ECOLE
POLYTECHNIQUE
DE BRUXELLES

ULB

UNIVERSITÉ LIBRE DE BRUXELLES, UNIVERSITÉ D'EUROPE

Spatial Allocation in Swarm Robotics

Mémoire présenté en vue de l'obtention du diplôme
d'Ingénieur Civil Informaticien

Jacopo De Stefani

Directeur
Professeur Mauro Birattari

Superviseur
Ir. Andreagiovanni Reina

Service
IRIDIA - CoDE

Année académique
2012 - 2013

"My family is my strength and my weakness."

— Aishwarya Rai Bachchan

"A friend is one that knows you as you are, understands where you have been, accepts what you have become, and still, gently allows you to grow."

— William Shakespeare

Grazie di tutto, Nonno.
Dedicated to the loving memory of Mario Enrico De Stefani.
α 1931 – Ω 2013

ABSTRACT

This thesis proposes three distributed methods to achieve the allocation of an homogeneous swarm of robots to spatially distributed tasks. Tasks are grouped together in space as to form clusters.

These methods have been developed in the form of probabilistic finite state machines, a microscopic level, behavior-based approach in Swarm Robotics.

The first method (Naive) simply consists of a greedy allocation of the robots to available tasks in space, as soon as they have been localized.

The second one (Probabilistic) improves the Naive one with probabilistic rules to avoid allocation conflicts and achieve a more uniform allocation.

The third (Informed) is built upon the Probabilistic one, using odometry to avoid redundant exploration.

The methods have been simulated on three different scenarios: Uniform, Biased and Corridor. We characterize the methods according to their allocation uniformity and allocation speed.

We show that there exists a trade-off between uniformity and speed for the developed methods. We also find that the positioning of the clusters has a strong impact on the performances of the methods. We conclude that the Informed method is the one having the best performances on the proposed scenarios.

KEYWORDS: Swarm Robotics, Task allocation, Decentralized control

MOTS CLÉS: Robotique en essaim, Répartition des tâches, Contrôle décentralisé

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [Knuth, 1974]

ACKNOWLEDGEMENTS

I would like to thank Mauro, for giving me the possibility to be at IRIDIA, and for the continuous and fruitful feedback he gave me through the preparation of this work.

Thanks also to Giovanni, who has always been there when I needed help on technical matters.

Special thanks to all the people that accompanied me during my journey in Brussels.

To my colleagues, Nadine, Dimitri and Robin, who have welcomed me since the beginning, gave me helpful advices concerning the university and helped improving my French skills.

To all the people I met throughout these two years that supported me and helped me grow as a person: Michele, Simona, Chiara, Laura, Thomas & Francesca among all.

A special mention for my "second family" in Brussels, Fabrizio, Giorgio, Laura, Eleonora, Blanca and Sinan.

Thanks to Gianluca, Davide & Anissa and Stefano and all my friends in Castelleone, who always made me feel at home, no matter how much time I spent away.

Last but not least, my heartfelt thanks to my parents, which gave me the possibility to leave for this two years experience in Brussels and which have always been supportive and present, regardless of the distance.

CONTENTS

1 INTRODUCTION 1

2 BACKGROUND & RELATED WORK 4

2.1 Concepts 4

2.1.1 Self-organization 4

2.1.2 Swarm Intelligence 6

2.1.3 Swarm Robotics 9

2.2 Spatial allocation 14

2.2.1 Related work 15

2.3 Materials 19

2.3.1 E-puck 19

2.3.2 IRIDIA Task Abstraction Module 22

2.3.3 Autonomous Robot Go Swarming 23

3 METHODOLOGY 26

3.1 Problem statement 26

3.2 Experimental setup 27

3.3 Methods 29

3.3.1 Overview 30

3.3.2 Naive 32

3.3.3 Probabilistic 34

3.3.4 Informed 36

3.3.5 Summary 38

4 RESULTS 40

4.1 Scenarios 40

4.1.1 Uniform 40

4.1.2 Biased 40

4.1.3 Corridor 42

4.2 Metrics 42

4.2.1 Allocated robots 42

4.2.2 Maximum error 43

4.2.3 Allocation levels 44

4.2.4 Cluster views 44

4.3 Allocation uniformity 45

4.3.1 Maximum error 45

4.3.2 Allocated robots 46

4.3.3 Allocation levels 47

4.4 Allocation speed 48

4.4.1 Scenario Uniform 49

4.4.2 Scenario Biased 49

4.4.3 Scenario Corridor 50

4.5 Scenario difficulty 58

4.5.1 Scenario Uniform 58

4.5.2 Scenario Biased 58

4.5.3 Scenario Corridor 59
5 CONCLUSION & FUTURE WORK 63
5.1 Conclusion 63
5.2 Future work 65
A ADDITIONAL PLOTS 66
A.1 Maximum error 67
A.2 Allocated Robots 70

BIBLIOGRAPHY 73

LIST OF FIGURES

Figure 1 Examples of self-organizing patterns emerging from morphogenesis 5
Figure 2 Examples of Swarm Intelligence 8
Figure 3 Overview of the typical development process in Swarm Robotics 11
Figure 4 Map of the relevant concepts related to Spatial Allocation 16
Figure 5 Annotated picture of the e-puck robot 19
Figure 6 Picture of the Task Abstraction Module (TAM) with an *e-puck* with the range and bearing board and the omni-directional camera extension 22
Figure 7 Architecture of ARGoS 24
Figure 8 Screenshot of the ARGoS simulator for the scenario B environment configuration 24
Figure 9 Schematic representation of the TAM disposition in a cluster 28
Figure 10 Finite state machine representing the TAM states 29
Figure 11 Finite state machine representing the *e-puck* behavioral rules 30
Figure 12 Deterministic finite state machine corresponding the implemented individual robot controller for the *Naive* method 32
Figure 13 Probabilistic finite state machine corresponding the implemented individual robot controller for the *Probabilistic* method 34
Figure 14 Probabilistic finite state machine corresponding the implemented individual robot controller for the *Informed* method 36
Figure 15 Representation of the cluster disposition in scenario Uniform 41
Figure 16 Representation of the cluster disposition in scenario Biased 41
Figure 17 Representation of the cluster disposition in scenario Corridor 42
Figure 18 Median values for the maximum error across clusters $e_{max}(t)$ and the number of allocated robots $R(t)$ on the scenario Uniform. 52
Figure 19 Empirical cumulative density functions for the allocation levels o_x $x \in 0.25, 0.50$ distributions of the three methods on the scenario Uniform. 53

Figure 20 Median values for the maximum error across clusters $e_{max}(t)$ and the number of allocated robots $R(t)$ on the scenario Biased. 54
Figure 21 Empirical cumulative density functions for the allocation levels o_x $x \in 0.25, 0.50$ distributions of the three methods on the scenario Biased. 55
Figure 22 Median values for the maximum error across clusters $e_{max}(t)$ and the number of allocated robots $R(t)$ on the scenario Corridor. 56
Figure 23 Empirical cumulative density functions for the allocation levels o_x $x \in 0.25, 0.50$ distributions of the three methods on the scenario Corridor. 57
Figure 24 Median value of $v_i(t)$ for $i \in \{1, \cdots, 4\}$ on scenario Uniform 60
Figure 25 Median value of $v_i(t)$ for $i \in \{1, \cdots, 4\}$ on scenario Biased 61
Figure 26 Median value of $v_i(t)$ for $i \in \{1, \cdots, 4\}$ on scenario Corridor 62
Figure 27 Comparison of the maximum allocation error e_{max} of the three methods on the scenario Uniform. 67
Figure 28 Comparison of the maximum allocation error e_{max} of the three methods on the scenario Biased. 68
Figure 29 Comparison of the maximum allocation error e_{max} of the three methods on the scenario Corridor. 69
Figure 30 Comparison of number of allocated robots $R(t)$ of the three methods on the scenario Uniform. 70
Figure 31 Comparison of number of allocated robots $R(t)$ of the three methods on the scenario Biased. 71
Figure 32 Comparison of number of allocated robots $R(t)$ of the three methods on the scenario Corridor. 72

LIST OF TABLES

Table 1 Summary of the applications of Swarm Robotics 13
Table 2 Overview of the technical specifications of the *e-puck* robot 21
Table 3 Overview of the technical specifications of the *TAM* device 23
Table 4 Clusters requests in Uniform,Biased and Corridor experiment setups 28
Table 5 Overview of the developed methods 38
Table 6 Summary of the values of the maximum allocation error $e_{max}(t)$ at $t = 10000$. 45
Table 7 Summary of the values of the number of allocated robots $R(t)$ at $t = 10000$. 46
Table 8 Summary of the probabilities to reach allocation levels o_{25} and o_{50} across 50 trials within 10000 time steps 47
Table 9 Summary of the median values of cluster views $v(t)$ at $t = 10000$ in scenario Uniform. 60
Table 10 Summary of the median values of cluster views $v(t)$ at $t = 10000$ in scenario Biased. 61
Table 11 Summary of the median values of cluster visits $v(t)$ at $t = 10000$ in scenario Corridor. 62

ACRONYMS

AI Artificial Intelligence

ARGoS Autonomous Robot Go Swarming

ER Evolutionary Robotics

IR Infrared

MAS Multi-Agent Systems

RL Reinforcement Learning

SI Swarm Intelligence

SO Self-organization

SR Swarm Robotics

TAM Task Abstraction Module

DCSP Distributed Constraint Satisfaction Problem

OAP Optimal Assignment Problem

CMOS Complementary Metal–Oxide–Semiconductor

LED Light Emitting Diode

CPU Central Processing Unit

XML eXtensible Markup Language

RAB Range and Bearing

1

INTRODUCTION

In nature, many animals societies exhibit forms of collective behavior. Such kind of behavior emerges from interactions occurring at animal level. One of the reasons for those interactions is that agents are often faced with demanding activities, whose complexity is generally beyond the capabilities of the individual. Hence, the only way to perform such kind of tasks is to require the intervention of other entities and join forces with them. When cooperation is triggered, the problem of dividing the global duty into more, simpler components and assign (i. e.allocate) them to the different agents arises.

Indeed, *task allocation* is a common problem that can be observed in different collaborative societies. The way the allocation is performed, however, varies across different species.

In humans, the most prominent example of this problem is the *division of labour*. This process consists of the decomposition a work activity into sub-tasks and their *allocation* to different individuals, in order to profit of the skills and capabilities possessed by each of them. The *allocation* is generally performed in a centralized manner, with a single entity that assign tasks and/or determine the degree of specialization of each agent. The presence of such entity, with a global knowledge of the agents and the tasks, allows, in most cases, for a quick and coherent allocation. However, a similar approach has some major drawbacks. First of all, the presence of a single unit with enhanced capabilities, introduces a single potential point of failure in the system. In fact, the allocation, thus the possibility to work of all the other agents depends on the "allocator". As soon as it will be unable to perform the allocation, the whole system will stop functioning. Furthermore, in many contexts it is impossible or impractical to gather a global knowledge of the system, hence limiting the effectiveness of the centralized solution.

The division of labour process in human societies has raised many philosophical and sociological issues, discussed by Plato, Adam Smith and Karl Marx, among others, whose discussion is beyond the scope of this thesis.

Surprisingly, successful examples of task allocation and task partitioning can be found in the animal kingdom. For instance, eusocial organisms (mostly insects, with some exceptions) are characterized by a reproductive division of labour, a specialization among reproductive and non-reproductive activities (caring for other individuals, nest building and defense, among others) that occurs at insect level. Moreover, some species of ants and bees are able to respond to changes in demand for particular activities by redistributing the available workforce. Here, task allocation is a product of the cooperation or, in other words, a consequence of the *self-organization* of the individuals. Even though each agent is a relatively simple entity,

with limited sensing capabilities, the task allocation for the complex activity can be achieved through processes and interactions locally occurring among them.

The study of collective behavior of decentralized, self-organized systems, natural or artificial, composed by a relevant number of homogeneous organisms, is the main focus of the of *Swarm Intelligence*, a research sub-field in Artificial Intelligence (AI). The goal of this research is to formulate models to explain the emergence of the behavior in order to be able to design effective, distributed algorithms for problem solving. In this thesis, we apply this paradigm to the *task allocation* problem. To be more precise, we focus on the problem of allocating agents to *spatially distributed* tasks.

Our study will be performed in the framework of Swarm Robotics, hence the agents will be a group of simple and (quasi) identical robots, with decentralized control and lack of synchronicity [Beni, 2005]. Given their characteristics, we decided to adopt is the E-puck, a small mobile robot, as the robotic platform for our experience. Tasks, on the other hand, are represented by IRIDIA Task Abstraction Modules, devices specifically designed to allow the *e-puck* to abstract tasks that are beyond its capability.

In our implementation of the problem, tasks are arranged to form a limited number of clusters in space.

Our purpose is to understand whether it is possible to achieve a uniform allocation of the robots across the clusters by relying only on *Self-organization*, without the need for any centralized solution.

In order to do so, we developed three methods. The first one (*Naive*) consists simply of a greedy allocation of the robots, as soon as the tasks are discovered, while the second one (*Probabilistic*) introduces a probabilistic mechanism to better distribute the robots across different clusters. The third (*Informed*) is built by adding an odometry-based memory of the last visited cluster to the *Probabilistic* one.

The performance of these methods are compared on three different scenarios: Uniform, Biased, Corridor, each one of them characterized by a precise disposition in space of the clusters and the initial deployment area of the robots.

THESIS LAYOUT

The thesis is structured in 5 chapters.

In BACKGROUND & RELATED WORK we start by presenting the reader the relevant notions related to the Swarm Intelligence (SI) field. Then, we discuss the problem of *spatial allocation* and we provide a summary of state-of-the-art on the problem (Spatial allocation). We conclude the chapter by providing the readers some specifics concerning the hardware (E-puck, IRIDIA Task Abstraction Module) and

software (Autonomous Robot Go Swarming) components used in the development of the thesis.

In METHODOLOGY, we first provide a formal Problem statement, we describe our Experimental setup and we illustrate the three developed methods, the Naive, Probabilistic and Informed one.

The chapter RESULTS is dedicated to a detailed analysis of the results and properties of the methods. First, we present the scenarios (Section 4.1) that we used to test our methods and the Metrics we measured in our experiments. Then, we focus on two relevant properties of our methods: the *uniformity* of the *speed* of the allocation, analyzed respectively in Sections 4.3 and 4.4. The chapter is concluded by an evaluation of the difficulty of the different scenarios (Section 4.5).

Finally, CONCLUSION & FUTURE WORK concludes the thesis with a discussion on the obtained results and suggestions for future research directions to explore.

2

BACKGROUND & RELATED WORK

The main purpose of this chapter is to present all the relevant knowledge required to understand the work described in METHODOLOGY. In Concepts we discuss the basic concepts upon which the thesis has been developed with a top-down approach. We begin by giving a quick overview of the Self-organization framework. Then, we proceed by illustrating the Swarm Intelligence field, focusing especially on the presentation of Swarm Robotics.

Once the context have been clarified, we state the objective of our work, the motivation behind it, its specific literature and related work in Spatial allocation.

In Materials, we conclude by giving a brief overview on the hardware and software components used in the development of the thesis.

2.1 CONCEPTS

2.1.1 *Self-organization*

A first, general notion that is required to fully understand the proposed methods is self-organization. Self-organization occurs naturally in a variety of biological, chemical, and social systems. Due to the heterogeneity and strong diversification among these categories, it is difficult to define precisely. For our scope, we borrow a working definition from Camazine et al. [2001]:

> *"Self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system.*
> *Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern."*

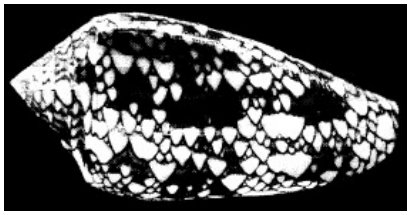
A *pattern* can then be seen as an organized disposition of the system components in space (or time).

Many processes in chemistry, for instance, are examples of how interactions at a microscopic level have a great influence on the final properties of a material at the macroscopic level.

One of them, *Crystallization*, is the process of forming solid crystals by precipitation from a (generally) liquid solution. It takes place when a solute (chemical compound) is mixed with a solvent, heated at high temperature. This causes the molecules of the compound to disaggregate and start floating around. By gradually reducing the

temperature of the compound, the free solute molecules, which cannot be held anymore in the solvent, start to aggregate. Not all the clusters of molecules become crystals, but only those who have reached a critical size (determined by the operating conditions of the process, namely temperature and pressure). The remaining molecules then join the "surviving" aggregates, arranging themselves in a defined periodical manner that gives the crystal its characteristic structure. It is indeed an example of *self-assembling* process, analogous to the process that binds the DNA molecules to form the characteristic double helical structure.

Without restraining ourselves to chemistry, similar activities can be observed in biology as well. Notable illustrations of the self-organizing phenomena are, indeed, morphogenesis, protein folding and homeostasis. *Morphogenesis* is a term used to identify the process through which a biological organism develops its final shape. Many particular patterns in nature, such as pigmentation on some species of shells (Figure 1a) and coat patterns in giraffes and zebras (Figure 1b) are examples of morphogenesis products.



(a) Pigmentation pattern on a cone shell (Courtesy of Stephan Wolfram)

(b) Pigmentation patter of a zebra coat (Courtesy of The Technium)

Figure 1: Examples of self-organizing patterns emerging from morphogenesis

In Turing [1952], Alan Turing explained the occurrence of a similar structures by means of the diffusion in the cells of chemical substances (morphogens). By modeling this phenomenon with systems of differential equations, he discovered that a particular pattern is the result of the interaction between different types of morphogenes, some promoting and other preventing cell growth. On the other hand, *Homeostasis* is the property of system to self-regulate its internal environment to maintain a stable, relatively constant state. This process, first studied by Bernard in the 18th Century (cf. Cooper [2008]), has been explained in the framework of dynamical systems (feedback cycles).

"Positive feedback isn't always negative"
–M.Resnick, Learning about Life

Even though, at a first glance, the aforementioned processes may seem really different among them, they can all be explained through the combination of *positive feedback*, that iteratively amplifies the re-

sponse of the system and *negative* one, which compensates by limiting it, leading the system towards stable equilibrium states.

From a different point of view, another relevant definition of self-organization has been given by [Ashby, 1962]:

> *"[...] the system that starts with its parts separate (so that the behavior of each is independent of the others' states) and whose parts then act so that they change towards forming connections of some type."*

Examples of self-organizing systems in this sense, are human neural networks, whose connections and capabilities evolve across time thanks to the interconnection with similar cells. It should be noted the latter definition does not address the question of the usefulness of the emerging organization and that both the definitions focus on the interaction (and consequent cooperation) among the parts of the system. In both cases, no constraints are imposed on the basic building blocks of the system, which could be either living or inanimate entities. Moreover, the definitions do not specify whether information is exchanged among the agents during their interaction and how such knowledge is used in the process.

The research in Swarm Intelligence, on the other hand, is focused on the study of self-organizing behaviors of a precise typology of entity: *agents* (i.e. entities capable of observing and modifying their environment by means of, respectively, sensors and actuators).

2.1.2 *Swarm Intelligence*

What is, then, an intelligent swarm? We can find a good answer in Beni [2005]:

> *"[...] a group of "machines" capable of forming "ordered" material patterns "unpredictably"."*

This definition allows us to immediately clarify the most important aspects concerning this discipline. First of all, the fact of dealing with systems composed by multiple individuals which are able to organize themselves autonomously. The intelligence of the swarm is expressed in terms of computational equivalence. Its unpredictability arises from the fact that it is as powerful as any other universal computational model.

Moreover, the notion of machine (as *"[...] an entity capable of processing matter/energy"*, from Beni [2005]) does not define the nature of the entity itself which could be either a living creature (*Natural* Swarm Intelligence) or a human artifact (*Artificial* Swarm Intelligence).

Natural Swarm Intelligence deals with the study of the collective behaviors emerging from the interactions of biological organisms (generally animals or insects).

The most fascinating examples in this category, under a visual point of view, are surely the flocking (schooling) behavior that can

be observed in groups of birds (fish). While displaying a similar behavior (Figures 2a, 2b), the group of animals moves as if it were a single, fluid entity. Furthermore, even though rapid changes of direction are triggered as a reaction to events in the environment (e.g. presence of a predator/obstacle) accidental collisions among the individuals are rare. Surprisingly, a limited set of rules (three, to be precise, cf. Reynolds [1987]) describing the inter-agent interactions and local sensing capability are sufficient to achieve this complex global behavior.

Insects are an interesting example of how organization can be achieved when the number of individuals scales up from tens to thousands of units in the group.

For instance, ants of the *Pheidole* genus are divided two distinct categories of individuals: minors and majors. Insects from the former category carry out the majority of the tasks required in the colony (grooming, brood care, foraging) while those in the latter focus on nest defense and seed milling. As shown in Wilson [1984], when the ratio minors:majors drops below a certain threshold, the repartition of the activities among the agents is modified in a way that majors compensate for the lack of minors by performing their tasks. If the ratio is increased again, the ants recover the original task allocation. An explanation of this adaptive unsupervised division of labor, has been given several years later with the response-threshold model (Theraulaz et al. [1998]), based only on features of each agent.

Bees, on the other hand, show us how the global structure of their hive can emerge from the independent, but self-organized work of a large number of individuals (Figure 2d). An honey bee colony presents in fact a characteristic pattern, made by three concentric zones containing respectively brood, pollen and honey (from the center to the exterior area). A first hypothesis that has been formulated to describe this phenomenon was that bees possessed some innate knowledge about how to arrange cells (e.g.blueprint or pattern). What Camazine [1991] has shown, is that brood is initially deposed in a compact way by the queen, while pollen and honey are dispersed randomly in the structure or in cells where the same substance is already present. The structure then emerges from a simple nourishment [1] displacement behavior performed by all the agents, based on brood quantity in the neighborhood, along with the different rates at which new substances are brought in the colony.

Evidence of the emergence of a collective intelligence through simple interactions in a large group of individuals have been observed in humans as well. In Krause et al. [2010] we find the application of this principle to different real situations such as the guess of the exact quantity of marbles in a jar, the management of a football team, the design of a new product. The main benefit of a similar cooperation

1. Pollen and honey.

in humans is the collective information processing to provide a new cognitive solution to a problem.

(a) Flock of auklets (seabirds). (Courtesy of D. Dibenski)

(b) School of fishes swimming together in a spherical formation.

(c) Major and minor ants of the *Pheidole* genus taking care of the colony brood. (Courtesy of Alexander Wild)

(d) Bees deposing honey in their colony. (Courtesy of Kamillo Kluth)

(e) Exemplification of the "Wisdom of crowds" concept.

Figure 2: Examples of Swarm Intelligence

Why are these behaviors widely observed in nature? Why there is an interest in their study from an engineering standpoint? An answer can be found in the characteristics that these systems possess.

First of all, they are completely *distributed*, in the sense that there are no leading individuals in the group that impose a certain behavior, neither locally nor globally. Furthermore, all the agents are relatively *homogeneous* (i.e., they present only limited variability in terms of morphology or capabilities) with only *local sensing* capabilities (that

is, they could only have insights on what is happening in their surroundings, but not at a global level). The combination of this features implies that no individual is essential to the survival of the group, thus ensuring the ability to cope with the loss of individuals (due to death or temporary/permanent inability), hence the *robustness* of the system. Moreover, through simple behavioral rules concerning the interaction with other agents and basic information transfer, both direct or indirect (e.g. stigmergy), a collective, intelligent organization emerges. This locality allows the system to be *resilient* with respect to both the fluctuations in the environment and in the swarm. As we may notice, all the aforementioned features brings a set of advantageous properties to the system.

The research in Swarm Intelligence is mainly twofold. On one hand, the aim is to understand the mechanisms underlying such kind of systems and provide a model which is able to explain the emergence of a collective behavior (*Scientific Swarm Intelligence*). Since Swarm Intelligence is a bio-inspired discipline, the largest part of this research is conducted by life scientist. On the other hand, the focus is how to apply the discovered knowledge to problems having a practical relevance (*Engineering Swarm Intelligence*). In this case, researchers in computer science, mostly in artificial intelligence and robotics, are using the knowledge gathered in other fields to create algorithms able to solve optimization problems, to perform realistic simulations of thousands of agents or develop efficient robotic systems.

2.1.3 *Swarm Robotics*

In the first years of 1990, while life scientists [Camazine, 1991] were observing and modeling the behavior of social insects, the first examples of multi-agent systems were being developed by computer scientists (Singh [1991]).

These artificial systems are composed by a group of autonomous entities interacting among them and with an environment, which can be either physical (e.g robots in a real environment) or virtual (software agents, for example). The nature (physical/virtual, human/artificial) and the complexity of the agents (active/passive, cognitive/reflexive) could be highly variable, yielding to heterogeneous sets of agents. Over the years, similar systems have been mainly used to perform distributed problem-solving (Colorni et al. [1991]) or multi-agent simulation of real world processes (Benenson [1998]).

In the years 2000, the biological and artificial research paths have joined in the Swarm Robotics field.

A more precise definition can be found in Şahin [2005]:

> *"Swarm robotics is the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local inter-*

actions among agents and between the agents and the environment."

A physical embodied agent is an entity whose behavior is affected by its morphological features and the environment it is situated in [Pfeifer et al., 2007]. In order for the agent to be embodied, it must be able to transfer and process matter, energy (through its actuators) and information (by means of its sensors and its internal architecture), hence it must be, according to Beni [2005], a robot.

As a consequence, we can observe from the definition that Swarm Robotics is a branch of collective robotics where a global behavior is achieved through the self-organization of the individuals. The autonomous agents are relatively simple in the sense that they are equipped with a minimal set of sensors and actuators, that prevents each agent from having a *global knowledge* of the system and the environment.

It is important to clarify that, while the definition deals with a large number of robots (from hundred to thousands of robots), the empirical experiments in Swarm Robotics are performed with group of at most 30 to 50 robots. This discrepancy is mainly due to practical and economical reasons. In fact, even though the technological evolution allowed to reduce the costs of the microelectronic components that constitute a robot, a single assembled agent still remains costly (from several hundred to several thousands euros). Furthermore, most of the robots used in the experiments are battery-powered, which constrains the autonomy of the robots to those of their power supply.

Despite of these limitations, why there is still an interest in Swarm Robotics? The answer to this question can be found by analyzing how self-organization can solve some common issues in collective robotic. For instance, the robot coordination in many application of collective robotics is achieved through a shared, centralized communication medium, which is accessed concurrently by all the agents. A similar solution has some major drawbacks: if the medium is unavailable, the system cannot operate correctly. Moreover, the quality of the communication decays as the number of robots in the system increases, due to the concurrency. Conversely, if the coordination is achieved through local interactions of the robots, the *robustness* of the system with respect to faults and disturbances can be achieved. The points of failure will then become distributed among the agents, lowering their influence on the overall performance of the system. Also, if the agents are homogeneous and relatively simple, the swarm becomes even more dependable by means of *redundancy*. In addition, the lack of a central communication mechanism favors the *scalability* of the system with respect to changes in the number of individuals. Another advantage of self-organization is that robots could operate autonomously, hence performing *parallel* activities which are

distributed in space and in time, improving the global efficiency of the system.

It is important to stress the fact that the aforementioned properties are not guaranteed as a consequence of self-organization.

2.1.3.1 *Development methodology*

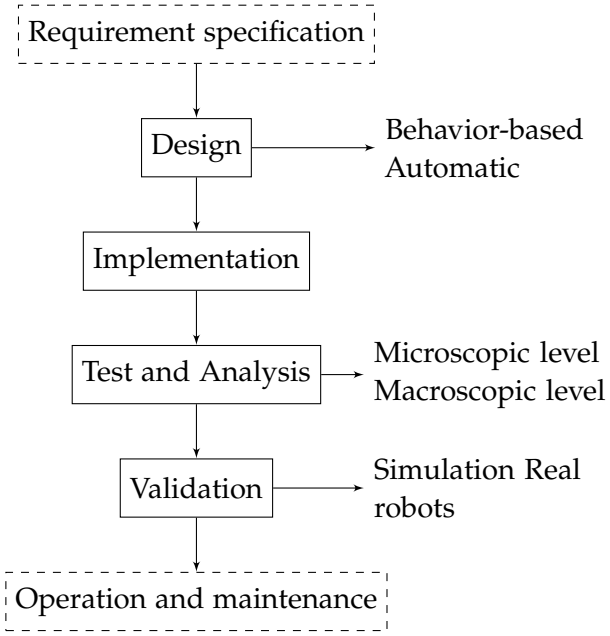


Figure 3: Overview of the typical development process in Swarm Robotics. Backward feedback loops are omitted for clarity. The elements having a dashed border are not generally performed in the context of Swarm Robotics.

Indeed, the main research question in Swarm Robotics (SR) is how to develop design methodologies at the individual level that will cause the emergence of a collective behavior exhibiting the aforementioned properties.

Figure 3 depicts the typical development process in Swarm Robotics. The systematic application of scientific and technical knowledge in a structured way in the development process is generally referred as Swarm Engineering.

Concerning the *design* step, there are two common approaches, *behavior-based* and *automatic*.

The *behavior-based* one is a process that consists in developing (often by hand, in a trial-and-error fashion) a specification of the agents behavior which can be easily implemented. According to Brambilla et al. [2013] the most used methods in this category are the *finite state machine* and *virtual physics* one. The *finite state machine* one is based on the definition of a set of relevant states for the robot and the corresponding transitions, based on the inputs from the sensors and

The interested reader could find all the details and relevant articles describing their application in Brambilla et al. [2013]

on the robot current state. *Virtual physics* [Spears et al., 2004] treats each agent as a virtual particle subject to virtual forces. Those forces are the result of the interaction of the robot with a virtual potential field, that can be perceived through its sensors.

As for *automatic* methods, the idea is to apply Artificial Intelligence techniques to the design process of the robot, without relying on human-based development. The most frequently used techniques are Reinforcement Learning (RL) and Evolutionary Robotics (ER).

With Reinforcement Learning [Sutton and Barto, 1998] the agent learns a desired individual behavior by means of iterated interactions with the environment, receiving a feedback on its actions. The polarity (positive or negative) of the feedback, determines whether the agent should retain or forget a certain component of the behavior.

In Evolutionary Robotics, evolutionary computation techniques [Goldberg and Holland, 1988] are applied to the swarm of robots. The individual behavior of the robots (identical for all the agents) is the basic component that is evolved through different iterations of the method. At the beginning, this behavior is generated randomly and the collective behavior produced by the simultaneous execution of it on all the robots is evaluated by means of a fitness function. At each iteration, the individual behaviors are modified by means of selection, recombination and mutation operations. The process is stopped once the desired collective behavior has been achieved.

After the design phase have been completed, the emerging behavior must be thoroughly analyzed in order to assess whether the desired properties hold or not, before actually validating it with tests on the real robots. The *analysis* is performed mainly by means of *simulations* and *mathematical models*.

Simulations offer a swarm representation which is as close as possible to the reality, by modeling each robot component at a microscopic level. Unfortunately, this accuracy comes at the price of computational complexity, which dramatically increases as the number of agents becomes larger, making technically unfeasible the simulation of numerous swarms.

On the other hand, *mathematical models* based on the theory of dynamical systems or on the control and stability theory can be used to describe the system at a macroscopic level. They can be studied regardless of the group size, but their extent is often restricted to simplified representations of the systems to study, limiting their practical utility.

The final step of this engineering process is the execution of the individual behavior by a swarm of real agents. Surprisingly, as Brambilla et al. [2013] shows, the validation on real robots is performed on less than half of the papers that the authors have reviewed.

The validation through simulation of the results presents some clear advantages: it prevents damages to the robot in case of a be-

SCIENTIFIC	ENGINEERING
– Aggregation – Pattern formation – Self-assembly – Coordinated motion	– Self-assembled motion – Area coverage – Collective indoor exploration – Collective transport – Consensus achievement – Task allocation – Fault detection – Group size regulation

Table 1: Summary of the applications of Swarm Robotics. The task typologies are those presented in Brambilla et al. [2013].

havior fault, it can be easily parallelized thus being executed faster, it does not require the actual deployment and maintenance of the robots. However, the lack of experiment in the real environment of the robots does not give any guarantees on the feasibility of the methods and on the realism of the underlying assumptions.

2.1.3.2 *Applications of Swarm Robotics*

As we discussed in Swarm Robotics, the affordability and the maintenance issues of a swarm of real robots are the problems that hinder the creation of swarm-based solutions that can operate in everyday life.

Nevertheless, the research in Swarm Robotics has provided valuable insights on the functioning of self-organized biological systems and proofs-of-concept regarding how agent cooperation can successfully tackle complex problems. Hence, according to the taxonomy defined in Dorigo and Birattari [2007], these results can be classified as *Artificial Swarm Intelligence* with both *Scientific* and *Engineering* purposes.

Under a *scientific* point of view, the most fruitful applications are related to collective behaviors that cause the emergence of patterns with a precise connotation in space. For instance, Garnier et al. [2005] have successfully designed a robot controller that causes the formation of group of agents in defined regions of the space (i.e. their *aggregation*), replicating the behavior of cockroaches. In addition, the *chaining* behavior of ants have been modeled and implemented on artificial agents, both with [Mondada et al., 2005] and without [Nouyan et al., 2008] physical connection among the agents. Also, the *flocking* behavior, the most known example of *coordinated motion*, depicted in figures 2a and 2b, has been achieved with a group of robots [Balch and Hybinette, 2000].

Studies in Swarm Robotics has not only proven to be useful in explaining the underlying mechanisms that trigger the emergence of Swarm Intelligence in animals and insects, but also in developing solutions to problems having a practical relevance (i.e. from a *engineering* standpoint).

Here, the most interesting results have been obtained while dealing with *environment exploration*, *object transport*, *swarm awareness* and *decision making* problems.

Within the *exploration* framework, solutions have been found by deploying robots in a way that they maximize the *area coverage* of the given environment, sometimes by having a connectivity constraint among the agents [Howard et al., 2002] or by using an heterogeneous swarm, in order to take advantage of the different characteristics of the individuals [Ducatelle et al., 2011].

Collective object transport has been successfully tackled by a group of robots, simply by relying on the force, position and orientation sensing of each agent [Donald et al., 1997].

Some methods have also been developed to make the swarm *aware* of its current state, for example, by being able to *detect* the presence of *faulty robots* [Christensen et al., 2009] or estimate and regulate its group size [Brambilla et al., 2009].

Moreover, the problem of having the swarm reach a *consensus* concerning a choice (e.g. moving direction, aggregation point) that will affect the performance of the whole group, have been solved using both direct [Gutiérrez et al., 2010] and indirect communication [Garnier et al., 2005].

Last but not least, the problem of *allocating* robots to different tasks (foraging against resting) in order to maximize the throughput of the system have been successfully addressed with an individual decision mechanism [Krieger and Billeter, 2000].

We are aware that the overview we presented may be incomplete, but its main purpose is to illustrate the most prominent results obtained in Swarm Robotics.

2.2 SPATIAL ALLOCATION

In this thesis we investigate the problem of allocating embodied agents to physical tasks distributed in space. A brief example might clarify this description.

Consider a search and rescue scenario: some people are dispersed in an hazardous environment (e.g. a building on fire) and they need help in the shortest possible time. The nature of the environment generally impedes a global exchange of information among the rescuers, making a centralized coordination of the process unfeasible. Moreover, the information concerning the topology of the environment and the number of people in distress may be unavailable or unreli-

able. If we consider that robots are, in general, faster, stronger, more resistant and more accurate than human, we could foresee the application of a swarm of robots to operate in the aforementioned dangerous environment. In any case, the only operable solution would be to let the rescue agents self-coordinate themselves during the process. The optimal allocation, in a similar scenario, would be the one allowing to rescue all the people in the shortest possible time.

Indeed, the fundamental aspects that characterize the performance of the method are the *uniformity* of the distribution of agents across the requests in the environment and the *time* needed to discover and allocate to the task.

The collective behavior that the robots should achieve is in between the *task allocation* and the *foraging* activity (cf. Figure 4). In fact, in the context of Swarm Robotics, the *task allocation* problem is usually related to the choice among different alternatives, which are known a priori, concerning the role of the agents in the swarm [Krieger and Billeter, 2000; Agassounon and Martinoli, 2002; Pini et al., 2009].

On the other hand, *foraging* is the process of collectively searching for resources scattered in the environment, harvesting them and depositing them at predefined collection points. This activity can be decomposed into four phases. First, the robots have to employ a search strategy (e.g. uninformed random walk) to *localize* the resource in space. Once the object has been found, the agent have to physically *collect* it. Then, it must apply a *navigation* strategy to head back to the collection point. Lastly, the robot *deposit* the gathered object and restart the process. A detailed taxonomy of the process can be found in Winfield [2009].

Moreover, according the taxonomy defined in Gerkey and Matarić [2004] our problem could be classified as ST-SR-IA (Single Task, Single Robot, Instantaneous Assignment). As a matter of fact, we analyze a problem in which homogeneous agents (i.e. no difference among them) should *localize* homogeneous tasks (that is, any robot could potentially allocate itself to any activity) in space and then *decide* whether their allocation to the task is required or not.

2.2.1 *Related work*

In our vision of the problem, the only common subtasks with the *foraging* activity are the *localization* and the *collection* one. The *navigation* and *deposit* operations are outside of the scope of our methods. As for the *localization* problem, several techniques of *collective exploration* such as *area coverage* [Spears et al., 2004] or *chain formation* [Nouyan et al., 2008] could be potentially employed. Instead, we choose to implement a simplifed and unbiased exploration technique: *random walk*. In addition, the *collection* operation will be abstracted by means of a specific device (i.e. *TAM*).

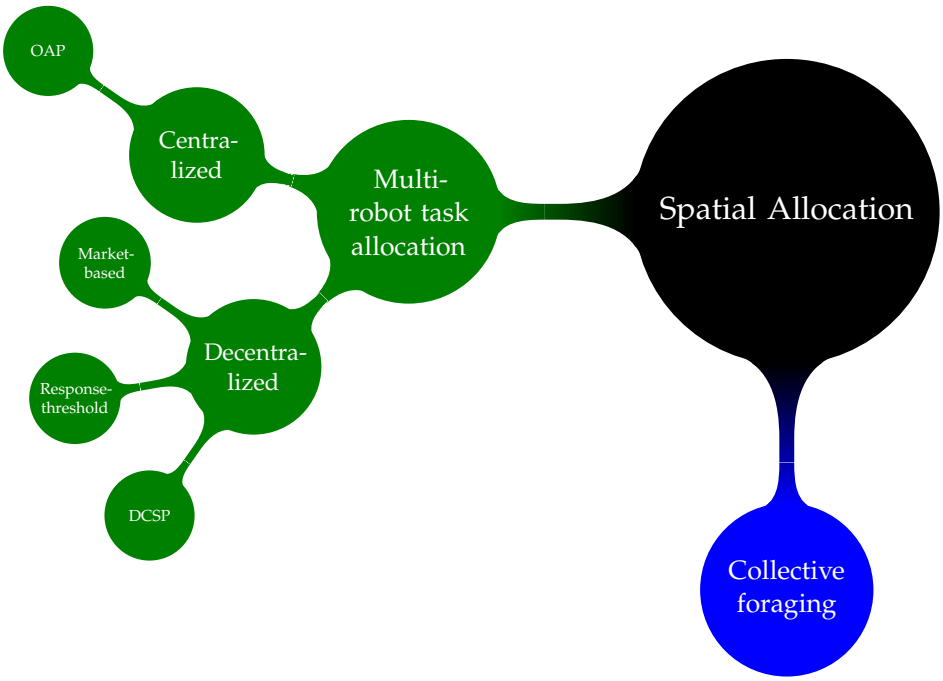


Figure 4: Map of the relevant concepts related to Spatial Allocation

For these reason, we choose to focus our related literature analysis on the *task allocation* instead of the *foraging* activity. Here, we can distinguish two different approaches: *centralized* and *decentralized*.

Centralized approaches are based on the modeling of the allocation activity as an optimization problem, the Optimal Assignment Problem (OAP) [Gerkey and Matarić, 2004].

In the classical formulation of the problem [Gale, 1960], there are m agents that have to be assigned to n jobs. Each agent i is characterized by a utility estimate value U_{ij} that predicts his performance on task j. The optimal solution of the problem consists in the allocation of agents to jobs that maximizes the system performance (utility). By gathering all the information concerning the agents in a single entity, which can be either a robot in the group or an external computation device, it is possible to obtain a solution using a combinatorial optimization algorithm (e.g. the Hungarian method Kuhn [1955]).

A similar approach requires complex communication capabilities and a global knowledge of the characteristics of the agents in order to be effective, requirements that are generally not met on real Swarm Robotics applications.

Moreover, in case of malfunctions or unavailability of the single allocator entity the whole system ceases to function. Also, the centralized collection of information introduces a bottleneck in the system. We can conclude that this approach lacks of robustness and scalability, thus being unsuitable for swarm implementations.

Decentralized approaches can be classified according to the type of coordination they employ. On one hand, we have *intentional coor-*

dination among the agents, as in *market-based* approaches, for whose detailed survey of can be found in Dias et al. [2006]. The underlying principle is similar to the OAP one.

Each agent possesses, in fact, an individual utility function that quantifies its preferences for the allocation to a certain task. In addition, the agent can estimate the cost (e.g. the distance, the energy consumption) related to the task it is seeking allocation for.

Whenever more than one agent is interested in a certain task, an auction process is started. On basis of the information it possess, it makes a certain bid for the task. The agent with the highest bid wins the auction process and gets the task.

Examples of approaches in this category are Lin and Zheng [2005]; Guerrero and Oliver [2003].

An advantage of this method is that the auction process does not necessarily require global or perfect information. Unfortunately, the bidding process is particularly demanding in terms of communication resources, since it requires an iterated exchange of messages by all the parties involved in the transaction and does not scale well as the number of robots increases.

On the other hand, *coordination emerges* from simple interactions among the agents, as in *response-threshold based* approaches, which draw inspiration from nature.

The response-threshold model, developed by Theraulaz et al. [1998], assumes that each task in the environment has an associated stimulus. Each agent has a different response threshold for the available tasks and its probability to engage in the activity is a function of the threshold and the value of the stimulus. The stronger the stimulus, the higher the likelihood that an agent will allocate itself to a certain task. The threshold can be either fixed (as in the original model) or adaptive (in a Reinforcement Learning fashion). In the latter case, the decrease of the threshold value for a certain activity can make an individual more sensible to the corresponding stimulus, resulting in more frequent allocation. Conversely, an agent will be more reticent to allocate to a certain task when the corresponding threshold is raised. Since the allocation decision is performed without direct communication among the agents, the approach scales well as the number of agents increases. Moreover, it is robust with respect to faults on the robots and flexible with respect to variations in the demand. The main drawback of this approach is the difficulty to predict the emergent behavior of the system, making the design of a system for a specific purpose problematic. A comparative analysis among the two mentioned approaches has been done by Kalra and Martinoli [2006], showing a better performance of the response-threshold approach with respect to the market-based one when the information is inaccurate.

The multi-robot task allocation problem can also be seen as an instance of the Constraint Satisfaction Problem. The problem consist in finding an assignment of values to a set of variables V from the corresponding domains set D that respect the set of constraints C. If we consider V as the set of tasks, D as the set of robots and we introduce the constraint that each robot must be assigned to exactly one task we obtain the single robot, single task allocation problem. In Shen and Salemi [2002], this problem is approached in a distributed fashion (Distributed Constraint Satisfaction Problem (DCSP)). The proposed solution however, is particularly demanding in terms of task modeling and information exchange among robots, thus difficult to apply on large groups of robots. Up to now, to the best of our knowledge, there are no studies that explicitly concentrate on the allocation of robots to physically situated activity with a self-organizing approach.

Despite having a completely different approach, the work from Hsieh et al. [2008] addresses a problem closely resembling ours. Their problem involves N agents to be distributed among M sites in an environment whose topology can be expressed as a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, in terms of sites ($\mathcal{V}$) and one-way connections among them ($\mathcal{E}$). The state of the system is represented by the relative number of agents at each site i ($x_i(t)$). The authors propose two strategies for redistribution: *baseline* and *quorum-based*. In the first one, the distribution of the agents occurs by means of transition probabilities per unit of time (k_{ij}) between nodes, defined on every edge in $\mathcal{E}$. In the second one, each site has an associated quorum value q_i. Whenever the current occupation $x_i(t)$ is above the quorum then one of the outgoing transitions rates from that node is multiplied of a coefficient α until the occupation descends again below the quorum value. In both cases, the entries of $\mathbf{K}$ (i.e. $k_{ij} \forall i, j$) are determined using Metropolis optimization [Landau and Binder, 2009].

Even though the model allows to obtain promising results, successfully achieving the redistribution of a swarm of 20000 robots, there are some strong assumptions that differentiate this approach from ours. First of all, the agents do possess a global knowledge of the topology of the environment ($\mathcal{G}(\mathcal{V}, \mathcal{E})$) and thus are capable of localizing themselves and navigating easily from one site to another. Moreover, the robots are supposed able to estimate the current occupation $x_i(t)$ of the site they are in. Also, the results are obtained only with simulations where real-world non-linear effects on the sensors and actuators along with realistic deployment dynamics for the robot have not been modeled. Although the stochastic transition polices are defined at an agent-level and do not require any wireless communication, the actual values of the transition probabilities are computed off-line and then distributed to all the agents, making the proposed solution de facto centralized.

In contrast to Hsieh et al. [2008], our purpose is to distribute uniformly agents across spatially distributed tasks, in real-time, without global information concerning the topology of the environment or the transition rates.

2.3 MATERIALS

Since we are going to develop a method in the context of Swarm Robotics, we must clarify two fundamental aspects: the *characteristics* of the embodied agents and the *development* methodology we are going to adopt.

Our embodied agents are mobile robots, the *e-pucks*, presented in detail in the E-puck section.

A novel point in our approach is that tasks are abstracted but embodied as well, by means of a specifically designed device, the *TAM*, whose specifications will be discussed in IRIDIA Task Abstraction Module.

As for the *development* methodology, we are going to adopt a behavior based approach (explained in METHODOLOGY) with a microscopic-level simulation-based analysis. All the relevant information concerning the multi-robot simulator can be found in Autonomous Robot Go Swarming.

2.3.1 *E-puck*

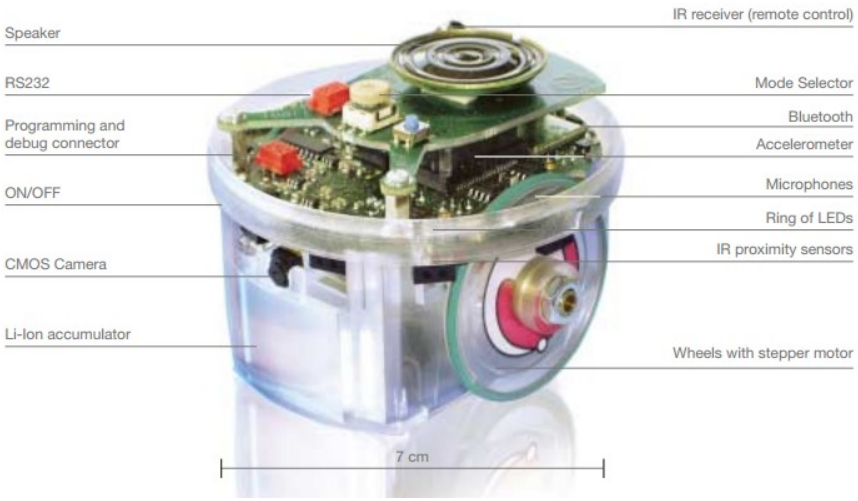


Figure 5: Annotated picture of the e-puck robot. (Courtesy of RoadNarrows)

The *e-puck* (Figure 5) is an open-source educational desktop mobile robot [Mondada et al., 2009]. It has been developed at the EPFL (Ecole Politechnique Federale de Lausanne), in order to provide a common

Only the relevant aspects for the thesis have been discussed here. For more details concerning the robot, visit `http://www.e-puck.org`

robot platform to be used in a broad range of university courses. The *educational* purpose of the robot has influenced its design process.

The compact and modular shape of the robot is thought to be *robust* with respect to the student use but easy to repair at the same time. Moreover, its size has been conceived make the robot usable also in relatively small environment, like a desk.

In addition, the robot is equipped with several different sensors and actuators as well as with an expansion board, making the robot flexible and usable in a wide range of contexts (from signal processing to automatic control, for example).

In order to favor the diffusion of knowledge and the improvement of the robot, the designers have given open access to its hardware specifications and distributed the related software with an open source license.

The publication of all the information concerning the robot has made the development of extension boards for the *e-puck* possible. Among the different available extensions, we focus only on those relevant for the development of our methods: the *omni-directional camera* and the *Range and Bearing (RAB)* board.

The *omni-directional camera* is build by encapsulating the standard *e-puck* camera in a glass cylinder, with an hyperbolic mirror on the opposite edge. By pointing the camera towards the mirror it is possible to obtain a 360 degrees viewing range around the robot. The board, containing the CMOS camera, a FIFO buffer and an additional microcontroller, is mounted on top of the robot and connected to it by means of a card-edge connector leaving the possibility to connect other extensions to the *e-puck*.

The *Range and Bearing* board is an extension developed to provide local communication capabilities to the robot [Gutiérrez et al., 2008]. By means of 12 IR emission/reception modules, nearly uniformly distributed on the perimeter of the board, the *e-puck* becomes capable of broadcasting up to 16 bit of information to the nearby robots. The board takes its name from the fact that, given the power of the received signal and the placement of the sensors/actuators, the embedded microcontroller on the board is able to compute the range (distance) and bearing (angle with respect to the receiver) of the sender robot.

Figure 6 shows an *e-puck* with both the aforementioned extensions installed.

We decided to choose the *e-puck* as the robotic platform for the development of our methods since it is a simple yet powerful device which best represent the notion of swarm agent.

If we look at the specifications (Table 2), we may notice the *limited sensing capability*, especially with respect to the sensors we are going to use in our method: the *proximity sensors* and the *camera*.

The affordability and reduced size (with respect to similar devices) makes also the creation of a large swarm of robots feasible.

Although the *e-puck* possesses a wide set of sensors and actuators, we are only going to use those required to perform displacements (*wheels*), obstacle avoidance (*proximity sensors*) and odometry (*encoders*), plus the functionalities offered by the extension boards to perform phototaxis (*omni-directional camera*) and local communication (*range and bearing system*).

Phototaxis is a kind of taxis, or locomotory movement, that occurs when a whole organism moves in response to the stimulus of light.

FEATURES	TECHNICAL INFORMATION
SIZE AND WEIGHT	70 mm diameter, 55 mm height, 150 g
PROCESSOR (ROBOT)	dsPIC 30F6014A @ 60 MHz (15 MIPS) 16 bit microcontroller with DSP core
CONTROLLER (OMNIDIRECTIONAL CAMERA)	dsPIC33FJ256 GP506 16 bit microcontroller with (Averlogic AL440B-24-PBF) (FIFO) frame buffer
CONTROLLER (RAB BOARD)	dsPIC 33FJ256 16 bit microcontroller
MOTORS	2 stepper motors with a 50:1 reduction gear
ENCODERS	One per wheel, pulses resolution: 0.13 mm
PROXIMITY SENSORS	8 IR sensors uniformly placed below the ring
CAMERA	2 CMOS VGA color camera, resolution: 480x640 pixels (4 fps at 40x40) (Front and omni-directional)
RAB EMISSION/RECEPTION MODULE	Infrared emitting diode with infrared modulated receiver and infrared photodiode.
MICROPHONES	3 omni-directional microphones for sound localization
ACCELEROMETER	3D accelerometer along the X, Y and Z axis
LEDS	8 red LEDs (ring), green LEDs (body), one strong red LED (front)
SPEAKER	On-board speaker (WAV and tone sound playback)
CONNECTIVITY	Serial port (RS232), Bluetooth, IR Remote control
PROGRAMMING LANGUAGES	C (General purpose), ASM (DSP)

Table 2: Overview of the technical specifications of the *e-puck* robot

2.3.2 *IRIDIA Task Abstraction Module*

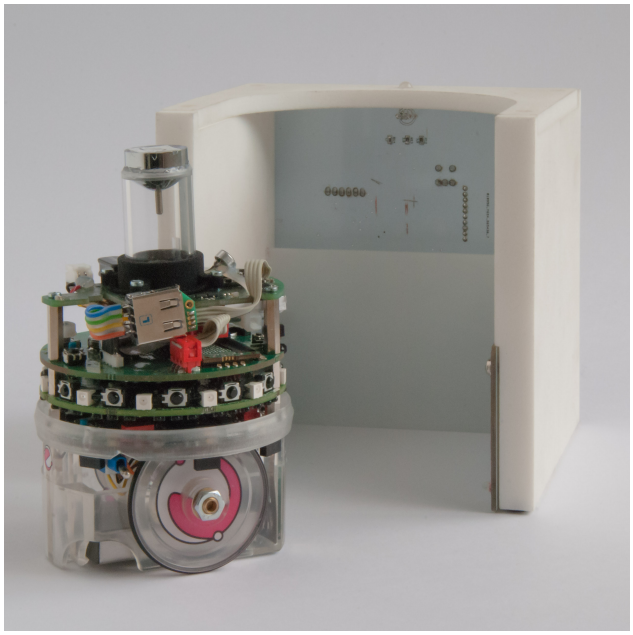


Figure 6: Picture of the TAM with an *e-puck* with range and bearing board and the omni-directional camera extension (Courtesy of Arne Brutschy)

The IRIDIA TAM is a device for task abstraction specifically designed for the *e-puck* robot [Brutschy et al., 2010]. Due to its physical structure, the robot cannot be extended by means of a manipulation device (e.g a gripper) thus limiting the number of activities that the robot could actually undertake.

The TAM comes from the idea that the tasks that are beyond the capabilities of the robot could be instead simulated. It has been designed as an U-shaped booth (Figure 6) that could welcome a robot, whose presence can be detected by means of the *light barrier*. The task abstraction occurs through the interaction between the device and the *e-puck*. In fact, the TAM is a programmable device which can display its internal state through the colors of the *RGB LEDs*.

It is a duty of the researcher to find a suitable representation of the task, given the functionalities of the aforementioned device. An example of a successful application of this device is Pini et al. [2011], where a group of *TAM* is used as a cache site to simulate the deposit and pick up of objects.

In the context of our work, we take advantage of the fact that a *TAM* could be seen as a physical representation of an activity, thus having a precise collocation in space. Indeed, *TAMs* will model the spatially distributed tasks that will require the allocation of a robot.

FEATURES	TECHNICAL INFORMATION
SIZES	120 mm x 120 mm x 108.3 mm (height x width x depth)
PROCESSOR	Atmel ATmega168 @ 20 MHz (20 MIPS) 8-bit micro-controller
BATTERY	5Wh LiION rechargeable and removable (~3 hours autonomy)
LIGHT BARRIER	IR LED (emitter,left side) with IR transistor (receiver,right side)
LEDS	3 RGB LEDs (left side, right side, top)
CONNECTIVITY	802.15.4 XBee DigiMesh (Wireless)
PROGRAMMING LANGUAGES	Processing, C, C++

Table 3: Overview of the technical specifications of the *TAM* device

2.3.3 *Autonomous Robot Go Swarming*

ARGoS [2] is a multi-robot simulator, written in C++, and released under GPL-3 license.

ARGoS 2.0 development has been discontinued, the new version of the simulator, still in beta testing phase, is available at `https://github.com/ilpincy/argos3`

It has been developed by Pinciroli et al. [2012] in the framework of the EU-Funded project Swarmanoid [3] which is also the official simulator of the EU-Funded projects ASCENS [4], H2SWARM [5], and E-SWARM [6].

Since the Swarmanoid project dealt with a swarm of heterogeneous robots, two main issues had to be addressed by the simulator: *flexibility* and *efficiency*.

In order to be *flexible* the simulator should allow the user to implement and integrate new features (e.g. sensors or robot models). In ARGoS, flexibility is achieved through a modular architecture on every level. Every simulated entity, from the sensors to the physics engine, is implemented as a plugin, which can be easily modified or extended by the end-user. A similar architecture supports the existence of different versions of the same components (e.g. actuators/sensors with-/without noise models, fine vs coarse grained physical simulations) making the simulation accuracy highly tunable.

Moreover, the plugin nature of the components (Figure 7) allow the user to load only the required ones, making the simulation more efficient. Efficiency is also achieved through the support for the simultaneous simulation of different physical engines, each one of them as-

2. http://iridia.ulb.ac.be/argos/home.php
3. http://www.swarmanoid.org
4. http://ascens-ist.eu
5. http://www.esf.org/activities/eurocores/running-programmes/eurobiosas/collaborative-research-projectscrps/h2swarm.html
6. http://www.e-swarm.org/

signed to non overlapping regions of the simulation space, resulting in an improved allocation of computational resources. Furthermore, the multi-threaded architecture of the simulator can profit of the presence of multiple core CPUs.

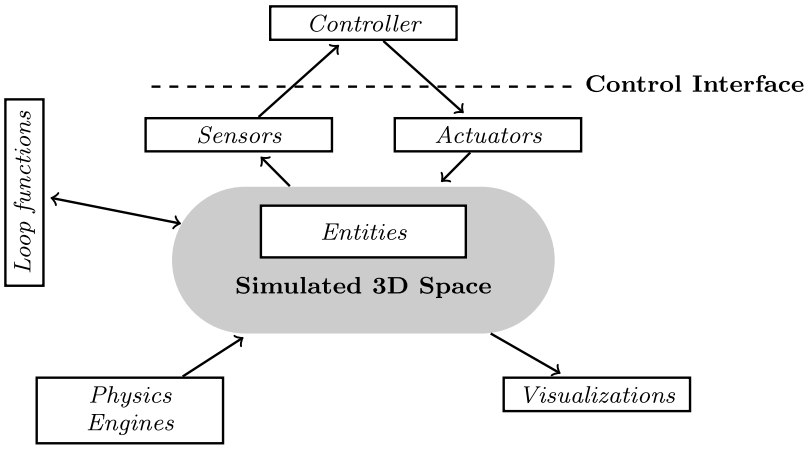


Figure 7: Architecture of ARGoS. The white boxes correspond to user-definable plug-ins.
From Pinciroli et al. [2012] in Swarm Intelligence, Volume 6, Issue 4, December 2012. Reprinted with permission from Springer Science and Business Media. All rights reserved.

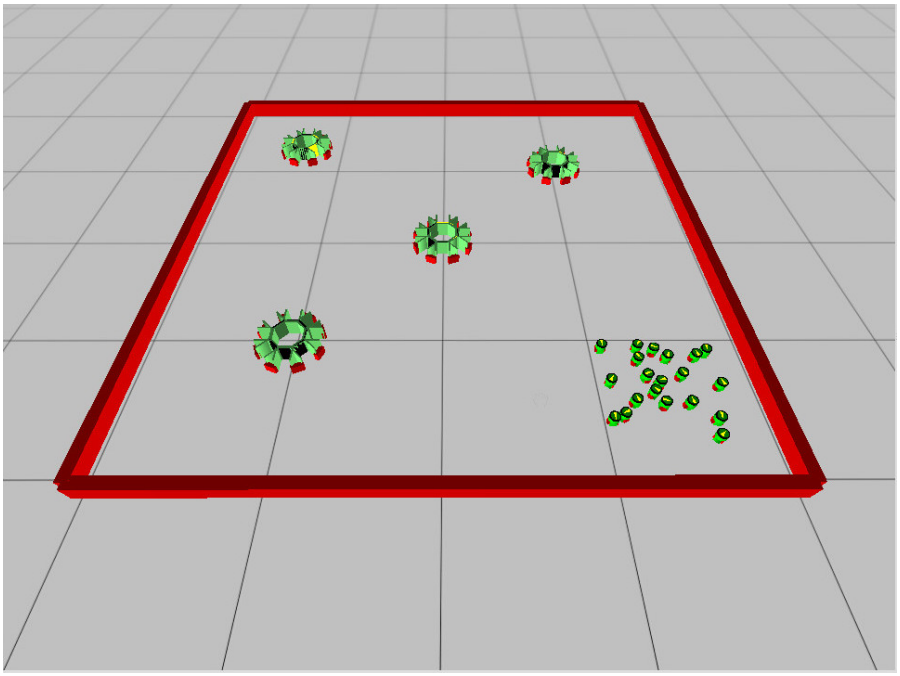


Figure 8: Screenshot of the ARGoS simulator for the scenario B environment configuration.

In addition to the aforementioned properties, we decided to use *ARGoS* since it offers full support for the *e-puck* robot (all the sensors/actuators are simulated).

In order to launch a working simulation, ARGoS requires two components to be provided: the *controllers* and the *configuration file*.

The *controllers* are plugins which make use of all the other components provided by the simulator to implement the individual behavior of an entity (*TAM* and *e-puck*, in our case). They consist of C++ source files (.cpp), with the respective headers (.h), which must be compiled prior to be dynamically loaded at runtime. The control interface to write robot controllers in *ARGoS* is common to both simulated and real robots. This means that the transition between simulated and real robots is seamless, requiring only a recompilation for a different target architecture (except for the *TAM*).

The *configuration file* is an XML file which describes the structure of the simulated environment (physics engines, placement of mobile and static objects, placement of the cameras) and which is used to map each simulated entity with the corresponding controller, giving also the possibility to pass parameters to it.

Another important functionality offered by ARGoS is the possibility to create *loop functions*. The simulation loop in *ARGoS* involves three phases: *sense and control*, *act*, *update*. In the first phase, the values of the *sensors* are read and passed to the code of the user defined *controller*, which is then executed, modifying the state of the actuators. Then, in the *act* phase, the actions stored in the actuators are executed. Lastly, in the *update phase*, the physics engine updates the state of the entities under its control.

Loop functions are user-defined functions hooks that are placed in precise points in the simulation loop (i.e. at initialization time, before and after the execution of the *update* phase). By means of this tool, the user is able to query the physics engine and to modify its state at run time. This allows us to collect data at each simulation step for further processing and visualization.

3

METHODOLOGY

The METHODOLOGY chapter contains an accurate description of the methodology that we adopted for development of our methods.

We begin by giving a formal definition to the allocation problem we are going to tackle (Problem statement).

In section Experimental setup, we describe how we have translated the problem statement into a physical implementation of the problem.

Then, in Methods, we present our contribution using a top-down approach. We start by presenting an Overview of underlying idea of the methods. Lastly, we conclude by characterizing our methods: Naive, Probabilistic and Informed.

3.1 PROBLEM STATEMENT

The class of problems we are interested in is the one related to the allocation of a swarm of robots to spatially distributed tasks.

We speak of class since there are several different features in the problem statement that may yield to different problem instances. Generally speaking, the problem can be described as:

Definition 1. *Given n robots and m tasks having a precise spatial distribution, determine a mapping of the robots to the tasks which optimizes a given allocation metric.*

Clearly, the parameters of this general definition are: the *number* of *robots*, the *number* of *tasks*, the *distribution* of *tasks* and the *allocation metric*.

Moreover, the *relation* between the *number* of *robots* and *tasks* is key factor in the problem statement, strictly related to the allocation metric.

For example, in the scenario where the number of robots is greater or equal than the number of tasks ($n \geqslant m$), if the experiment duration is sufficiently long, the allocation of all the agents will be eventually reached. There, it could be interesting to analyze the speed with which the complete allocation is achieved.

On the other hand, if the number of tasks is greater than the number of agent ($n < m$), the allocation metric could measure how evenly are the robots spread over the task in space or how fast the swarm could reach some allocation "milestones" (e.g. 25%, 50%, 75% of the tasks).

Regarding the *spatial distribution* (in a two-dimensional space) we could foresee two possible categories of problems: *random* and *deterministic* distribution.

With a *random* distribution, the positions of the tasks are determined by a bivariate random distribution (e.g. normal or uniform).

Conversely, a *deterministic* distribution consist of a precise disposition of the tasks in space, as to form a *grid* or *clusters*.

In the light of this classification, we can give a precise definition of our problem:

Definition 2. *Given* n *robots and* m *tasks* ($n < m$) *clustered in space, determine a mapping of the robots to the tasks which is as uniform as possible across clusters.*

We assume that the values of n and m remain constant for all the duration of the experiment. Furthermore, robots are assumed to be *identical* among them and tasks are considered *homogeneous* (i.e. there are no difference among tasks in terms of required skills to be performed) and *independent* of each other (i.e. there are no relations among the tasks).

The allocation of a robot to a task automatically prevents the allocation of other robots to the tasks. As a consequence, any robot could perform any available task.

In addition, the m tasks to be performed are distributed across c clusters in space. For each cluster i, we can define the *request* r_i and the *occupation* $o_i(t)$.

The *request* r_i consists of the fraction of the total number of tasks m that belongs to the cluster i.

The *occupation* $o_i(t)$ corresponds to the number of tasks of the cluster i that are being performed by a robot at time t.

Consequently, we have $m < r_i <= o_i(t), \forall\, i, t$.

Given the two measures, we can compute a third one, the *error*, which will serve as a measure for the quality of the allocation.

$$e_i(t) = r_i - o_i(t) \tag{1}$$

Indeed, if a cluster request will be completely satisfied, the error will be null. A thorough discussion of the measures of allocation quality is made in the RESULTS chapter.

3.2 EXPERIMENTAL SETUP

As described in definition 2, our problem consist in allocating robots to tasks that are distributed in space but grouped in a limited number of clusters. In our experimental setup, the tasks are represented through TAMs and a cluster consists of a circular arrangement of tasks. Moreover, the cluster is able to broadcast information concerning itself (as shown in Figure 9), namely its id i, the number of requests r_i

and its current occupation o_i, in a limited surrounding area. In the simulation, the information transfer occurs only when the robot is within a circular range (radius: 51cm) around the cluster. Although this cluster-to-robot communication is only performed in simulation, this operation could be easily implemented with real TAMs and real robots by means of a local communication device such as the Range and Bearing board by Gutiérrez et al. [2008]. The shape of the cluster has been specifically designed to give the robots the possibility to navigate around it, either to assess the cluster occupation or to direct to an available task.

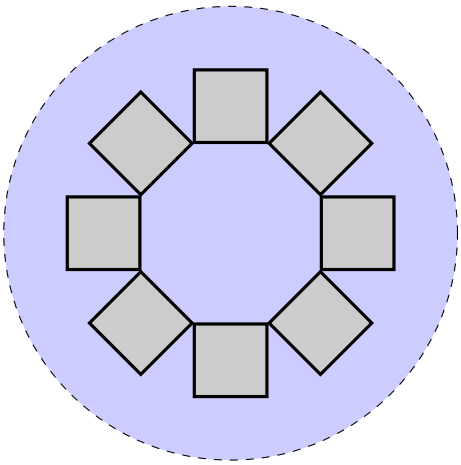


Figure 9: Schematic representation of the TAM disposition in a cluster. Each square box represent a single TAM entity. The blue circle corresponds to the area (r=51cm, in our experiments) within which the robots are able to sense informations concerning the cluster.

In our experimental setup we are going to use 4 clusters, each one composed by 8 TAMs. The requests r_i of the clusters will be distributed as follows:

CLUSTER	TAMS	REQUESTS
1	8	7
2	8	5
3	8	8
4	8	5
	TOTAL	25

Table 4: Clusters request in Uniform, Biased, Corridor experiment setups.

Given the technical equipment of the TAM (Table 3), we decided to make use of the RGB LEDs to make the internal state of the device detectable by the robots, as shown in Figure 10.

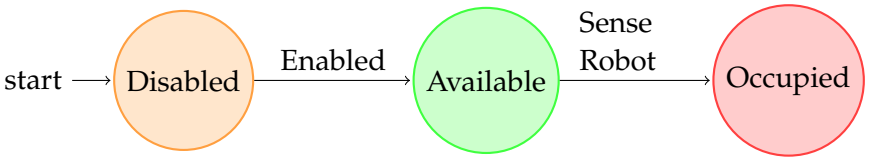


Figure 10: Finite state machine representing the TAM states. The color of the states corresponds to the actual color displayed by the TAM LEDs.

At the beginning of the experiment all the TAMs are initialized in the *Disabled* state. Then, a number of TAMs corresponding to the requests r_i defined in Table 4 is enabled in each cluster, making the device *available*. Finally, when a robot is sensed in the TAM by means of the light barrier, the device becomes *Occupied*. Clearly, in each cluster i, $8 - r_i$ cluster will remain disabled, thus unaccessible by the robots.

We decided to represent the tasks as *sporadic* (i.e. not occurring periodically), *atomic* (i.e. they cannot be suspended and later resumed) in order to test the capability of the swarm to dynamically adapt to changes in configurations instead of learning periodic patterns.

As we need to have a *number of tasks greater* than the *number of robots* (i.e. $n < m$), we will use 20 *e-pucks* randomly deployed in a predefined area of the environment. The initial position of the robots will be determined by drawing the x and y coordinates of the robot from a uniform distribution within the range depicted. It should be noted that, given their technical specifications, the chosen number of robots is not sufficient to perform a complete coverage of the environment, thus requiring the *e-pucks* to explore it.

3.3 METHODS

Unlike many methods in Swarm Robotics, ours are not inspired by any natural phenomenon. Instead, simplicity was the principle that guided our development.

We started by focusing ourselves on the less complex (i.e. Naive) method that achieved a reasonable allocation of the robots.

Once that has been found, we incrementally built a second method (Probabilistic) upon the basic one, trying to devise specific measures to overcome its limitations.

Lastly, we tried to further improve the second method by adding navigation information to it (Informed).

All the methods are based on the *sense*, *think*, *act* paradigm. At each simulation step, the robot first collects the information gathered through the *sensors*, then, according to its internal controller, deter-

mine the values to transmit to the actuators. Here, for the sake of clarity, we focus only on the *think* phase, giving a brief description of the robot controllers.

Among the different behavior-based developing approaches described in Development methodology, we decided to use the probabilistic finite state machine one. Finite state machines allow to decouple a complex problem in several easier sub-problems, which can be tackled separately and independently of each other. Furthermore, they provide an elegant and clear representation of the robot controller, in terms of internal states of the robot and transitions, based on both the state and the currently sensed values of the robot.

In order to operate correctly, all the methods require the use of a minimal set of sensors and actuators: the *wheels* to make the robots move in the environment, the *omni-directional camera* to localize the tasks, the *proximity sensors* to perform obstacle avoidance and the *range and bearing* board to receive the informations transmitted by the cluster. In addition, the Informed method requires the use of the encoder sensors on the wheels, as described in the homonym section.

3.3.1 *Overview*

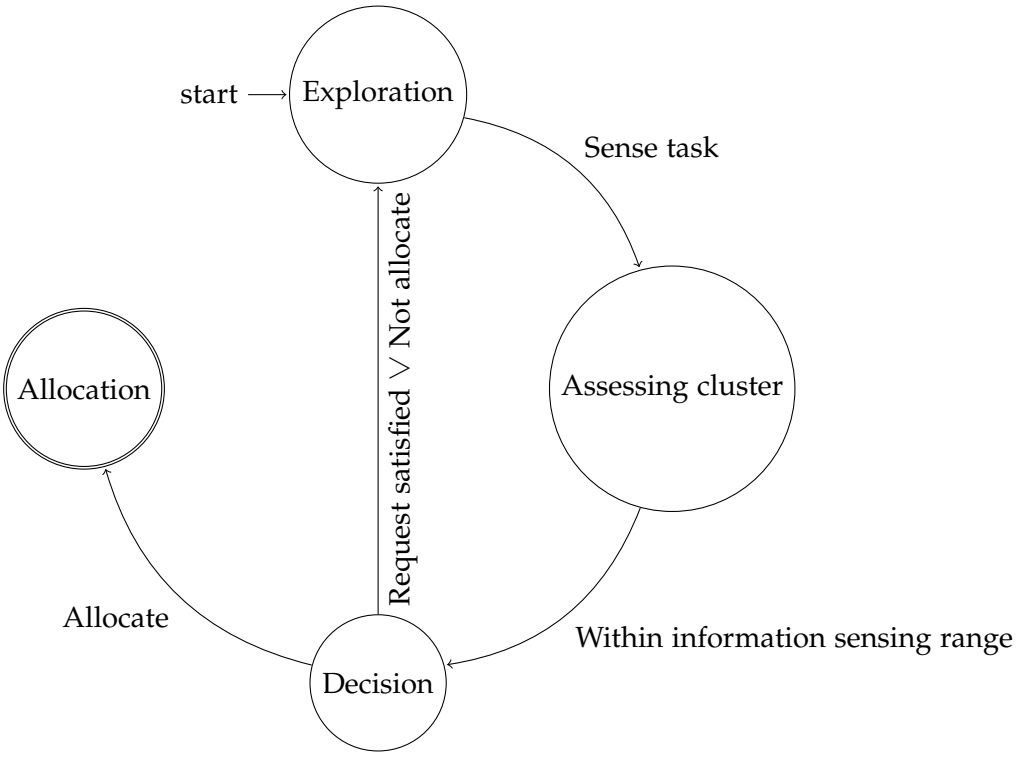


Figure 11: Finite state machine representing the *e-puck* behavioral rules

Figure 11 summarizes the high-level behavior implemented in our methods.

All the robots are initially placed in a common deployment area, where they start their *Exploration* of the environment. As soon as they visually detect (i.e *sensed*) the cluster by means of their camera, the robots direct themselves towards the cluster in order to collect information, thus performing a *cluster assessment*. When the information has been gathered, the robots enter the *decision* phase. In the case of a positive outcome of the *decision* phase, the robot decides to allocate itself to the task, thus completing its duty. Otherwise, the robots goes back to the *exploration* phase to find another available activity to perform.

As we discussed in section Spatial allocation, our problem can be decoupled into two sub-problems: *task localization* and *task allocation*.

The *task localization* problem is addressed in the *Exploration* state, whereas the *task allocation* is performed in the *Assessing cluster*, *Decision* and *Allocation* states.

Instead of implementing collective navigation techniques, such as area coverage or chain formation, we chose to perform the *Exploration* phase with a random walk.

Our choice was made taking into account the advantages offered by such a method: its simplicity, its minimal requirements in terms of sensors and computational resources and the absence of bias towards some preferred directions. The Naive and Probabilistic methods implements an uninformed version of random walk, while the Informed one makes use of odometric information to guide the exploration.

The *decision* mechanism for the allocation is the key feature that distinguishes the different methods.

The Naive method applies a greedy allocation rule: as soon as an available task has been detected, the robot tries to allocate to it.

On the other hand, the Probabilistic and Informed methods introduce an actual probabilistic decision phase, prior to the allocation.

Every time, during the assessment phase, that a change in the current cluster occupation is sensed, a stochastic decision mechanism is triggered. The robot decides whether to leave or not the cluster with a probability equal to the cluster's current relative occupation (i.e. $\frac{o_i(t)}{r_i}$). Through this simple decision rule, we would like to prevent a concentration of the robots on a single cluster and stimulate a more uniform allocation.

In addition to this probabilistic rule, in every method, the decision to leave the cluster is taken anytime a robot detects that the cluster current occupation equals the requests (i.e. $o_i(t) = r_i$ for the cluster i being assessed) or, in other words, when the cluster request is satisfied.

Whenever the decision to leave is taken, the robot enters a temporary blind state (not depicted on the state machine). The robot remains in this state for 100 time steps, during which it ignores the readings coming from the camera. This simple mechanism has been

developed to prevent a robot from being attracted by a cluster that it has just decided to leave, without resorting to more sophisticated solutions (e.g. odometry).

3.3.2 *Naive*

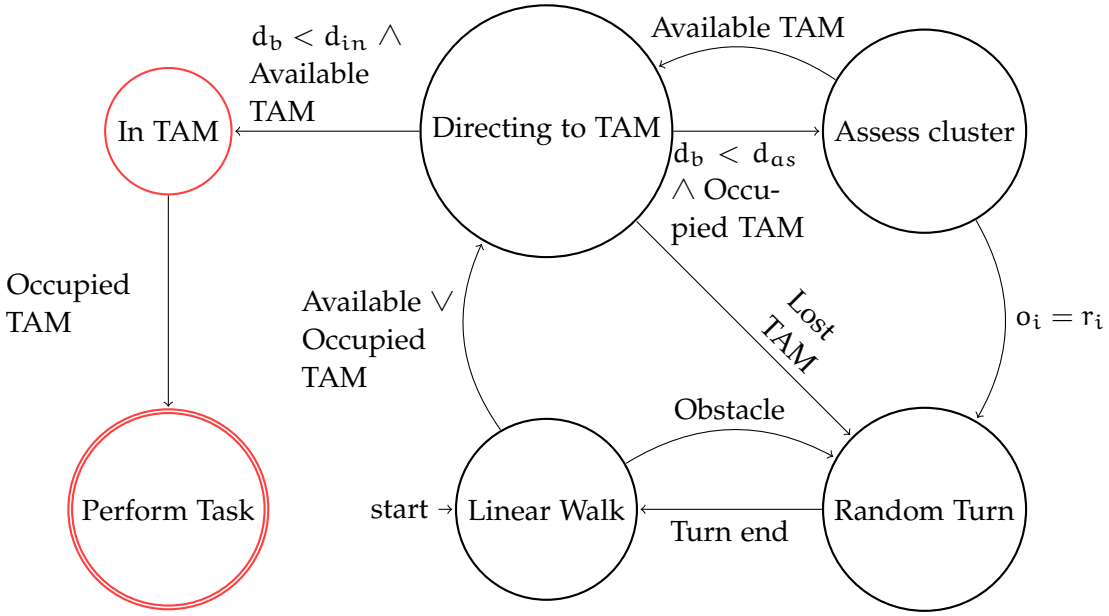


Figure 12: Deterministic finite state machine corresponding the implemented individual robot controller for the *Naive* method. The red circles corresponds to the states where the robot has light up the red LEDs on its body.

Figure 12 provides a detailed description of the robot controller for the *Naive* state, based on the high level description depicted in Figure 11.

Here, the uninformed *random walk* is performed by combining two simple behaviors: *Linear walk* and *Random turn*. *Linear walk* consists of letting the robot move in a straight line, i.e actuating the same speed on both the wheels. As soon as the proximity sensors detect the presence of an obstacle (within a range of 15 cm), the robots enter the *Random turn* state. A *random turn* is done by first choosing a rotation direction (clockwise or counterclockwise) and then pivot for a random number of time steps.

The robot is initialized in the *Linear walk* state, but as soon as a task (both available or occupied) is detected, the robot *directs* itself towards it. As explained in Section 3.2, the robot is able to determine the state of a task by perceiving the corresponding color with its omni-directional camera within a range of 50 cm. In our simulation, the resulting readings from the omni-directional camera are colored points, characterized by their color, their distance and angle

with respect to the direction where the robot is heading. Thanks to this information, the robot can easily rotate and head in the same direction as the sensed TAM. As soon as an enabled TAM, regardless of its internal state, is detected by means of the camera, the robots enters the *Directing to TAM* state, and starts moving in the direction of the perceived color point. The sensed TAM could be either *available* or already *occupied*. Since the *Naive* allocation rule is *greedy*, if the abstracted task is available, the robot directs towards it.

When the distance of the color point d_b it perceives is smaller than the TAM depth (d_{in}=10.83 cm) the robot lights up the red LEDs on its body to prevent other robots from allocating to the same task and enters the *In TAM* state, stopping inside the TAM. Once the TAM has detected the presence of the *e-puck* by means of its light barrier, it signals the change in its internal state by changing the color of its RGB LEDs to red. This operation consists indeed, in the abstraction of the activity that the *e-puck* should perform. In response to this state transition, the robot moves from the *In TAM* state to the the final state *Perform Task*. From the *e-puck* point of view, in our experimental setup, the abstraction of a task consists of remaining idle inside the TAM.

On the other hand, if the sensed TAM is unavailable, while being in the *Directing to TAM* state, the *e-puck* moves in the direction of the cluster to be able to *assess* its occupation. When the robot arrives closer to the TAM then the assessing distance threshold (i.e. when the distance of the perceived color point d_b is smaller than $d_{as} = 25$cm), the robot's internal state changes to *Assessing Cluster*. The assessment phase has two possible outcomes: either there are still tasks available in the cluster or the cluster request have been satisfied.

In the case of the presence of available TAMs, the robot starts navigating around the cluster, performing a circular motion, remaining inside the assessing range, until it perceives a green color point, corresponding to the *available* task. Then he moves to the *Directing to TAM* state, eventually entering the TAM as described above.

Otherwise, if the cluster request have been satisfied (i.e. $o_i(t) = r_i$), the *e-puck* performs a random turn and restart the exploration of the environment, before moving back to the *Linear Walk* state.

3.3.3 *Probabilistic*

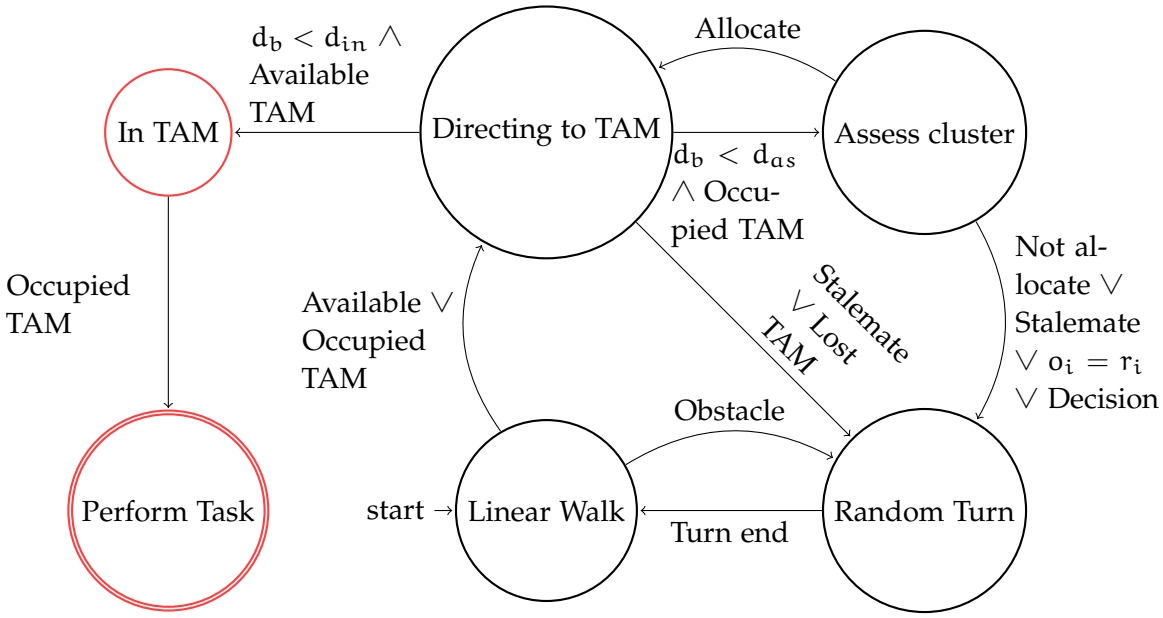


Figure 13: Probabilistic finite state machine corresponding the implemented individual robot controller for the *Probabilistic* method.
The red circles corresponds to the states where the robot has turned on the red LEDs on its body.
Allocate and *Not Allocate* transitions represent the two possible outcomes of the probabilistic decision.

Since the *Probabilistic* method (Figure 13) is incrementally built upon the *Naive* one, the general dynamic of the controller is similar to what has been explained in the Naive section. The behavior of the robot is identical to that of the *Naive* method in the *In TAM*, *Perform Task*, *Linear walk* and *Random turn*.

The *exploration* technique is indeed an uninformed random walk.

The *allocation* rule, on the other hand, is modified in order to address the two main issues of the *naive* approach: the occurrence of a *stalemate* and the *lack* of an allocation rule to obtain a more *uniform* task allocation.

A *stalemate* may occur whenever two robots decide to allocate themselves to the same task. In that case, both the robots will start moving towards the task. If the robots would arrive closer enough (around 30cm) to the TAM at the same time, they will start trying to avoid each other. Since there is no direct communication among the robots, there is no possibility of an explicit agreement on which robot should perform the task.

Our solution for this issue has been developed by implementing a counter which is started once the robot enters the *Directing to TAM* or *Assessing Cluster* states and incremented each simulation time step. When the counter surpass a predefined threshold (100 time steps, in

our implementation), the robots decide stochastically whether to wait or leave the cluster. The probability to leave (p_l), by performing a *random turn* and starting a blind exploration, is equal to 0.01. The decision process is then repeated at each time step, until either one of the robots decides to leave before the other, thus allowing the remaining one to allocate itself to the task, or both will leave the cluster.

On the other hand, the *greedy* allocation rule of the *Naive* method is substituted by a probabilistic one.

In the *Probabilistic* method, a robot in the *Linear walk* state still heads towards a TAM as soon as it has perceived it, regardless of its state, but behaves differently during the *assessment* phase.

In fact, if the perceived task is directly available, the robot will immediately try to allocate itself to it.

Otherwise, if the robot will enter the *Assessing Cluster* state, the probabilistic decision will be triggered. Here, the robot will either decide to *allocate* or to *not allocate*.

In the first case, in the same way as in the *Naive* method, the robot will turn around the cluster until the first available task is found.

In the second one, the robot will move to the *Random turn* state, perform a random turn and start a temporary blind exploration.

Moreover, the *decision* phase will occur whenever a change in the occupation of the currently assessed cluster is sensed.

The idea of introducing a stochastic component in the decision rule arises from the objective of achieving a uniform allocation of the robots across the cluster. One way of doing so, is allowing the robots to move from already crowded clusters to those that are still almost empty, thus balancing the occupation among them. With this idea in mind, we tried to devise a new allocation rule.

A deterministic rule has been immediately discarded since it lacked of flexibility with respect to differences in the size of the cluster or the number of robots. If the rule would have been based on an absolute occupation threshold (e.g. leave the cluster if $o_i(t) > X_i$), it would have required a global knowledge of the environment, in order to determine a priori the optimal values to achieve a uniform allocation. With a rule based on relative occupation (e.g. leave the cluster if $\frac{o_i(t)}{r_i} > X_i$)), in addition to the global knowledge requirement, once the threshold would have been met in all the clusters, it would not have been possible to allocate the remaining robots.

Thus, we decided to implement a probabilistic rule based on a simple intuition: limiting the allocation of robots to cluster whose occupation is already high. In order to do so, we propose an abandon probability, computed every time that a robot enters the *decision* phase. The abandon probability a_i for a certain robot assessing cluster i, at time t is defined as:

$$a_i(t) = \frac{o_i(t)}{r_i} \tag{2}$$

The occupation o_i is normalized by the cluster request in order to obtain a value in the $[0, 1]$ range.

It should be noted that: the higher the number of robots currently being allocated to tasks belonging to the cluster, the higher the likelihood of leaving the cluster.

The advantages of this allocation rule is that it is *completely distributed*, with *minimal* requirements in terms of communication (i.e. only the information on the occupation $o_i(t)$ must be transferred to the robot) and computational capabilities and *flexible* with respect to the different requests of the clusters.

3.3.4 *Informed*

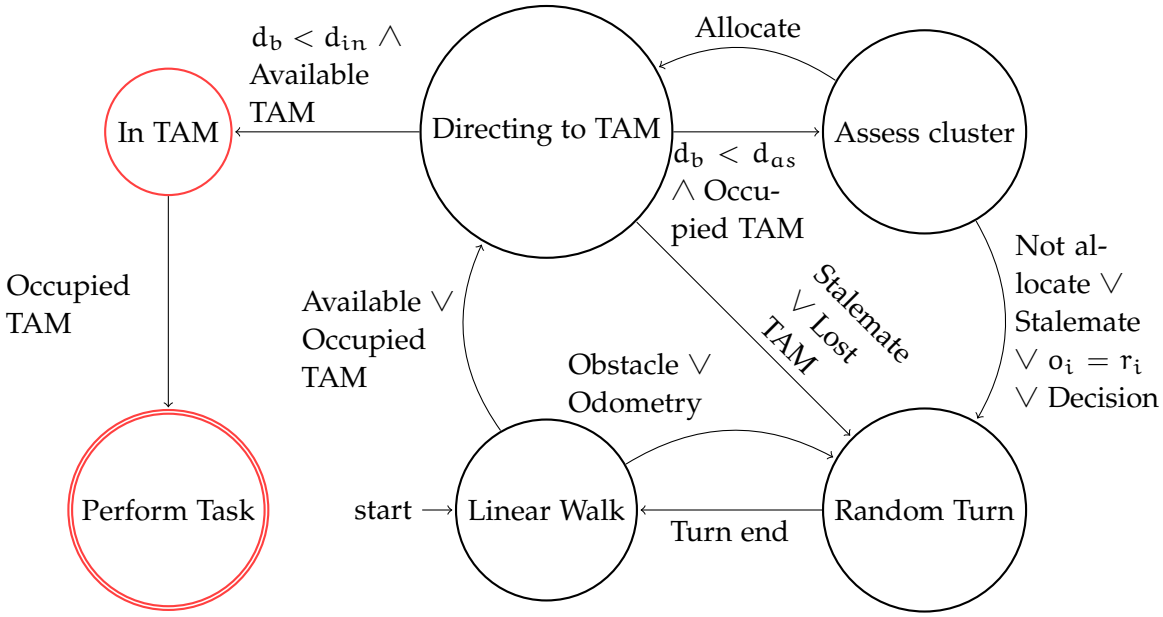


Figure 14: Probabilistic finite state machine corresponding the implemented individual robot controller for the *Informed* method.The red circles corresponds to the states where the robot has light up the red LEDs on its body.

The *Informed* method is built upon the *Probabilistic* one, hence the behavior of the robots in all the states is the same as described in section Probabilistic, including the mechanism to solve the stalemate issue and the probabilistic allocation rule. With the *uniform allocation* problem being tackled with the probabilistic rule introduced in Equation 3.3.3, the only difference with the previous method is how the *exploration* is performed. While visualizing the first experiments with the *Probabilistic* method, we discovered a minor issue in the exploration phase. In fact, every time a robot decides to leave a cluster, after having completed the blind exploration phase, there is no guarantee that the it will actually direct towards another cluster without

coming back to the previously left one. Indeed, a robot may assess several times the cluster it has just left due to the lack of available tasks, thus having a redundant and inefficient behavior. This could happen as a consequence of the obstacle avoidance with respect to other robots or the arena walls. Given the sensory equipment of the *e-puck*, we decided to propose a solution to this issue using the odometry. Odometry is a technique to estimate the change of position of the robot with respect to a known position, using moving sensors. In our solution, as soon as a robot decides to leave a cluster, it starts keeping track of the position of the cluster $\mathbf{p}$. This relative localization with respect to the robot is then update at each time step with the information coming from the sensors.

A detailed explanation of the model can be found in Lucas [2001]

Odometry is required since the displacement and the rotation of the robot cause its reference frame to move and rotate accordingly. Thus, if a position of a fixed point is not translated into the new reference frame, its exact position could not be tracked anymore. Through the readings of the encoder sensors mounted on the wheels it is possible to determine the distance traveled by each wheel of the robot d_l and d_r during each simulation step. Since the development of our methods has been performed in simulation, there is no error in the readings coming from the sensors. However, the readings from the sensors on real robots are perturbed by noise. Here, we propose to model the noise as an additive gaussian noise with mean 0 and standard deviation 0.2. Thanks to these values, knowing the inter-wheel distance of the robot d_{iw}, it is possible to estimate the displacement d_d and rotation ϑ of the robots:

$$\begin{aligned} d_d &= \frac{d_l + d_r}{2} \\ \vartheta_d &= \frac{d_l - d_r}{d_{iw}} \end{aligned} \tag{3}$$

The displacement magnitude and angle can be combined to form a displacement vector: .

The vectors are expressed in polar coordinates, in the form (magnitude,angle) *or, equally* (r, ϑ)

$$\mathbf{d} = (d_d, \vartheta_d) \tag{4}$$

The vector is than used to perform the roto-translation required to correctly update of the stored position of the previous cluster $\mathbf{p}$:

$$\begin{aligned} \mathbf{p} &= \mathbf{x} + \mathbf{d} \\ \mathbf{p} &= \begin{bmatrix} \cos(\vartheta_d) & -\sin(\vartheta_d) \\ \sin(\vartheta_d) & \cos(\vartheta_d) \end{bmatrix} \cdot \mathbf{p} = (d_p, \vartheta_p) \end{aligned} \tag{5}$$

By performing the sequence of operations described in Equations 3, 4, 5 at each time step it is possible to maintain a reasonable estimate of the position of the most recently left cluster. We speak of "a

METHOD	EXPLORATION	DECISION RULE (AT TIME t^*)
Naive	Uninformed random walk	Greedy. Leave if at time t^*, $r_i(t^*) = o_i(t^*)$.
Probabilistic	Uninformed random walk	Probabilistic with abandon probability $a_i(t^*) = \frac{o_i(t^*)}{r_i(t^*)}$. Probabilistic stalemate prevention rule. [1] Leave if at time t^*, $r_i(t^*) = o_i(t^*)$.
Informed	Informed random walk using odometry	Probabilistic with abandon probability $a_i(t^*) = \frac{o_i(t^*)}{r_i(t^*)}$. Probabilistic stalemate prevention rule. [2] Leave if at time t^*, $r_i(t^*) = o_i(t^*)$

Table 5: Overview of the developed methods.

reasonable estimate" since the presence of the error in the readings does not allow to precisely track the position $\mathbf{p}$ of the cluster.

This information is used in the *Linear walk* state to perform an informed random walk. In fact, whenever the cluster estimated orientation ϑ_p is included in the range $(h - \frac{\pi}{6}, h + \frac{\pi}{6})$, with h being the angle of the direction where the robot is currently heading and the cluster estimated distance d_p is smaller than two times the omnidirectional camera range (i.e. $1m$), the robot performs a change of direction.

The change of direction (marked with *Odometry* in Figure 14) is implemented by changing the state of the robot to *Random Turn*.

We believe that, even though our solution could not completely profit from the advantages brought by the use of an exact odometry, the increased occurrence of direction changes due to this mechanism will result in a better exploration of the environment and possibly, in a more even distribution of the robots across clusters.

3.3.5 *Summary*

1. *Probabilistic stalemate prevention rule:* After the 100^{th} time step spent in directing state, decide every time step whether to leave with probability $p = 0.01$.
2. See 1.

Our starting point was the problem of uniformly allocating robots to spatially distributed tasks (cf. Definition 2)

We clarified this statement by fixing constraints on the number of agents (20), the number of tasks (25) and the arrangement of the tasks (*4 clusters*).

From this statement, we devised a physical implementation of the tasks, the TAM and their arrangement to form clusters (Figure 9).

Once having defined a concrete setup for the problem, we began the development of our methods by means of an iterative process based on simulations.

We developed three robot controllers, presented here as finite state machines, that tackles separately the two sub-problems that characterizes our definition of the problem : *exploration* and *allocation*.

A summary of the relevant features of the solutions we implemented in our methods to tackle these problem is presented in Table 5.

In chapter 4 we present the results we obtained by implementing the the same controller on all the robots in the swarm and launching simulations of 1000s (10000 time steps) each.

4

RESULTS

The RESULTS chapter is dedicated to the discussion of the results we obtained through the application of our method to the Experimental setup presented in the homonym section.

We start by describing the different scenarios (Section 4.1) that characterizes our experimental setup: scenario Uniform (Section 4.1.1), Biased (Section 4.1.2), Corridor (Section 4.1.3).

Then, we present the Metrics that we have devised to evaluate the three most important properties of our methods: *Allocation uniformity*,*Allocation speed*.

The core of the chapter is represented by sections Allocation uniformity and Allocation speed, devoted to a thorough analysis of the aforementioned properties.

We conclude by evaluating the difficulty of the scenario (Section 4.5).

4.1 SCENARIOS

The problem statement (cf. Definition 2) gives us several degrees of freedom in the definition of an experimental setup. In section Experimental setup, we clarified the number of robots, the number of tasks and the presence of aggregates of clusters. However, the distribution of these groups of tasks in space and the initial distribution of robots has not been defined. By varying these two aspects, we devised three different scenarios. The purpose of this choice is to test the performances of the methods in different environments, in order to determine the weaknesses and the strong points of each method.

4.1.1 *Uniform*

The notion of difficulty of a scenario will be clarified in Metrics

The scenario Uniform (Figure 15) is characterized by having the deployment area in the center of the arena, thus being equally distant from all the clusters. This is the first scenario we developed and the simplest one, which will serve as a baseline for comparison with the other ones.

4.1.2 *Biased*

In the scenario Biased (Figure 16), the deployment area is moved in the bottom left corner of the arena and cluster 2 is placed near the opposite corner of the environment. This results in having three

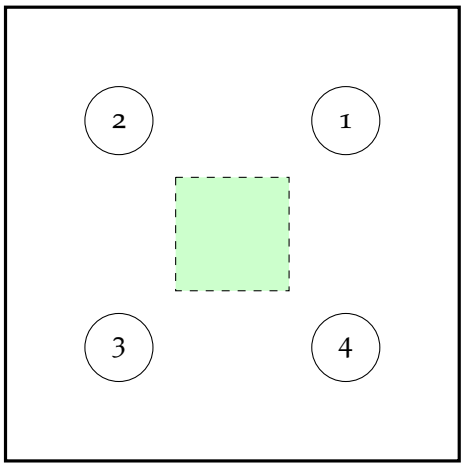


Figure 15: Representation of the cluster disposition in scenario Uniform.
Clusters are represented by circles, while the dashed rectangle indicates the robot deployment area.
The arena size is 4m x 4m, while the deployment area is 1m x 1m.

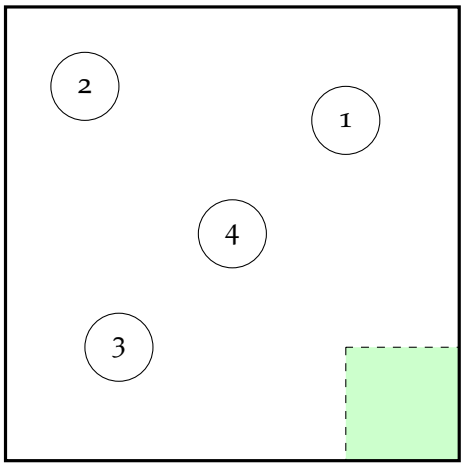


Figure 16: Representation of the cluster disposition in scenario Biased.
Clusters are represented by circles, while the dashed rectangle indicates the robot deployment area.
The arena size is 4m x 4m, while the deployment area is 1m x 1m.

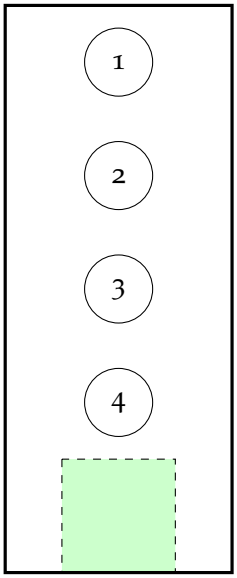


Figure 17: Representation of the cluster disposition in scenario Corridor. Clusters are represented by circles, while the dashed rectangle indicates the robot deployment area.
The arena size is 2m x 5m, while the deployment area is 1m x 1m.

clusters closer to the deployment site and the fourth one far away. We introduced this bias in the environment in order to test the influence of a non-uniform positioning of the clusters to the performances of the method.

4.1.3 *Corridor*

In scenario Corridor (Figure 17) we placed the clusters in a narrow rectangular arena, closely resembling a corridor. The clusters are equally spaced among them and the deployment area is located at one edge of the corridor. The motivation of the scenario Corridor is to test the impact of inter-robot interference on the performance of the methods. In fact, the environment presents a limited space around the clusters, due to the proximity of the walls and the cluster themselves, which will cause the robots to aggregate and potentially interfere with each other.

4.2 METRICS

4.2.1 *Allocated robots*

Since we are dealing with methods to allocate robots to tasks, the most intuitive aspects we can analyze are, respectively: the number of robots that could be actually allocated and the time required to

achieve such allocation. By indicating with $s_i(t)$ the internal state of robot i at time t, we can define the number of allocated robot at each time step t as:

$$R(t) = \sum_{i=1}^{20} (s_i(t) == \textit{Perform Task}) \tag{6}$$

We analyze this metric under two distinct points of view. On one hand, we look at the number of robots allocated at each time step, analyzing, for instance the maximum and minimum number of robots allocated at a given time step t across a set of trials. On the other hand, by looking at the evolution in time of this allocation function, we can easily compare the performances of the methods.

4.2.2 *Maximum error*

In section Problem statement, we defined the allocation error $e_i(t)$ for each cluster as the difference between the desired number of robots in the cluster (i.e. the request r_i) and the current number of robots in the cluster (i.e. the occupation $o_i(t)$). This measure easily allow to understand and visualize how well the robots are allocated on a certain cluster.

Our goal, on the other hand, is to distribute the robots uniformly across the clusters, hence this value is of a little interest to us. Nevertheless, by computing an aggregate measure from all the clusters, we could have a better view on the overall allocation.

Hence, we compute the maximum error across cluster:

$$e_{max}(t) = \max_{i \in \{1,\cdots,4\}} e_i(t) = \max_{i \in \{1,\cdots,4\}} (r_i - o_i(t)) \tag{7}$$

Indeed, the value of $e_{max}(t)$ corresponds to the greatest difference between request and occupation in all the clusters. Moreover, a uniform allocation of the robots is the one that better distributes the robot across the clusters. This entails that, in case of a uniform allocation, the error $e_i(t)$ in each cluster will not be null, but the maximum error will be reduced, since the fair distribution of the robot will reduce the differences $r_i - o_i(t)$ in each cluster.

The notion of fair distribution is applied to determine the lower bound for this metric, the optimal allocation error e_{opt}:

$$e_{opt} = \lfloor \frac{m-n}{C} \rfloor + 1 \tag{8}$$

Here, m corresponds to the global number of tasks, n is the number of robots and C the number of clusters in the environment. The optimal allocation error e_{opt} corresponds to the value of $e_{max}(t)$ that would be obtained by allocating iteratively one robot per cluster until their complete allocation.

4.2.3 *Allocation levels*

Another way to measure how evenly are the robots distributed across cluster is to determine if certain allocation levels are reached. The concept of allocation level is defined with respect to the relative occupation $\frac{o_i(t)}{r_i}$ of each cluster. As a matter of fact, if the relative occupation of all the clusters is greater than a certain value x, we could say that the allocation level x has been reached. Whenever this occurs, we are also able to determine the time step o at which the level is attained:

$\bigwedge_i$ corresponds to the logical AND operation extended to every cluster i.

$$o_x = \arg\min_t \Big(\bigwedge_{i=1}^{4} \frac{o_i(t)}{r_i} \geqslant x \Big) \qquad x \in 0.25, 0.50 \tag{9}$$

o_x corresponds to the first time step at which the relative allocation of all the clusters is above the desired threshold x.

The values of x are selected in order to have representative and feasible measures for the cluster allocation. For this reason, given the distribution of requests r_i and the optimal allocation error e_{opt}, the value of 0.75 has been discarded since it was incoherent with respect to the notion of fair allocation (i.e. the relative allocations of each cluster in the case of a fair allocation are not guaranteed to be above the aforementioned threshold). It should be noted that, given the previous definition of fair allocation, it is required to attain at least a relative occupation of all the clusters greater or equal than 0.5, in order to achieve the optimal allocation error e_{max}.

4.2.4 *Cluster views*

In section Scenarios we presented the developed scenarios and the motivations behind their structure.

What we actually did, at design time, was formulating *hypothesis* on how the structure of the environment could potentially affect the performances of the methods.

In order to *test* our assumptions, we decided to measure the number of times that cluster i has been seen at time t:

The cluster-robot distance is measured from the robot to the closest TAM belonging to cluster i.

$$v_i(t) = \sum_{s=0}^{t} \sum_{k=1}^{20} (d_{ik}(s) \leqslant c_r) \tag{10}$$

Here, $d_{ik}(s)$ represents the distance of robot k from cluster i at time s, while c_r corresponds the range of the omni-directional camera (50 cm in our experiments). Through the analysis of the number of visits, we expect to assess whether the number of visits to a certain cluster is biased by the nature of the scenario and whether the differences in the methods have an impact on the exploration of the environment.

4.3 ALLOCATION UNIFORMITY

The main goal of our methods is to diffuse the robots evenly across the spatially distributed clusters.

In order to achieve these results, we improved the *Naive* method by introducing probabilistic mechanisms (*Probabilistic*) and informed decisions *Informed*) to favor the redistribution of the robots from crowded clusters to the empty ones.

Here, we analyze the performances of the three methods, by focusing on the final allocation achieved by each of them on a set of 50 simulations. Each simulation is characterized by a seed value, which will be used to initialize the simulator's internal random number generator, thus influencing the stochastic behavior of the method. As a matter of fact, the seed will influence the initial robot placement and orientation and all the probabilistic decisions made by the methods. We run each simulation for 1000 s, corresponding to 10000 simulated time steps, using the same set of 50 seeds for all the different methods, on the same scenario.

In order to evaluate the final allocation, we compute the maximum error across cluster $e_{max}(t)$ and the number of allocated robots $R(t)$ for $t = 10000$. Instead of presenting the whole distribution of values, we decided to aggregate the values using non-parametric statistics, namely median value and interquartile range. Moreover, since we are dealing with integer values (e.g. number of tasks or robots) the computation of some statistics (e.g. mean) could results in decimal values having no real, physical meaning.

The results are shown in Tables 6 and 7.

4.3.1 *Maximum error*

METHOD	SCENARIO					
	UNIFORM		BIASED		CORRIDOR	
STATISTIC	Median	(q_{25}, q_{75})	Median	(q_{25}, q_{75})	Median	(q_{25}, q_{75})
NAIVE	3	(3,4)	4	(4,5)	6	(6,6)
PROBABILISTIC	3	(3,3)	3	(3,4)	5	(4,5)
INFORMED	3	(3,3)	3	(3,3)	5	(4,5)

Table 6: Summary of the values of the maximum allocation error $e_{max}(t)$ at $t = 10000$ for a swarm of 20 *e-pucks* with 25 available tasks.
Median values and corresponding inter-quartile ranges are computed across 50 trials of 10000 time steps each.
The optimal allocation error e_{opt} is equal to 2.

Table 6 reports the median maximum error across the clusters at the end of the experiment.

The inter-quartile range is added in order to give informations concerning the dispersion of the values (by definition, half of the values of the sample are included in the inter-quartile range).

At a first glance we can observe that no method is able to achieve the optimal allocation error e_{max} of 2 tasks, but on scenario Uniform and scenario Biased the *Probabilistic* and *Informed* methods are able to reach an error of 3 tasks, with no variability.

Furthermore, by comparing the methods we are able to see that the enhanced methods (i.e. *Probabilistic* and *Informed*) have similar performances and both clearly outperform the *Naive* one.

Also, by looking at the different scenarios, we can see an increase in the values of the median maximum error and in its variability, indicating a growing complexity of the environments.

4.3.2 *Allocated robots*

METHOD	SCENARIO					
	UNIFORM		BIASED		CORRIDOR	
STATISTIC	Median	Range	Median	Range	Median	Range
NAIVE	20	(12,20)	20	(9,20)	19	(6,20)
PROBABILISTIC	20	(20,20)	20	(20,20)	20	(18,20)
INFORMED	20	(20,20)	20	(20,20)	20	(18,20)

Table 7: Summary of the values of the number of allocated robots $R(t)$ at $t = 10000$ for a swarm of 20 *e-pucks* with 25 available tasks. Median values and corresponding ranges (min,max) are computed across 50 trials of 10000 time steps each.

Table 7 offers a complementary vision on the final allocation, displaying the median number of allocated robots.

Here, instead of focusing on the inter-quartile range, the whole range of the 50 sampled values is presented, in order to verify whether all the methods are able to allocate all the robots or some problems arise.

As for the median values, all the methods are able to achieve the complete allocation of the robot, except the *Naive* method on *scenario Corridor*.

The same trend described in Maximum error can be observed here: there is a clear difference between the *Probabilistic* and *Informed* methods' performances and the *Naive* one.

The remarkably low minimum number of allocated robots of the *Naive* method can be explained through the absence of the *stalemate* check. In fact, since no scenario presents obstacles or considerably narrow passages that could cause the robots to get stuck in them, we could safely suppose that after a sufficiently long time all the robots will eventually be allocated. Moreover, the same figures for the minimum numbers of allocated robots are not present in the *Probabilistic* and *Informed* method, where the *stalemate* condition is used.

Another view of the data concerning the number of allocated robot $R(t)$ can be found in the annex, section A.2.

4.3.3 *Allocation levels*

METHOD	SCENARIO					
	UNIFORM		BIASED		CORRIDOR	
LEVEL	0.25	0.50	0.25	0.50	0.25	0.50
NAIVE	0.900	0.660	0.240	0.020	0.200	0.020
PROBABILISTIC	1.000	0.800	0.800	0.440	0.800	0.040
INFORMED	1.000	0.740	0.900	0.680	0.800	0.060

Table 8: Summary of the probabilities to reach allocation levels o_{25} and o_{50} across 50 trials within 10000 time steps for a swarm of 20 *e-pucks* with 25 available tasks.

Table 8 summarizes the probabilities to reach the allocation levels 25% and 50% within the chosen experiment duration (10000 time steps). The analysis of this results allows us to highlight the limitations of the proposed methods. In fact, no method is able to achieve high probabilities for the allocation level 50% (necessary condition for an optimal allocation), on all the scenarios. Nevertheless, the *Informed* and *Probabilistic* methods are able to successfully achieve the 25% allocation level on all the trials, and the 50% level in a relevant number of simulations.

The *Naive* method also have fairly good performances on the *scenario Uniform*, but the probabilities have a significant drop on scenarios *Biased* and *Corridor*. We suppose that this variation is determined by the combination of the biased placement of the clusters across the *scenario Biased* and the greedy allocation rule of the *Naive* experiment, which hinder the possibility of having a uniform allocation. Another possible explanation for the performances of the *Naive* method on scenario Corridor, especially for the allocation level 25%, can be given by looking at Table 10. In case *Naive-scenario Corridor* we can observe a high variability on the number of allocated robots, reaching a min-

imal value of 6 units. By referring to Table 4, we can observe that this value is smaller than the number of requests of a single cluster, thus making the likelihood of a uniform allocation across the clusters really low, using a greedy approach.

It should be noted that the performances of the other two methods are also affected by the change in scenarios. However, the impact of this change is less dramatic than the *Naive* one, with the *Corridor* still being the most difficult to tackle (i.e. the one having the smallest values for the cumulated probabilities of o_{25} and o_{50}).

Again, by comparing the performances of the different methods, we notice the similarity between the *Probabilistic* and *Informed* methods' results. The only remarkable difference is the higher value for the probability of reaching 50% allocation in scenario *B*. This result can be related with the use of the *odometry* in a reasonably large environment, as the one depicted in Figure 16. The wider nature of this environment limits the number of encounters among the robots, after the initial deployment phase. Considering our implementation of the *random walk* in both the *Probabilistic* and *Informed* methods, less encounters between the robots implies a reduced number of random turns, which in turn limits the exploration of the environment. Due to the *odometry*-based change of direction, to avoid returning to already visited clusters, the *Informed* method presents an higher probability to make random turns, thus it should better explore the environment and potentially achieve a more uniform distribution.

4.4 ALLOCATION SPEED

The Allocation uniformity section has been devoted to the evaluation of the uniformity of the allocation across clusters, mainly through the analysis of the final results achieved by the methods.

Another interesting point-of-view on our data can be given by analyzing how the methods have achieved these results, instead of restraining ourselves to the final values of the proposed metrics.

The simulation setup is the same as above, 50 simulations of each method, on each scenario of 10000 time steps each (1000 s), using the same set of seeds on each scenario to ensure the same initial condition to all the methods.

We try to gather information on the speed of the allocation by looking at the evolution of the maximum error across clusters $e_{max}(t)$ and the number of allocated robots $R(t)$ for $t \in \{1, \cdots, 10000\}$. The two metrics allows us to focus on both the *uniformity* aspect and the allocation *velocity* one. On one hand, the evolution of $e_{max}(t)$ allows us to understand how much time do the methods require to bring the overall error across clusters below a certain threshold, explaining also how uniformly are the robots distributed across clusters at a certain moment in time. On the other hand, $R(t)$, captures how many robots

are allocated to a task at a certain time step, making no distinction among clusters. Through the comparison of the number of allocated robots for the different methods, we are able to evaluate how fast is each method in assigning the robots to the tasks. In addition, we analyze the empirical cumulative distribution functions of the times needed to reach the allocation levels 25% and 50% (o_{25} and o_{50}) in order to have a different standpoint on the *uniformity* of the allocation.

As for the $e_{max}(t)$ and $R(t)$, side by side comparisons of the plots including all the quantiles and the mean values can be found in the annex, chapter A.

4.4.1 *Scenario Uniform*

By looking at Figure 18, we can see that, considering the median number of allocated robots, *Naive* method is able to achieve the fastest allocation in *scenario Uniform* (Figure 18b), with all the the 20 robots successfully performing a task in less than 1500 time steps. Up to 500 time steps, the *Probabilistic* and *Informed* curves are paired but eventually, the *Probabilistic* method achieves a faster allocation than the *Informed* one.

However, the fastest allocation is not necessarily the fairest. As we see in Figure 18a, the *Naive* method is also the one presenting the highest median maximum error across clusters, while the plots of the *Probabilistic* and *Improved* methods are superposed. The analysis of the empirical cumulative distribution functions (Figure 19) confirms this trade-off between speed and quality of the allocation.

A greedy method (i.e. *Naive*) is the one ensuring the fastest allocation of the robots to the tasks, since the robots spend less time exploring the environment. Nevertheless, given the structure of *scenario Uniform* (Figure 15), with a central deployment area, equally distant from all the cluster and the uniform distribution of the robot at the beginning of the experiment, relatively high probabilities of satisfying at least the 50% of the occupation of the cluster (i.e. o_{50}) can still be obtained. It should be noted that, by looking at the *uniformity* of the allocation, even on the simplest scenario, methods *Probabilistic* and *Informed* attain a smaller value of e_{max} faster than the *Naive* one, but finally, all the curves converge to the value of 3.

4.4.2 *Scenario Biased*

Figure 20 gives additional evidence in support of the hypothesis of the existence of a trade-off between *speed* and *allocation quality*.

Again, the *Naive* method represents the upper bound in terms of allocation velocity while being the lower bound in terms of maximum median allocation error across cluster.

By looking at Figures 20a and 20b, no remarkable differences can be seen concerning the performances of *Probabilistic* and *Informed* methods.

With respect to *scenario Uniform* (Figure 18), we can observe that, in *scenario Biased*, more time is required to reach the same levels of robots allocation error. We explain this shift in the curves with the differences in the nature of the environment, more precisely the placement of cluster 2. While in *scenario Uniform* it is as distant as all the other clusters from the deployment area, in *scenario Biased* it is the farthest one. As a consequence, more time is required to navigate through the arena to actually reach tasks belonging to the second cluster, before eventually deciding to allocate to them.

In addition, we suppose that the bias in the positioning of the clusters could be a potential cause of the highest median maximum error across cluster e_{max} reached by the *Naive* method at the end of the experiment.

In Figure 21, we can observe that the *Naive* method is able to reach the 25% and 50% allocation levels in a smaller number of trials with respect to the other methods. Moreover, whenever those levels are reached, the time required to reach these allocation thresholds is higher than both the *Probabilistic* and *Informed* methods.

Thus, we can conclude that the performances of the *Naive* method are definitely worse than those of the enhanced methods on *scenario Biased*.

4.4.3 *Scenario Corridor*

Figure 22 highlights the relative improvements the introduction of probabilistic mechanisms and the use of odometry have brought to the *Naive* method.

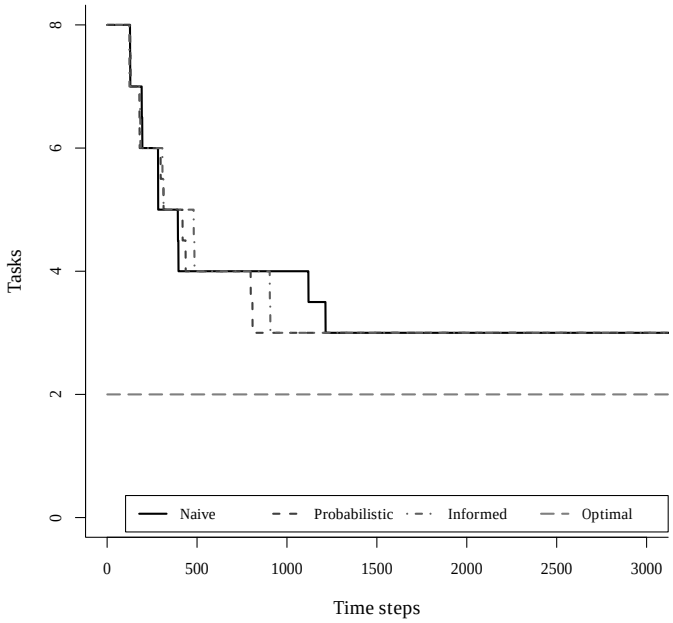
In Figure 22b we can see that the *Naive* method is slower than the *Probabilistic* and *Informed* one and it is not able to reach the complete allocation of the robots to the tasks. On the contrary, the enhanced methods are able to attain this objective, with the *Informed* one being the fastest.

Although there is an improvement in the velocity of the allocation, it should be noted that the median maximum error across clusters, even for the *Informed* method, converges to a value considerably higher with respect to the optimal one.

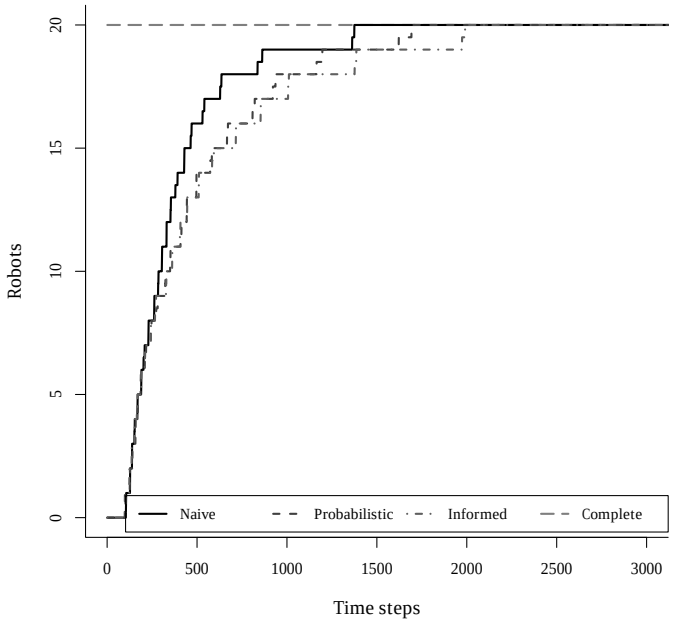
Furthermore, the time needed to achieve the complete allocation of the robots is more than four times higher than the one of *scenario Biased*. Since this delay is observed in all the methods, we take it as an evidence supporting our hypothesis of increasing difficulty of the scenarios.

Figure 23 confirms the better overall performances of the *Informed* method. In Figure 23a the steeper increase of the empirical cumula-

tive density function shows that the 25% allocation level is reached more often and in less time than the other methods. While the same final probability value is reached by the *Probabilistic* method as well, here the *Naive* methods displays its limits. Nevertheless, Figure 23b attests that the 50% allocation level is rarely reached by all the methods, confirming the impossibility to achieve a uniform allocation in this scenario.

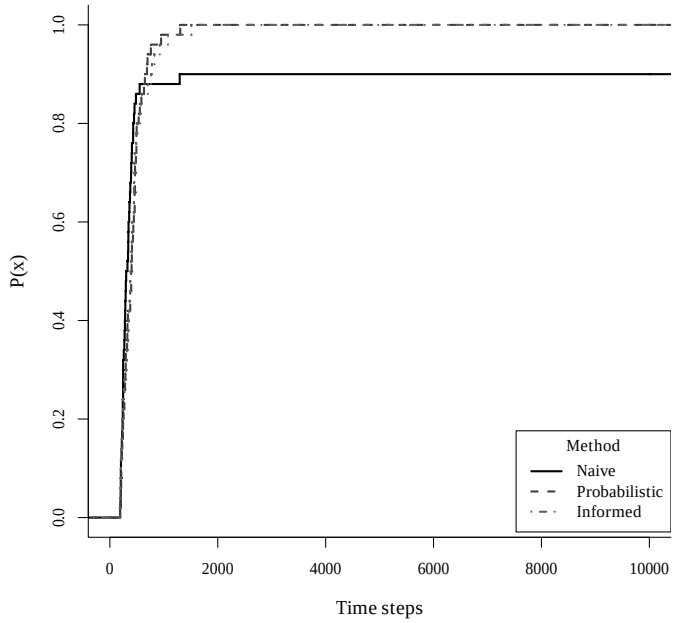


(a) Median $e_{max}(t)$. The dashed grey line represents the optimal allocation error e_{opt}. The differences among the curves are statistically significant. (Wilcoxon signed-rank test, $p \ll 0.05$).

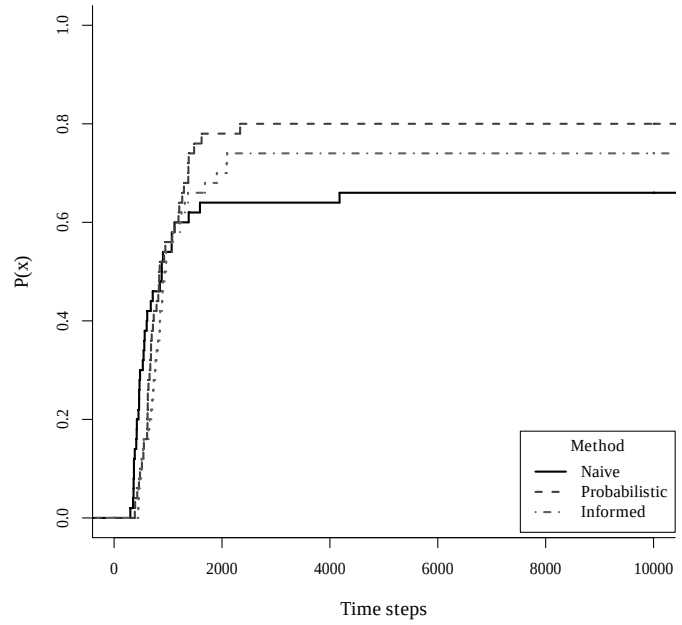


(b) Median $R(t)$. The dashed grey line represents the total number of robots in the experiment. The differences among the curves are statistically significant. (Wilcoxon signed-rank test, $p \ll 0.05$)

Figure 18: Median values for the maximum error across clusters $e_{max}(t)$ and the number of allocated robots $R(t)$ for each of the three methods on the scenario Uniform on 50 trials of 1000 s (10000 simulation steps) each. Each trial is performed with 20 robots and 25 available tasks.

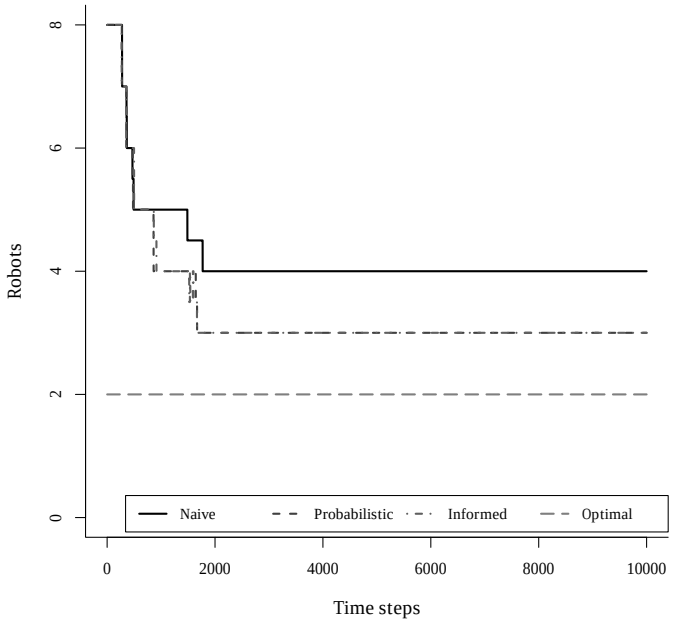


(a) Allocation level: 25%.The differences among the curves are statistically significant. (Mann-Whitney test, $p \ll 0.05$)

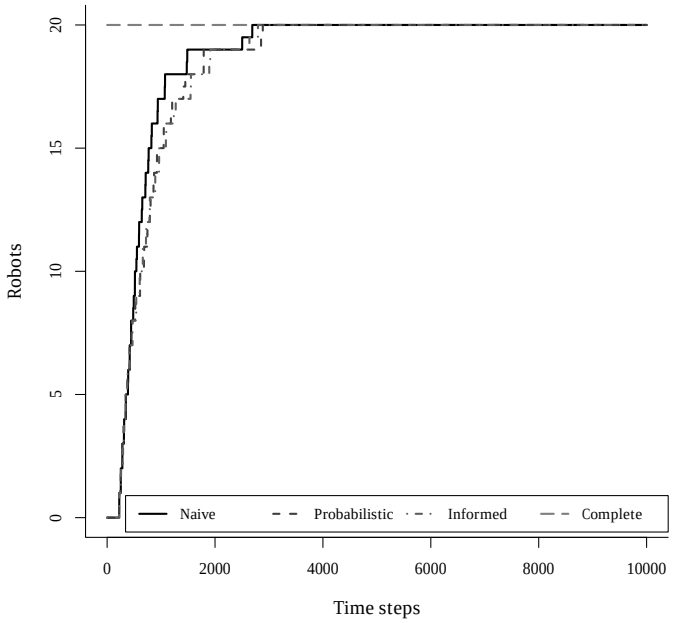


(b) Allocation level: 50%. The differences among the curves are statistically significant. (Mann-Whitney test, $p \ll 0.05$)

Figure 19: Empirical cumulative density functions for the allocation levels o_x $x \in 0.25, 0.50$ distributions of the three methods on the scenario Uniform on 50 trials of 1000 s (10000 simulation steps) each. Each trial is performed with 20 robots and 25 available tasks.

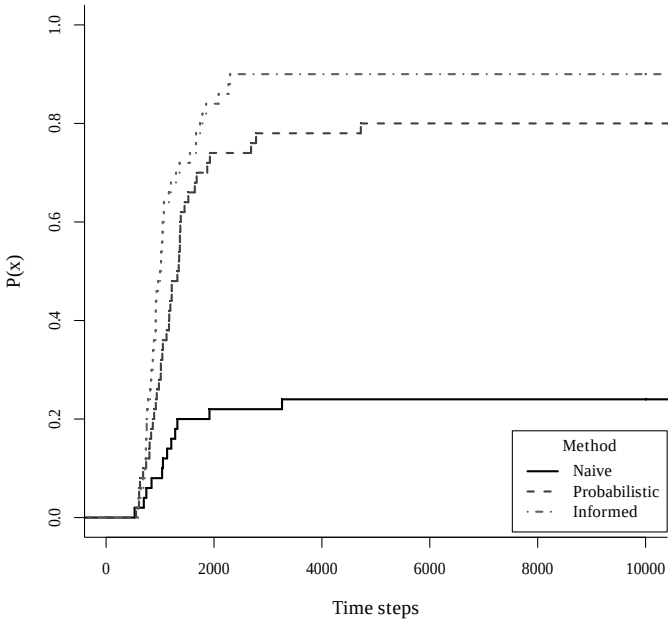


(a) Median $e_{max}(t)$.Only differences among the *Naive* and *Probabilistic* and *Naive* and *Informed* curves are statistically significant. (Wilcoxon signed-rank test, $p \ll 0.05$)

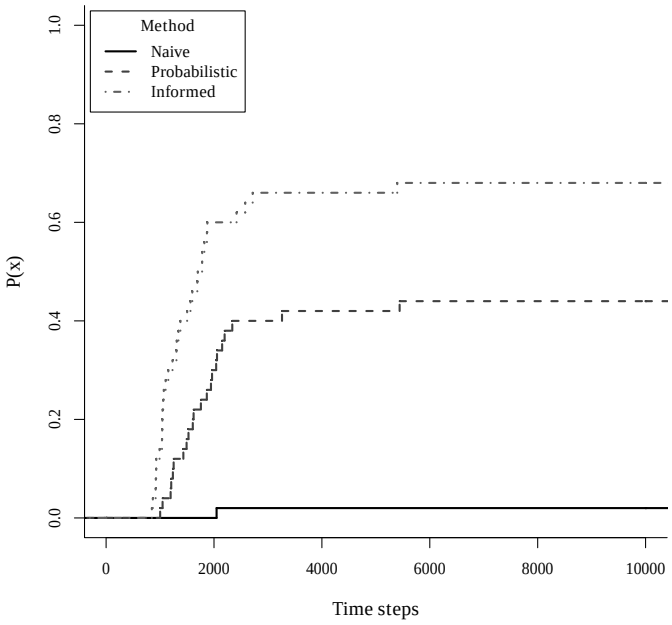


(b) Median $R(t)$.Only differences among the *Naive* and *Probabilistic* and *Naive* and *Informed* curves are statistically significant. (Wilcoxon signed-rank test, $p \ll 0.05$)

Figure 20: Median values for the maximum error across clusters $e_{max}(t)$ and the number of allocated robots $R(t)$ for each of the three methods on the scenario Biased on 50 trials of 1000 s (10000 simulation steps) each. Each trial is performed with 20 robots and 25 available tasks.

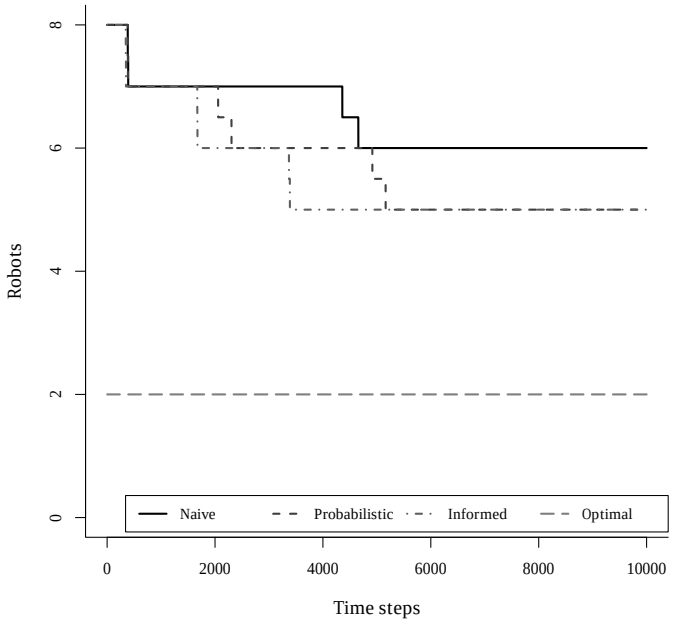


(a) Allocation level: 25%. The differences among the curves are statistically significant. (Mann-Whitney test, $p \ll 0.05$)

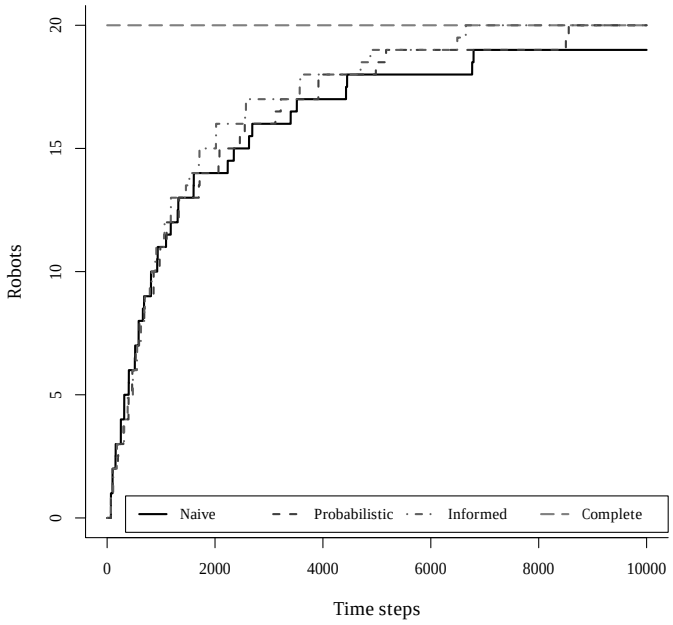


(b) Allocation level: 50%. The differences among the curves are statistically significant. (Mann-Whitney test, $p \ll 0.05$)

Figure 21: Empirical cumulative density functions for the allocation levels o_x $x \in 0.25, 0.50$ distributions of the three methods on the scenario Biased on 50 trials of 1000 s (10000 simulation steps) each. Each trial is performed with 20 robots and 25 available tasks.

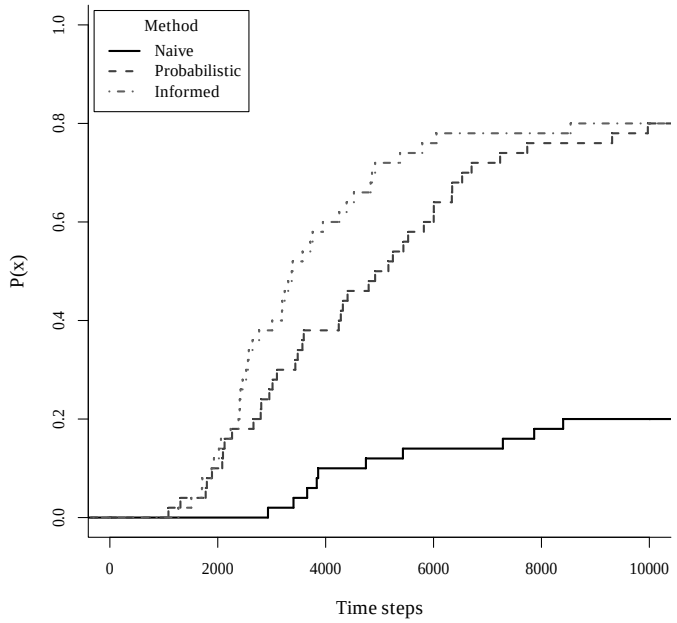


(a) Median $e_{max}(t)$. The differences among the curves are statistically significant. (Wilcoxon signed-rank test, $p \ll 0.05$)

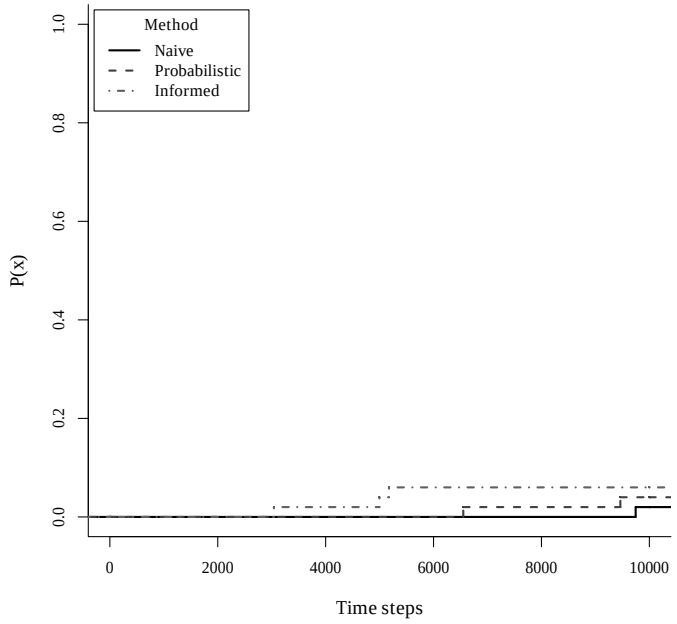


(b) Median $R(t)$. The differences among the curves are statistically significant. (Wilcoxon signed-rank test, $p \ll 0.05$)

Figure 22: Median values for the maximum error across clusters $e_{max}(t)$ and the number of allocated robots $R(t)$ for each of three methods on the scenario Corridor on 50 trials of 1000 s (10000 simulation steps) each. Each trial is performed with 20 robots and 25 available tasks.



(a) Allocation level: 25%. The differences among the curves are statistically significant. (Mann-Whitney test, $p \ll 0.05$)



(b) Allocation level: 50%. The differences among the curves are statistically significant. (Mann-Whitney test, $p \ll 0.05$)

Figure 23: Empirical cumulative density functions for the allocation levels o_x $x \in 0.25, 0.50$ distributions of the three methods on the scenario Corridor on 50 trials of 1000 s (10000 simulation steps) each. Each trial is performed with 20 robots and 25 available tasks.

4.5 SCENARIO DIFFICULTY

After having discussed the properties of the different methods, we now focus on these of the scenarios, by trying to characterize their difficulty with respect to the achievement of a uniform allocation.

We choose to measure this difficulty in terms of the number of the times that the different clusters are seen by the robots, following a simple intuition: the higher the number of times a cluster is viewed, the higher the likelihood a robot will decide to allocate itself to the cluster. This naturally implies that, whenever there is a remarkable difference in the number of views of the clusters, it would be difficult to achieve a uniform allocation, since there exists a bias towards one or more of the clusters.

Moreover, the distribution of the views across the clusters depends on the exploration strategy adopted by the different methods. Since the $v_i(t)$ metric consists of a cumulative sum of the number of views of the cluster, we can assess the effectiveness of the exploration technique of the different methods by evaluating the magnitude of $v_i(t)$.

4.5.1 *Scenario Uniform*

The assumption of the simplicity of the scenario we made during the design phase is confirmed by both Figure 24 and Table 9. For all the methods, we can observe a quasi-uniform distribution of the median values of the views at time $t = 10000$ to the clusters, for every method. Regarding the number of times the cluster have been seen, the *Naive* method is the one having the most limited variability across clusters. Concerning the magnitude of the number of views, we can observe a non-negligible difference among the enhanced methods (*Probabilistic* and *Informed*) with respect to the *Naive* one.

4.5.2 *Scenario Biased*

The *scenario Biased* is characterized by having cluster $1, 3, 4$ closer to the deployment area than cluster 2. This evident cluster distribution bias is confirmed by the magnitude of the median number of views, as shown in Table 10, which, for all the methods, is similar for the three closer clusters and completely different from that of the isolated one. The difference is clearly displayed in Figure 25. Moreover, the *Naive* method has a median number of views of the cluster 2 which is ten times smaller than those of the *Probabilistic* and *Informed* ones. We believe that the systematic ignorance of the isolated cluster arises from the greedy allocation strategy without probabilistic redistribution mechanisms adopted by the first method. Given the nature of the *Naive* method, in fact, the robots will try to go past clusters $1, 3, 4$ only when their requests will be completely satisfied.

4.5.3 *Scenario Corridor*

The distinctive feature of *scenario Corridor* is the rectangular shape of the arena, with a slightly smaller area than the other two. The cluster are arranged linearly in descending order from the deployment area: the cluster having the greater id is the closest one.

The narrower profile of the arena has a strong impact on the magnitude of the median number of views of the clusters (Table 11), which, for example, for the *Naive* method, on cluster 4, becomes ten times bigger than the corresponding value in *scenario Uniform*.

At a first glance, the fact that cluster 3 has a greater number of views than cluster 4 for all the methods seems surprising. A possible justification for this behavior is that the closer cluster is the one having the higher likelihood to be seen and also the higher likelihood to become fully occupied in a short time. As shown in Figure 26, once its request has been satisfied, the cluster rebounds the robots elsewhere in the environment, thus favoring the exploration of other clusters.

Also, the method having the higher number of views on the clusters closer to the deployment area is the *Naive* one, while $v_1(t)$ (the farthest one) is the smallest one. This can be explained by the fact that the *Naive* method do not possess any mechanism to perform an efficient exploration of the environment. For this reason, especially in a narrow environment like *scenario Corridor*, it is likely that a robot will keep viewing the clusters that it has already decided to leave, thus increasing the number of views without actually allocating.

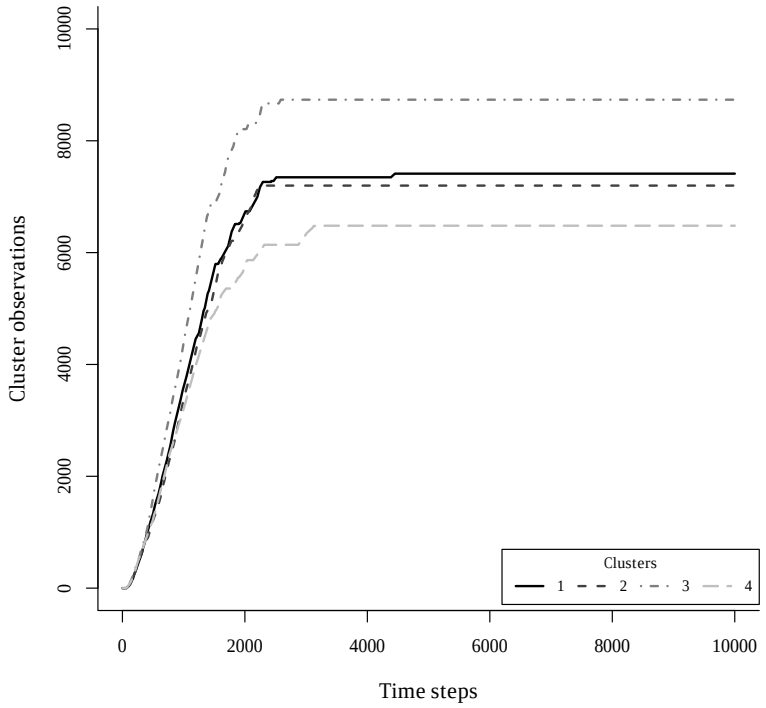


Figure 24: Median value of $v_i(t)$ for $i \in \{1, \cdots, 4\}$ on scenario Uniform across 50 trials of 10000 time steps each. Each trial is performed with 20 robots and 25 available tasks. The cluster disposition can be seen in Figure 15. The differences among the curves are statistically significant. (Wilcoxon signed-rank test, $p \ll 0.05$)

	CLUSTER			
METHOD	1	2	3	4
NAIVE	5650	4673	5464	5070
PROBABILISTIC	7866	5942	7420	6716
INFORMED	7412	7198	8734	6481

Table 9: Summary of the median values of cluster views $v(t)$ at $t = 10000$ in scenario Uniform. Median values are computed across 50 trials of 10000 time steps each. Each trial is performed with 20 robots and 25 available tasks.

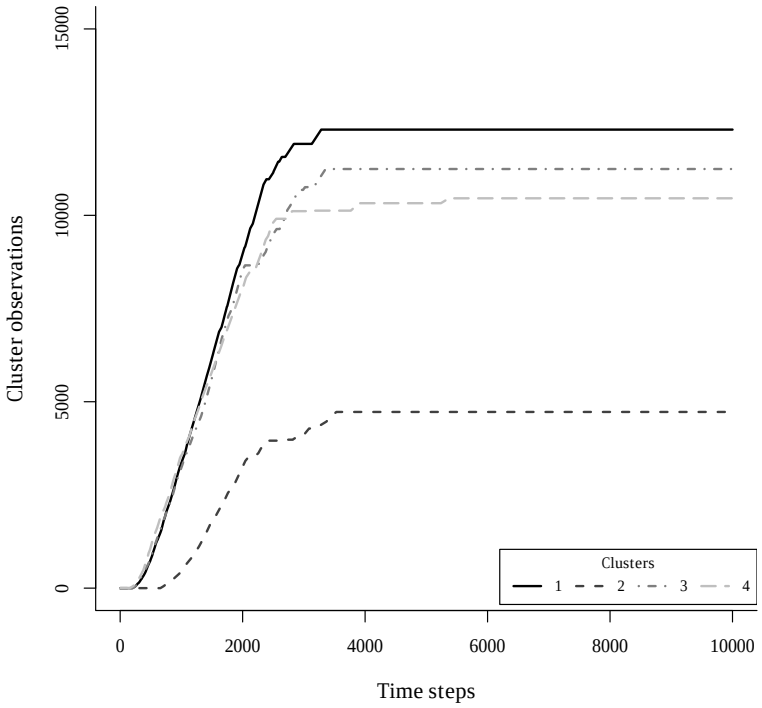


Figure 25: Median value of $v_i(t)$ for $i \in \{1, \cdots, 4\}$ on scenario Biased across 50 trials of 10000 time steps each. Each trial is performed with 20 robots and 25 available tasks. The cluster disposition can be seen in Figure 16. The differences among the curves are statistically significant. (Wilcoxon signed-rank test, $p \ll 0.05$)

	CLUSTER			
METHOD	1	2	3	4
NAIVE	13805	498	13231	11854
PROBABILISTIC	13390	4118	14405	12924
INFORMED	12299	4726	11242	10457

Table 10: Summary of the median values of cluster views $v(t)$ at $t = 10000$ in scenario Biased. Median values are computed across 50 trials of 10000 time steps each. Each trial is performed with 20 robots and 25 available tasks.

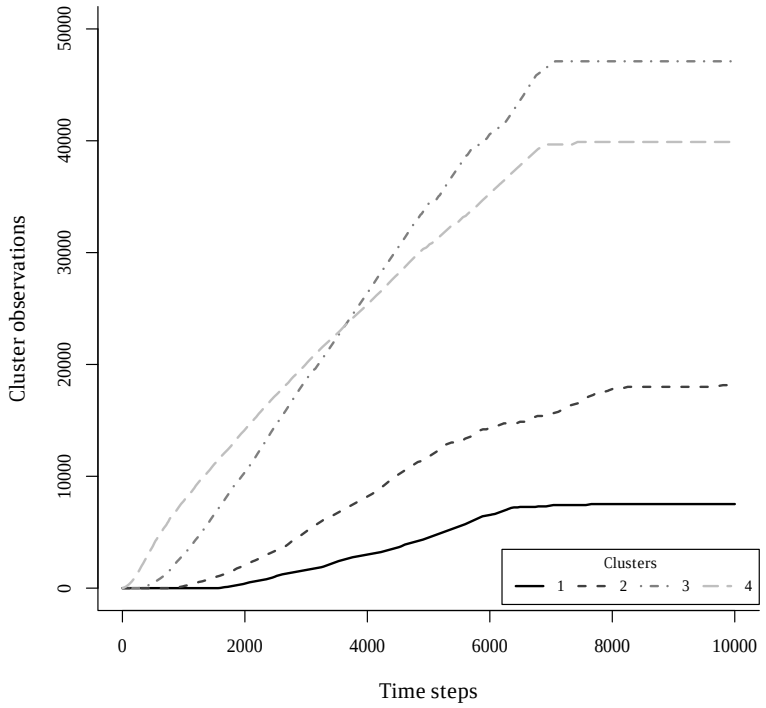


Figure 26: Median value of $v_i(t)$ for $i \in \{1, \cdots, 4\}$ on scenario Corridor across 50 trials of 10000 time steps each. Each trial is performed with 20 robots and 25 available tasks. The cluster disposition can be seen in Figure 17. The differences among the curves are statistically significant. (Wilcoxon signed-rank test, $p \ll 0.05$)

	CLUSTER			
METHOD	1	2	3	4
NAIVE	4363	28015	73242	57410
PROBABILISTIC	7247	23554	62802	49276
INFORMED	7516	18148	47106	39893

Table 11: Summary of the median values of cluster visits $v(t)$ at $t = 10000$ in scenario Corridor. Median values are computed across 50 trials of 10000 time steps each. Each trial is performed with 20 robots and 25 available tasks.

5

CONCLUSION & FUTURE WORK

5.1 CONCLUSION

The problem of allocating agents to tasks having a precise collocation is space is a recurrent problem in collective robotics.

In environmental monitoring, for instance, we may observe the problem of having robots deployed in an unknown environment in order to check for the presence of dangerous substances. Generally, the size and the morphology of the environment prevents the robots from having a global communication and from sharing knowledge about the environment. Hence, each robot needs to rely only on its own capabilities or eventually, communicate with the neighboring robots. Clearly, a centralized solution is technically unfeasible and presents several disadvantages: *single point of failure* and *lack of scalability* among all. Thus, devising a distributed method, possibly emerging from the local coordination among the agents becomes the only possible solution.

Our research question mainly arises from the need of finding efficient solutions to such problems.

In this thesis, we consider a specific instance of the problem of allocating robots to spatially distributed tasks, by fixing some constraints. In our vision of the problem, we have a finite number of robots (n) and a finite number of tasks m, with $n < m$. The tasks are distributed in space according to precise circular patterns, denominated clusters. Each cluster is characterized by the number of tasks it is composed of (i.e. the cluster request) and the number of robots currently being allocated to tasks belonging to the cluster (i.e. the cluster occupation). The robots, on the other hand, are homogeneous agents, with a minimal set of sensors, endowed with local sensing and local communication capabilities.

To the best of our knowledge, we are not aware of other studies in the literature that try to solve a similar problem. For this reason, we developed the *Naive* method to serve as a baseline for comparison with our main contributions: the *Probabilistic* and *Informed* one. We have decomposed the spatial allocation problem into two distinct subproblems, to be tackled independently and sequentially: *task localization* and *task allocation*. Each method is characterized by a different strategy to solve the aforementioned subproblems.

The *Naive* method simply consists of a random exploration of the environment to find the tasks and a greedy allocation to them as soon as they are detected.

The *Probabilistic* method maintains the uninformed random walk from the *Naive* method as exploration technique, but substitutes the purely deterministic allocation rule with a probabilistic decision mechanism. Here, the probability of leaving a cluster to further explore the environment is proportional to the cluster occupation. The use of this stochastic decision mechanism should in principle, favor the redistribution of the robots across the clusters, yielding to a more uniform distribution. Moreover, a check on the occurrence of a stalemate, a situation that may occur when two robots decide to allocate to the same task, has been introduced, along with a probabilistic decision mechanism to overcome this issue and increase the number of allocated robots.

The *Informed* method is built upon the *Probabilistic* one, preserving the probabilistic allocation mechanism but upgrading the uninformed random walk to an informed one through the use of odometry. Odometry is introduced to prevent multiple visits to clusters whose occupation corresponds to the request (i.e not requiring additional robot to be allocated).

We decided to evaluate the performances of our methods with respect to two relevant aspects: *allocation uniformity* and *allocation speed*.

In order to do so, we devised three scenarios: *Uniform*, *Biased* and *Corridor*. Each one of them is characterized by the disposition of the clusters in space and the positioning of the area where the robots are initially deployed.

The *Uniform* one has a central deployment area which is equally distant from all the clusters in the environment, while the *Biased* one presents some clusters closer to the deployment zone than the others. The *Corridor* one consists of a narrow arena, with the clusters linearly deployed in the middle of it.

Concerning *allocation uniformity*, the *Informed* method performs better than the *Naive* and *Probabilistic* one on all the designed scenarios. However, no method is able to reach the ideal uniform allocation on any of the scenarios. Regarding *allocation speed*, the *Naive* method achieves a faster robot allocation than the *Probabilistic* and the *Informed* one on scenario *Uniform* and *Biased*, but it is outperformed by both of them on the *Scenario C*. Furthermore, we observed that there exist a trade-off between the two aspects we are interested in, (i.e. a fast allocation comes at the expense of evenness in the distribution of robots and viceversa). Our analysis showed that the Informed *method* is the one achieving the better balance between *uniformity* and *velocity*.

We also devised a metric to assess the difficulty of a scenario with respect to the problem of achieving a uniform allocation. In fact, by looking at the number of times that every cluster is seen by the robots, we are able to determine whether the distribution of this views is even across cluster or it is biased towards certain ones. In the latter

case, we can safely assume that the scenario is more difficult since the applied method has to compensate for the differences in the views repartitions.

To summarize, our contribution consists of three methods to achieve the allocation of robots to tasks distributed in space and aggregated in clusters. In terms of relative performances among the methods, the *Naive* method is the one achieving the fastest allocation while the *Informed* one allows to better distribute the robots across the clusters. Nevertheless, there is a strong influence of the type of scenario on the performance of the system.

5.2 FUTURE WORK

Since the development and test of the methods has occurred only by means of simulation, our plan is to run experiments using the real *e-puck* robots and the real *TAM*s.

Even though the error in the odometry has been modeled and no communication is employed, further studies to verify the impact of the real noise on the swarm performance should be performed.

Another interesting possibility would be to evaluate different probabilistic rules for the allocation of the robots, in order to have a more fine-grained control on the distribution of the robots across clusters.

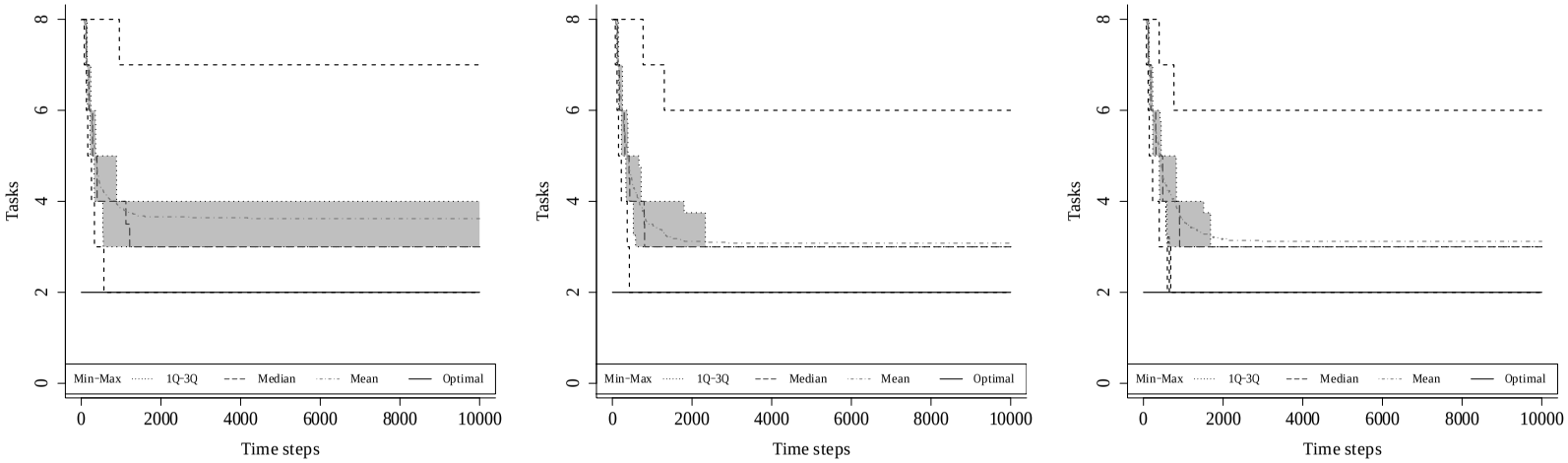
A further study that could be made concerns the use of an inter-cluster recruiting behavior, as occurring in some species of ants, to direct robots from a cluster to another.

Last but not least, tests on the flexibility (i.e. the capacity of the swarm to dynamically adapt to changes in the response), scalability with respect to the number of tasks and robots and robustness of the methods with respect to faults should necessarily be performed.

A

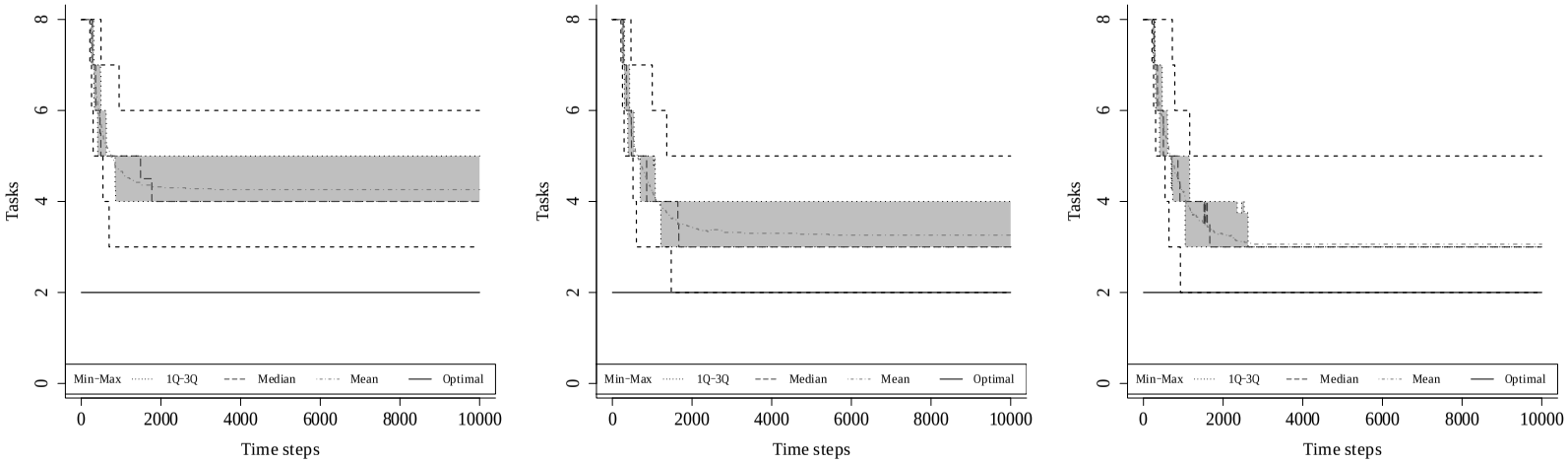
ADDITIONAL PLOTS

This appendix includes the figures that have been discarded from the thesis due to the low readability on the paper version. The relevant data from this plots have been represented in the thesis in a more suitable way. In Maximum error the plots of the quartiles and means of the maximum error across cluster are displayed side by side for a better visual comparison, divided by scenario. In Allocated Robots the plots of the quartiles and means of the number of allocated robots are displayed side by side for a better visual comparison, divided by scenario.



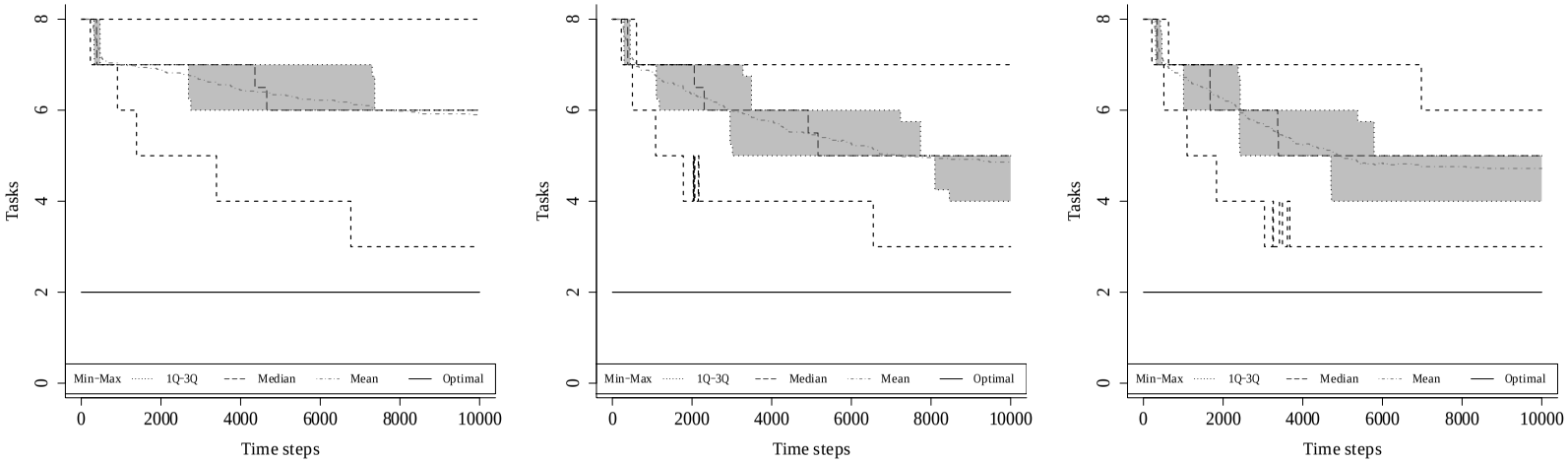
(a) Naive method (b) Probabilistic method (c) Informed method

Figure 27: Comparison of the maximum allocation error e_{max} of the three methods on the scenario Uniform on 50 trials of 1000 s (10000 simulation steps) each. The grey area corresponds to the interquantile range.



(a) Naive method (b) Probabilistic method (c) Informed method

Figure 28: Comparison of the maximum allocation error e_{max} of the three methods on the scenario Biased on 50 trials of 1000 s (10000 simulation steps) each. The grey area corresponds to the interquantile range.



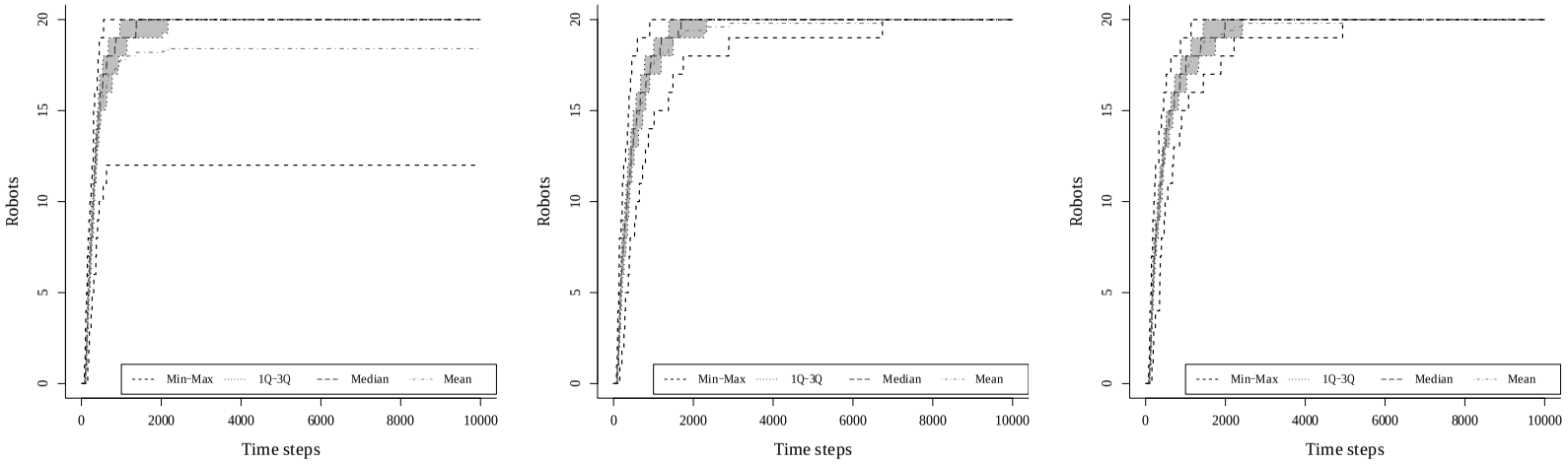
(a) Naive method

(b) Probabilistic method

(c) Informed method

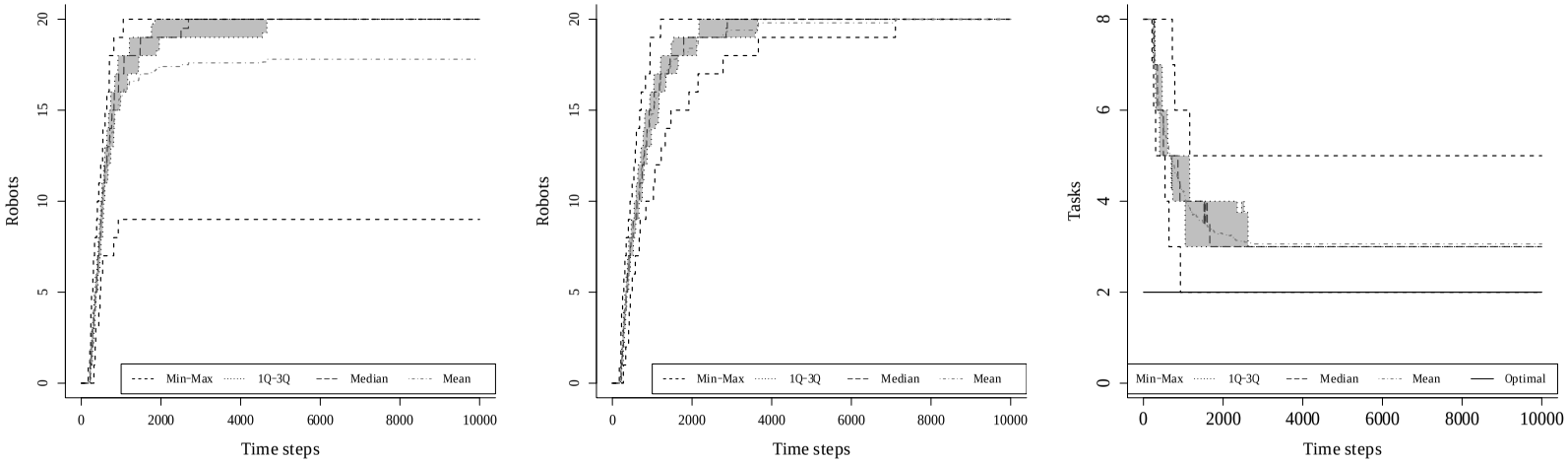
Figure 29: Comparison of the maximum allocation error e_{max} of the three methods on the scenario Corridor on 50 trials of 1000 s (10000 simulation steps) each. The grey area corresponds to the interquantile range.

A.2 ALLOCATED ROBOTS



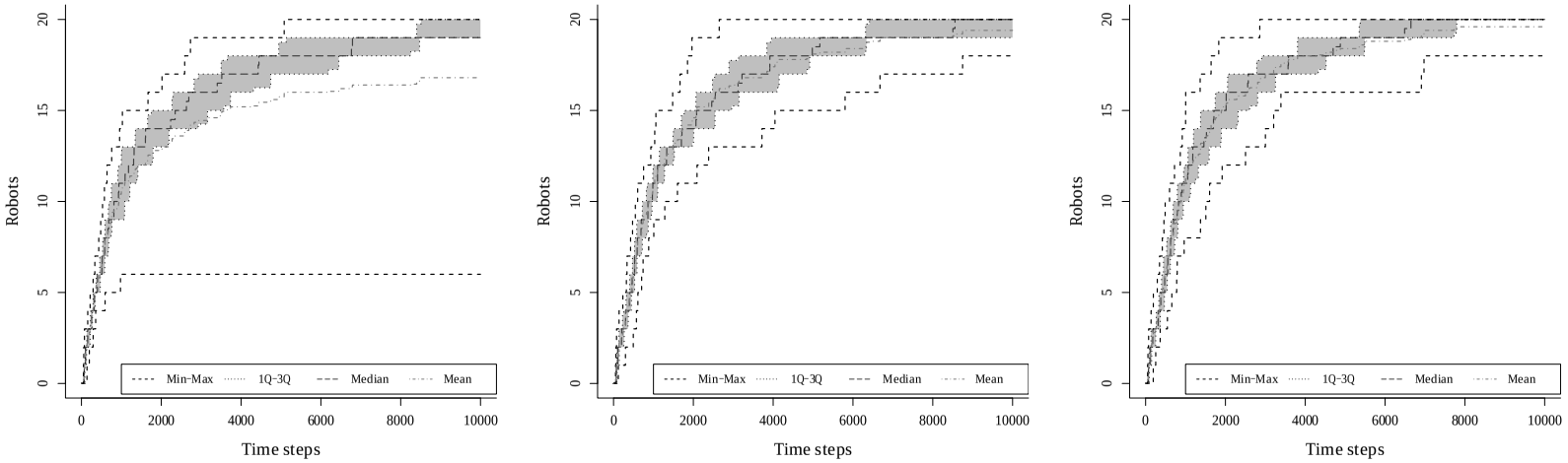
(a) Naive method (b) Probabilistic method (c) Informed method

Figure 30: Comparison of number of allocated robots $R(t)$ of the three methods on the scenario Uniform on 50 trials of 1000 s (10000 simulation steps) each. The grey area corresponds to the interquantile range.



(a) Naive method (b) Probabilistic method (c) Informed method

Figure 31: Comparison of number of allocated robots $R(t)$ of the three methods on the scenario Biased on 50 trials of 1000 s (10000 simulation steps) each. The grey area corresponds to the interquantile range.



(a) Naive method (b) Probabilistic method (c) Informed method

Figure 32: Comparison of number of allocated robots $R(t)$ of the three methods on the scenario Corridor on 50 trials of 1000 s (10000 simulation steps) each. The grey area corresponds to the interquantile range.

BIBLIOGRAPHY

William Agassounon and Alcherio Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1090–1097. ACM, 2002.

p. 15.

W Ross Ashby. Principles of the self-organizing system. *Principles of Self-organization*, pages 255–278, 1962.

p. 6.

Tucker Balch and Maria Hybinette. Social potentials for scalable multi-robot formations. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 73–80. IEEE, 2000.

p. 13.

Itzhak Benenson. Multi-agent simulations of residential dynamics in the city. *Computers, Environment and Urban Systems*, 22(1):25–42, 1998.

p. 9.

Gerardo Beni. From swarm intelligence to swarm robotics. In *Swarm Robotics*, pages 1–9. Springer, 2005.

pp. 2, 6, and 10.

Manuele Brambilla, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. A reliable distributed algorithm for group size estimation with minimal communication requirements. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6. IEEE, 2009.

p. 14.

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

pp. 11, 12, and 13.

Arne Brutschy, Giovanni Pini, Nadir Baiboun, Antal Decugnière, and Mauro Birattari. The IRIDIA TAM: A device for task abstraction for the e-puck robot. Technical Report TR/IRIDIA/2010-015, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2010.

p. 22.

Scott Camazine. Self-organizing pattern formation on the combs of honey bee colonies. *Behavioral ecology and sociobiology*, 28(1):61–76, 1991.

pp. 7 and 9.

Scott Camazine, Jean-Louis Deneubourg, N Franks, J Sneyd, Eric Bonabeau, and Guy Theraulaz. *Self-Organization in Biological Systems*. Princeton University Press, 2001.

p. 4.

Anders Lyhne Christensen, Rehan O'Grady, and Marco Dorigo. From fireflies to fault-tolerant swarms of robots. *Evolutionary Computation, IEEE Transactions on*, 13(4):754–766, 2009.

p. 14.

Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France, 1991.

p. 9.

Steven J Cooper. From claude bernard to walter cannon. emergence of the concept of homeostasis. *Appetite*, 51(3):419–427, 2008.

p. 5.

M Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

p. 17.

Bruce Randall Donald, James Jennings, and Daniela Rus. Information invariants for distributed manipulation. *The International Journal of Robotics Research*, 16(5):673–702, 1997.

p. 14.

M. Dorigo and M. Birattari. Swarm intelligence. 2(9):1462, 2007.

p. 13.

Frederick Ducatelle, Gianni A Di Caro, Carlo Pinciroli, and Luca M Gambardella. Self-organized cooperation between robotic swarms. *Swarm Intelligence*, 5(2):73–96, 2011.

p. 14.

David Gale. Theory of linear economic models, the. 1960.

p. 16.

Simon Garnier, Christian Jost, Raphaël Jeanson, Jacques Gautrais, Masoud Asadpour, Gilles Caprari, and Guy Theraulaz. Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. In *Advances in artificial life*, pages 169–178. Springer, 2005.

pp. 13 and 14.

Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

pp. 15 and 16.

David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

p. 12.

José Guerrero and Gabriel Oliver. Multi-robot task allocation strategies using auction-like mechanisms. *Artificial Research and Development in Frontiers in Artificial Intelligence and Applications*, 100:111–122, 2003.

p. 17.

Alvaro Gutiérrez, Alexandre Campo, Marco Dorigo, Daniel Amor, Luis Magdalena, and Félix Monasterio-Huelin. An open localization and local communication embodied sensor. *Sensors*, 8(11):7545–7563, 2008.

pp. 20 and 28.

Álvaro Gutiérrez, Alexandre Campo, Félix Monasterio-Huelin, Luis Magdalena, and Marco Dorigo. Collective decision-making based on social odometry. *Neural Computing and Applications*, 19(6):807–823, 2010.

p. 14.

Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002.

p. 14.

M Ani Hsieh, Ádám Halász, Spring Berman, and Vijay Kumar. Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence*, 2(2-4):121–141, 2008.

pp. 18 and 19.

Nidhi Kalra and Alcherio Martinoli. Comparative study of market-based and threshold-based task allocation. In *Distributed Autonomous Robotic Systems 7*, pages 91–101. Springer, 2006.

p. 17.

Donald E. Knuth. Computer Programming as an Art. *Communications of the ACM*, 17(12):667–673, December 1974.

p. vi.

Jens Krause, Graeme D Ruxton, and Stefan Krause. Swarm intelligence in animals and humans. *Trends in ecology & evolution*, 25(1): 28–34, 2010.

p. 7.

Michael JB Krieger and Jean-Bernard Billeter. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30(1):65–84, 2000.

pp. 14 and 15.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

p. 16.

David P Landau and Kurt Binder. *A guide to Monte Carlo simulations in statistical physics*. Cambridge university press, 2009.

p. 18.

Liu Lin and Zhiqiang Zheng. Combinatorial bids based multi-robot task allocation method. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1145–1150. IEEE, 2005.

p. 17.

GW Lucas. A tutorial and elementary trajectory model for the differential steering system of robot wheel actuators. *The Rossum Project*, 2001.

p. 37.

Francesco Mondada, Michael Bonani, André Guignard, Stéphane Magnenat, Christian Studer, and Dario Floreano. Superlinear physical performances in a swarm-bot. In *Advances in Artificial Life*, pages 282–291. Springer, 2005.

p. 13.

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, 2009.

p. 19.

Shervin Nouyan, Alexandre Campo, and Marco Dorigo. Path formation in a robot swarm. *Swarm Intelligence*, 2(1):1–23, 2008.

pp. 13 and 15.

Rolf Pfeifer, Max Lungarella, and Fumiya Iida. Self-organization, embodiment, and biologically inspired robotics. *science*, 318(5853): 1088–1093, 2007.

p. 10.

Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence*, 6(4):271–295, 2012.

pp. 23 and 24.

Giovanni Pini, Arne Brutschy, Mauro Birattari, and Marco Dorigo. Interference reduction through task partitioning in a robotic swarm. In *Sixth International Conference on Informatics in Control, Automation and Robotics–ICINCO*, pages 52–59, 2009.

p. 15.

Giovanni Pini, Arne Brutschy, Marco Frison, Andrea Roli, Marco Dorigo, and Mauro Birattari. Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, 5(3-4):283–304, 2011.

p. 22.

Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM, 1987.

p. 7.

Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm robotics*, pages 10–20. Springer, 2005.

p. 9.

Wei-Min Shen and Behnam Salemi. Towards distributed and dynamic task reallocation. *Intelligent Autonomous Systems: IAS-7*, page 289, 2002.

p. 18.

Munindar P Singh. Towards a formal theory of communication for multi-agent systems. In *IJCAI*, volume 91, pages 69–74. Citeseer, 1991.

p. 9.

William M Spears, Diana F Spears, Jerry C Hamann, and Rodney Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2-3):137–162, 2004.

pp. 12 and 15.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

p. 12.

Guy Theraulaz, Eric Bonabeau, and JN Deneubourg. Response threshold reinforcements and division of labour in insect societies. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265 (1393):327–332, 1998.

pp. 7 and 17.

AM Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237 (641):37–72, 1952.

p. 5.

Edward O Wilson. The relation between caste ratios and division of labor in the ant genus pheidole (hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 16(1):89–98, 1984.

p. 7.

Alan FT Winfield. Towards an engineering science of robot foraging. In *Distributed Autonomous Robotic Systems 8*, pages 185–192. Springer, 2009.

p. 15.