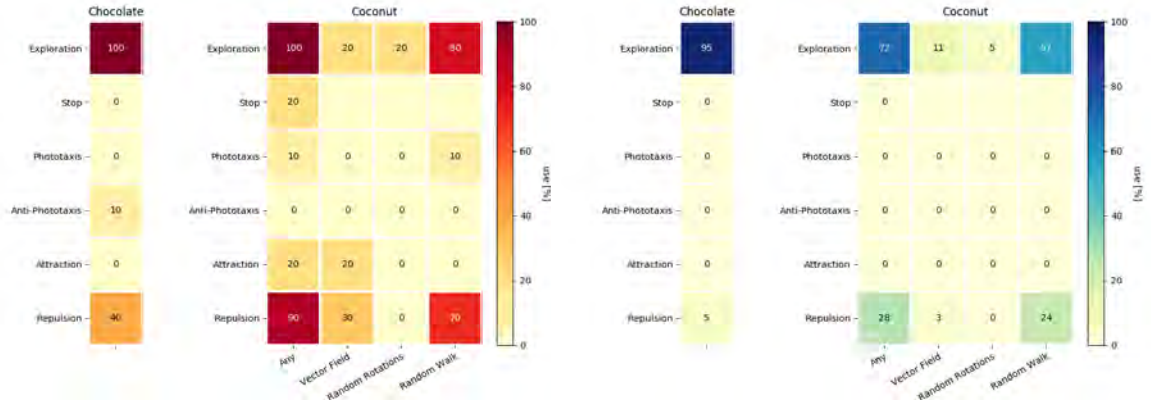
(a) Modules use in the finite state machines (b) Modules use during the mission

Figure A.4: Use of the modules for FORAGING with one missing wall.



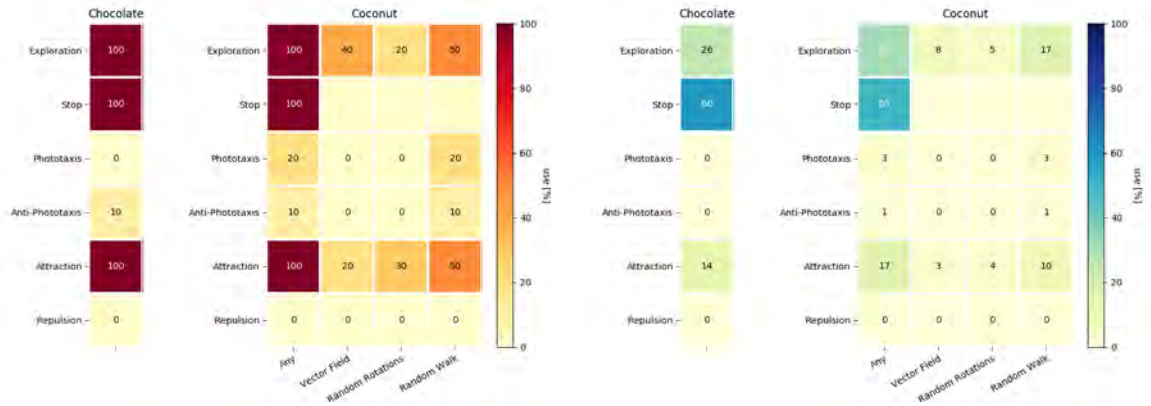
(a) Modules use in the finite state machines (b) Modules use during the mission

Figure A.5: Use of the modules for FAST POINT FINDING with one missing wall.



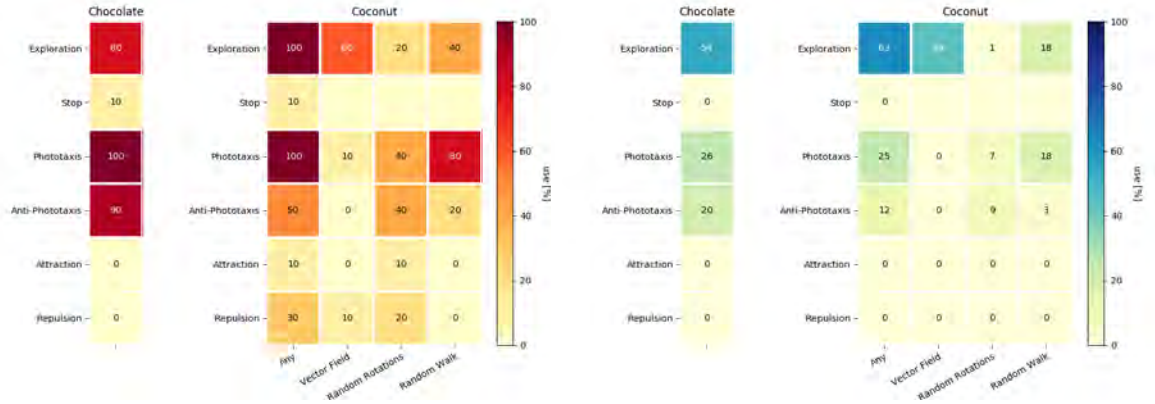
(a) Modules use in the finite state machines (b) Modules use during the mission

Figure A.6: Use of the modules for GRID EXPLORATION with one missing wall.



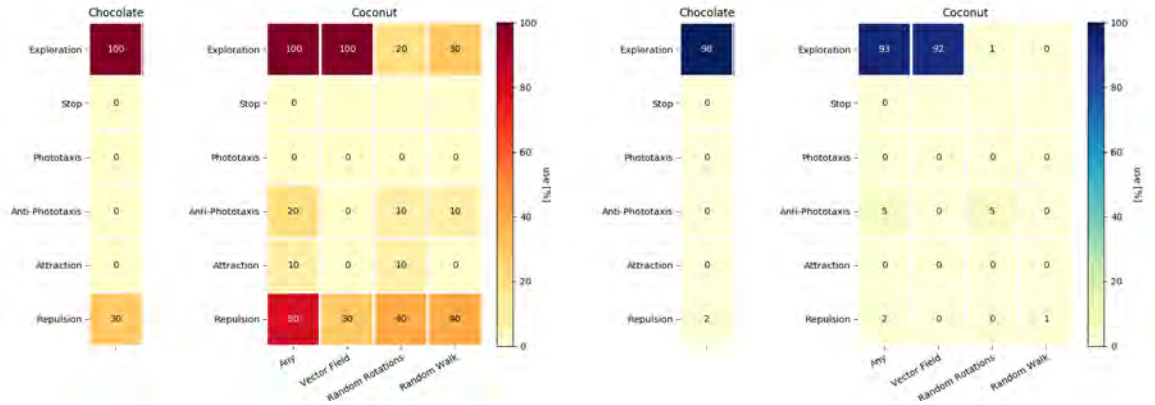
(a) Modules use in the finite state machines (b) Modules use during the mission

Figure A.7: Use of the modules for AGGREGATION with three missing walls.



(a) Modules use in the finite state machines (b) Modules use during the mission

Figure A.8: Use of the modules for FORAGING with three missing walls.



(a) Modules use in the finite state machines (b) Modules use during the mission

Figure A.9: Use of the modules for FAST POINT FINDING with three missing walls.