ECOLE
POLYTECHNIQUE
DE BRUXELLES

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

Automatic Design of Robot Swarms by Demonstration: Addressing Sequences of Missions via Multi-Criteria Design

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

Jeanne Szpirer

Directeur
Professeur Mauro Birattari

Superviseur
David Garzòn Ramos

Service
IRIDIA

Année académique
2022 - 2023

ULB UNIVERSITÉ LIBRE DE BRUXELLES

ULB UNIVERSITÉ LIBRE DE BRUXELLES

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

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

CONSULTATION DU MEMOIRE/TRAVAIL DE FIN D'ETUDES

Je soussigné

NOM :
SZPIRER

PRENOM :
JEANNE

TITRE du travail :
Automatic Design of Robot Swarms by Demonstration: Addressing Sequences of Missions via Multi-Criteria Design

AUTORISE*

~~**REFUSE***~~

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

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

Fait en deux exemplaires, **Bruxelles, le** 21/08/2023

Signature

Formulaire version 04/2015

** Biffer la mention inutile*

** Biffer la mention inutile*

Acknowledgements

First of all, I would like to thank my promoter, Prof. Birattari, who never ceased to show interest and enthusiasm for my work. He really encouraged me to expand the boundaries of my subject to make it more interesting and meaningful.

Secondly, I could never have achieved the result I did without the help of David Garzòn Ramos, who was extremely patient and available throughout the year. I would like to thank him once again for all his invaluable advice, his ideas when I was short of them and his precious time. I could not have imagined a better supervisor.

Thanks also to Ilyes Gharbi, who took the time to help me use all the tools I needed to get started. He and the whole IRIDIA team made me feel welcome in the department where I worked for a year.

During my five years of engineering studies, I had the opportunity to learn a lot more about myself than I thought I would, and to carry out projects that I could not have imagined myself capable of, thanks to the Ecole Polytechnique de Bruxelles and the Cercle Polytechnique.

To my two best friends, Emilie and Elisabeth, who supported me every day and with whom I shared everything, even if one of them was on the other side of the world, thank you, I cannot say it enough.

A warm thank you to all my friends, especially Francois, Eliot and Alexis, with whom I have been privileged to share the Cooloc experience, and Alex with whom I have been lucky enough to share many hours of work (and fun, mostly fun).

Special thanks to Romain who supported, encouraged and listened to me when I needed it, and will continue to do so for a while I am sure.

Finalement, je remercie de tout mon coeur mes parents et mon frère qui m'ont accompagnée, soutenue, conseillée, aimée durant toutes ces années, je n'y serais pas arrivé sans vous.

UNIVERSITÉ LIBRE DE BRUXELLES (ULB)

Résumé

Master en Ingénieur Civil en Informatique

Automatic Design of Robot Swarms by Demonstration: Addressing Sequences of Missions via Multi-Criteria Design

by Jeanne Szpirer

Le design automatique est une approche prometteuse pour la réalisation d'essaims de robots. La littérature traditionnelle dans ce domaine s'est toujours concentrée sur les missions de robotique en essaim qui considèrent un but unique, qui est typiquement défini par l'optimisation d'une fonction objective. Dans ce mémoire, ces deux hypothèses de travail sont remises en question. Une méthode de conception automatique appelée `Demo-Fruit` est introduite pour produire des logiciels de contrôle pour les essaims de robots. La particularité de `Demo-Fruit` est qu'elle peut fonctionner sur des missions de robotique en essaim qui sont spécifiées comme des séquences de sous-missions. En outre, les missions à effectuer sont présentées au système par des démonstrations intuitives du positionnement souhaité des robots dans les deux sous-missions.

L'adressage de séquences de missions est un problème multicritère : chaque sous-mission de la séquence est un critère à satisfaire pendant l'exécution de la mission. En ce sens, l'objectif de `Demo-Fruit` est de produire des essaims de robots dans lesquels les robots exécutent bien les deux sous-missions, avec un compromis neutre. Dans ce mémoire, une analyse comparative de deux méthodologies d'optimisation multiobjectif a été réalisée: la méthode de la somme pondérée, qui simplifie le problème en un contexte mono-objectif, et un processus d'optimisation récemment développé basé sur `irace`, AutoMoDe-`Mandarina`. L'apprentissage par démonstration de `Demo-Fruit` est réalisé à l'aide de l'apprenticeship learning via l'apprentissage par renforcement inverse.

Le problème de conception abordé avec `Demo-Fruit` n'a pas été traité dans le passé, il était nécessaire de créer un protocole d'évaluation pour cette méthode. Pour ce faire, des efforts ont été consacrés à la conception d'un dispositif expérimental et d'un protocole d'évaluation. Les expériences sont menées à l'aide de simulations, elles portent sur douze paires de sous-missions qui posent différents défis à l'essaim de robots. Les résultats montrent que `Demo-Fruit` a la capacité d'aborder des séquences de missions. En outre, il est possible d'étendre l'approche de l'apprentissage par démonstration pour indiquer les tâches séquentielles que les robots doivent effectuer. Les deux approches d'optimisation étudiées produisent des comportements collectifs comparables et peuvent aborder les missions proposées de manière similaire.

Mots-clés : robotique d'essaims; `Demo-Cho`; design multi-critère; optimisation multi-objectif; AutoMoDe.

Abstract

Automatic design is a promising approach for the realization of robot swarms. Traditional literature in this domain has always focused in swarm robotics missions that consider a single goal, which is typically specified by a mathematical formulation in the form of an objective function to be optimized. In this thesis, these two working hypothesis are challenged.
We introduce an automatic design method named `Demo-Fruit` to produce control software for robot swarms. The particularity of `Demo-Fruit` is that it can operate over swarm robotics missions that are specified as sequences of sub-missions. That is, missions in which the swarm must perform a first task for some time, and then transition to a second task. Moreover, the missions to be performed are presented to the system by intuitive demonstrations of the desired positioning of the robots in the two sub-missions.

The problem of addressing sequences of missions is a multi-criteria problem: each sub-mission in the sequence is a criteria to be satisfied during the execution of the mission. In this sense, the goal of `Demo-Fruit` is to produce robot swarms in which the robots perform well the two sub-missions, with a neutral compromise. In this master thesis, a comparative analysis of two multi-objective optimization methodologies is conducted: the weighted sum method, which simplifies the problem into a single-objective context, and a recently developed optimization process based on `irace`, AutoMoDe-`Mandarina`. The learning by demonstration component of `Demo-Fruit` is performed with apprenticeship learning via inverse reinforcement learning.

The design problem addressed with `Demo-Fruit` has not been addressed in the past and, therefore, there was a need to create a evaluation protocol to assess the performance of the method. To do so, in this thesis, efforts were devoted to conceive and implement an experimental set-up and evaluation protocol that would facilitate the study the automatic design of robot swarms in sequences of missions and by using demonstrations. Experiments are conducted with simulations. The experiments consider twelve pairs of sub-missions that pose different challenges to the robot swarm. The results show that `Demo-Fruit` has capabilities to tackle the sequences of missions. Moreover, the research showed that it is possible to extend the learning by demonstration approach to indicate multiple task that the robots must perform in a single deployment—although, limited so far to tasks to be performed in sequence. The results showed also that the two optimization approaches under study produce comparable collective behaviors, and can tackle the proposed missions in a similar way.

Keywords: swarm robotics; `Demo-Cho`; multi-criteria design; multi-objective optimization; AutoMoDe.

Contents

Chapter 1

Introduction

Swarm robotics [1, 2] involves the use of swarm intelligence [3] to autonomously control groups of robots, orchestrating them to carry out specific missions, assignments that the robots have to carry out and can be assessed through a performance measure. The main feature is that robots have exclusive access to local information about their peers and the surrounding environment, resulting in a decentralized, self-organizing system. This particular behavior generates a number of advantages, including scalability, fault tolerance and flexibility.

Despite the advantages arising from decentralized, self-organizing system, a challenge inherent to the field of swarm robotics is surfacing, namely the problem of the micro-macro linkage [4]. This highlights the link between the desired collective behavior exhibited by a group of robots and the formulation of individual behaviors. Currently, a general methodology for designing the behavior of individual agents within a swarm of robots undertaking a specific task remains absent from the literature.

Various methods have been presented throughout the study of this field. Firstly, manual design, where an expert designer uses trial and error to produce control software for robots [5]. This method does not always work in a reasonable amount of time, and requires a great deal of human resources, as well as expertise in this restricted field.

Automatic methods, without human intervention, have been proposed to overcome the above-mentioned problems. In most cases, neuroevolutionary robotics techniques [6] are used. The controllers of the various agents are artificial neural networks (ANNs) which use the values returned by the sensors present on the robots as input, and output the behaviour to adopt. The neural network weights are optimized using a performance measure specific to the mission for which the control software is to be supplied. This performance is measured at the level of the swarm, thus avoiding an explicit switch from group to individual behavior. ANNs can be adapted to any mission constrained by sensors and actuators, but the adaptation to the real situation is compromised by the reality gap. Indeed, ANNs tend to overfit during simulations, increasing variance and creating non-scalable controllers.

A way of reducing this variance and increasing the bias in automatically generated control software has been proposed: automatic modular design. AutoMoDe [7] offers a family

of these automatic modular designs. The basic principle is to control the agents using probabilistic finite state machines (PFSMs) composed of limited collection of states and transitions defined for each flavour of the family. By limiting the possibilities, the capacity for perfect representation in simulation is reduced and thus so is the reality gap.

However, there remains a challenge for these two automatic methods of design creation: the performance measurement, usually presented in the form of an objective function, has to be designed manually. It is not trivial to define this function for missions that may appear simple, but are carried out by a swarm. Even experts encounter challenges as missions evolve to encompass greater degrees of complexity.

`Demo-Cho` [8] is an alternative solution to this problem. This new method adapts apprenticeship learning to the field of swarm robotics, to create control software capable of performing a task demonstrated by an expert. Experts no longer have to formalize missions, but rather show the swarm the desired result. The objective function is an implicit reward function learned by the algorithm thanks to the demonstrations. This solution has shown good results on a series of missions often used in the literature concerning swarm robotics.

In order to go further, to extend the possibilities of using the demonstration for design automation and to meet future expectations, it is interesting to consider sequences of missions. Instead of having one final objective, two objectives can be designed and must be achieved sequentially. The questions to be asked are how to demonstrate a mission sequence, what technique to use to represent the implicit objective function and then optimize it, and how to tell the robots to change mission?

To answer these questions, `Demo-Fruit` was developed, an extension to `Demo-Cho` that optimizes automatic designs for mission sequences based on demonstration sequences. This new method relies on `Demo-Cho` for the overall structure, including demonstrations, apprenticeship learning and feature representation. A new flavour of AutoMoDe was used to build the final PFSMs, `TuttiFrutti` [9] enabling communication using color emission and perception.

The main challenge is that demonstration sequences, by their very nature, introduce a notion of selection preference within a multi-criteria problem, each sub-mission of the sequences being a new criteria. Different ways of optimizing PFSMs are used in order to compare the different existing methods, even though few studies have been carried out on this subject. For instance, should a controller that optimally enhances the initial part be deemed superior to one that enhances both part but not to a perfect degree? The two methods tested are the aggregation of information into a weighted sum, thus working with a single-objective optimization, and the non-aggregation method, which forces to solve a multi-objective optimization problem.

Finally a method evaluation protocol was set-up using different metrics and visual inspection to assess the performance of `Demo-Fruit` since no methodology has been proposed in the literature for this type of experiment. The protocol is composed of a series of mission sequences also specifically designed to evaluate `Demo-Fruit`.

The master thesis develops in the following sections, first, a state of the art of the existing methods and the concepts needed to understand the project. Then, the methods used to construct `Demo-Fruit` from the demonstrations to the optimization of PFSM. The experimental set-up and procedural protocol are formulated before presenting the results obtained using the two distinct optimization methodologies. Finally, these results are discussed and conclusions are presented.

Chapter 2

Related Work

Evolutionary algorithms make it possible to represent any mission and adapt the behavior of a swarm of robots automatically in simulation, but suffer from a significant reality gap. On the other hand, automatic modular design introduces the bias needed to reduce this reality gap. However, both methods still rely on the knowledge of experts capable of formulating objective functions to characterize the swarm's performance. Demonstrations can be a solution for eliminating the need to explicitly formulate reward functions.
In addition, the missions chosen may have one or more objectives, raising the question of single or multi-optimization of the control design of a swarm of robots.

The aim of this master thesis is to investigate the possibility of extending demonstration-based learning to sequences of missions to be carried out by swarms of robots.

This chapter is structured as follows: the section 2.1 begins by defining swarm robotics and its main features. Then, the design of swarms of robots is tackled in the section 2.2 and covers the different existing techniques, i.e. manual design and automatic design via evolutionary algorithms, automatic modular design and learning by demonstration. The single and multi-optimization techniques used for robot swarm design are presented respectively in sections 2.3 and 2.4. Finally, the use of mission sequences for swarm robotics in the literature is presented in section 2.5.

2.1 Swarm Robotics

Swarm robotics uses swarm intelligence, the collective behaviour of decentralised, self-organised systems [2], to design and control groups of robots performing the same task. For these two concepts, systems are generally simple agents that interact locally with each other. In the specific context of robots, agents are simple robots whose behaviour results from local interactions between them and their environment.
Robot swarms are defined by different criteria:

- Robots and their possible behaviors are simple.
- Robots are homogeneous (swarms with heterogeneous robots also exist [10]).
- Robots interact only locally with themselves and their environment, they do not have access to global information.

- Global behaviour emerges from these interactions.

The above characteristics can be used to generate group behaviour with the following properties:

- Fault tolerant: decentralisation means that all the robots are interchangeable; they are not individually responsible for overall behaviour, there is no leader. If one is defective, it will be replaced by another agent.
- Scalable: robots only use local information. This assumes that the available neighborhood information remains virtually unchanged when the number of robots in the swarm changes, without going to extremes.
- Flexibility: robots themselves decide to dynamically allocate themselves to different tasks to achieve the objective, this is self-organization. If the environment changes, each robot will adapt its behaviour to best match the new requirements and conditions in which it finds itself.

2.2 Design of Robot Swarms

So far, there is no specific, defined way of designing a swarm robot. The first technique developed is the manual design of the behaviour of a swarm by describing the behaviour of a robot, observe the result and adapt the control software until the desired collective behaviour is reached. This method can be beneficial in cases where knowledge is necessary and known, but most of the time it is a long and inefficient process. This is called behavior-based design [5].

In order to be able to carry out increasingly complex tasks without increasing the knowledge required, it was necessary to develop automatic swarm design methods. All evolutionary algorithms have been adapted to suit the design of a group of robots. However this technique shows a large reality gap. In this context, automatic modular design has been developed to reduce this gap. However, expert knowledge is still required to formulate the reward function associated with each assignment. Design by demonstration is a new alternative that avoids the explicit formulation of objective functions. These three methods are presented in more detail below.

2.2.1 Evolutionary Robotics

Evolutionary robotics (ER) is an approach using Darwin's principles of natural selection to generate controllers for robots [11], so entities adapt to their environment autonomously. Controller optimization begins with the generation of random designs represented as genotypes containing all the parameters to optimize with regards to some evaluation metrics. The worst ones are eliminated and replaced by combinations or mutations of other individuals in the population exactly as in evolutionary computation [12].
However, the design of control software for mobile robots interacting with their environment is complicated and time-consuming. One way of improving the algorithm is to introduce artificial neural networks (ANN) for each of the robots [13]. This particular way of optimizing an autonomous system is called neuroevolutionary [6, 14].

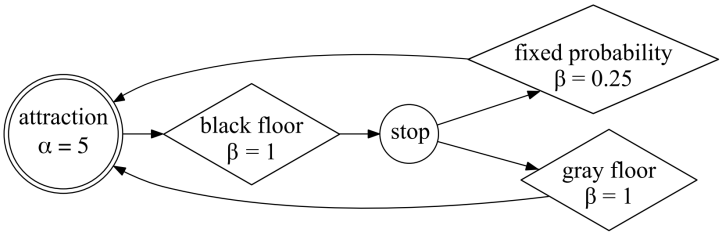


Figure 2.1: Example of a probabilistic finite state machine [23]. Circles represent atomic behaviors and diamonds are conditional state transitions.

The techniques of evolutionary robotics were then adapted to swarms to create complex behaviours for sets of autonomous robots [15, 16]. For swarms of robots, the algorithm works in the same way, ANN makes the link between sensored measures and behaviour. A genotype represents the parameters of an ANN used to control a swarm agent, and is optimised through iterations of the genetic algorithm. To test an ANN and evaluate its performance, it is applied to the agents in a swarm simulating a mission several times. The final performance is the average of the performances of the different runs.
Nonetheless ERs try to minimise the impact and therefore the bias introduced by designers on controllers [17, 18]. However, reducing the bias increases the variance, leading to non-scalable controllers and, for example, large drop in performance due to the reality gap [19].
An application of ER to swarm robotics called `EvoStick` was presented by Francesca *et al.* [7].

2.2.2 Automatic Modular Design

In order to increase the robustness of the controller design method, work has been carried out on the bias-variance tradeoff described for machine learning [20]. It has been shown that introducing a bias reduces the variance and therefore the effects of the reality gap. Indeed, the variability of the biases reduces overfitting, which prevents the system from adapting to different environments and possible setups [21]. In this context, Francesca *et al.* [7] have developed a more robust controller design method called AutoMoDe (automatic modular design).

The basic principle is to offer a robot controller in the form of a probabilistic finite state machine (PFSM) [22]. The controller is therefore made up of a set of predefined behaviour modules called *atomic behaviors* and linked together by *conditional state transitions*, an example can be seen at the figure 2.1. In order to find the optimal PFSM, the search space, composed of all possible PFSM, needs to be explored by one the many optimization algorithms, the ones used more often are detailed in the 2.3 section.

Different versions of AutoMoDe were developed through the years, these are briefly presented in the next section.

AutoMoDe

Different specializations corresponding to different versions of the epuck-robot [24] [25]. These specializations, or flavours, are listed below.

AutoMoDe-`Vanilla` [7] AutoMoDe's first flavour is a proof-of-concept that automatically generates PFSMs to control a swarm of robots. The probabilistic finite state machine is constructed from the tunable modules listed in Table 2.1. As the optimization algorithm, F-Race 2.3.2 is used. Two distinct tasks were employed to compare the design of control software for a collective of robots: specifically, FORAGING and AGGREGATION. The missions were carried out in simulation and in real-life situations to measure the effect of the reality gap. In order to obtain the expected performance of AutoMoDe-`Vanilla` and `EvoStick`, 20 different experiments were launched for each of the methods, in simulation and in reality. Only one run was used to assess these stochastic optimization methods [26]. AutoMoDe-`Vanilla` and `EvoStick` were also compared with human experts, Francesca *et al.* [27] proposed two different manual methods : `U-Human` and `C-Human`. The former allows experts to implement control software without constraint, while the latter must be a combination of the modules available with AutoMoDe-`Vanilla`. `C-Human` performs significantly better than AutoMoDe-`Vanilla`, which in turn performs significantly better than `U-Human` and `EvoStick`, these results are obtained by applying a Friedmann test [28] with the task as blocking factor. These results highlight the main advantage of AutoMoDe-`Vanilla`, which gives better software control than `EvoStick`, but also the limitation introduced by the use of F-Race, which was improved in subsequent research to obtain better results.

AutoMoDe-`Chocolate` [23] The results of AutoMoDe-`Vanilla` [7] showed a weakness in the algorithm, which failed to outperform human experts. In order to confirm two hypotheses: improving the optimization algorithm yields better results and, under certain conditions, a design control software will be able to outperform `C-Human` [7]; AutoMoDe-`Chocolate` has been developed. The only difference with `Vanilla` is the optimization algorithm, since Iterated F-Race is used. For tests and results, the same missions as those proposed by Francesca *et al.* [7] are used to compare the different methods. Finally, under certain conditions, `Chocolate` outperforms `Vanilla` and `C-Human`. It is the first automatic design to achieve this objective.

AutoMoDe-`Gianduja` [29] `Gianduja` is an extension to AutoMoDe-`Chocolate` that allows robots to communicate by sending and receiving one single message. This message allows a reaction when it is received, but also the approach behaviour of peers who spread the message. Two behaviours (, attraction-to-message and repulsionfrom-message) are added to those listed in the Table 2.1, along with two transitions, message-count and inverted-message-count. AutoMoDe-`Gianduja`'s performance is assessed through three missions: AGGREGATION, STOP and DECISION [29], selected on the basis of the impact that communication should have on the robots' ability to perform the desired behaviour. It is compared with the performance of AutoMoDe-`Chocolate` and `EvoCom`, an extension of `EvoStick` that allows the use of a message, on the same tasks. The results show that AutoMoDe-`Gianduja` performs significantly better than AutoMoDe-`Chocolate` and

Atomic Behavior	**Parameters**	**Description**
EXPLORATION	$\tau \in [0, 100]$	movement by random walk
STOP	*None*	standstill state
PHOTOTAXIS	k fixed to 5	movement towards the nearest light if perceived; otherwise, straight movement
ANTI-PHOTOTAXIS	k fixed to 5	movement away from the nearest light if perceived; otherwise, straight movement
ATTRACTION	$\alpha \in [1, 5]$ and k fixed to 5	movement towards the center of mass of the robots
REPULSION	$\alpha \in [1, 5]$ and k fixed to 5	movement away from the center of mass of the robots
State Transition	**Parameters**	**Description**
BLACK-FLOOR	$\beta \in [0, 1]$	black floor beneath the robot
GRAY-FLOOR	$\beta \in [0, 1]$	gray floor beneath the robot
WHITE-FLOOR	$\beta \in [0, 1]$	white floor beneath the robot
NEIGHBOR-COUNT	$\eta \in [0, 20]$ and $\xi \in \{0, 10\}$	value of $z(n) = \frac{1}{1+e^{(\eta(\xi-n))}}$, where n is the number of neighboring robots
INVERTED-NEIGHBOR-COUNT	$\eta \in [0, 20]$ and $\xi \in \{0, 10\}$	value of $1-z(n)$, where n is the number of neighboring robots
FIXED-PROBABILITY	$\beta \in [0, 1]$	value of β

Table 2.1: Atomic Behaviors and Conditional State Transitions of the AutoMoDe-Vanilla flavour

EvoCom using meaningfully communication. However, AutoMoDe-Gianduja is more sensitive to the noise perceived by the ground sensor 3.2.1.

AutoMoDe-Maple [30] Behaviour trees, defined by Marzinotto *et al.* [31], are constructed using the modules available with AutoMoDe-Chocolate, and the results obtained on two missions: FORAGING and AGGREGATION are compared with AutoMoDe-Chocolate and EvoStick. The aim is to explore the possibility of using behaviour trees to create automatic modular design. In the end, the results are similar to those obtained using Chocolate. Further experiments are needed to study Maple's potential more fully.

AutoMoDe-Waffle [32] When AutoMoDe-Waffle was introduced, the aim was to show that the hardware could also be built using predefined modules. This new flavour is an extension of AutoMoDe-Chocolate, with the same modules, transitions and optimization algorithm. The choice of hardware is usually fixed and does not allow any flexibility at this level. Three missions are used to determine Waffle's performance: ANYTIME SELECTION, END TIME AGGREGATION and FORAGING [32]. During testing, constraints may or may not be imposed on the software and hardware control search algorithm; these may be monetary or power constraints. The results show that software and hardware control are highly dependent on the mission chosen and the budget imposed.

AutoMoDe-IcePop [33] AutoMoDe-IcePop differs from AutoMoDe-Chocolate only in the optimization algorithm used. In this new flavour, component-based analysis [34] is used to optimize the finite state machine made up of the same modules as AutoMoDe-Chocolate. The comparison with AutoMoDe-Chocolate was based on two missions: AGGREGATION WITH AMBIENT CUE (AAC) and FORAGING. The experimental results suggest that IcePop is a viable solution for improving the optimization algorithm, but other variants will have to be studied to differentiate the performance of the different algorithms.

AutoMoDe-Coconut [35] AutoMoDe-Coconut is the first flavour for which the influence of the exploration scheme (such as random walks [36] [37]) has been studied. In this proposal, the exploration scheme is configurable and AutoMoDe-Coconut should select the most appropriate exploration to carry out three missions: FORAGING, AGGREGATION and GRID EXPLORATION. Another specification is that the missions are also tested on an unbounded version of the arena. The comparison with AutoMoDe-Chocolate does not allow to clearly identify a difference in performance, the gaps in bounded arena are created by the difference of size of the search space. When results on unbounded arena are analyzed performance and use of the modules still are the same. AutoMoDe-Chocolate works well because of the combination of the modules and not on the schemes of exploration.

AutoMoDe-Cedrata [38] Like AutoMoDe-Maple, AutoMoDe-Cedrata creates a behaviour tree. The update consists of including new modules to allow the swarm to handle a message transfer and creating new modules that are made for constructing behaviour trees. The comparison is done on three missions : FORAGING, AGGREGATION MARKER and STOP; with three different ways to produce a control software, AutoMoDe-Maple that creates behaviour trees with other modules, experts that are specialized in swarm robotics but not in behaviour trees and a reference behaviour tree done by experts in this

field. The flavour is able to produce well-performing instances of control softwares, in two of the three missions AutoMoDe-`Cedrata` outperforms AutoMoDe-`Maple`. For the last one, AutoMoDe-`Maple` outperforms everything because of its access to the ambient cue. The optimization algorithm does not allow to have better results than the ones found by humans. The behaviour trees can contain a lot of superfluous modules and Iterated F-Race cannot distinguish that. Futhermore the communication works only if the modules are tuned at the same signal which can happen by chance but is not something Iterated F-Race is always trying to reach.

AutoMoDe-`Arlequin` [39] AutoMoDe-`Arlequin` is very similar to AutoMoDe-`Chocolate` except that the six hand-coded modules are neural networks. To create these new behavioral modules objective functions describing the original six modules were given to `EvoStick`. In order to assess the performance and bias/variance tradeoff, the results on two missions : AGGREGATION XOR and FORAGING; were compared with AutoMoDe-`Chocolate` and `EvoStick`. In simulation, AutoMoDe-`Arlequin` produces similar results to AutoMoDe-`Chocolate` and `EvoStick`, but the reality gap is very wide, so AutoMoDe-`Chocolate` remains the best design.

AutoMoDe-`Mate` [40] AutoMoDe-`Mate` was introduced to complete missions that mainly concern the spatial organization of a swarm of robots. The modules are the same as AutoMoDe-`Chocolate`, with the addition of FORMATION, which enables the robots to form patterns with each other. The missions chosen to assess AutoMoDe-`Mate`'s performance place particular emphasis on the positioning of the robots in space and their ability to cover certain areas or not. In addition to AutoMoDe-`Chocolate`, `EvoSpace` was used to compare results. `EvoSpace` is a neuro-evolutionary method for generating fully-connected feed-forward artificial neural network by including the physical modules available in AutoMoDe-`Mate`. For the three missions chosen AutoMoDe-`Mate` outperformed the other methods in the comparison and gave similar results when tested with physical robots.

AutoMoDe-`TuttiFrutti` [9] AutoMoDe-`TuttiFrutti`'s special feature is the use of robots capable of emitting and perceiving colours using RGB LEDs. The modules used are extensions of those available with AutoMoDe-`Vanilla` for using colours. The missions chosen to evaluate the new method use an arena that also emits colour. The performance comparison is made with `EvoColor`, an extension of `EvoStick` that allows the use of LEDs and colour sensors. AutoMoDe-`TuttiFrutti` uses colour-based information for its chosen missions, whereas `EvoColor` does not necessarily produce collective behaviour using the emission and perception of colour. Overall, AutoMoDe-`TuttiFrutti` outperforms `EvoColor` and is less affected by the reality gap.

2.2.3 Design by Demonstration

In all the design of swarm robotics presented before, an objective function needed to be defined in order to assess the performance of the swarm and adapt the control software. Only it is very complicated to define a task-specific objective function for a swarm, even by an expert [4]. One way of overcoming this definition problem is to use the Apprenticeship Learning algorithm and AutoMoDe-`Chocolate`, which were applied by Gharbi to

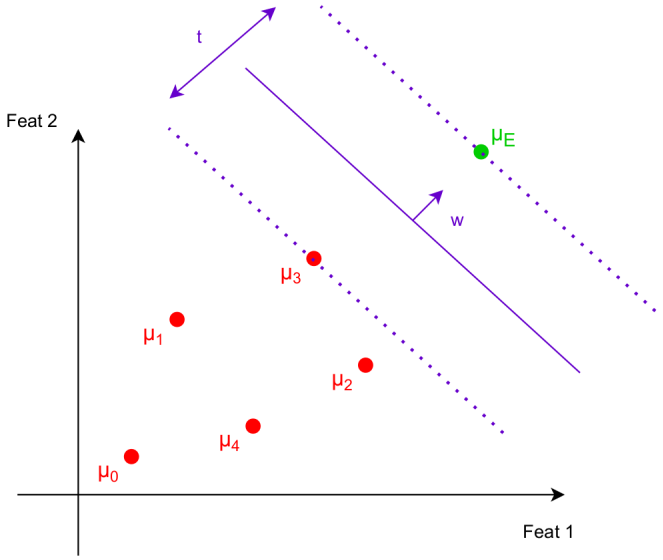


Figure 2.2: Graphical representation of the four first iterations of the apprenticeship learning algorithm in a two-dimensional feature space. The green point, expected value, is linearly separated thanks to a SVM from the red points, produced policies [8].

develop `Demo-Cho` [8].
Apprenticeship learning [41] means implicitly learning an objective function, to avoid having to build one first. This has several advantages, notably the avoidance of the objective function, but also the ability of the algorithm to adapt to a wide range of tasks. In order to produce the implicit objective function, features must be given to describe the behaviour of the expert to reproduce. In the case of `Demo-Cho`, the assumption was that features could be calculated from the robots' final positions after an experiment. The second hypothesis made is that the final objective function is a linear combination of the features. In order to construct this implicit reward function with the expected values and the features of the experiments over time, the linear kernel SVM concept is applied [42, 43]. The figure 2.2 represents how the objective function production algorithm works after a few iterations.

Once an objective function is proposed to describe the mission to be carried out it is given to `irace` (section 2.3.4) to produce a PFSM with the modules of AutoMoDe-`Chocolate`. The whole process is iterated a fixed number of times in order to increase the quality of the proposed PFSM.

2.3 Single-Objective Optimization in the Automatic Design

Each of the previously introduced methods for automatic design creation integrates single-objective optimization notably through the implemented version of Iterated F-Race, `irace`. Below, a review is provided concerning the established techniques utilized for

the optimization of designs within the domain of swarm robotics.

Algorithms for solving standard optimization problems are generally used to determine the optimum value for a large number of parameters specific to the problem in question. Improving the performance of these algorithms is possible, but may result in the determination of hundreds of thousands of parameters [44, 45, 46, 47].
For a long time, the parameters of these algorithms were tuned manually during a tuning phase before the algorithm was used. However, this method has a number of disadvantages in terms of time and the bias introduced by the designer [48]. Automation has rapidly come to play an increasingly important role in determining the parameters of these algorithms. Several methods can be cited, including experimental design techniques [49, 50], racing approaches [51] and statistical modelling approaches [52, 53].

Automatic algorithm configuration can be seen as the determination of optimal parameters using training instances to generate an algorithm that performs optimally on unknown instances, as a machine learning problem [54]. Two approaches can be used: offline or online [48]. For the offline approach, two stages are clearly defined: the tuning phase and the testing phase. In the first stage, configuration takes place on training instances before testing on similar but unknown instances. On the other hand, when the online approach is used, the parameters are adapted during the resolution of an instance [55, 56]. Some parameters still need to be determined offline, which is why the two approaches can be complementary [46].

In the context of modular automatic design, the automatic optimization algorithms used are based on racing approaches. These will be detailed in the following sections in the case of single-objective operations. The aim of the problem of optimizing a single objective is to find the solution which minimises or maximises a criterion, an objective function for example [57]. The multi-objective optimization (MOO) consists of the simultaneous and continuous optimization of several objective functions [58] and this will be discussed in the next section.

2.3.1 Racing Approach

When the parameters are optimized offline, a large number of parameter combinations and training instances are available for choosing the best configuration during the tuning phase. Racing allows parallel comparisons of the different models and quickly determines which are better than the others, so that the worst performers can be eliminated [60]. This kind of approach was first introduce in the machine learning field by Maron *et al.* [61] to solve the model selection and multimodel inference problems [62]. The racing approach assesses a finite number of configurations at each step of the algorithm which will be evaluated on the instance corresponding to the step. At each stage, each instance, the remaining configurations are evaluated using a number of statistical tests and the poorer candidates are eliminated. These eliminations allow to concentrate the efforts on the most promising candidates. The number of stages is not determined in advance, but adapts according to the statistical data collected as the computation progresses. The

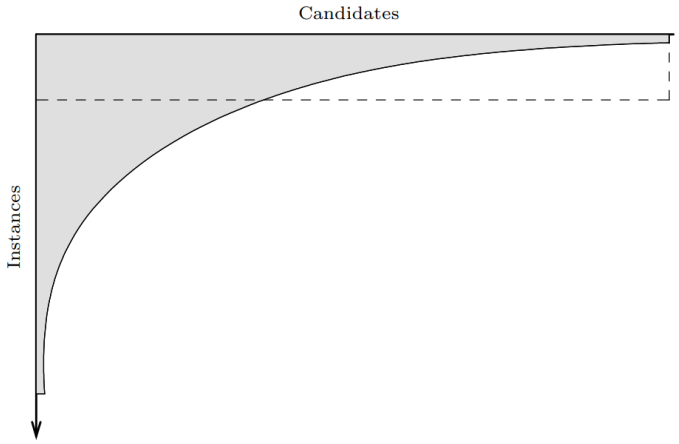


Figure 2.3: Graphical representation of the number of assessments required for the racing (shadowed figure) and brute-force approaches (dashed rectangle). For the racing approach, as soon as the statistical tests provide sufficient evidence, some candidates can be eliminated. On the other hand, the brute force approach evaluates all the candidates with the same number of instances. The two areas are identical because the number of simulations is determined [59].

graphical comparison between brute-force and racing approaches can be seen on figure 2.3.

A formal definition of the racing algorithm is proposed by Birattari *et al.* [51]. A sequence of k instances i_k is generated from the desired instance class I according to the probability model P_I. The cost of a single run for a candidate configuration θ on an instance i_k is noted as c_k^θ. At step k of the algorithm, the evaluation sequence is as follows:

$$c_k(\theta) = (c_1^\theta, c_2^\theta, ..., c_k^\theta) \tag{2.1}$$

A sequence of nested sets of configuration is generated as each step is completed.

$$\Theta_0 \supseteq \Theta_1 \supseteq \Theta_2 \supseteq ... \tag{2.2}$$

where Θ_k is the set of remaining candidates after k steps. Details about the selection of initial candidates can be found in [59]. The transition from Θ_{k-1} to Θ_k is achieved by the possible elimination of suboptimal candidates.
At step k, the remaining candidates forming Θ_{k-1} are evaluated on a new instance i_k. Each cost c_k^θ is added to the list $c_{k-1}(\theta)$ to form the new sequence $c_k(\theta)$ for each θ in Θ_{k-1}. The step ends by defining Θ_k leaving out configurations of Θ_{k-1} that are considered suboptimal by statistical tests applied to the array $c_k(\theta)$ for all θ in Θ_{k-1}.
This step is iterated until only one configuration remains, the number of instances to be tested has been reached or the budget allocated at the start has been exceeded.

2.3.2 F-Race

F-Race is a particular racing algorithm based on on the non-parametric Friedman's two-way analysis of variance by ranks [63] or just Friedman test. This algorithm was first proposed by Birattari *et al.* [51]. It is used as optimization algorithm for the AutoMoDe-`Vanilla` automatic modular design.

A description of the F-Race algorithm can be given by means of an example, assuming that step k has been reached and n represents the number of remaining configurations, $n = |\Theta_{k-1}|$. The Friedman test assumes that the costs are k mutually independent and n-variate random variables

$$\begin{array}{lcl} b_1 & = & (c_1^{\theta_1}, \ c_1^{\theta_2}, \ \ldots, \ c_1^{\theta_n}) \\ b_2 & = & (c_2^{\theta_1}, \ c_2^{\theta_2}, \ \ldots, \ c_2^{\theta_n}) \\ \vdots & \quad \vdots & \quad \vdots \quad \ddots \quad \vdots \\ b_k & = & (c_k^{\theta_1}, \ c_k^{\theta_2}, \ \ldots, \ c_k^{\theta_n}) \end{array} \tag{2.3}$$

where each b_l is called a block [64] and represents the computational results on an instance i_l by the remaining configurations at step k.
Within these blocks, costs are listed in ascending order, average ranks are used in case of ties. R_{lj} denotes the rank of the configuration θ_j in block b_l, $R_j = \sum_{l=1}^{k} R_{lj}$ being the sum of the ranks over all the instances i_l, $1 \leq k \leq l$. The Friedman test made to do the comparisons is the following [63]:

$$T = \frac{(n-1)\sum_{j=1}^{n}(R_j - \frac{k(n+1)}{2})^2}{\sum_{l=1}^{k}\sum_{j=1}^{n} R_{lj}^2 - \frac{kn(n+1)^2}{4}} \tag{2.4}$$

Under the null hypothesis that all candidates in a block are equally matched, T approximately follows an χ^2 distribution with $n-1$ degrees of freedom [65]. If T is more than the $1-\alpha$ quantile of this distribution, the null hypothesis can be rejected at the approximate α level meaning that at least one of the candidates can be considered significantly better than the others.
If the null hypothesis is rejected, comparisons can be made between the configurations to determine which is better. The Student's distribution t [63] is used to determine if configurations θ_j and θ_h are different. The condition is the following

$$\frac{|R_j - R_h|}{\sqrt{\frac{2k(1-\frac{T}{k(n-1)})(\sum_{l=1}^{k}\sum_{j=1}^{n} R_{lj}^2 - \frac{kn(n+1)^2}{4})}{(k-1)(n-1)}}} > t_{1-\frac{\alpha}{2}} \tag{2.5}$$

If the null hypothesis is not rejected, every candidates of the Θ_{k-1} set are in the Θ_k set. In the other case, the best candidate is pairwise compared with all the others. Every configuration that is determined different so significantly worse by the equation 2.5 is discarded from the set of remaining configurations. If only two candidates are remaining the Friedman test becomes the binomial sign test for two dependent samples [66]. Nevertheless, in the F-Race algorithm, Wilcoxon matched pairs signed-ranks test [63] is used because it is more adapted in this case [67].
Only ranking is used in the F-Race algorithm because it has two main advantages : the non-parametric nature of the ranking and the implementation of a blocking design [68]. The blocking design plays the role of normalisation by working on each instance at a time.

2.3.3 Iterated F-Race

The improvement to the algorithm previously presented is called Iterated F-Race [69] and consists of an iterative application of F-Race where candidates eliminated in the previous

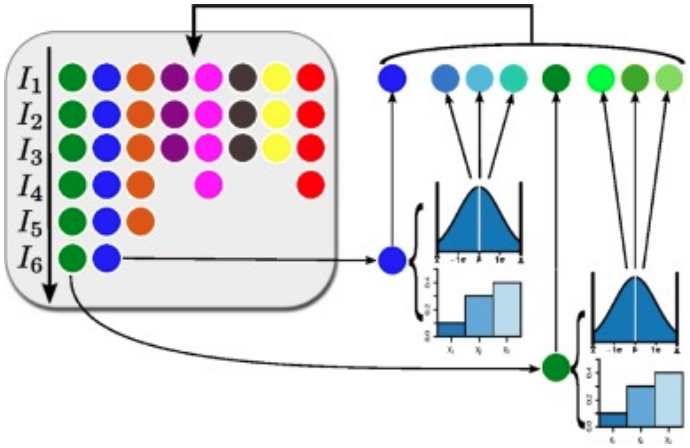


Figure 2.4: Scheme of the iterated racing algorithm [71]. After six problem instances tested, the two best control software's are selected to create new control software for the next iteration. The goal is to concentrate the search around the promising candidates while adding bias.

iteration are added to increase the bias at each iteration. With this change, Iterated F-Race follows the model-based approach [70], which is built in three stages: first, the construction of a set of candidates using a probability model; then, the evaluation of the candidates; and finally, the selection of the candidates by adapting the probability model with the best candidates to inject bias. The procedure is represented on figure 2.4. These steps are repeated until a termination criterion is reached. The pseudo-code of the general framework of this new algorithm is given in 1 by [59].

Algorithm 1 Iterated F-Race [59]

Require: parameter space X, a noisy objective function black-box f.
  initialize probability model P_X for sampling from X
  set iteration counter l = 1
  repeat
    sample the initial set of configurations Θ_0^l based on P_X
    evaluate set Θ_0^l by f using F-Race
    collect elite configurations from F-Race to update P_X
    $l = l + 1$
  until termination criterion is met
  identify the best parameter configuration x^*
  return x^*

The difficulty of Iterated F-Race is that a sufficiently number of configurations must be given to the F-Race algorithm for it to work well while the number of iterations should remain low because of the need of keeping a small number of steps given by the budget allocated to the configuration process. To overcome this limitation, an ad-hoc method was proposed for biasing the sampling in [69] but it was only tested with numerical parameters. Some issues are discussed in [59], conclusions are listed below:

- For a given computational budget, the number of iterations is determined by the number of wanted candidate configurations at each iterations
- A way of allocating the budget among the iterations is to divide equally but other strategies can be adopted
- As the configurations become more and more similar as the iterations increase, it is more interesting to have more configurations at the beginning
- An F-Race run, and therefore an iteration, stops if the budget allocated to that iteration is exceeded or if a minimum number of remaining configurations is reached. This minimum is calculated as a function of the dimension of the parameter space.
- The probability model used to select configurations from the parameter space depends on the desired exploration/exploitation trade-off.

2.3.4 `irace` Package

The implementation of the iterated F-Race is done through the `irace` package [71]. The use of this package can be found in the user guide [72]. The first step is to estimate the number of iterations required to optimize all the parameters. The estimation is $N^{iter} = \lfloor 2 + log_2 N^{param} \rfloor$. When the size of the parameter space increases, so does the number of iterations. The run performed at each of the iterations has a maximum budget $B_j = (B - B_{used})/(N^{iter} - j + 1)$, for $j = 1, ..., N^{iter}$. Each run starts with a set of configurations Θ_j which size depends on the iteration j, $|\Theta_j| = N_j = \lfloor B_j/(\mu + T^{each} \cdot min\{5, j\}) \rfloor$, where T^{each} is the number of instances at each of the iteration. The number of configurations is decreasing when going through the iterations in order to have more evaluations per configuration. The parameter μ is by default the number of instances needed for the first iteration (T^{first}). This formula goes in the direction of one the conclusions made when discussing Iterated F-Race (section 2.3.3).

The algorithm begins by sampling a set of configurations in parameter space (first determining the unconditional parameters). All configurations are then tested via a run on the first instance. The algorithm continues instance testing with a subset of the configurations until it reaches the maximum number of instances (T^{first}). At the end of each race, the Friedman test [63] or the paired t-test [73] is applied to determined if some configurations can be discarded.

The first statistical test is performed after the first evaluation. Then it may be useful to evaluate the configurations several times before applying the statistical test. Instances can belong to different classes, and tests are applied after configurations have been run on all classes.

A race stops when the bugdet does not allow to evaluate all the remaining configurations on the next instance or when the number of configuration has reached the minimum defined before (N^{min}). Each configuration is then associated to a rank r_k which is the sum of ranks or the mean cost, depending on the statistical test. The configurations with the lowest rank, $N_j^{elite} = min\{N_j^{surv}, N^{min}\}$ form the elite configurations Θ^{elite}. The number of new configurations for the next iteration is determined by $N_{j+1}^{new} = N_{j+1} - N_j^{elite}$. The

new configurations are constructed thanks to parents sampled in the elite configurations of the previous iteration.

The algorithm stops when the budget has been used up or the number of candidate configurations is no greater than the number of elite configurations in the previous iteration. If the number of iterations estimated at the beginning is reached but the remaining budget is enough to start a new iteration, N^{iter} is increased.

The `irace` parameters have been chosen to correspond to as many scenarios as possible. A study has been carried out to determine the best combination [74], but modifications can be made if necessary.

2.4 Multi-Objective Optimization in the Automatic Design

Previous optimization methods considered problems for which a single objective had to be achieved. Increasingly, however, the missions of swarms of robots are becoming more complex, and several objectives are being identified. This section first presents current methods for optimizing the parameters of an algorithm tackling multi-objective problems, and then presents the multi-objective optimizations currently used in neuroevolutionary robotics.

2.4.1 Performance Assessment of Multi-Objective Optimizers

In order to operate, a metric must be given to the automatic configuration, yet it is difficult to assess an optimization problem with several objectives. Bezerra *et al.* [75] reviewed some automatic configuration of multi-objective optimizers as well as their solution to tackle the performance assessment problem.

Ideally, when comparing two multi-objective optimizers, one should outperform every front of other one. However, this is rarely the case, which is why other methods to evaluate approximation fronts have been proposed to enable this comparison to be made.

The first methodology cited is the dominance rankings proposed by Knowles *et al.* [76]. Let's take two multi-objective optimizers to compare, Θ_1 and Θ_2, and define $A^1_{\Theta_1}$, $A^2_{\Theta_1}$,..., $A^r_{\Theta_1}$ and $A^1_{\Theta_2}$, $A^2_{\Theta_2}$,..., $A^r_{\Theta_2}$ their approximation fronts over r runs. Pareto optimality [77] is used to rank the fronts over the combined collection C of every approximation sets. Each set of approximation fronts is transformed into a set of ranks describing how good the fronts of that optimizer are compared to all the others. The rank sets can then be compared using a statistical rank test [63] to determine whether one optimizer has particularly smaller ranks than another. This method works well when the performance of the optimizers is significantly different.

Zitzler *et al.* [77] proposed the quality indicators to assess the performance of multi-objective optimizers. The metrics can either analitycally measure some characteristics a high-quality front should present or analitycally determine the difference between two

fronts and in the same way measure the difference with the Pareto front. These two cases describe two types of metrics: unary and binary metrics. As part of the comparison with the Pareto front, it is interesting to be able to measure the agreement of an indicator with this front. Formally, let $\mathcal{I} : \Omega \to \mathbb{R}$ be a quality indicator, which is to be maximized. $\mathcal{I}$ is said to be Pareto-compliant if, and only if, for every pair of approximation fronts (A, B) $\in \Omega$ for which $I(A) \geq I(B)$, it also holds that $B \not\preceq A$ [77].
True unary indicators do not provide good comparisons while respecting Pareto dominance, so binary indicators are more appropriate. However, the amount of information given when analysing several algorithms can be excessive. This is why the two ways of using quality indicators correctly would be to use binary indicators as a supplement to do rankings or to transform binary indicators into unary indicators by considering the comparison between the fronts and a reference front, the ideal being the Pareto front.

2.4.2 Automatic Configuration of Multi-Objective Optimizers

In this section the case of *multi-objective configuration* [78] will not be detailed. It considers that the configuration is itself a multi-objective problem with, more often than not, contradictory criteria such as quality and calculation time.

The focus will be on approaches proposed to solve the problem of *automatically configuring a multi-objective optimizer*, most often a metric is used to transform the front into a value. For example, Wessing *et al.* [79] proposed using the quality indicator. In this case, the multi-objective nature of the optimizer no longer applies to the algorithm. As the study of the automation of multi-objective optimization is fairly recent, few approaches have been proposed to solve this problem. Moreover, only `irace` allows complex performance metrics to be computed while the algorithm is running. Some proposals are described below, they are the first to tackle the problem of automatically configuring a multi-objective optimizer.

The multi-objective ant colony optimization (MOACO) framework was first introduced by López-Ibáñez *et al.* [80] and is an extension of the ant colony optimization (ACO) [81] for a biojective optimization problem, in particular the traveling salesman problem (TSP). It is the first template-based automatic algorithm design. This proposal is based on the separation of the MOACO into different components. The details of the components are not defined; they are inspired by the elements defined by the ACO in order to be able to concentrate on the multi-objective nature of the problem. The components concern the construction of the solution, the update of the pheromones and the multiple colonies. The aim of the algorithm is to characterise each of these components without modifying the features specific to the ACO.

The TP+PLS framwork proposed by Dubois-Lacoste *et al.* [82] is a combination of the two-phase local search (TPLS) and the Pareto local search (PLS) frameworks. TPLS uses the transformation of the multi-objective problem into a sequence of scalarisations solved via algorithms for the resulting single-objective problems. On the other hand, PLS is a local search method that uses Pareto dominance and an acceptance criterion. The final algorithm is a combination of these two phases.

Bezerra *et al.* [83] proposed the AutoMOEA framwork which exploits the predefinition of some bounds on solution quality in order to be able to discard the clear outliers. This new configurable framework is constructed with the application of the methodology on multi-objective evolutionary algorithms. It also takes into account the separation between algorithm and multi-objective parameters.

2.4.3 Multi-Objective Optimization in Neuroevolutionary Robotics

Trianni and López-Ibáñez [84] have identified several cases in which multi-objective optimization is used for neuroevolutionary robotic systems. These are the following:

- to solve single-objective design problems by transforming them into multi-objective; i.e. multiobjectivization;
- to solve design problems through a multi-objective approximation by proxies;
- to solve design problems that have multi-objective by nature.

Handl and Knowles [85] give a detailed description of these resolution techniques.
But most of the time, designers do not specify which technique is used to solve the problem described. For example, a collision term is often added to an objective function [86, 87], but it is not clear why. It may be a choice on the part of the designers, because they know that this term makes it possible to accomplish the mission more efficiently, but it could also be part of the mission's requirements, although this is not clear.

The studies proposed using AutoMoDe are clearer in this respect, and the designers use weighted sums to describe the specifications of the task in hand. These are made up of sub-missions to be carried out simultaneously [23] or in sequence [9]. For example, in Francesca *et al.* study [23], one of the tasks is based on two sub-tasks: coverage of the perimeter and area of two surfaces simultaneously. The objective function describing this mission combines the objectives of the two sub-missions into a weighted sum. This method therefore relies on the aggregation of objectives to work with a single objective function.

Two other techniques can also be used to manage multi-criteria control software production problems: illumination [88] and quality diversity [89] methods. While standard neuroevolution algorithms are based on observable behaviors in nature, they only take into account the optimization of mission performance, without considering the simultaneous diversification present in natural evolutionary processes [90]. Evolutionary algorithms exploiting this diversifier point of view are used as a basis for the development of quality diversity methods: novelty search [91], novelty search, with local competition [92] and MAP-Elites [88]. In the case of QD algorithms, unlike other methods that return several solutions, diversification is based on behavior. The designer selects behavioral features forming a subset in which the algorithm can work, in addition to the usual mission specifications. Features are used to form different behaviors, and results are obtained by searching the design space for predefined behaviors that perform well. The control softwares produced are more or less different from one another according to the behavioral features given, and the designer finally chooses which instance will be the best. The criteria are handle independently, without any aggregation.

Although examples exist, few studies have been carried out on the advantages and limitations of solving multi-criteria problems using multi-objective optimization. Trianni and López-Ibáñez [84] were the first to address this issue, comparing weighted sum, a way of aggregating information, with a multi-objective approach based on the estimation of the Pareto set. For the weighted sum, several sets of weights are tested, and for the multi-objective method, the hypervolume measure [93] is estimated to find the size of the design space drawn by the solutions obtained.

This study concluded that the weighted sum is most appropriate when the designer is able to choose the right set of weights, which limits the use of this technique in algorithms that are increasingly automatic and no longer require human intervention. On the other hand, the multi-objective technique can be used to solve non-linearly solvable problems for which the designer has no particular knowledge.

2.5 Sequence of Missions with Robot Swarms

For a long time, studies on robot swarms focused on the accomplishment of a single task, which consisted in performing a geometric or spatial task or mechanical abilities. However, it's becoming increasingly interesting to be able to combine different tasks and do them in sequence. This can also be useful when a large and complex job has to be solved all at once. A number of studies have been carried out to assess the performance of a swarm when several tasks have to be completed by switching from one to another [94] [86]. In these studies, the order of the assignments and the transitions, i.e. the time allocated to each assignment, is defined in advance by the designer as well as the controller used to realize the sequence. The next step proposed by Garattoni *et al.* [95] would be to automate the control design configuration enabling a mission sequence to be carried out. To carry out this task, the TAM [96] concept is used and presented below, as is the idea of autonomous task sequencing.

2.5.1 TAM

Task abstraction has been used a lot in the context of robot swarms in order to assess the performance of a design, the important feature is the interaction between robots and with the environment, not the details of the task itself so the abstraction is often used. Simulation allows a task to be carried out without actually consuming resources, but does not take into account the reality gap [97]. The specificity comes from relying on an ad hoc solution which makes the abstraction difficult to generalize and therefore unusable in situations other than the one in which it was created. As ad hoc solutions are rather impromptu, they do not make it easy to describe the tasks to be performed by a set of robots. This is why a new method has been proposed for task abstraction, based on an object called TAM (Task Abstraction Module) [96].
The task abstraction for a single-robot is defined as the occupation of the robot for a certain time at a certain place at a certain time [96]. The TAM, illustrated at figure 2.5, is able to represent this. To represent more complex tasks carried out by groups of robots, these tasks need to be divided into simple missions that can be represented by a TAM each time. Each TAM abstracts a single-robot subtask and they are then all linked together

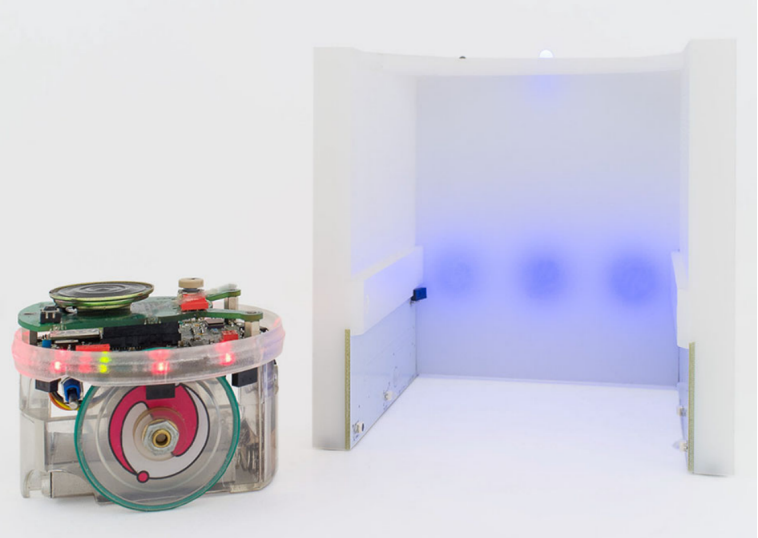


Figure 2.5: TAM and an epuck robot. The robot can fit into the TAM box, which contains RGB LEDs. Once the robot is completely inside, it can be considered to be performing the task abstracted by TAM. Several TAMs can be linked together to create relationships between abstracted tasks[96].

to represent the connection between the subtasks as well. The new approach to realize these two steps and abstract complex missions is composed of the TAM, the modeling tools for complex tasks, and the control framework for controlling the collection of TAM's.

When tasks are divided into sub-tasks, they can be done in the same place or in different locations in the arena. In this latter case, unlike single-robot tasks, the missions would no longer be stationary. The complex task is modelled on two distinct levels. The high level is used to determine the hierarchy between the subtasks and interrelationships, but without defining the details of the single-robot stationary sub tasks. The low-level model is the stage at which the details of the subtasks are defined. The control framework is a centralized framework that manages all the TAMs according to the interrelationships defined by the high-level model. Thanks to this comprehensive framework, all complex tasks can be modelled.

2.5.2 Autonomous Task Sequencing

Garattoni *et al.* [95] proposed a new swarm, TS-Swarm, to collectively achieve sequencing tasks in an unknown order. Instead of coding the transition before execution, TS-Swarms sequences the missions at run time and autonomously. The two existing paradigms deliberative [98] (sense-model-plan-act) and reactive [99] (sense-act) are mixed so that the model and plan of sequencing comes from the interactions at the collective level.
They chose to create a swarm that can do a certain number of tasks without repetition in an unknown order by one or multiple robots. The different tasks have to be done in a specific area. Some robots in TS-Swarm create chains to help mission robots knowing in which area they are and keep track of the order in which the missions are and have to be

done. The robots are not aware of their location in the arena, they can not know where they are and they are not able to sequence the tasks individually on the basis of their order execution.
Chaining in swarm robotics is not a new feature, it already has been used to create waypoints [100, 94]. The robots can follow the chain and know what mission to execute next. As soon as the mission is done a feedback is given to the robot for it to know if it was the right mission to execute or not. The different types of TS-Swarm consider three or four missions and an immediate or final feedback, which is given a the end of the sequence. TAMs 2.5.1 are used to abstract the subtasks.

The results show first that the three-task sequences can be extended to four tasks. When feedback is only given at the end of the sequence the chains help the robots to locate themselves and construct also different permutations of sequences tested initially. This allows the swarm to explore different possibilities if the feedback is negative. All tests allow to conclude that the sequencing of tasks is possible even if the feedback is given after the whole sequence and if the number of tasks go from three to four. Some limitations and solutions are proposed at the end including the modification of the swarm to enable the autonomous determination of the number of tasks.

This novel swarm is investigated within the domain of automated sequencing of established abstract tasks. The robots are tasked with determining the optimal task allocation and temporal arrangement. Notably, the robots are not required to possess comprehension regarding the task execution methodology. This distinction is notable when compared to `Demo-Fruit`, wherein all essential task-related information is provided a priori, yet the instructions on task execution or the triggering mechanism for transitioning between tasks are not imparted to the robots.

Chapter 3

Methods

The aim of this master thesis is to propose an automatic controller design for robots belonging to a swarm and having to carry out a sequence of two missions, without giving access to an explicit objective function describing one or both objectives. This section describes the methodology used to create this automatic design. Firstly, by presenting the extension of Gharbi's work on automatic design by demonstration [8], and secondly, by talking about Demo-Fruit, the framework used to switch from demonstration sequences to the swarms robot controller carrying out mission sequences.

3.1 Automatic Design through Sequences of Demonstrations

The aim of this work is to extend the `Demo-Cho` [8] principle to mission sequences in order to better represent the demands and needs for the use of swarms robots. Adaptation begins with the demonstration of sequences of missions, as detailed in the first section below. These new demonstrations are then integrated into the process put in place for the simple missions.

3.1.1 Notion of Sequences of Demonstrations

Instead of trying to build complicated objective functions to apply to robot swarms, it is easier to pass on expert knowledge directly. One way of doing this is to show the robots what to do. This is the principle used by `Demo-Cho` and extended here to mission sequences. The robots will have to change their behaviour after half of the steps of the mission in order to complete a second task in an equivalent time, respecting the neutral compromise.

For single-task missions, the demonstration made by an expert to show the robots the task to be carried out is the set of final positions expected in a specific context like the arena where the robots are going to conduct the experiments. An expert positions the robots within the swarm at the designated final location corresponding to the conclusion of the allocated mission time. By doing so, he or she defines the spatial organization of the swarm for the mission. This is one of the limitations of the type of mission: a demonstration cannot describe a mission in several stages, at different places. The pursuit of

expanding the capabilities of `Demo-Cho` to address more intricate missions, particularly those involving temporal considerations, prompted the adoption of a strategy involving the division of intricate missions into sequences comprised of individual single-task missions.

In this context of task sequences, it is essential to adapt the demonstrations provided to the swarm. The final mission position is no longer sufficient. Instead, a demonstration must be given for each single-task mission. The concept remains unchanged; the final positions are defined by an expert. For this master thesis, a sequence of two demonstrations is provided for each mission. The order of the sequence is important. More precisely, this sequence of demonstrations is a concatenation of the robot positions after half the steps allocated for the mission and after all the steps. The time allocated for each task is the same, to avoid making assumptions about the importance or difficulty of a task and make the automatic modular design the most general.

3.1.2 Application in the Automatic Design

For the algorithm, the positions must be pre-processed to become features that describe the final state of the swarm robot. The features used are the same as those defined by Gharbi [8].

Abbeel *et al.* [41] defined the feature vector ϕ as $\phi : S \mapsto [0,1]^k$ where S is the set of final positions and k is the number of features. In the case of sequences of n tasks, the number of features is $k = nK$ where K is the number of features of a single-task mission. As the work is done by a swarm of robots, the feature vector of each task is already a concatenation of information about each robot.

The expected feature vector, representing the behaviour to reproduce is called μ and is the average of several ϕ vectors for several demonstrations or sequences of demonstrations. As this is the average, both notations are considered equivalent for the following explanations. Features have to be based on the position of the robots because it is the only available information. The first idea of features was to compute multiple features for each robot, one feature per robot per patch included in the arena. This feature is computed as follows:

$$\mu_{patch_{ij}} = \begin{cases} 1 & \text{if robot i is inside patch j;} \\ 0 & \text{if an obstacle is between robot i and patch j;} \\ e^{-\frac{2ln(10)}{d} x_{patch_{ij}}} & \text{otherwise.} \end{cases} \tag{3.1}$$

Where $i = 1, ..., n$ is the number of robots composing the swarm, $j = 1, ..., m$ is the number of patches present in the arena, d is the diameter of the arena and $x_{patch_{ij}}$ is the the ith smallest distance from a robot of the swarm to the jth floor patch. If the ith nearest robot is inside the patch, the feature $\mu_{patch_{ij}}$ is set to 1. The higher the value, the closer the robot. So maximizing the value of $\mu_{patch_{nj}}$ ensures that all feature values related to this patch j will be maximized because the distance between the patch j and the furthest robot will be minimized.

The other type of feature per robot concerns the relations between the robots of the

swarm, the distance between themselves. It is more specially proportional to the distance with the closest neighbor:

$$\mu_{neigh_i} = e^{-\frac{2ln(10)}{d} x_{neigh_i}}, \tag{3.2}$$

where $i = 1, ..., n$ is the number of robots composing the swarm and x_{neigh_i} is the ith smallest distance between one robot and its closest neighbor in the swarm.

The feature vector for one sub-mission is

$$\mu_s = (\mu_{patch_{11}}, ..., \mu_{patch_{1m}}, ..., \mu_{patch_{nm}}, \mu_{neigh_1}, ..., \mu_{neigh_n}), s \in \{1, 2\} \tag{3.3}$$

The value of s is either 1 or 2 because there are two steps in the sequences. The distance values are the one corresponding to the positions of the robots at the half of the mission if $s = 1$ and at the end of the mission if $s = 2$.

Some additional features were proposed and tested on one particular mission for which the above were not enough. Details can be found in the appendix A.

3.2 Demo-Fruit

`Demo-Fruit`, the method we propose, is an automatic modular design for generating control software for robots in swarms that execute mission sequences. The robots are able to perceive and emit RGB color via LEDs. More precisely, `Demo-Fruit` is a combination of the adaptation of the apprenticeship learning algorithm via inverse reinforcement learning [41] to robot swarms and an automatic modular design from the AutoMoDe family, in this case the AutoMoDe flavour needed to include a way of triggering the change in behaviour, `TuttiFrutti` or `Mandarina` are composed of modules able to handle the communication through colors. A color change serves as a trigger to warn robots of the swarm.

Five fundamental components characterize this automatic modular design: the robot platform, the simulator to execute experiments, the interface to make the demonstrations, the modules and transitions available for building PFSMs, and the design process that optimizes control software. In the following subsections, these components are explained and their links highlighted.

3.2.1 Robot platform

`Demo-Fruit` produces control software for a version of the epuck robots [25]. The extended version that is used, shown in the figure 3.1, are simple robots that roll on two wheels and are regularly used for swarm robotics studies. The reference model RM 3, an extension of RM 1.1, is represented in table 3.1. The notion of reference model is a formalization of the capabilities of a robotic platform used to perform experiments (in simulation and in reality)[101].

The different inputs and outputs are abstraction of the sensors and actuators of the epucks. Inputs represent sensors, they can only be read by the control software. On the

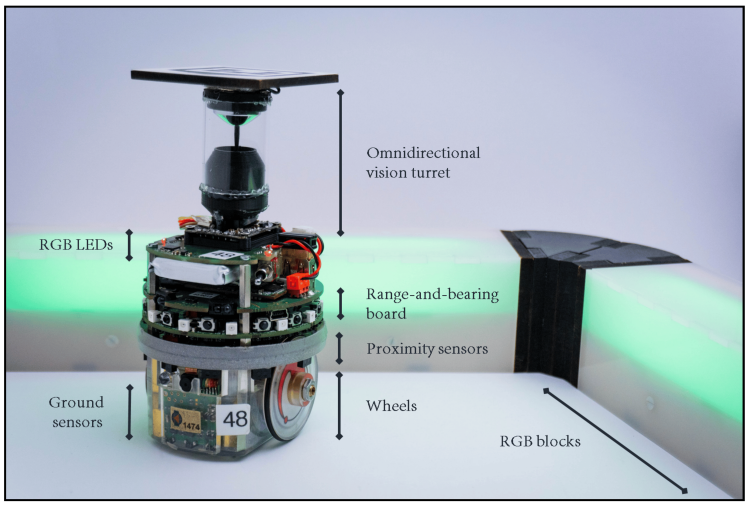


Figure 3.1: Extended version of epuck used to develop Demo-Fruit. The various sensors and actuators are shown in the photo, along with the RGB blocks used for TuttiFrutti [9].

other hand, outputs are the actuators and they can only be written. Every 100ms, every inputs and outputs are updated.

The proximity sensors are represented by the variable $prox_i$ which has a range value between 0 and 1, $prox_i = 0$ if nothing is perceived in a circle of 0.3m around the epuck. The value is 1 if an object is perceived at less than 0.1m. The three ground sensors, noted gnd_j, are able to detect the color of the floor below the robot. The floor can be gray, white or black depending on the patches that are used for the experiments, most of the time the background is gray and patches are white or black. To perceive the neighboring epucks, robots use their range-and-bearing board, n represents the number of close robots. The epucks don't have access to the angle and distance to all the neighboring robots but only to the vector of attraction V_n that gives the direction to the center of mass of the close robots. The big difference with reference model RM 1.1 is that the epucks are not able to detect the ambient light, instead they can perceive colors. The omnidirectional vision turret acts like a light sensor for different colors : red (R), green (G), blue (B), cyan (C), magenta (M) and yellow (Y). The turret allows to perceive these colors in a circle of radius 0.5m around the robot. Every cam_C is linked to an attraction vector to the robot or wall that displays the color C.

The actuators to control the speed and the direction of the robot are accessible for the control software through two variables : v_l and v_r. These variables take the values, between -0.12 and 0.12m/s, to describe the velocities of the two wheels of the epuck, left and right. The LEDs can be used by the control software to display cyan, magenta or yellow to enable a new sort of communication between robots.

The reference model RM 3 was first used by `TuttiFrutti`. Before this novelty, `Vanilla`, `Chocolate` and `Maple` are using RM 1.1. RM 2 can handle the exchange of binary messages and is therefore used by `Gianduja`.

Input	Value	Description
$prox_{i\in1,...,8}$	$[0, 1]$	reading of proximity sensor i
$gnd_{j\in\{1,...,3\}}$	$\{Gray, White, Black\}$	reading of ground sensor j
n	$\{0, ..., 20\}$	number of neighboring robots detected
V_n	$([0.5, 20]; [0, 2]\pi$ rad)	their relative aggregate position
$cam_{C\in\{R,G,B,C,M,Y\}}$	$\{yes, no\}$	colors perceived
$V_{C\in\{R,G,B,C,M,Y\}}$	$(1.0; [0, 2])\pi$ rad	their relative aggregate direction
Output	**Value**	**Description**
$v_{k\in\{l,r\}]}$	$[-0.12, 0.12]$m/s	target linear wheel velocity
$LEDs$	$\{\varnothing, C, M, Y\}$	color displayed by the LEDs

Table 3.1: Reference model RM 3 [101]. Inputs and output concerning colors are new compared to reference model RM 1.1, differences are highlighted in cyan. Robots are able to perceive red (R), green (G), blue (B), cyan (C), magenta (M) and yellow (Y) and they can emit no color ($\varnothing$, cyan (C), magenta (M) and yellow (Y).

3.2.2 ARGoS

ARGoS [102] has been developed to simulate the behavior of epucks in experiments. ARGoS is multi-physics simulator designed for swarm robotics. The platform is used to simulate robot swarm missions with epucks, figure 3.1, through the avalaible epuck plugin [24]. ARGoS was used for most of the experimental simulations as part of the development of AutoMoDe flavors. An experiment to be simulated with ARGoS is described by an XML file (.argos) containing all the necessary information to realize a mission. ARGoS also enables the visualization of the simulation in order to evaluate the behaviour of the robots visually.

The following tags are always present in the .argos file:

- The framework containing the mission length and the number of steps available for the sequence.
- The loop function is used to run the simulation, describing the arena and the environment in which the robots evolve: arena shape, size and patches, as well as the actions to be performed at each step and at the end of the mission. It is also in this section that the positions of the expert demonstrations made with orchestra are included.
- The controllers containing information on software design control and epuck controller.

- The arena gives more information about the arena, in this case the position and the intensity of the light (not used in the modules for `Demo-Fruit`. The arena also contains the epuck tags that provides all information about the epucks, their positions and the controller to use for example.
- The physics engines describes what engine is used for the simulation.
- The media tag includes the specifications of LEDs and range-and-bearing of the epuck.
- The visualisation tag enables the visualisation of ARGoS used to do visual inspection.

3.2.3 Orchestra

Before simulating experiments with ARGoS, the demonstrations must be made and converted into set of positions so that they can be written to an .argos file. Orchestra is used for this purpose.

Orchestra is the interface used and developed by IRIDIA using game engine Unity. The aim is to monitor a mission carried out by a swarm of robots. ROS [103] is used for communications between the robots and Orchestra. During a mission, the user can retrieve sensor values from epucks to analyze their behavior. One or more robots can also be controlled directly by a user. Thanks to ROS, several users can interact with the same swarm.

Gharbi [8] has extended Orchestra's capabilities by adding an arena builder and a demonstration builder. The interface can be used to create an arena for experiments by placing walls and patches, and can also be used to place robots on an arena and record their positions for demonstrations.

3.2.4 Set of Modules

For this new automatic modular design of control software, modules and transitions of `TuttiFrutti` [9] are used because a visual trigger was needed to signal a sub-mission change halfway through the mission. In order to handle the perception and emission of colors, low-level behaviors and transitions had to be modified. AutoMoDe-`TuttiFrutti` is an extension of AutoMoDe-`Vanilla` and is conceived to work with RM 3.

Low-level behaviors

The first four behaviors in the table 3.2 were already presented with the AutoMoDe-`Vanilla` flavour. The `EXPLORATION` behavior describes the straight movement of the robot before detecting an obstacle with the proximity sensor in front, $prox_i$. If the detection occurs, it rotates on itself for a certain number of times determined by $\tau \in \{0, ..., 100\}$. The epuck does not move if it is in the `STOP` behavior. `ATTRACTION` and `REPULSION` are linked together and work almost the same way. For these behaviors, robot moves in the V_d or $-V_d$ direction meaning closer or farther from the other robots, respectively. The

Atomic Behavior	**Parameters**	**Description**
EXPLORATION	$\{\tau, \gamma\}$	movement by random walk
STOP	$\{\gamma\}$	standstill state
ATTRACTION	$\{\alpha, \gamma\}$	movement towards the center of mass of the robots
REPULSION	$\{\alpha, \gamma\}$	movement away from the center of mass of the robots
COLOR-FOLLOWING	$\{\delta, \gamma\}$	steady movement towards robots/objects of color δ
COLOR-ELUSION	$\{\delta, \gamma\}$	steady movement away from robots/objects of color δ
State Transition	**Parameters**	**Description**
BLACK-FLOOR	$\{\beta\}$	black floor beneath the robot
GRAY-FLOOR	$\{\beta\}$	gray floor beneath the robot
WHITE-FLOOR	$\{\beta\}$	white floor beneath the robot
NEIGHBOR-COUNT	$\{\xi, \eta\}$	number of neighboring robots greater than ξ
INVERTED-NEIGHBOR-COUNT	$\{\xi, \eta\}$	number of neighboring robots lower than ξ
FIXED-PROBABILITY	$\{\beta\}$	transition with a fixed probability
COLOR-DETECTION	$\{\delta, \beta\}$	robots/objects of color δ perceived

Table 3.2: Atomic Behaviors and Conditional State Transitions of the AutoMoDe-TuttiFrutti flavour. It is an extension of the AutoMoDe-Vanilla, the differences are highlighted in cyan [9].

velocity is calculated with n, the number of neighboring epucks and a parameter α. If case $n = 0$, the movement is simply straight. `COLOR-FOLLOWING` and `COLOR-ELUSION` follow the same principle as the last two behaviors. The velocity is constant in the direction (V_C) or away from ($-V_C$) the color C that was detected cam_C, in case no color is perceived, the robot goes straight. The first parameter characterizes the color detected by the sensor is noted $\delta \in \{R, G, B, C, M, Y\}$. Colors are not all emitted by all the elements in the arena, robots can only display $\delta \in \{C, M, Y\}$ and objects should be able to emit the three other colors $\delta \in \{R, G, B\}$. The obstacle avoidance used for `EXPLORATION`, `ATTRACTION`, `REPULSION`, `COLOR-FOLLOWING` and `COLOR-ELUSION` is the one proposed by Borenstein *et al.* [104]. The parameter γ present for every behavior describes the color emitted by the robot itself. All the parameters are part of the elements tuned by the optimization process to create the best PFSM for the mission.

Transitions

Each transition is accompanied by a probability that the behaviour will change if the transition condition is met. For the first three transitions, `BLACK-FLOOR`, `GRAY-FLOOR` and `WHITE-FLOOR`, the transition occurs with a probability β if the robot is wheeling respectively on a gray, black or white patch. The probability of transitioning for the `NEIGHBOR-COUNT` and `INVERTED-NEIGHBOR-COUNT` transitions is calculated from two parameters and the number of neighboring robots, n. It is calculated as $z(n) = \frac{1}{1+e^{\eta(\xi-n)}}$, where $\xi \in \{0, 10\}$ and $\eta \in [0, 20]$ are tunable parameters for the optimization algorithm. The `FIXED-PROBABILITY` is only about the probability to transition. The last transition is the `COLOR-DETECTION` is about the eventual color perceived by the color sensor cam_C. The color to react to is described by $\delta \in \{R, G, B, C, M, Y\}$ and the change in behavior occurs with probability β.

With the new possibility to perceive and emit colors, `TuttiFrutti` was extended with two new behaviors `COLOR-FOLLOWING` and `COLOR-ELUSION` and one new transition, `COLOR-DETECTION`. All the transitions are exactly the same as those implemented in the AutoMoDe-`Vanilla`. `EXPLORATION`, `STOP`, `ATTRACTION` and `REPULSION` were modified to correspond better to this automatic design method.

3.2.5 Design Process

The design process is separated into two phases, first the application of the apprenticeship learning [41] to find the implicit objective function and the optimization with the use of `irace` package (section 2.3.4), the implementation of Iterated F-Race (section 2.3.3). These two phases are iterated a fixed number of times in order to find the best PFSM possible.

Apprenticeship learning

The first phase concerns the use of features generated with the final positions to create the implicit objective function. Features have been modified and adapted to correspond to the context of swarm robotics and to the experimental setup. The objective function is a linear combination of the different features calculated before. The weighted sum is constructed thanks to a Support Vector Machine [105].

A linear Support Vector Machine supposes that the features are linearly separable. This is a strong hypothesis that could be more studied to verify if a non-linear model would be better to define some missions. In this case, the feature space has 80 dimensions and the frontier between the two classes is an hyperplane. The two classes are the expected value, μ_E and the experimental values, $\mu^{(i)}$ where i is represents the iteration at which the experimental value was calculated. The experimental values are evolving at each iteration and the goal is to be as close as possible to the expected value. The SVM uses a simple hyperplane equation to determine the two domains for the two classes.

$$\begin{aligned} D_1 &= \{\mathbf{x} : \mathbf{w}\mathbf{x}' + \gamma \leqslant 0\} \\ D_2 &= \{\mathbf{x} : \mathbf{w}\mathbf{x}' + \gamma > 0\} \end{aligned} \tag{3.4}$$

From these equations can be derived two parametrization optimization objectives by considering that the label of the first domain is 1 and -1 for the second one.

$$\begin{aligned} \mathbf{w}\mathbf{x}' + \gamma &= 1, \mathbf{x} \in D_1 \\ \mathbf{w}\mathbf{x}' + \gamma &= -1, \mathbf{x} \in D_2 \end{aligned} \tag{3.5}$$

The goal of these equations is to determine the boundaries between the two classes by using two hyperplane. The objective function is the distance between the domains and the SVM tries to maximize it which is why the experimental points being closer the expected value should give more accurate objective function. The lines are parallel so the distance can be expressed like

$$d = \frac{\pm 2}{\sqrt{\mathbf{w}\mathbf{w}' + 1}} \tag{3.6}$$

This distance is used as measure for the optimization problem so no generality is lost if [105]

$$d = \frac{\pm 2}{\sqrt{\mathbf{w}\mathbf{w}'}} \tag{3.7}$$

Finally, the norm notation is the preferred one when writing the Support Vector Machine optimization problem. For this work, we will use the term margin and this notation to describe the distance between the hyperplanes.

$$t = \frac{2}{||\mathbf{w}||} \tag{3.8}$$

The development above allows to understand how the features and the SVM are used to construct the implicit objective function needed for the apprenticeship learning algorithm. This reward function is written $R = w.\mu$ where μ is the feature vector and w the set of weights compiled with the SVM. The evolution of w can be seen on the figure 3.2. The expert value, mean of the five demonstrations given to the algorithm, is labeled $+1$ and noted μ_E. The experimental values are noted $\mu(\pi^{(i)})$ on the figure and just μ_i then and labeled -1. The stop criteria of the algorithm used in `Demo-Fruit` is the same as the one used in `Demo-Cho` and is just the number of iterations determined before execution.

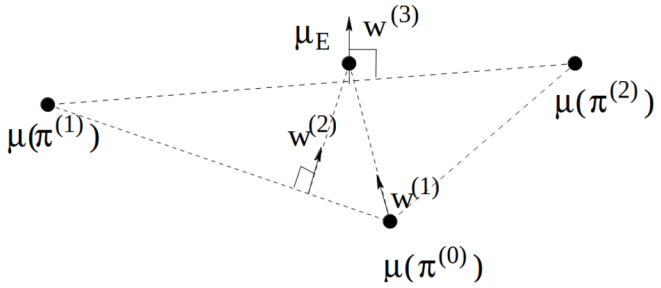


Figure 3.2: Graphical representation of the three first iterations of the apprenticeship learning algorithm. Each iteration contributes to the elaboration of a weights vector distributing better and better the objective to reach [41].

The adapted apprenticeship algorithm can be resumed with these steps

1. Demonstrate the two-tasks mission that as to be accomplished by the robot swarm and determine the number N of iterations.
2. Compute the features of the expert demonstrations for each of the submissions, calculate the mean values, μ_{1E} and μ_{2E} and label them $+1$ in the $PFSM$ history.
3. Set $i = 0$, generate a random $PFSM_i$ and compute the resulting feature vectors. Label the latter as -1 in the $PFSM$ history.
4. Use the SVM to compute $t_s^{(i)}$, the margins and $w_s^{(i)}$, the weight vectors to construct $R_s^{(i)}$, $\forall s = \{1, 2\}$.
5. If $i = N - 1$, stop the algorithm and return the $PFSMs$, post-processing should be applied to select the best $PFSM$.
6. Run AutoMoDe-`TuttiFrutti` with the reward $R^{(i)}$ in the arena that serves to demonstrate to compute the $PFSM^{(i+1)}$. Step to be modified according to the optimization process chosen.
7. Compute the resulting feature vector, label $PFSM^{(i+1)}$ -1 in the $PFSM$ history and increment the i value.
8. Go back to step 4.

Optimization

There are two different ways tested to optimize the PFSM in the `Demo-Fruit` algorithm, single-objective optimization and multi-objective optimization.

First, the modification of the reward function in order to keep a single-objective for the optimization step (6 in the pseudo-code). The scalarization is a common used way to tackle

multi-objective problems and it is the most represented in the field of swarm robotics [84]. Each part of the sequence is modelized by a reward function to be minimized.

$$\begin{array}{ll} min & R_1 = \sum_i w_{1i}.\mu_{1i}, \quad i = 1, ..., 80 \\ min & R_2 = \sum_i w_{2i}.\mu_{2i}, \quad i = 1, ..., 80 \end{array} \tag{3.9}$$

The idea with the scalarization is to construct only one objective function for the two parts like

$$min \quad R = \sum_i w_{1i}.\mu_{1i} + \sum_i w_{2i}.\mu_{2i} \quad i = 1, ..., 80 \tag{3.10}$$

In the optimization step, $R^{(i)}$ is the scalarized version of the sequence reward function. There is no additional weights in the weighted sum of the reward functions because of the automatic nature of the `Demo-Fruit` algorithm and the neutral compromise. If a weight is needed for one or two of the submissions, it will depend on the submission itself and the combination of the sequence. The hypothesis for now is that the two tasks have the same importance and the same difficulty or differences sufficiently low to be handled by a simple weighted sum. A good way of improving `Demo-Fruit` could be to design an automatic method for adapting weights as iterations progress if the equivalent weights are insufficient.

The second method exploits the `irace` package features to optimize a multi-objective problem, is called `Mandarina` [106] and is in developpement right now. The iterated F-Race algorithm optimizes, fine tunes the PFSM according to mission-specific criteria : the objective function and the design criteria. The created PFSMs are made of maximum four states and every transition links two different states.

The `irace` package that implements iterated F-Race begins by looking for the search space of the finite-state machines useful for the mission. It performs different simulations using ARGoS3 (section 3.2.2) to assess the performance of the different PFSM tested. The number of simulations allowed at each iteration of the process is determined before the execution. When the maximum number of simulations is reached, the best PFSM so far is returned to the apprenticeship algorithm.

The difference in `Mandarina` is that the specifications of the missions given to `irace` characterize bi-objective optimization problems. The only modification from the usage in AutMoDe-`Chocolate` or AutoMoDe-`TuttiFrutti` is that the candidate solutions are evaluated for both the objectives and not only one. The comparison between the original and the new racing algorithm is shown at figure 3.3.
As explained in the section 2.3.3, the iterated F-Race works in three phases :

1. Generating control software from a determined distribution.
2. Selecting the best control software from the sampled ones.
3. Adapt the distribution in order to bias the sampling towards the best control software found.

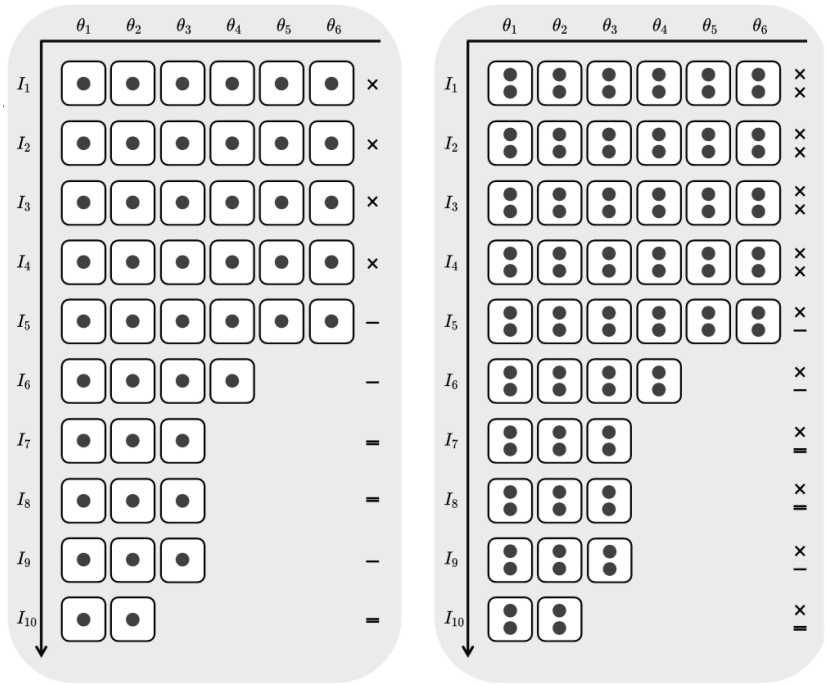


Figure 3.3: Comparison between the original racing algorithm (on the left) and the `Mandarina` version (on the right) to handle bi-objective optimization problems and evaluate two objective functions. In both cases candidate solutions θ_i are evaluated on a certain number of problem instances I_k. Squares represent the result of the evaluation of a candidate solution on a problem instance. Inside the squares, the dots (•) show the number of objective functions evaluated. Statistical tests are done to determine if the solution should be discarded or not, symbols at the right of the tables show the decision. '×' means that no test was performed, '−' stands for the elimination of at least one candidate and when no candidate is discarded it is noted '=' [106].

Difference with the original version is that the tests return two values, one for each of the objectives. The iterated F-Race then performs the Friedman's test on the two values. The goal of this technique is to avoid trying to compare the performance values of a candidate on a problem instance. The iterated F-Race algorithm does not work with the performance values but more the average rank of the solutions. The results of `Mandarina` were better than `TuttiFrutti`, `EvoColor`, `NEAT-Color` and `TuttiFrutti-HV` so the expectation is that this method will also work better than a weighted sum with `Demo-Fruit`.

The condition for `Mandarina` to be able to handle a biobjective optimization problem is that the biobjective function can be expressed as a set of two independent objective functions.

The code to use `Demo-Fruit` can be found here.

Chapter 4

Experimental Set-Up

This chapter describes the experiments carried out to measure and study the performance of two optimization methods on biobjective swarm robot missions. As there is no model to study this performance for the automatic design of robot swarms, a discussion on performance measurement was conducted. Several methods are used, including performance aggregation for each sub-mission and visual inspection. The set of missions used to assess `Demo-Fruit` is also presented.

4.1 Missions Framework

The basic principle of the mission framework is to compose sequences using a set of submissions. The peculiarity is that no objective function is provided to measure performance, only demonstrations are given to the algorithm.

In order to better understand and imagine the concrete use of mission sequences, the robot swarms evolve in a event room that can be used for several purposes. The swarm of robots takes care of customers, with two missions to be carried out in sequence for each type of event. This environment does not mean that `Demo-Fruit` cannot be used in any particular setting, it is just a way of explaining the missions and scenarios more easily. Each mission must be carried out within a defined and equivalent timeframe for both parts. It is assumed that `Demo-Fruit` makes no assumptions about the importance or difficulty of a task, respecting the neutral compromise. The aim, especially when no objective function is to be given, is to be as general as possible and adaptable to any situation.

4.1.1 Experimental Environment

The event room abstraction is built to allow robots to evolve, and consists of a bar, a kitchen side for preparing food, a kitchen side for washing up and twenty epucks (section 3.2.1). The experimental environment, simulation visible on figure 4.1, is an hexagonal arena of 2.60m^2 composed of RGB blocks that are aggregated in six walls. The two black floor patches represent the two areas of kitchen; on the left, the place to do the dishes and on the right, the spot to cook the meals. The white patch is the bar where the drinks are prepared. The gray floor is the room to which customers have access. The walls are made up of four 0.25 m RGB blocks. Each part of the wall can be controlled independently of

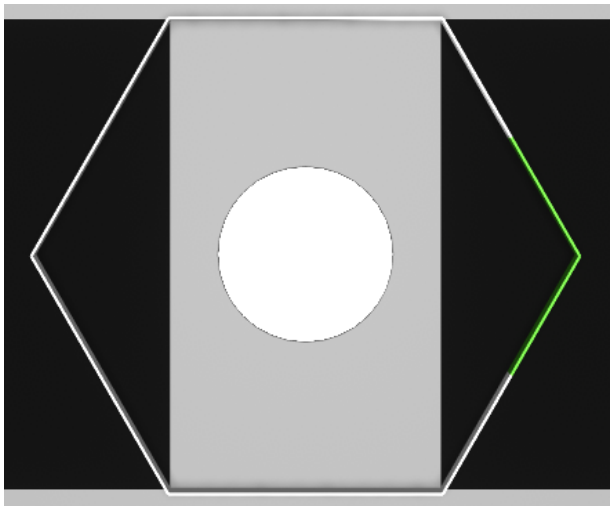


Figure 4.1: Simulation of the abstraction of the event room with floor patches and RGB blocks to create an arena for the twenty epucks. The green walls do not change of color, other walls are used as triggers to switch from one mission to another by changing color.

the others. The four green blocks remain green throughout the sequence to differentiate the two black patches, while all other wall blocks are red for 600 time steps and then blue for 600 time steps. This indicates to the robots what mission they should perform. An interesting and exploitable aspect of this arena is the orthogonal symmetry of the horizontal axis passing through the center.

4.1.2 Sub-Missions

In order to achieve a sufficient number of mission sequences, a set of six missions has been proposed. The missions are as follows: BAR, WELCOME, CLEANING, DISHES, MEALS and SERVING. These missions have been designed to suit the environment in which the twenty epucks with reference model 3 will operate. They are inspired by missions used to assess the performance of design control software from the AutoMoDe family [7, 23, 29, 30, 32, 33, 35, 38, 40].
The missions are just a part of what `Demo-Fruit` should be able to perform. Each of the sub-mission is associated with a description and a demonstration to illustrate the final position of the swarm doing the mission. There is no clear performance metric, this will be discussed later. Descriptions and demonstrations are listed below:

- BAR (m_{B}): the robots have to prepare abstract drinks in the bar to serve the clients. The preparation is done if the robots are in the center of the arena. They have to gather on the white patch. Robots should use local information on wall color, floor color and number of neighbors to accomplish the mission. The five figures 4.2 represent the expected behavior at the end of the time dedicated to the mission.
- WELCOME (m_{W}): the robots welcome customers as they enter or accompany those leaving at certain edges of the room. To complete this mission, robots must stick

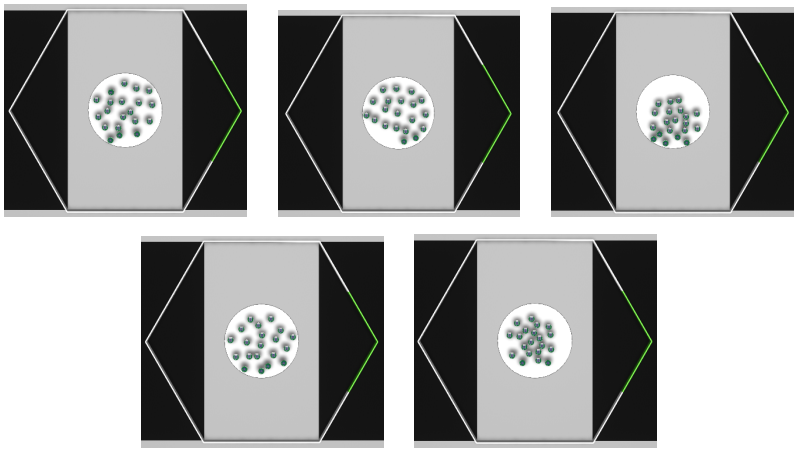


Figure 4.2: Five demonstrations of the end positions the robots have to take for the BAR mission. The expected behavior is the aggregation on the white patch.

to blocks that are not part of the two rightmost walls (containing green). The color of the walls and the neighboring count should give enough information to complete the WELCOME task. The five demonstrations represented on figure 4.3 show that the symmetry allows to have pretty different demonstrations and positions to define one behavior. The prupose of this particular mission is to see if symmetry and demonstrations that are more different than usual are well managed.

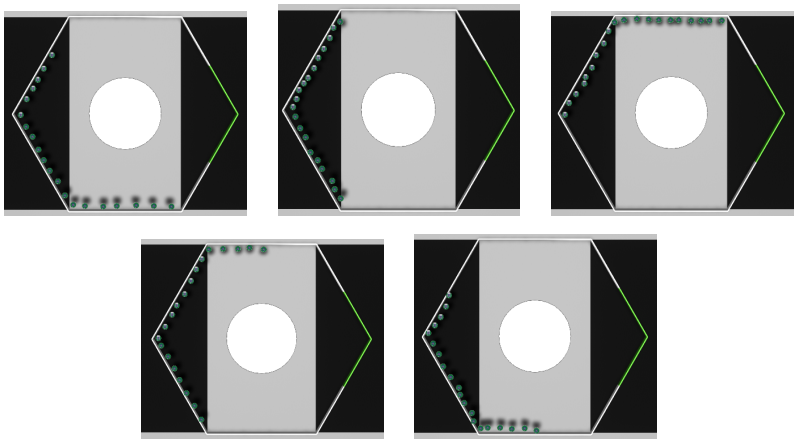


Figure 4.3: Five demonstrations of the end positions the robots have to take for the WELCOME mission. The expected behavior is to line up against the appropriate walls.

- CLEANING (m_{C}): the robots have to clean all the room covering all the space. They are able to do that by walking around all the arena and do coverage. The cleaning is more effective if the inter-robot distance is maximized, the minimum distance between robots should be as high as possible during the mission and more specifically at the end of the time. The five figures 4.4 give indication on the expert demonstrations used to build the implicit objective function.

- DISHES (m_{D}): The robots have to do the dishes in the appropriate side of the kitchen. To achieve this mission the robots are considered doing the dishes when they gather on the left black patch. The difference with the welcome mission is

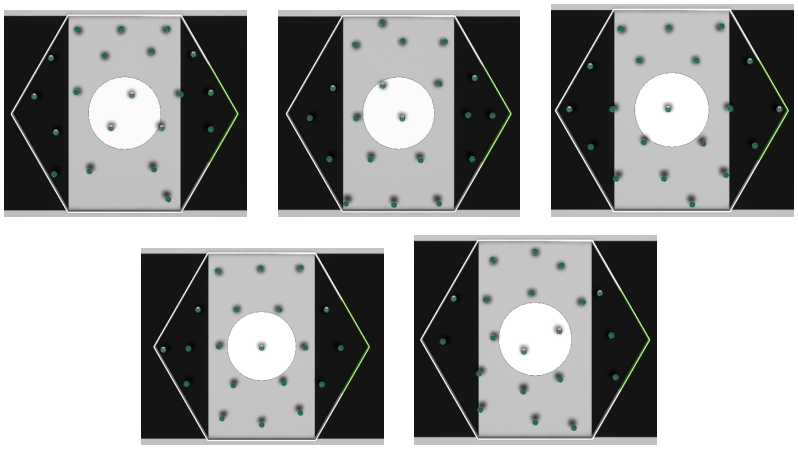


Figure 4.4: Five demonstrations of the end positions the robots have to take for the CLEANING mission. The expected behavior is to cover the whole arena.

that the dishes should be done more effectively when the robots are really close to each other. The epucks should use the color of the walls, of the floor and the indication about neighboring to do this task. The five figures 4.5 show the different demonstrations to describe the dishes.

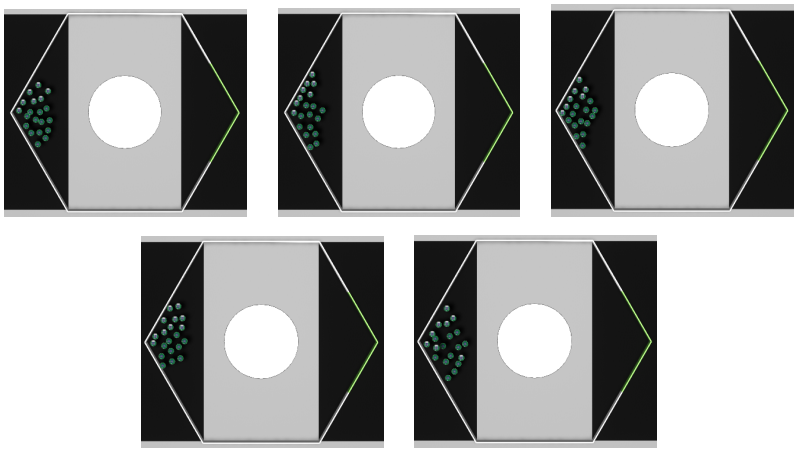


Figure 4.5: Five demonstrations of the end positions the robots have to take for the DISHES mission. The expected behavior is to aggregate on the left black patch.

- MEALS (m_{M}): the robots should prepare the meals in the kitchen, not on the dishes side. The abstraction of this task is gathering on the right black patch in the arena, going to the green lights. Like the dishes, meals are prepared more effectively when the maximum distance between robots is low. The green light is the most important feature the robots have to consider, it is really the element that differentiate the two sides of the kitchen. The five figures 4.6 represent the demonstrations given to construct the expert features for the algorithm.

- SERVING (m_{S}): the robots take care of the clients in the room where the latter are eating and drinking. The eating zone is the gray floor area, robots have to navigate in it and do not go into the black or white patches. The minimum distance between robots should be as high as possible. The robots should use the floor sensor to detect

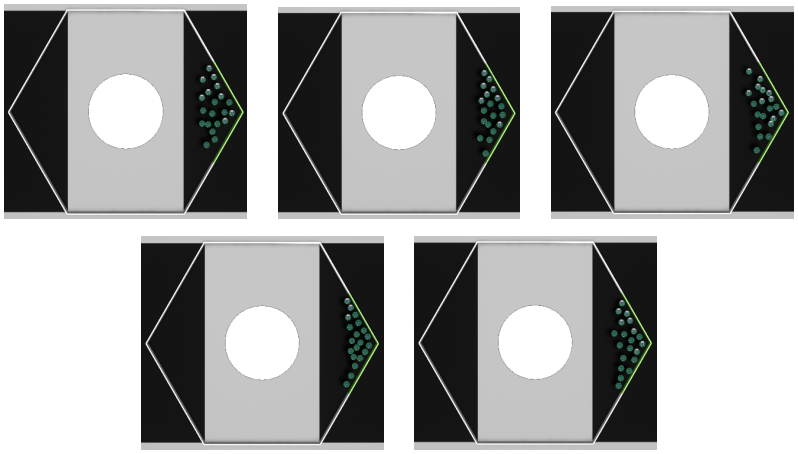


Figure 4.6: Five demonstrations of the end positions the robots have to take for the MEALS mission. The expected behavior is to aggregate on the right black patch, against the green blocks.

if they are on black or not. The five figures 4.7 show the different demonstrations to describe the dishes. In the end, this mission was not used in sequence training, as its optimization did not work with the calculated features. Additional features were proposed to solve this problem, details of which can be found in appendix A.

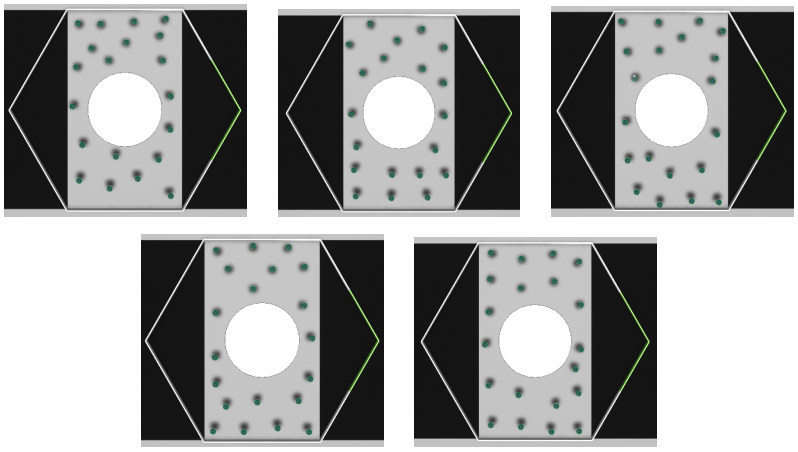


Figure 4.7: Five demonstrations of the end positions the robots have to take for the SERVING mission. The expected behavior is to cover the gray area.

4.1.3 Sequences of Missions

The purpose of `Demo-Fruit` is to solve bi-objective optimization problems, the way of producing bi-criteria missions is by pairing sub-missions with each other. The order of the missions is important because they have to be executed in sequence, one after the other. The set of bi-criteria missions is composed of twelve pairs and noted $M = \{m_{\text{B·W}}, m_{\text{W·B}}, m_{\text{C·D}}, m_{\text{D·C}}, m_{\text{B·M}}, m_{\text{M·B}}, m_{\text{W·M}}, m_{\text{M·W}}, m_{\text{C·W}}, m_{\text{W·C}}, m_{\text{D·M}}, m_{\text{M·D}}\}$. For example, the sequence mission $m_{\text{B·W}}$ stands for the combination of the two missions BAR and WELCOME where the robots have to first prepare the drinks and then, welcome the clients for the evening to begin. The bi-criteria missions are executed within 120s and

each of the sub-mission has a time T to be finished. This time T is equal to 60s. The two missions should be assess and the performance metrics found are returned for each part after the execution of `Demo-Fruit`.

In order to make the mission sequences more realistic and to be able to imagine a context of use, they have been divided into different types of events. When organizers want to offer a reception, missions $\{m_{\text{B·W}}, m_{\text{W·B}}, m_{\text{W·M}}, m_{\text{M·W}}, m_{\text{C·W}}, m_{\text{W·C}}\}$ can help by having evenings divided in two where epucks either welcome and serve drinks or meals to customers (or start by serving and then escorting customers out) or, welcome the clients, let them enjoy the event and then, clean the space. For a restaurant or cafe, it is interesting that the swarm can be used to serve food and then wash dishes or serve drinks. The associated missions are therefore $\{m_{\text{D·M}}, m_{\text{M·D}}, m_{\text{B·M}}, m_{\text{M·B}}\}$. The final category just contains one mission which represents the cleaning of the event room by doing the dishes and clean $\{m_{\text{D·C}}, m_{\text{C·D}}\}$. This one can be used the day after a event that didn't include cleaning in the sub-missions.

Each of the mission is represented in the table 4.1 with a color associated to the first part and another one for the second part. This represents the change of color of the walls that are not green. The robots should identify that the change of color is the trigger to make them change what they are doing.

4.2 Metrics

The problem with this new approach is that no metrics have been defined to measure the performance of a swarm of robots in carrying out a sequence of missions not assessed by any objective function. Part of the work was therefore to find a way of determining whether one PFSM was better than another. The first step is to correctly select the performance measure for the sub-tasks that make up the sequences. Then try to find a way of aggregating these results and use them to assess the quality.

4.2.1 Performance of the Sub-Missions

First of all, PFSM results are evaluated according to the implicit objective function produced by apprenticeship learning [41] in `irace`. In apprenticeship learning, the way in which quality is measured is by measuring the margin, the distance between the hyperplanes t. The problem with this method is that the margin is influenced by the possible outliers. This is represented on the figure 4.8. The second graph represents the classification made by a linear SVM with an outlier, a point that is far from the group of points of the same class. In this case it allows to reduce the margin which should give a better implicit objective function, closer to the expert demonstration but actually the behavior is further in terms of features.

The solution I proposed is to use the distance in the feature space. The hypothesis is that a good behavior is a behavior that produces features close to the expert features. The Euclidian distance is used to compute the similarity instead of the margin.

No.	Mission	Combination
1	$m_{\text{B·W}}$	BAR → WELCOME
2	$m_{\text{W·B}}$	WELCOME → BAR
3	$m_{\text{C·D}}$	CLEANING → DISHES
4	$m_{\text{D·C}}$	DISHES → CLEANING
5	$m_{\text{B·M}}$	BAR → MEALS
6	$m_{\text{M·B}}$	MEALS → BAR
7	$m_{\text{W·M}}$	WELCOME → MEALS
8	$m_{\text{M·W}}$	MEALS → WELCOME
9	$m_{\text{C·W}}$	CLEANING → WELCOME
10	$m_{\text{W·C}}$	WELCOME → CLEANING
11	$m_{\text{D·M}}$	DISHES → MEALS
12	$m_{\text{M·D}}$	MEALS → DISHES

Table 4.1: The set of missions M is composed of five different sub-missions BAR (B), WELCOME (W), CLEANING (C), DISHES (D) and MEALS (M). These sub-missions are paired into sequences of missions $m_{i \cdot j}$. The colors red ( ) and blue ( ) are representing the color of the walls in each part of the mission. The sub-missions are executed as sequences following the order described by the (→). Each part has half of the time of the mission to be finished and is assessed with an independent performance metric. After the execution, the performance metrics are returned.

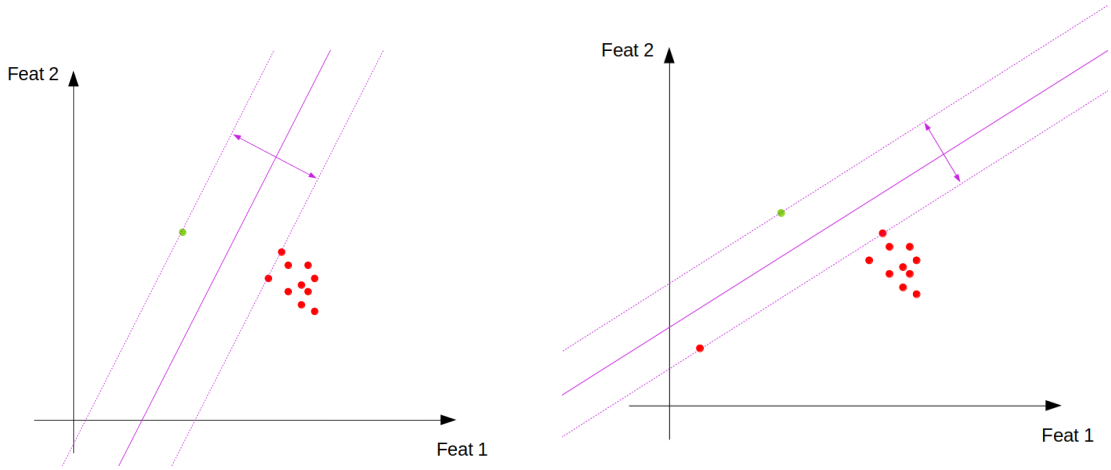


Figure 4.8: Consequence of the presence of an outlier in the demonstration feature vectors. The margin can be decreased but the produced behavior is farther from the expert feature vector. The t-value is then not a perfect measure of quality.

The latest PFSM proposed by apprenticeship learning is not necessarily the one closest to expert demonstrations. In fact, even if the margin is reduced a priori from iteration to iteration , the distance between produced and expert in the feature space may be greater at the end of the execution. Distance is a performance measure used between independent designs, but also to select the best PFSM for each design.

4.2.2 Performance of the Sequences

Like sub-tasks, performance must first be used to select the best PFSM by independent design. Each iteration of each design returns a PFSM optimized by `irace` on the basis of an objective function generated by apprenticeship learning, as well as two distances, one for each part of the mission sequence. In order to compare the different PFSMs, it was therefore chosen to aggregate the two distances to produce a value. Instead of making a choice and adding more bias, several techniques are used and benchmarked. Two score-based techniques : arithmetic mean and L2 norm and the last PFSM produced to have an idea of the accuracy of the last objective function proposed by the algorithm. Salman *et al.* [107] proved the effectiveness of these two simple aggregation methods for feature selection.

The formulas of the aggregation techniques are the following, for the arithmetic mean

$$\frac{d_1 + d_2}{2} \tag{4.1}$$

and for the L2 norm

$$\sqrt{d_1^2 + d_2^2} \tag{4.2}$$

where d_1 and d_2 are the distances between expert and produced for the first and the second part, respectively. The arithmetic mean is very easy to understand but it probably does not penalise sufficiently the PFSMs that are perfect for one of the parts and really bad for the other. This will be analyzed in the next chapter.

In addition to the scores that only consider the end of missions (such as demonstrations), it can be interesting to analyze robot behavior throughout the process. Two kind of visual inspection were therefore also carried out on all the best PFSMs in each of the designs. The additional evaluation gives a score ranging from 0 for random behavior to 4 for a perfect behavior to each part and allows to see if the trigger was considered or not. In some cases the trigger is not visible but present in the PFSM, this could not be assess with the score-based techniques. The second kind is an external evaluation done by a mix of people who know the world of swarm robotics and those who don't. In order to gather their opinions, a survey was sent to them. Designed via the Wooflash website, it features two videos per mission to be evaluated; one for each part of the mission and five designs were randomly selected to represent a pair of sub-missions. After viewing the two videos, the externals are asked to rate the simulation according to its distance from the demonstrations: from very far away to very close. More information about the survey is available in the appendix B and the videos are available here.

4.3 Protocol

In a more practical and concrete way, an evaluation protocol has been set up to compare the two optimization proposals and determine whether `Demo-Cho` can be adapted to mission sequences.

The experimental protocol is as follows

1. The arena of the chosen mission is constructed in the arena builder Orchestra.
2. Five demonstrations are done for each part of the mission selected using the Demonstration builder Orchestra.
3. Ten instances of control software of `Demo-Fruit` with the single-objective optimization are computed. Each independent design is launched for 15 iterations with a budget of 100,000 executions for each in `irace`.
4. Ten instances of control software of `Demo-Fruit` with `Mandarina` optimization are computed. Each independent design is launched for 15 iterations with a budget of 100,000 executions for each in `irace`.
5. For each instance, score-based and visual evaluation are performed.
6. Results are reported as graph performance of the PFSM's selected with the different score-based methods and heatmaps of the weights calculated through the SVM.

Chapter 5

Results

This chapter presents an analysis of the results obtained after applying the protocol described earlier. A qualitative and quantitative analysis is carried out for each of the proposed methods, weighted sum and `Mandarina`, and for each of the mission.

The performance metrics are presented for each sequence proposed and evaluated by the protocol. These metrics include the representation of the distances between each sub-mission and the corresponding demonstrations in feature space and heatmaps illustrating the mean weights attributed to calculated features across all designs within a given sequence. Only the L2 norm metric is retained to select the best PFSMs for further analysis, as it allowed to keep the best and most meaningful PFSMs for both optimization methods. The scores of the other selection techniques are available in appendix C. Furthermore, visual inspection offers a more comprehensive understanding of robot behavior.

Following each individual analysis, a comparison is conducted between the outcomes derived from the weighted sum approach and those generated by Mandarina. This comparison is presented through two graphical representations: the distribution of optimal PFSMs relative to the distances associated with the two sub-missions, and the representation of L2 norm values utilizing notched boxplots.

Lastly, the evaluations conducted by external observers are presented. These assessments exclusively pertain to the outcomes yielded by the weighted sum method, given its better overall performance.

The following section exposes all the results.

5.1 Missions with m_B and m_W

This section includes the two missions composed of the BAR and WELCOME sub-missions, in which the robots have to aggregate together in the center and stick to certain walls, respectively.

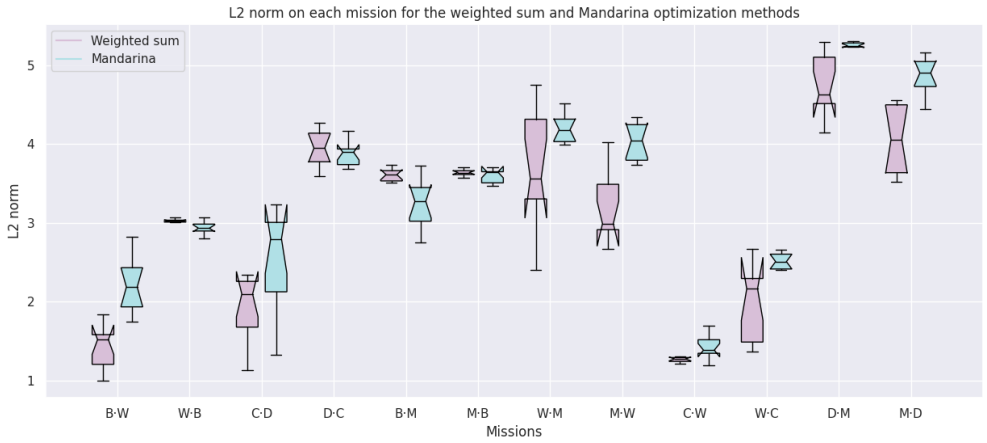
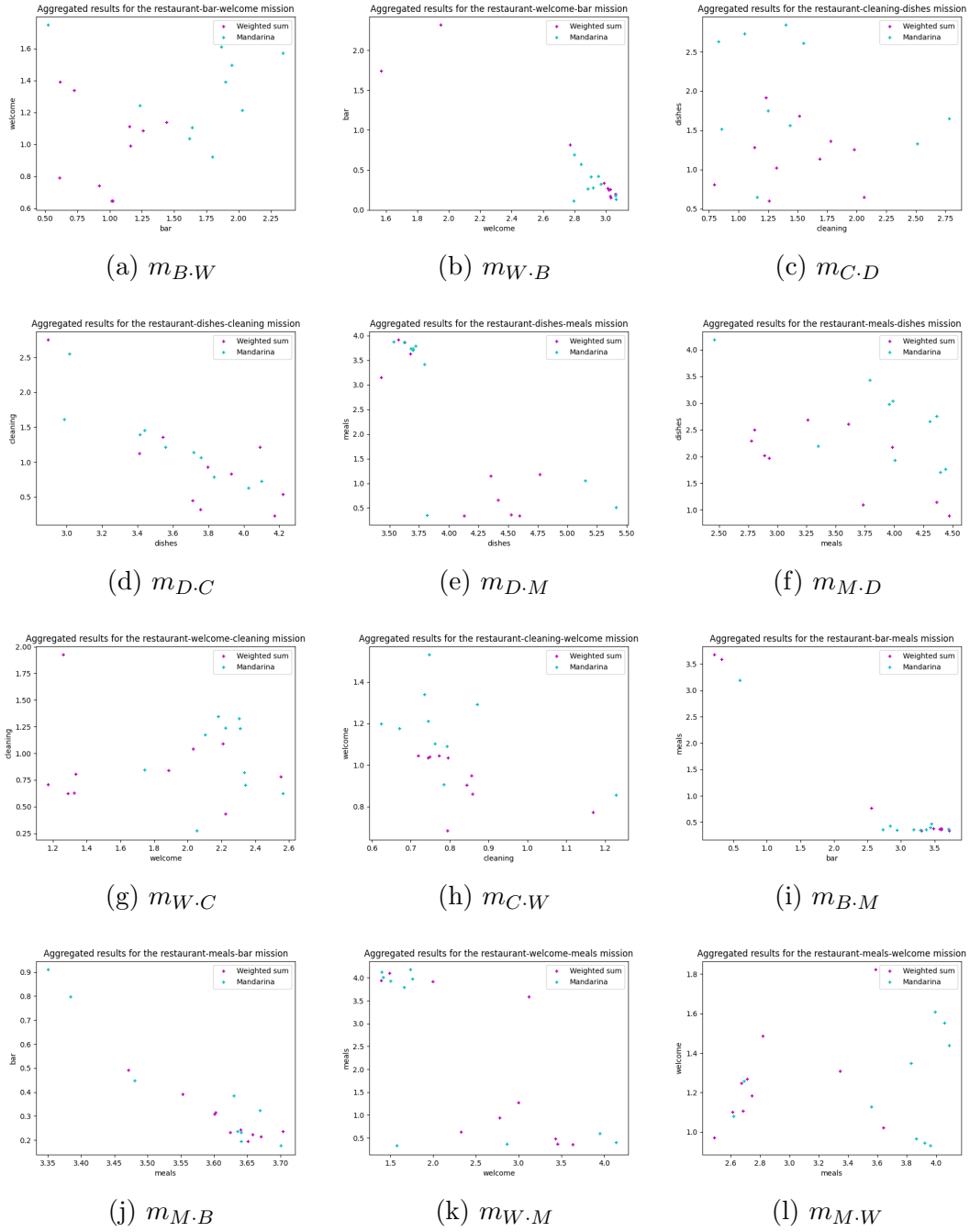


Figure 5.1: Plot showing distances aggregated using L2 norms for each mission and optimization method. The aim is to have the smallest possible value. The results are fairly similar for the different missions, although `Mandarina` tends to perform slightly less well.

BAR-WELCOME

For both optimization methods, the visual examination indicates a proficient comprehension of the demonstrations, as evidenced by the incorporation of triggers in all superior PFSMs of the results. Regarding subtask performance, the latter is consistently prioritized.
For the weighted sum, the figure 5.2a shows that the best PFSMs selected thanks to the L2 norm have small distances to the two sub-missions. When analyzing more closely the PFSM that is the closest to the bottom left corner on the figure 5.2a, the swarm first aggregates in the center because of the lack of blue detected but has a tendency to derive to one of the black patches because no condition is specific to the white area and then go to the blue walls. The result is good when comparing only the frames that represent the final positions of each sub-task but the robots do not seem to understand all of the mission. The visual inspection of the other PFSMs allows to find a different behavior where the robots understand that they have to stay in the center of the arena by avoiding the red color for the first part. Even if the distance is worse, the behavior is better in terms of spacial constraints. Moreover, externals also tend to consider behaviors that respect spatial constraints as better, even if aggregation is not as good, for example.
When using `Mandarina`, the results on the figure 5.2a are slightly worse but still optimize both parts of the mission. This observation is confirmed by visual analysis and small and slightly different values on the graph 5.1.

The four heatmaps displayed on the top of the figure 5.3 confirm that the mission representation BAR mainly involves the separation distance from black patches and the aggregation of robots for both optimization methods.
For the WELCOME part, the accentuated feature is very precise for aggregation by maximizing proximity to the farthest robot, which maximizes them all. However, the distance between individual patches is less clear. The differences between the two methods are very slight, with `Mandarina` tending to move away from the left patch towards the right one, but this is not particularly noticeable when observing the behavior of the swarm.



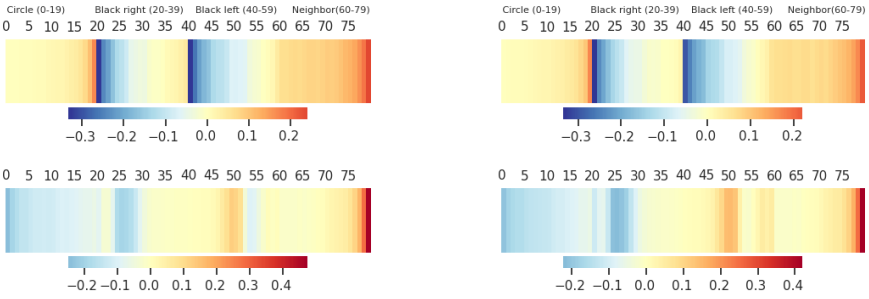
(a) $m_{B \cdot W}$ (b) $m_{W \cdot B}$ (c) $m_{C \cdot D}$

(d) $m_{D \cdot C}$ (e) $m_{D \cdot M}$ (f) $m_{M \cdot D}$

(g) $m_{W \cdot C}$ (h) $m_{C \cdot W}$ (i) $m_{B \cdot M}$

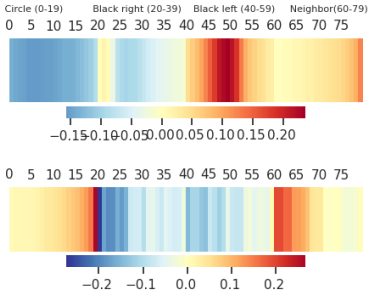
(j) $m_{M \cdot B}$ (k) $m_{W \cdot M}$ (l) $m_{M \cdot W}$

Figure 5.2: Graphs representing the best PFSM for each design of each mission selected with the score-based L2 norm method for weighted sum and `Mandarina`. The closer the points are to the origin, the better the behavior, since it minimizes the distances for each of the sub-missions making up the mission under consideration.

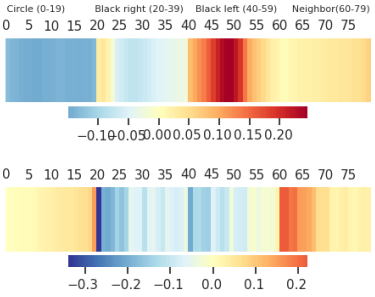


(a) $m_{B.W}$ with weighted sum

(b) $m_{B.W}$ with `Mandarina`



(c) $m_{B.W}$ with weighted sum



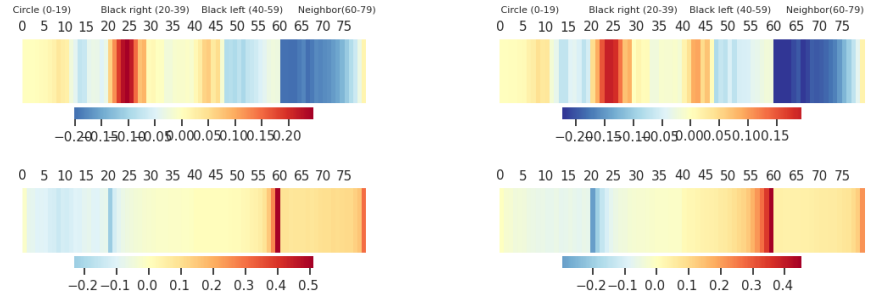
(d) $m_{W.B}$ with `Mandarina`

Figure 5.3: Heatmaps of the average weights for the BAR and WELCOME missions.

WELCOME-BAR

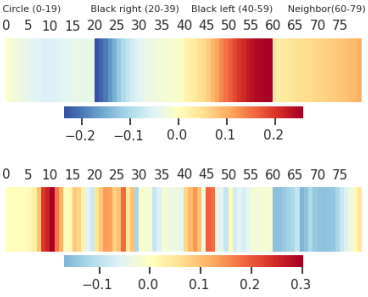
For this mission, the second part is almost the only one optimized. All the methods use to assess the performance have the same conclusion. On the figure 5.2b, almost all the points are in the bottom right corner for the weighted sum and `Mandarina`. Only weighted sum produces two PFSMs that optimize the first part of the mission. `Mandarina` produces only PFSMs that favor the BAR part.
Visual inspection confirms that the robots aggregate directly in the center of the arena without even moving towards the walls for the first 600 steps. For the weighted sum, only one experiment takes the trigger into account. `Mandarina` only produces PFSMs that behave perfectly at the BAR, as if there were no first part of the mission. The external inspection almost always qualifies the simulation as being far removed from the demonstrations, sometimes even very far removed since only one sub-mission is optimized. The sub-missions in this order appear to be contradictory.

The heatmaps of both optimization methods visible on the figure's bottom 5.3 show a certain degree of disparity with regard to execution of the required tasks. In the context of WELCOME mission, disadvantage is indicated for all positions close to the white circle. Conversely, the mission BAR shows a very precise characterization, underlining the importance of the robot positioned furthest from the white circle, which forces it to get as close to it as possible. At the same time, a reduction in the value associated with robots located close to the black patches is observed, encouraging them to stay away from these patches. The distribution of weights in the context of `Mandarina` is characterized by a certain degree of fuzziness with regard to the WELCOME mission, while being particularly

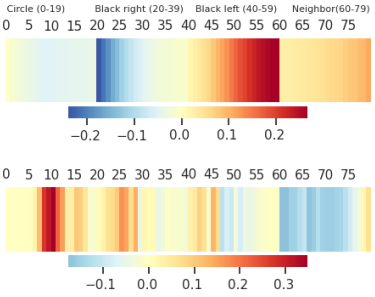


(a) $m_{C \cdot D}$ with weighted sum

(b) $m_{C \cdot D}$ with `Mandarina`



(c) $m_{D \cdot C}$ with weighted sum



(d) $m_{D \cdot C}$ with `Mandarina`

Figure 5.4: Heatmaps of the average weights for the CLEANING and DISHES missions.

accurate with regard to the BAR mission, same conclusion can be made with the weighted sum. This precision suggests a higher level of optimization within the BAR mission, aiming to find exactly the features that matter.

5.2 Missions with m_C and m_D

This section includes the two missions composed of the CLEANING and DISHES sub-missions, in which the robots have to cover the whole arena and aggregate in the left black patch.

CLEANING-DISHES

With the weighted sum, out of the ten independant design initiated, only one yielded a PFSM that neglected the trigger variable, whereas all other designs manifested a shift in behavior in response to change in wall color. Optimum visual results were achieved consistently with relatively close PFSM configurations. In this context, the role of the trigger is more associated with a reactive mechanism activated by the presence of the color blue, a facet that is not applicable in the case of red walls, during the first phase. As a result, the robots start the mission with a random walking behavior, leading to a final position close to that of the mission CLEANING, but without being perfect. When the blue hue appears, the robots engage the corresponding motion module in the direction of the blue walls, while maintaining a distance from the green corner. Figure 5.2c shows that the weighted sum method tends to optimize both parts, unlike `Mandarina` which

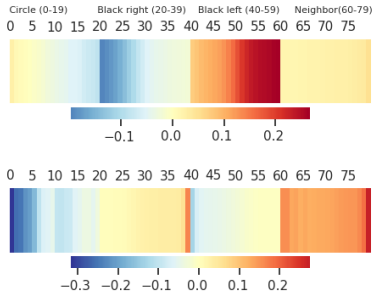
creates PFSMs optimizing only one part. However, visual analysis of `Mandarina` shows similar behavior to that observed with the weighted sum. Opinions of external people on the behaviors produced by the weighted sum are rather mixed, with the results falling more into the two most neutral responses, which denote average swarm quality.

The heatmaps situated on the top of the figure 5.4 shows that the weight allocation for both methods exhibits a notable similarity, with a pronounced focus on inter-robot distances for the CLEANING mission. Other weight values are relatively smaller in magnitude or are allocated to non-strategic features. The amplification of proximities on the right black patch is applied to robots located at a greater distance. For the DISHES mission, a difference can be observed between the two optimization methods: the weighted sum assigns more precise weights, while retaining the same logic as `Mandarina`: minimizing the distance to the left patch and maximizing that to the right patch, while also giving some weight to aggregation.

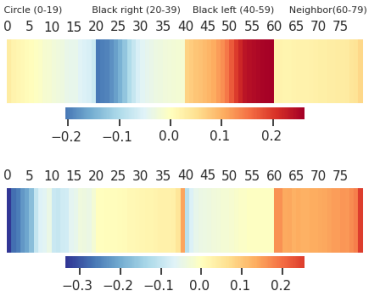
DISHES-CLEANING

When the sequence of subtasks is inverted, both performance and behavior undergo a complete transformation. Robots almost never react to color changes, never with the weighted sum and with only three designs with `Mandarina`. Nevertheless, with the weighted sum, flawless execution of the second sub-mission remains impossible if the first is totally ignored. Instead, a fusion of the mission parts is adopted. The robots behave in a way that facilitates mutual repulsion, while moving away from the color green, towards the dishwasher side. `Mandarina` focuses mainly around a complete optimization of the second part of the mission, displaying a relative disregard for the initial part. As a result, the PFSMs generated tend to neglect the aggregation in the black area on the left to simply adopt a random walk right from the start, aiming to perform the textsccleaning mission. Once again, external opinions are mixed, but remain fairly neutral, suggesting that executing just one of the two sub-missions does not make the simulation very far from the demonstrations for them. As shown in figure 5.2d, the points are located in a space where the distance is large with DISHES and very small with CLEANING. The PFSMs of the two methods are intertwined on this graph and the aggregated performances are really similar, see on figure 5.1.

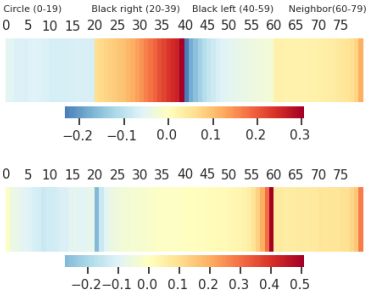
The heatmaps at the bottom of the figure 5.4 show an almost perfect resemblance between the weights assigned to mission DISHES by the weighted sum and `Mandarina`. These weights make sense in promoting distance from the black right patch and proximity to the left patch, but the weights are distributed more diffusely than when dishes were the second part and exhibits a reduced emphasis on the attribute of inter-robot distance. More differences emerge during the analysis conducted on the CLEANING mission. The weight allocation demonstrates a greater degree of randomness, with an objective of maximizing inter-robot distances while concurrently minimizing the distance of robots from the white patch within a range that avoids extreme proximity or distance. Distances to black patches are unclear and of less importance. The similarity exposed in the heatmaps can also be seen in the graph in the figure 5.1, where the results of the two methods for this mission are extremely close.



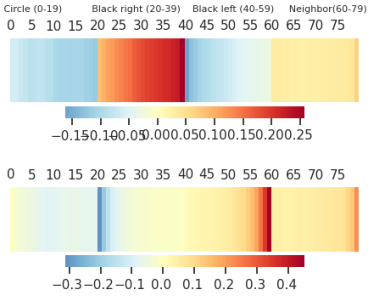
(a) $m_{D \cdot M}$ with weighted sum



(b) $m_{D \cdot M}$ with `Mandarina`



(c) $m_{M \cdot D}$ with weighted sum



(d) $m_{M \cdot D}$ with `Mandarina`

Figure 5.5: Heatmaps of the average weights for the DISHES and MEALS missions.

5.3 Missions with m_D and m_M

This section includes the two missions composed of the DISHES and MEALS sub-missions, in which the robots have to aggregate in the left black patch and aggregate in the right black patch close by the green light.

DISHES-MEALS

Visual inspection clearly shows that the sequence of the two sub-missions poses problems for `Demo-Fruit` with both optimization methods. With the weighted sum, the robots react to the color trigger, but are unable to compose a behavior that fulfills the two given objectives. If the second part is optimized (more than the first), the first part of the simulation will show robots that do not quite go to the black patch on the right, but put themselves in a position that facilitates the transition: all of the robots very close to the red walls or some of them near the right black patch to attract the others when the color changes, for example. In the opposite case, the first part is prioritized, and the result and performance are poorer. The robot's behavior is more closely associated with the WELCOME mission, which has certain points in common with DISHES. With `Mandarina` as the optimization approach, the outcomes demonstrate a prevailing poor visual performance, wherein only four out of the total of ten PFSM exhibit success in enhancing a single mission part, namely the second one. None of the optimization methodologies manage to yield a genuinely satisfactory PFSM, as evidenced by the figure 5.2e wherein the data points are distinctly grouped into two clusters positioned significantly away from the diagonal axis. For this mission, external opinions are clear: the simulations are far,

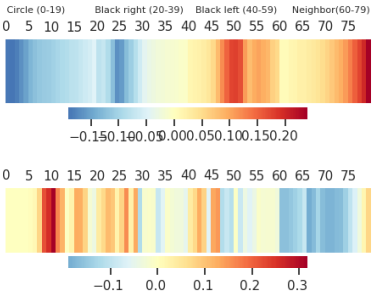
even very far, from the demonstrations given by the experts.

The heatmap pairs at the top of the figure 5.5 justify the behaviour of the robots during the different parts of the mission. For the first part, the importance of robots close to the black patch on the left and the distance from the black patch on the right are evident, while the emphasis on aggregation is relatively discrete. This suggests that robots tend to disperse along the walls in the left zone, facilitating a smoother transition because all the robots are sure to detect the trigger. In the second part, aggregation takes importance without being precise, closely followed by positioning in relation to the black patch on the right and distance from the other patches. The goal of the MEALS part is conveyed most effectively by the assigned weights. The only small difference between the optimization methods is the value of the weights for features 39 and 40, which is marginally greater for the weighted sum approach.

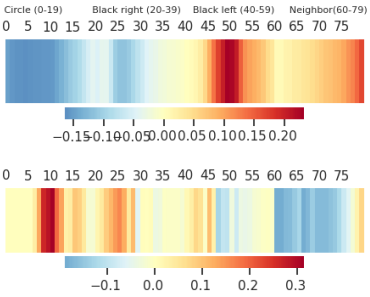
MEALS-DISHES

From a visual point of view, the deductions mirror those previously obtained for the mission, although they involve the same sub-tasks in a reversed sequence. This similarity implies the combination of tasks that appear contradictory and complex when performed sequentially. This alignment with expectations is consistent with the nature of the missions, compounded by the need for the robots to traverse the entire arena. The second part has priority, and the behaviour of the robots during the first phase is clearly influenced by the need to get to the black patch on the left immediately afterwards. One noticeable difference between the alternative orders lies in the distribution pattern of the data points on the graph 5.2f. Rather than appearing as two distinct clusters, the data points are more closely aligned along the diagonal axis and are distributed across all regions within the area characterized by large distances for both sub-missions. This trend aligns with both optimization methods, indicating a reduction in overall performance for the second part, with the first part failing to achieve optimization in both cases, even if visual inspection shows an understanding of the sub-mission. The robots do not go exactly where they're supposed to, but they are always attracted by the green during the first part of the mission. The external opinions do not allow to draw any new conclusions, since they are in line with the observations already made, describing the simulations as rather far from the demonstrations.

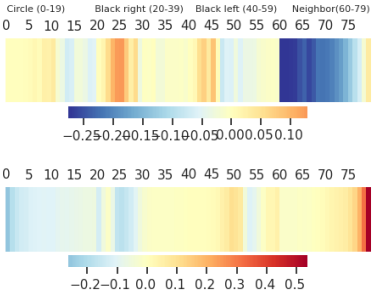
The heatmaps at the bottom of the figure 5.5 are again very similar, showing an understanding of the missions to be carried out. The weights in the MEALS part tend to favor moving away from the left black patch and towards the right black patch, while the aggregation is more accentuated with the weighted sum method. In the context of the DISHES mission, specific features exhibit noteworthy weight allocations that effectively target optimization of mission-relevant facets, namely proximity to the right black patch, distance from the left, and aggregation behavior. Overall, the distribution of weights is slightly wider when `Mandarina` is the optimization method used.



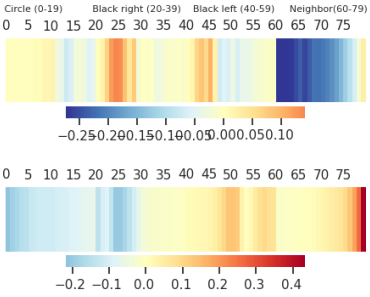
(a) $m_{W \cdot C}$ with weighted sum



(b) $m_{W \cdot C}$ with `Mandarina`



(c) $m_{C \cdot W}$ with weighted sum



(d) $m_{C \cdot W}$ with `Mandarina`

Figure 5.6: Heatmaps of the average weights for the WELCOME and CLEANING missions.

5.4 Missions with m_W and m_C

This section includes the two missions composed of the WELCOME and CLEANING sub-missions, in which the robots have to stick to certain walls and cover the whole arena respectively.

WELCOME-CLEANING

As part of this mission, the robots acquire the ability to execute both parts fairly using a weighted sum optimization approach. The trigger variable is integrated consistently, with neither part systematically taking priority over the other. The most exemplary PFSM in terms of visual representation introduces a behavioral pattern in which robots repel each other near the baseline, but when they enter a black zone, they are attracted by the red walls. Although PFSM doesn't explicitly feature the color blue, its structural design facilitates the transition: when there is no more red, the robots adopt random behavior on the black patches too, occasionally leaving them and repelling each other, they clean up. The majority of simulations are described as close or very close to demonstrations by external survey participants.
On the contrary, the results obtained with `Mandarina` are much poorer, even though they appear to be equivalent in the figure 5.2g. Only two PFSMs give satisfactory results when both missions are carried out, taking the trigger into account. The distinct characteristic among the remaining PFSMs with `Mandarina`, which maintain an acknowledgment of the WELCOME part, is that the robots exhibit a uniform behavior throughout the entirety of the mission entailing a random walking pattern away from the green area, which aligns

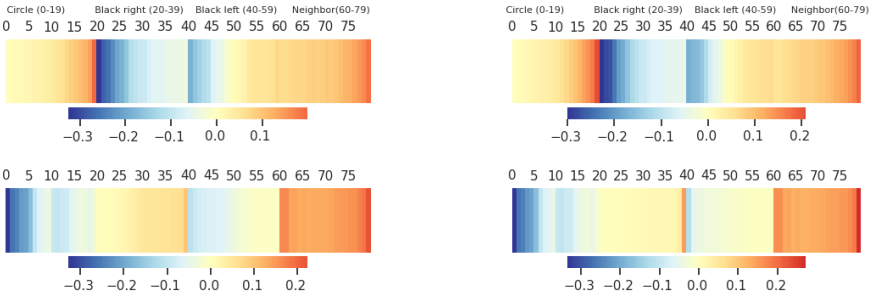
with the WELCOME part. Instead of having two behaviours, `Mandarina` constructs one that gives fine results for the two parts, mixing the goals.

The heatmaps positioned on top of the figure 5.6 facilitate a comparison of weight distributions, providing insights into the reasons behind the comparatively inferior results observed with `Mandarina`. In the context of the WELCOME part, across both methodologies, emphasis is placed on the proximity between the robots and with the left black patch, while concurrently seeking to maintain a certain distance from the central white circle. This weight allocation effectively encapsulates a comprehensive representation of the WELCOME mission. The difference is the feature selection, `Mandarina` sees a much less precise distribution, each peak is surrounded by a wide standard deviation, to describe the behavior many features are maximized or minimized. The heatmaps corresponding to the two optimization methodologies exhibit a notably higher degree of similarity in the context of the CLEANING mission. Here, the primary focus resides in the inter-robot distance.

CLEANING-WELCOME

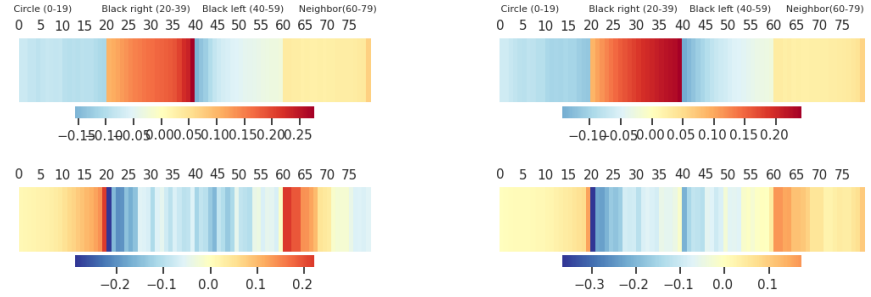
The robots behave very well visually and the trigger is correctly understood and taken into account for both the optimization methods. The second part is always perfect, the robots systematically go towards the walls when turning blue and the first part is almost perfect but often not as good as the second one. With the weighted sum, the two best visual PFSMs have a very simple PFSM that represents exactly the desired behaviour with a repulsion that only changes if blue is detected and transforms into a `GoToColor` towards blue. The same kind of PFSM is observed with `Mandarina` optimization. The only thing that is not perfect for the WELCOME part is that the robots should not go to the blue walls touching the right black patch. External opinions consider these simulations to be close or very close to the demonstrations, confirming the good understanding of the two sub-tasks.
The figure 5.2h accentuates the proximity between data points representing distances associated with the two mission parts for the optimal PFSMs produced by each optimization approach. While it might seem that `Mandarina` performs less effectively, it's important to note that the distance values are exceedingly minimal, comparison can be made with the graph of $m_{W.C}$ 5.2g. Consequently, both methodologies yield highly convergent behaviors with a slightly enhanced comprehension of the WELCOME part in the weighted sum approach.

The heatmaps are much more similar for this order of sub-missions, and can be seen at the bottom of the figure 5.6. Both behaviors are so well represented by heatmaps that they could be guessed just by using these tools. For CLEANING, a maximization of the distance between the robots being closest to ensure optimal coverage and a less interpretable distribution for the distance to patches. This is to be expected, since all robots must occupy the entire arena. WELCOME's heatmaps are very different from those of the opposite order, while retaining the same message, i.e. a certain aggregation of robots away from the white circle. The divergence lies in the allocation of weights, which is executed across a considerably reduced set of features. For instance, the accentuation of aggregation is achieved by maximizing proximity to the furthest robot, relying solely on a singular feature. The distance from the white circle and the right black patch are



(a) $m_{B \cdot M}$ with weighted sum

(b) $m_{B \cdot M}$ with `Mandarina`



(c) $m_{M \cdot B}$ with weighted sum

(d) $m_{M \cdot B}$ with `Mandarina`

Figure 5.7: Heatmaps of the average weights for the BAR and MEALS missions.

also maximized.

5.5 Missions with m_B and m_M

This section includes the two missions composed of the BAR and MEALS sub-missions, in which the robots have to aggregate on the white patch and aggregate on the right black patch to the green walls respectively.

BAR-MEALS

On visual inspection, the different designs all showed similar behaviour, with only one of the two parts being optimized by both the optimization methods. For the weighted sum, the second part, MEALS, is usually perfect, and in two cases it's the BAR. In addition, the trigger is almost never taken into account, which shows that the optimization algorithm is ignoring one of its missions, leading to poor evaluation from the external point of view in most of the designs presented. The best ones are considered close to the demonstrations but never very close.
For `Mandarina`, it is slightly different, since even if the second part is almost always perfect, the first part is also taken into account. In fact, the PFSMs push the robots away from the red walls, closer to the white circle for the first part, and then, after the first 600 steps, simply to the green walls. Even if the first part is not perfect - some robots are already heading for the green walls - it is not totally ignored. Within the figure 5.2i, the data points exhibit a division into two discernible clusters, with approximately com-

parable values. Nonetheless, `Mandarina` succeeds in shifting a subset of points nearer to the graph's origin. This alteration is aligned with the observation that, while maintaining minimal distances to the MEALS demonstrations, the distances from the BAR demonstrations are also diminished, consistent with visual assessments and boxplots position on figure 5.1 where the mean L2 norm of `Mandarina` is slightly smaller.

The heatmaps on the top of the figure 5.7 are very similar, with only the scales differing slightly. The BAR mission is characterized by maximizing the aggregation, the proximity to the white patch and distance from the right black patch, more than the left one. For `Mandarina`, the values of the key features are higher, which may explain the better execution of the first part of the mission. In the context of the MEALS mission, the primary focus resides in the distance from the white circle and the aggregation of robots. Additionally, two features pertaining to proximity with the right black patch and distance from the left black patch are also accentuated, with `Mandarina` displaying the strongest emphasis on these features.

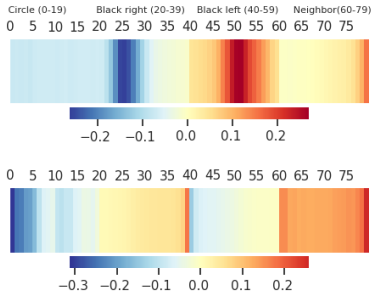
MEALS-BAR

Visual examination of the PFSM produced by the two optimization methodologies reveals that only the second mission part is considered. The robots directly navigate to the center of the arena within the white circle, without engaging in the tasks associated with the MEALS mission. The trigger is always imperceptible and only occasionally detectable in PFSM. However, it has no discernible influence or effect.
On the figure 5.2j, the points form a single cluster without taking the two outliers into account. However it is important to keep in mind the scale of this graph. Indeed, the distances to MEALS are around 3.5, which is enormous and ensures a poor understanding of the mission. On the other hand, distances with BAR are always very small, in line with visual observations. Opinions are more than similar for this mission, with all users considering the simulations far from the demonstrations.

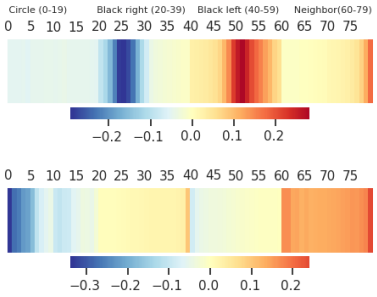
As for the heatmaps displayed in the lower part of figure 5.7, the MEALS part shows uniformity of behavior between the weighted sum and `Mandarina`. The distribution of weights is particularly scattered, but there's a general tendency to favor proximity to the right black patch while ensuring distance with the other patches. Differences can be seen in the BAR part. The weighted sum places great emphasis on minimizing the distance between the white spot and the farthest robot, and maximizing the distance between the right black spot and the nearest robot. The inter-robot distance is less affected by the direct weightings, but rather a consequence of the proximity of the white spot. For `Mandarina`, the same features are highlighted with comparatively lower values. This decrease in values in the patch features is counterbalanced by the importance (even if seemingly low) attached to optimizing inter-robot proximity.

5.6 Missions with m_W and m_M

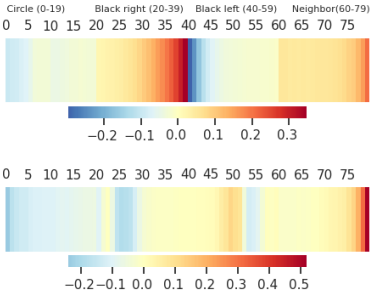
This section includes the two missions composed of the WELCOME and MEALS submissions, in which the robots have to stick to certain walls and aggregate on the right black patch to the green walls respectively.



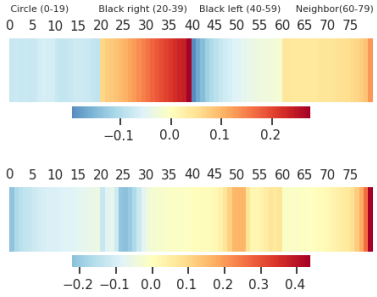
(a) $m_{W \cdot M}$ with weighted sum



(b) $m_{W \cdot M}$ with `Mandarina`



(c) $m_{M \cdot W}$ with weighted sum



(d) $m_{M \cdot W}$ with `Mandarina`

Figure 5.8: Heatmaps of the average weights for the WELCOME and MEALS missions.

WELCOME-MEALS

From a visual point of view and with the help of the illustrative diagram 5.2k, it becomes clear that optimization has been selectively applied to one part, while the other is not enhanced when using a weighted sum approach, resulting in an absence of points along the diagonal axis. Although the trigger variable is duly taken into account, it should be noted that the initial phase of the mission is often left incomplete by robots. The most exemplary PFSM in the visual representation, which skilfully accomplishes the second part and partially responds to the first, presents a distinct pattern of behavior. In this scheme, the robots strategically employ a WELCOME maneuver to approach the designated area on the right, minimizing the distance they must travel once the perimeter walls turn blue. Even so, with the robots showing a tendency to complete the first part, external observations do not all agree that the simulations are far from the demonstrations, rather there is a mix with also opinions saying that the simulations are close.
For `Mandarina`, the result is the same: the two parts of the mission are never both optimized, except for one PFSM. In the scenario involving this particular PFSM, the collective behavior pertaining to the mission is nearly impeccable. The robots exhibit movement towards the red walls, with the color alteration acting as a trigger that facilitates a transition directing them towards the green zone. Although this outcome may not be representative, it nonetheless constitutes an encouraging result.

The upper heatmaps in figure 5.8 support the distribution of points for the mission. One of the main observations is the similarity of these heatmaps between the two optimization methods: weighted sum and `Mandarina`. In the context of the WELCOME part,

the focus is on the maximal distance to the right black patch and the minimal distance to the left black patch, while giving importance to aggregation behavior. For the MEALS part, robots are asked to move away from the left black spot and the white spot towards the right black spot. The features describing the distances between the robots are more or less all maximized, which reduces the impact of this feature. A priori, aggregation is ensured by proximity to the right patch.

MEALS-WELCOME

Even if the points are not perfectly on the diagonal, since the distance is much smaller for the WELCOME part, the distribution is better than with the previous order. This can be seen on the figure 5.2l. The concentration of points with better values and smaller distances is higher for the weighted sum, but `Mandarina`'s best PFSMs are better visually. Rather than executing a straightforward movement towards the blue walls upon their transition, following a period of random walking during the red phase, the robots also exhibit an attraction towards the green walls during the first part. While the MEALS mission is not executed flawlessly, it is visible within the PFSMs. Of the 5 designs, two are considered far and three rather close to the demonstrations. The analysis would probably have been even better with PFSM of Mandarina.

The heatmaps at the bottom of figure 5.8 support the assumption that `Demo-Fruit` has captured the purpose of the sub-mission MEALS, as shown by the allocation of weights accentuating proximity to the right black patch, distance from the left black patch and, to a minor extent, robot aggregation. However, despite the presence of this intention, the prevailing conditions are inadequate to guarantee accurate execution of this specific part by the robots, even if the weighted sum constitutes the most accurate allocation. Similarly, the heatmaps relating to the WELCOME phase give the impression of a clever use of weights, with greater emphasis placed on the aggregation aspect, while comparatively less weight is assigned to the radial distance from the white circle and the right patch. For this part, `Mandarina` accentuates certain features (distance to patches) more than the weighted sum.

Chapter 6

Discussion

The primary objective of this master thesis is to investigate whether it is possible to automatically design the control for a robot swarm that must execute missions expressed as a sequence of sub-missions, in particular, by avoiding the need of explicitly defining an objective function for these sub-missions. To do so, `Demo-Fruit` was introduced and tested in the past sections by applying the protocol set up as part of this master thesis. This section is dedicated to the analysis and interpretation of the results presented in the preceding section, with the aim of elucidating the underlying factors contributing to these outcomes. Additionally, recommendations for enhancing the `Demo-Fruit` system are presented as a concluding part of this discussion.

Before discussing the results presented, it is worth mentioning that the experiments presented here miss a comparison with a reference method, the usual protocol for determining whether a new method offers satisfactory results. `Demo-Fruit` is a method that tackles a design problem for which no previous references exist. The specifications given in the experiments do not have a mathematical formulation, and therefore, it is not possible to compare `Demo-Fruit` against traditional automatic methods that rely on objective functions to drive the optimization process. To do so, we would have to apply these objective functions to the PFSMs generated with `Demo-Fruit`. Gharbi [8] also concluded this when introducing `Demo-Cho`, the method that inspired to build `Demo-Fruit`. A well-established protocol to assess design by demonstration is still missing in the swarm robotics literature.

Within a wider context, the L2 norm-driven notched boxplots representing distance aggregations on figure 5.1 illustrate that the two optimization methodologies (weighted sum and Mandarina) yield results that are comparable, if not similar. It is noteworthy that no existing methodology was available for the comparative evaluation of these optimizers within the sphere of multi-optimization for swarm control software. Hence, the method devised in this master thesis to address this challenge proves applicable and efficacious for both optimization techniques.

Examination of the heatmaps generated for each mission and sub-mission demonstrates the effectiveness of applying learning to mission sequences in this context. The weights allocated are oriented towards features that elucidate the behaviors predicted in the demonstrations. In this context, the choice of sub-missions was aligned with the use of `Demo-Cho` while adhering to its linearity assumption, inherent to the linear support vector machines

(SVMs) used. However, it is not logical to assume that, in all cases, the mission specification will be the expression of a linearly separable problem, to which a linear SVM can be applied. Consequently, future efforts are needed to create a more comprehensive method capable of dealing with nonlinearities in feature space. A single step to extend `Demo-Fruit` in this direction could be, for example, the application of the kernel trick [108]. In addition, increasing the feature set could be a viable approach to this challenge, as shown by the case of the SERVING sub-mission, which required a modification of the features to account for matching, details in A.3.

The results show that the optimization of the second part of the sequences is more efficient than that of the first part, as illustrated by the results shown in the graphs describing the distances between sub-missions and demonstrations. A possible hypothesis for these results is that `irace` aims to minimize the possible loss of performance that occurs during the transition between the first and second sub-missions. It is reasonable to assume that in some missions, the transition between the two may be complex (requiring several behavior modules). In addition, the initial positioning of the robots could complicate the execution of the first part of the mission, with the BAR mission for example, the effort to deviate from the center is probably too great compared to the gain of staying close to the initial positions and thus directly carrying out the second sub-mission. This hypothesis could lead us to think that the optimization process could possibly ignore the first part of the mission and move straight on to the execution of the second part, in order to minimize the loss of performance. However, this conjecture cannot be proven with the protocol presented in this thesis, and further research is needed to investigate this issue.

The analyses further revealed that the MEALS mission is not well-aligned with the functionality of the current version of `Demo-Fruit`. When this particular sub-mission is positioned within the initial part, it tends to be disregarded. Conversely, when placed in the second part, the optimization process tends to favor one of the sub-missions, whether it is the first or second. Two prospective resolutions have been discerned to enhance performance. Primarily, the inherent complexity of the arena itself poses a challenge, given that the robots become confined to an arena section that may not facilitate the detection of color triggers. Secondly, and as mentioned earlier, the existing linear optimization techniques may not be inherently suitable, prompting an avenue of inquiry into methods conducive to solving non-convex multi-objective problems, which could prove particularly insightful. Nevertheless, it's worth noting that this issue persists when employing `Mandarina`, suggesting that linear estimation of the implicit objective function may not be well-suited for this particular sequencing of tasks. In this scenario, the same potential solutions as previously discussed warrant exploration: a reevaluation of feature definitions and the potential utilization of kernel-based techniques.

Finally, a problem already raised in the analysis of `Demo-Cho` is still present: as the implicit objective function does not change much over the iterations if the PFSMs do not reduce the margin, the `Demo-Fruit` strategy does not evolve. This brings a diversification problem, and a means of exploring different solutions could be put in place to ensure that the best implicit function is found.

Chapter 7

Conclusions

In this master thesis, we introduced `Demo-Fruit`, aiming to formulate a novel automatic design of control software tailored for orchestrating mission sequences executed by robot swarms.

The initial phase of this master thesis involved the modification of `Demo-Cho`—a method that serve as inspiration—facilitating its compatibility with `AutoMoDe-TuttiFrutti` modules in lieu of the `AutoMoDe- Chocolate` counterpart. Subsequently, an augmentation was introduced to enable the optimization of mission sequences. In pursuit of this objective, two methodologies for handling multi-criteria designs were explored: the widely employed weighted sum approach, conventionally employed in the realm of multi-criteria design within swarm robotics, and the pioneering automatic design based on `irace`, named `Mandarina`.

To evaluate the effectiveness of the two defined optimization techniques, a comprehensive testing protocol was developed. This protocol was designed as part of this master thesis, starting with the contextualization and precise delineation of the sub-missions. Subsequently, these sub-missions were matched to formulate the ultimate biobjective missions. The test protocol also incorporated an assortment of innovative metrics, carefully formulated to provide a quantitative measure of performance. These metrics were used alongside a detailed visual inspection of the robot's behavior.

The evaluations carried out showed that both optimization methodologies produced probabilistic finite state machines (PFSMs) with similar performance characteristics. A subset of missions gave very encouraging results, even if some seemed ill-suited to the current version of `Demo-Fruit`. Avenues for improvement were suggested to strengthen `Demo-Fruit` and attempt to resolve the problems identified.

In conclusion, new automatic control software design suitable for robot swarms tasked with executing sequences comprising two distinct missions was designed. In addition, an evaluation protocol was developed to provide guidelines for future work on the automatic design of robot swarms by demonstration. This undertaking highlighted the ability of `Demo-Fruit` to produce results with a varying degree of satisfaction through the use of two distinct optimization techniques.

Bibliography

[1] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.

[2] Gerardo Beni and Jing Wang. Swarm Intelligence in Cellular Robotic Systems. In Paolo Dario, Giulio Sandini, and Patrick Aebischer, editors, *Robots and Biological Systems: Towards a New Bionics?*, pages 703–712, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[3] Marco Dorigo, Mauro Birattari, Simon Garnier, Heiko Hamann, Marco Montes de Oca, Christine Solnon, and Thomas Stützle. Swarm intelligence. *Scholarpedia*, 2(9):1462, 2007.

[4] Heiko Hamann. *Swarm robotics: A formal approach*, volume 221. Springer, 2018.

[5] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7:1–41, 2013.

[6] Kenneth Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1, January 2019.

[7] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari. AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112, June 2014.

[8] Ilyes Gharbi. Intuitive mission specification for robot swarm by learning from demonstration: inverse reinforcement learning for robot swarms. 2022.

[9] David Garzón Ramos and Mauro Birattari. Automatic design of collective behaviors for robots that can display and perceive colors. *Applied Sciences*, 10(13), 2020.

[10] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, Daniel Burnier, Alexandre Campo, Anders Lyhne Christensen, Antal Decugniere, Gianni Di Caro, Frederick Ducatelle, Eliseo Ferrante, Alexander Forster, Javier Martinez Gonzales, Jerome Guzzi, Valentin Longchamp, Stephane Magnenat, Nithin Mathews, Marco Montes de Oca, Rehan O'Grady, Carlo Pinciroli, Giovanni Pini, Philippe Retornaz, James Roberts, Valerio Sperati, Timothy Stirling, Alessandro Stranieri, Thomas Stutzle, Vito Trianni, Elio Tuci, Ali Emre

Turgut, and Florian Vaussard. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, 2013.

[11] Josh C. Bongard. Evolutionary robotics. *Communications of the ACM*, 56(8):74–83, August 2013.

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

[13] Dave Cliff, Inman Harvey, and Phil Husbands. Explorations in Evolutionary Robotics. *Adaptive Behavior*, 2:73–110, June 1993.

[14] Seyed Mohammad Jafar Jalali, Sajad Ahmadian, Abbas Khosravi, Seyedali Mirjalili, Mohammad Reza Mahmoudi, and Saeid Nahavandi. Neuroevolution-based autonomous robot navigation: A comparative study. *Cognitive Systems Research*, 62:35–43, 2020.

[15] Vito Trianni. *Evolutionary Swarm Robotics*, volume 108 of *Studies in Computational Intelligence*. Springer, Berlin, Heidelberg, 2008.

[16] Vito Trianni, Elio Tuci, Christos Ampatzis, and Marco Dorigo. Evolutionary swarm robotics: A theoretical and methodological itinerary from individual neurocontrollers to collective behaviours. *The horizons of evolutionary robotics*, 153, 2014.

[17] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems*, 20(2):205–224, 1997.

[18] Stefano Nolfi and Dario Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[19] Ken Hasselmann, Antoine Ligot, Julian Ruddick, and Mauro Birattari. Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms. *Nature Communications*, 12(1):4345, July 2021. Number: 1 Publisher: Nature Publishing Group.

[20] Stuart Geman, Elie Bienenstock, and René Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4:1–58, January 1992.

[21] Dario Floreano, Phil Husbands, and Stefano Nolfi. Evolutionary robotics. *Handbook of robotics*, 2008.

[22] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R.C. Carrasco. Probabilistic finite-state machines - part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025, July 2005.

[23] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Lorenzo Garattoni, Roman Miletitch, Gaëtan Podevijn, Andreagiovanni Reina, Touraj Soleymani, Mattia Salvaro, Carlo Pinciroli, Franco Mascia, Vito Trianni, and Mauro Birattari. AutoMoDe-Chocolate: automatic design of control software for robot swarms. *Swarm Intelligence*, 9(2-3):125–152, September 2015.

[24] Lorenzo Garattoni, Gianpiero Francesca, Arne Brutschy, Carlo Pinciroli, and Mauro Birattari. *Software infrastructure for e-puck (and TAM).* IRIDIA, Institut de Recherches Interdisciplinaires et de Développements en . . . , 2016.

[25] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

[26] Mauro Birattari. On the estimation of the expected performance of a metaheuristic on a class of instances. how many instances, how many runs? Technical report, IRIDIA, Université Libre de Bruxelles, Belgium, 2004.

[27] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Lorenzo Garattoni, Roman Miletitch, Gaëtan Podevijn, Andreagiovanni Reina, Touraj Soleymani, Mattia Salvaro, Carlo Pinciroli, et al. An experiment in automatic design of robot swarms: Automode-vanilla, evostick, and human experts. In *Swarm Intelligence: 9th International Conference, ANTS 2014, Brussels, Belgium, September 10-12, 2014. Proceedings 9*, pages 25–37. Springer, 2014.

[28] W.J. Conover. *Practical Nonparametric Statistics [By] W.J. Conover*. Wiley, 1971.

[29] Ken Hasselmann, Frédéric Robert, and Mauro Birattari. Automatic Design of Communication-Based Behaviors for Robot Swarms. In Marco Dorigo, Mauro Birattari, Christian Blum, Anders L. Christensen, Andreagiovanni Reina, and Vito Trianni, editors, *Swarm Intelligence*, pages 16–29, Cham, 2018. Springer International Publishing.

[30] Jonas Kuckling, Antoine Ligot, Darko Bozhinoski, and Mauro Birattari. Behavior Trees as a Control Architecture in the Automatic Modular Design of Robot Swarms. In Marco Dorigo, Mauro Birattari, Christian Blum, Anders L. Christensen, Andreagiovanni Reina, and Vito Trianni, editors, *Swarm Intelligence*, pages 30–43, Cham, 2018. Springer International Publishing.

[31] Alejandro Marzinotto, Michele Colledanchise, Christian Smith, and Petter Ögren. Towards a unified behavior trees framework for robot control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5420–5427, 2014.

[32] Muhammad Salman, Antoine Ligot, and Mauro Birattari. Concurrent design of control software and configuration of hardware for robot swarms under economic constraints. *PeerJ Computer Science*, 5:e221, September 2019.

[33] Jonas Kuckling, Keneth Ubeda Arriaza, and Mauro Birattari. Simulated annealing as an optimization algorithm in the automatic modular design of control software for robot swarms. *BNAIC 2019: Artificial Intelligence*, 2019.

[34] Alberto Franzin and Thomas Stützle. Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research*, 104:191–206, 2019.

[35] Gaëtan Spaey, Miquel Kegeleirs, David Garzón Ramos, and Mauro Birattari. Evaluation of alternative exploration schemes in the automatic modular design of robot swarms. In *Artificial Intelligence and Machine Learning: 31st Benelux AI Conference, BNAIC 2019, and 28th Belgian-Dutch Machine Learning Conference, BENELEARN 2019, Brussels, Belgium, November 6-8, 2019, Revised Selected Papers 28*, pages 18–33. Springer, 2020.

[36] Cristina Dimidov, Giuseppe Oriolo, and Vito Trianni. Random Walks in Swarm Robotics: An Experiment with Kilobots. In Marco Dorigo, Mauro Birattari, Xiaodong Li, Manuel López-Ibáñez, Kazuhiro Ohkura, Carlo Pinciroli, and Thomas Stützle, editors, *Swarm Intelligence*, pages 185–196, Cham, 2016. Springer International Publishing.

[37] Miquel Kegeleirs, David Garzón Ramos, and Mauro Birattari. Random Walk Exploration for Swarm Mapping. In Kaspar Althoefer, Jelizaveta Konstantinova, and Ketao Zhang, editors, *Towards Autonomous Robotic Systems*, pages 211–222, Cham, 2019. Springer International Publishing.

[38] Jonas Kuckling, Vincent Van Pelt, and Mauro Birattari. Automatic modular design of behavior trees for robot swarms with communication capabilites. In *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24*, pages 130–145. Springer, 2021.

[39] Antoine Ligot, Ken Hasselmann, and Mauro Birattari. Automode-arlequin: Neural networks as behavioral modules for the automatic design of probabilistic finite-state machines. In Marco Dorigo, Thomas Stützle, Maria J. Blesa, Christian Blum, Heiko Hamann, Mary Katherine Heinrich, and Volker Strobel, editors, *Swarm Intelligence*, pages 271–281, Cham, 2020. Springer International Publishing.

[40] Fernando J. Mendiburu, David Garzón Ramos, Marcos R. A. Morais, Antonio M. N. Lima, and Mauro Birattari. AutoMoDe-Mate: Automatic off-line design of spatially-organizing behaviors for robot swarms. *Swarm and Evolutionary Computation*, 74:101118, 2022.

[41] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[42] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[43] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[44] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Automated Configuration of Mixed Integer Programming Solvers. In Andrea Lodi, Michela Milano, and Paolo Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 186–202, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[45] Frank Hutter, Manuel López-Ibáñez, Chris Fawcett, Marius Lindauer, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. AClib: A Benchmark Library for Algorithm Configuration. In Panos M. Pardalos, Mauricio G.C. Resende, Chrysafis Vogiatzis, and Jose L. Walteros, editors, *Learning and Intelligent Optimization*, pages 36–40, Cham, 2014. Springer International Publishing.

[46] Manuel López-Ibáñez and Thomas Stützle. Automatically improving the anytime behaviour of optimisation algorithms. *European Journal of Operational Research*, 235(3):569–582, 2014.

[47] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013.

[48] T Stutzle. Lecture notes in Heuristic Optimization, Automatic Algorithm Configuration, 2011.

[49] Belarmino Adenso-Diaz and Manuel Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations research*, 54(1):99–114, 2006.

[50] Steven P Coy, Bruce L Golden, George C Runger, and Edward A Wasil. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7:77–97, 2001.

[51] Mauro Birattari, Thomas Stützle, Luis Paquete, Klaus Varrentrapp, et al. A racing algorithm for configuring metaheuristics. In *Gecco*, volume 2. Citeseer, 2002.

[52] Thomas Bartz-Beielstein and Mike Preuss. Experimental research in evolutionary computation. In *Proceedings of the 9th annual conference companion on genetic and evolutionary computation*, pages 3001–3020, 2007.

[53] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pages 507–523. Springer, 2011.

[54] Mauro Birattari and Janusz Kacprzyk. *Tuning metaheuristics: a machine learning perspective*, volume 197. Springer, 2009.

[55] Roberto Battiti, Mauro Brunato, and Franco Mascia. *Reactive search and intelligent optimization*, volume 45. Springer Science & Business Media, 2008.

[56] Giorgos Karafotias, Mark Hoogendoorn, and Ágoston E Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2014.

[57] João M. P. Cardoso, José Gabriel F. Coutinho, and Pedro C. Diniz. Chapter 8 - Additional topics. In João M. P. Cardoso, José Gabriel F. Coutinho, and Pedro C. Diniz, editors, *Embedded Computing for High Performance*, pages 255–280. Morgan Kaufmann, Boston, 2017.

[58] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26:369–395, 2004.

[59] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. F-race and iterated f-race: An overview. *Experimental methods for the analysis of optimization algorithms*, pages 311–336, 2010.

[60] Oden Maron and Andrew W Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11:193–225, 1997.

[61] Oded Maron and Andrew Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. *Advances in neural information processing systems*, 6, 1993.

[62] D Anderson and K Burnham. Model selection and multi-model inference. *Second. NY: Springer-Verlag*, 63(2020):10, 2004.

[63] William Jay Conover. *Practical nonparametric statistics*, volume 350. john wiley & sons, 1999.

[64] Angela Dean and Daniel Voss. *Design and analysis of experiments*. Springer, 1999.

[65] Athanasios Papoulis and S Unnikrishna Pillai. *Probability, random variables and stochastic processes*. 2002.

[66] David J Sheskin. *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2020.

[67] Siegel Sidney. Nonparametric statistics for the behavioral sciences. *The Journal of Nervous and Mental Disease*, 125(3):497, 1957.

[68] Oscar Kempthorne. *The design and analysis of experiments*, volume 73. LWW, 1952.

[69] Prasanna Balaprakash, Mauro Birattari, and Thomas Stützle. Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. In *Hybrid Metaheuristics: 4th International Workshop, HM 2007, Dortmund, Germany, October 8-9, 2007. Proceedings 4*, pages 108–122. Springer, 2007.

[70] Mark Zlochin, Mauro Birattari, Nicolas Meuleau, and Marco Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131:373–395, 2004.

[71] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[72] Manuel López-Ibáñez, Leslie Pérez Cáceres, Jérémie Dubois-Lacoste, Thomas G Stützle, and Mauro Birattari. *The irace package: User guide*. IRIDIA, Institut de Recherches Interdisciplinaires et de Développements en ..., 2016.

[73] Henry Hsu and Peter A Lachenbruch. Paired t test. *Wiley StatsRef: statistics reference online*, 2014.

[74] Leslie Pérez Cáceres, Manuel López-Ibáñez, and Thomas Stützle. An analysis of parameters of irace. In *Evolutionary Computation in Combinatorial Optimisation: 14th European Conference, EvoCOP 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers 14*, pages 37–48. Springer, 2014.

[75] Leonardo CT Bezerra, Manuel López-Ibáñez, and Thomas Stützle. *Automatic configuration of multi-objective optimizers and multi-objective configuration*. Springer, 2020.

[76] Joshua D Knowles, Lothar Thiele, and Eckart Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. *TIK-report*, 214, 2006.

[77] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117–132, 2003.

[78] Johann Dréo. Using performance fronts for parameter setting of stochastic metaheuristics. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2197–2200, 2009.

[79] Simon Wessing, Nicola Beume, Günter Rudolph, and Boris Naujoks. Parameter tuning boosts performance of variation operators in multiobjective optimization. In *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I 11*, pages 728–737. Springer, 2010.

[80] Manuel López-Ibánez and Thomas Stutzle. The automatic design of multiobjective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6):861–875, 2012.

[81] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[82] Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. Automatic configuration of state-of-the-art multi-objective optimizers using the tp+ pls framework. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 2019–2026, 2011.

[83] Leonardo CT Bezerra, Manuel López-Ibánez, and Thomas Stützle. Automatic component-wise design of multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(3):403–417, 2015.

[84] Vito Trianni and Manuel López-Ibáñez. Advantages of task-specific multi-objective optimisation in evolutionary robotics. *PloS one*, 10(8):e0136406, 2015.

[85] Julia Handl and Joshua Knowles. Modes of problem solving with multiple objectives: Implications for interpreting the pareto set and for decision making. In *Multiobjective Problem Solving from Nature: From Concepts to Applications*, pages 131–151. Springer, 2008.

[86] Miguel Duarte, Vasco Costa, Jorge Gomes, Tiago Rodrigues, Fernando Silva, Sancho Moura Oliveira, and Anders Lyhne Christensen. Evolution of collective behaviors for a real swarm of aquatic surface robots. *PloS one*, 11(3):e0151834, 2016.

[87] Christos Ampatzis, Elio Tuci, Vito Trianni, Anders Lyhne Christensen, and Marco Dorigo. Evolving self-assembly in autonomous homogeneous robots: Experiments with two physical robots. *Artificial Life*, 15(4):465–484, 2009.

[88] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[89] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[90] Marc Kirschner and John Gerhart. Evolvability. *Proceedings of the National Academy of Sciences*, 95(15):8420–8427, 1998.

[91] Joel Lehman, Kenneth O Stanley, et al. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

[92] Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218, 2011.

[93] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

[94] Shervin Nouyan, Roderich Groß, Michael Bonani, Francesco Mondada, and Marco Dorigo. Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711, 2009.

[95] Lorenzo Garattoni and Mauro Birattari. Autonomous task sequencing in a robot swarm. *Science Robotics*, 3(20):eaat0430, 2018.

[96] Arne Brutschy, Lorenzo Garattoni, Manuele Brambilla, Gianpiero Francesca, Giovanni Pini, Marco Dorigo, and Mauro Birattari. The tam: abstracting complex tasks in swarm robotics research. *Swarm Intelligence*, 9:1–22, 2015.

[97] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life: Third European Conference on Artificial Life Granada, Spain, June 4–6, 1995 Proceedings 3*, pages 704–720. Springer, 1995.

[98] Nils J Nilsson et al. Shakey the robot. 1984.

[99] Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991.

[100] Valerio Sperati, Vito Trianni, and Stefano Nolfi. Self-organised path formation in a swarm of robots. *Swarm Intelligence*, 5:97–119, 2011.

[101] Ken Hasselmann, Antoine Ligot, Gianpiero Francesca, and M Birattari. Reference models for automode. *IRIDIA, Université libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2018-002*, 2018.

[102] Antoine Ligot, Ken Hasselmann, Brian Delhaisse, Lorenzo Garattoni, Gianpiero Francesca, and Mauro Birattari. *AutoMoDe, NEAT, and EvoStick: implementations for the e-puck robot in ARGoS3*. IRIDIA, Institut de Recherches Interdisciplinaires et de Développements en . . . , 2018.

[103] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[104] Johann Borenstein and Yoram Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[105] Shan Suthaharan and Shan Suthaharan. Support vector machine. *Machine learning models and algorithms for big data classification: thinking with examples for effective learning*, pages 207–235, 2016.

[106] David Garzon Ramos, F Pagnozzi, T Stützle, and M Birattari. Iterated f-race for the automatic design of robot swarms with respect to multiple criteria. 2023.

[107] Reem Salman, Ayman Alzaatreh, and Hana Sulieman. The stability of different aggregation techniques in ensemble feature selection. *Journal of Big Data*, 9(1):1–23, 2022.

[108] MN Murty, Rashmi Raghava, MN Murty, and Rashmi Raghava. Kernel-based svm. *Support vector machines and perceptrons: learning, optimization, classification, and application to social networks*, pages 57–67, 2016.

Appendix A

Additional Features

When testing the feasibility of the missions used as sub-tasks, one of the missions was problematic and the algorithm could not optimize it properly. This mission is the SERVING where the robots have to cover the gray part of the arena. The given demonstrations can be seen on the figure A.1.

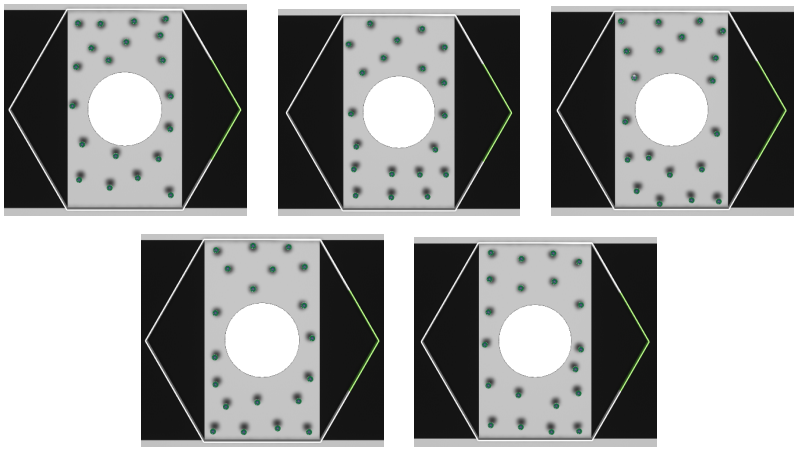


Figure A.1: Five demonstrations of the end positions the robots have to take for the SERVING mission. The expected behavior is to cover the gray area.

The problem is that the robots don't stay in the gray area, so there is almost no difference

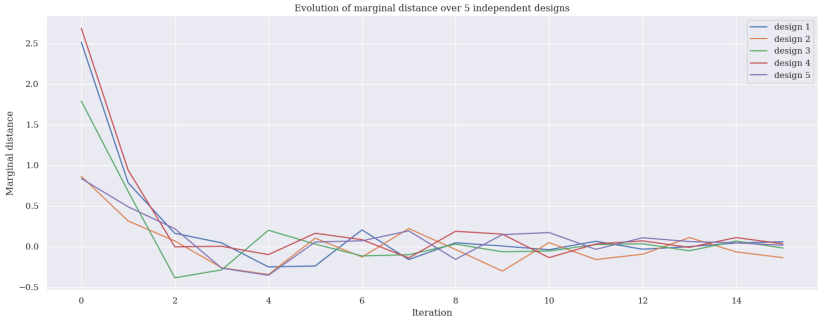


Figure A.2: Margin evolution over iterations for mission SERVING optimization. The margin value converges towards 0.

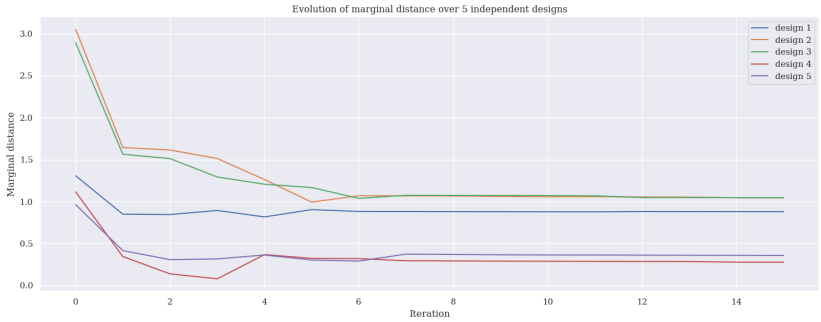


Figure A.3: Margin evolution over iterations for mission SERVING optimization with the new features. The margin value decreases from the first iteration but does not converge to one value.

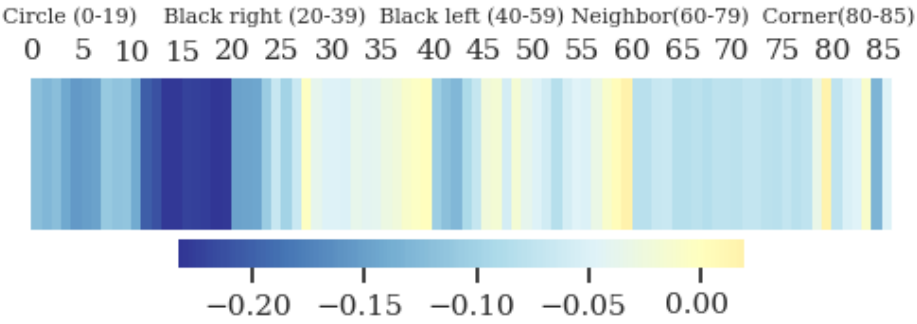


Figure A.4: Heatmap of the average weight values of the last iteration over the 5 designs.

with repulsion over the whole arena. However, the graph of the margin value A.2 did not identify the problem, only visual inspection. This is further proof that metrics are lacking to assess `Demo-Fruit`'s performance. The idea to solve the problem is to introduce a new feature describing the distance of the swarm to each of the corner of the arena. The new feature is calculated as follows:

$$\mu_{corner_i} = e^{-\frac{2ln(10)}{d} x_{corner_i}}, \tag{A.1}$$

where $i = \{0, 1, 2, 3, 4, 5\}$ is the number of corners of the arena, six corners in this case and x_{corner_i} is the distance between the closest robots of the swarm to the $corner_i$.

It is harder for the margin value to converge to a single clear value A.3, but it's still decreasing, and an improvement is visible for the area in which the robots have to navigate. The robots do not go into the black areas even if they do not really do repulsion in the gray area and the worst results are generated by the PFSM associated with the lowest values of t. The heatmap available in the figure A.4 shows that weight is given to two of the new features, the two corners from which the robots must move away as far as possible.

Appendix B

Survey for Visual Inspection

As there is no general method for assessing the performance of a swarm that performs a sequence of sub-missions learned from demonstrations and with no objective function available, several methods have been proposed, including visual inspection. But by keeping only my visual inspection, I introduce a bias since I know exactly what I wanted the robots to do, and I risk looking for any sign that might point in that direction. One way of countering this problem is to ask people outside the research field to rate the performance of the swarms.

The way the questions were asked had to be thought through and adapted to make the survey interesting without requiring too much time or effort. First of all, I decided to propose five designs for each mission, rather than the ten created for the test protocol. Secondly, even though the mission videos have been speeded up, seeing both parts in one go was complicated by the need to cram a lot of information into one go. The videos were therefore split in two, each representing half a sequence. For the evaluation itself, the aim was not to have the possibility of voting for a "neutral" score, not too bad and not really good, so four levels of quality are proposed to characterize the similarity with the demonstrations: very far, far, close and very close. Finally, I chose to evaluate only the mission as a whole, because if an evaluation were requested for each of the sub-missions and for the whole, the last measure would probably simply be an average of the first two. And evaluating each of the sub-missions is less interesting than evaluating the mission as a whole, since the distances already allow us to qualify the sub-missions individually.

In order to carry out this survey, the Wooflash platform was chosen, as it allowed me to create the type of question I needed, and above all to add images and videos to the questions. A screenshot of the question type can be seen on figures B.1

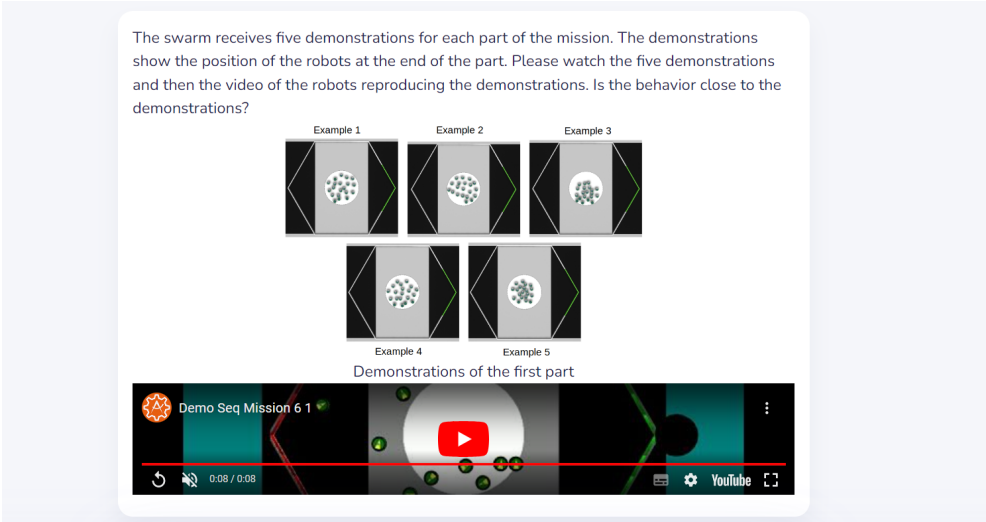


Figure B.1: Screenshot of the first part of the questions asked to external people to evaluate the performance of the swarm performing a mission sequence. The demonstrations provided to the algorithm to describe the behavior can be seen just above the video showing the final behavior of the robots. Here it is the sub-mission BAR that is shown.

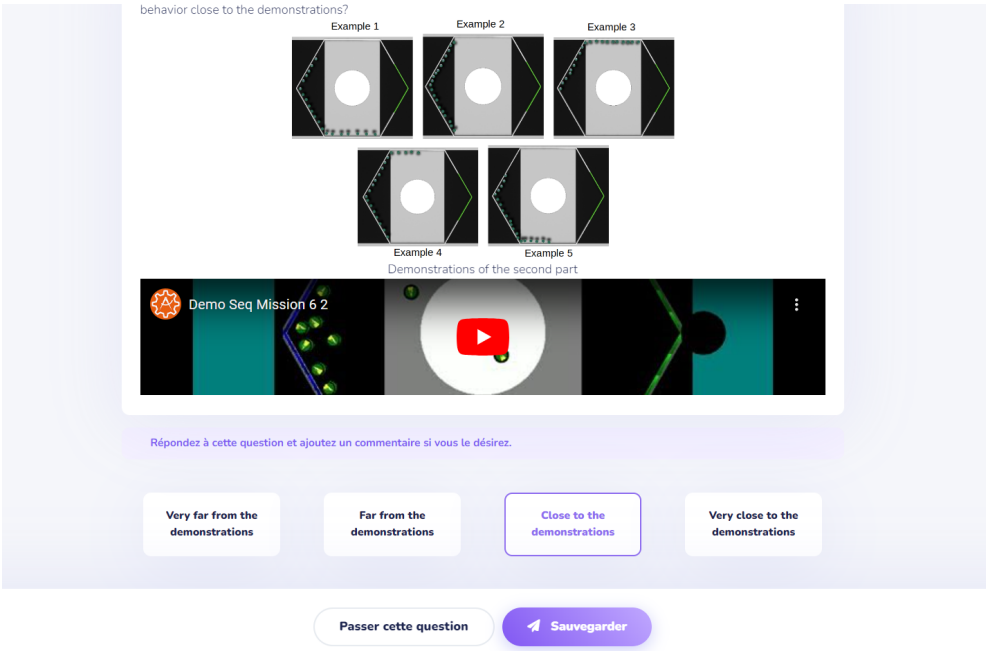
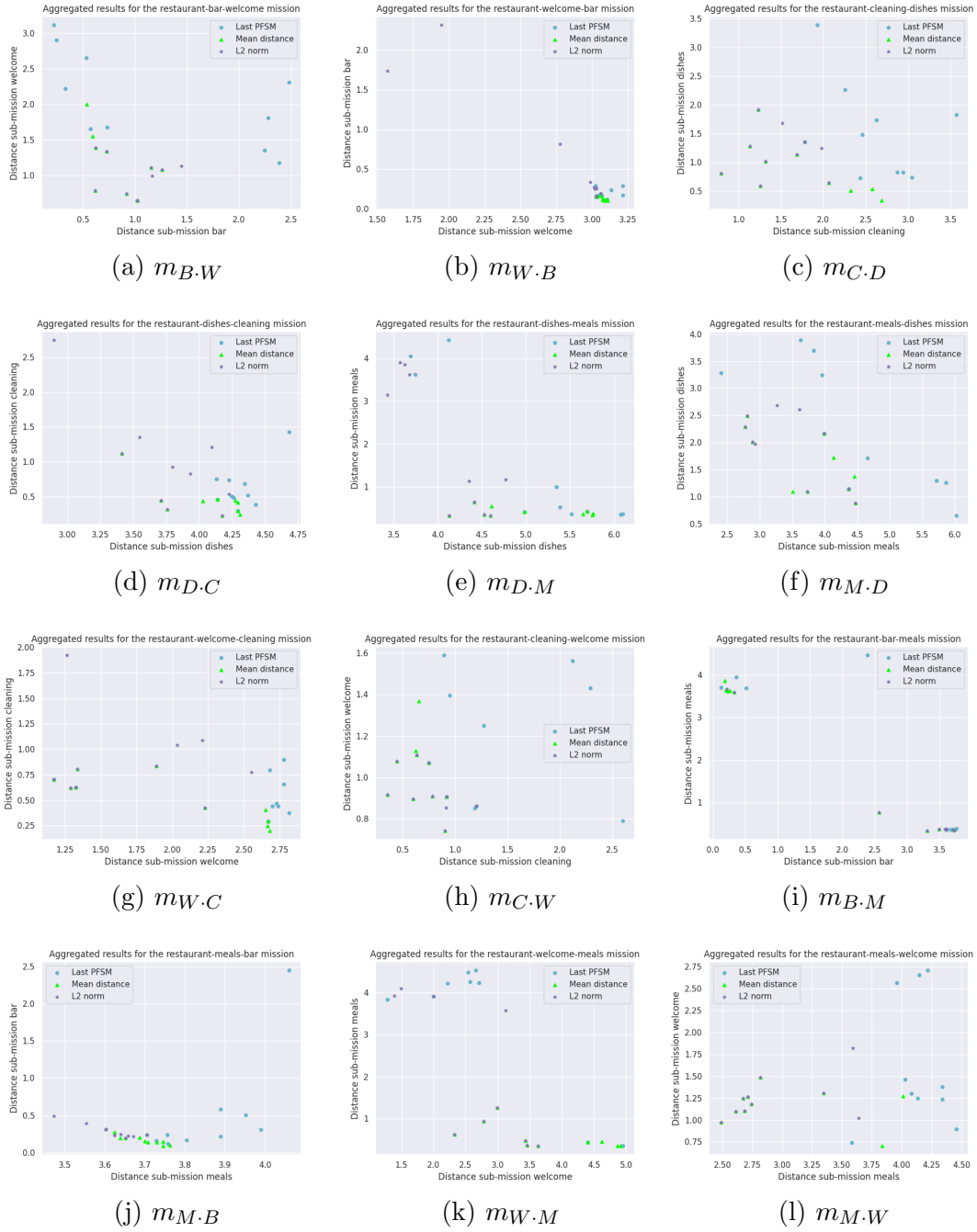


Figure B.2: Screenshot of the second part of the questions asked to external people to evaluate the performance of the swarm performing a mission sequence. The demonstrations provided to the algorithm to describe the behavior can be seen just above the video showing the final behavior of the robots. Here it is the sub-mission WELCOME that is shown.

Appendix C

Additional Graphs

Below are graphs showing the different ways of selecting the best PFSM generated by designing a mission solved with the weighted sum and then with `Mandarina`.



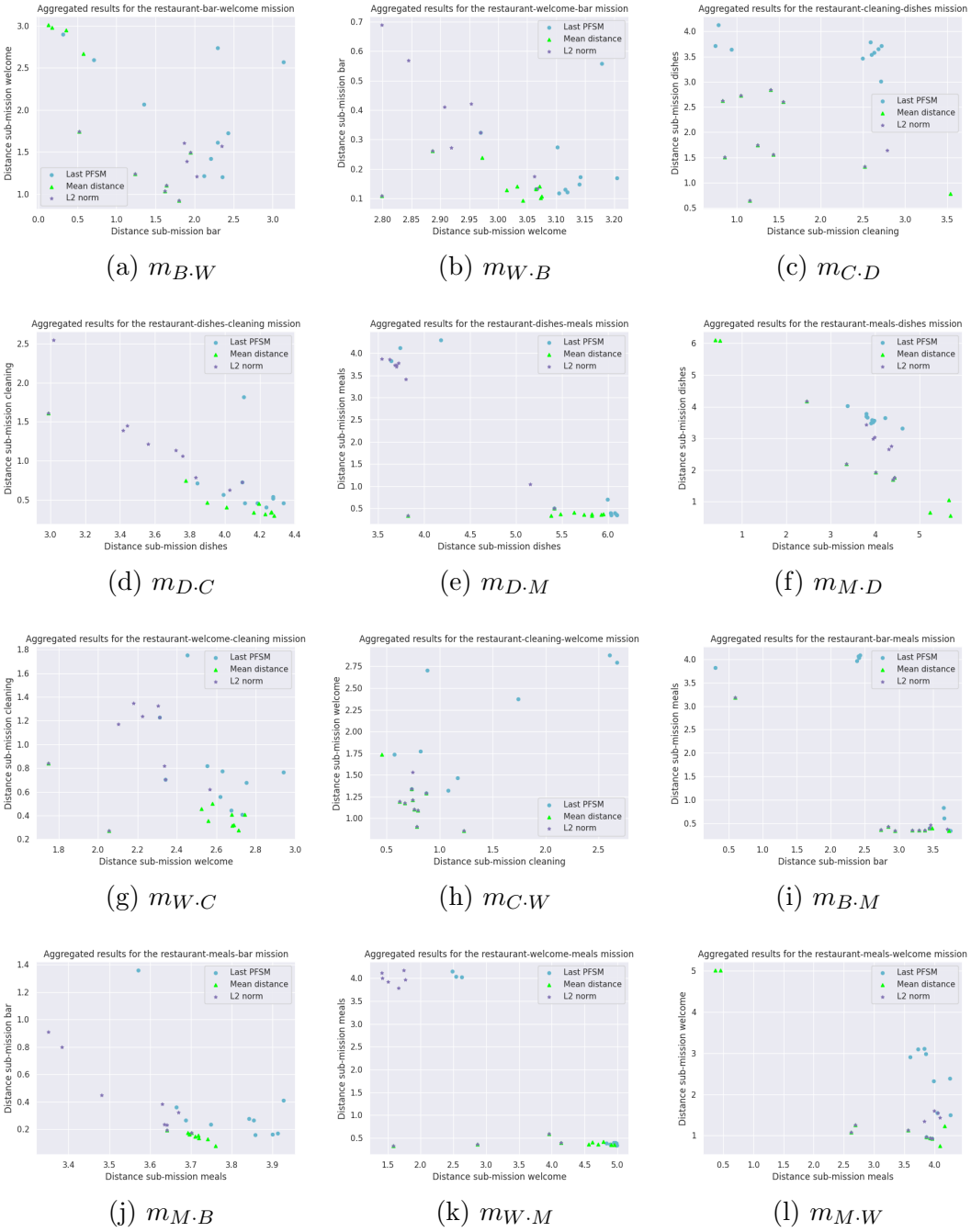
(a) $m_{B \cdot W}$ (b) $m_{W \cdot B}$ (c) $m_{C \cdot D}$

(d) $m_{D \cdot C}$ (e) $m_{D \cdot M}$ (f) $m_{M \cdot D}$

(g) $m_{W \cdot C}$ (h) $m_{C \cdot W}$ (i) $m_{B \cdot M}$

(j) $m_{M \cdot B}$ (k) $m_{W \cdot M}$ (l) $m_{M \cdot W}$

Figure C.1: Comparison of different ways of selecting the best PFSMs for each of the missions solved with the weighted sum.



(a) $m_{B \cdot W}$ (b) $m_{W \cdot B}$ (c) $m_{C \cdot D}$

(d) $m_{D \cdot C}$ (e) $m_{D \cdot M}$ (f) $m_{M \cdot D}$

(g) $m_{W \cdot C}$ (h) $m_{C \cdot W}$ (i) $m_{B \cdot M}$

(j) $m_{M \cdot B}$ (k) $m_{W \cdot M}$ (l) $m_{M \cdot W}$

Figure C.2: Comparison of different ways of selecting the best PFSMs for each of the missions solved with `Mandarina`.