Task Allocation in Robot Swarms for Time-Constrained Tasks

Yara Khaluf

Department of Design of Distributed Embedded Systems

University of Paderborn

A thesis submitted for the degree of

Dr.rer.nat

2014

Supervisors of Dissertation:

Prof. Franz-Josef Rammig

Prof. Marco Dorigo

To the soul of my grandfather *Antonious Haddad* $1931 - 2011$

Acknowledgements

First and foremost, I would like to thank my supervisor professor *Franz Rammig* for his excellent guidance, encouragement and patience. He was a perfect mentor from whom I learnt a lot as a teacher (through my Master studies) and as a supervisor (through my research time). Thanks for his flexibility and continuous understanding.

My second big and special gratitude, I would like to express to my second supervisor professor *Marco Dorigo* for the valuable chance and the rich experience I obtained through him. Marco has offered me, first, a 4 months visit to the Artificial Intelligence research laboratory of the Université Libre de Bruxelles, *IRIDIA*, in 2011, for which I will stay thankful. My first visit to IRIDIA was one of the most important stations during my 3 years of research and probably the most enjoyable part of the journey. I owe Marco many thanks for his guidances and the time he dedicated for our discussions, even for his continuous criticism of my English by writing.

After 2011, I have visited IRIDIA for several times in 2012 (up to 6 months) and here I would like to direct my next thanks to my collaborator *Dr. Michele Pace* for the intensive work we did together, for his friendship and for the nice pauses I have shared with him and with *Franco Mascia* playing darts. A deep gratitude goes also to *Carlo Pinciroli*, the kind PhD student (soon Dr. Carlo Pinciroli) who has received me in IRIDIA and introduced me the whole. Thanks for his continuous help with the robotic simulator designed mainly by him *ARGOS* and for our long and effective discussions. Another person in IRIDIA to whom I owe a deep thank, is *Dr. Mauro Birattari* for our useful discussions and for his shaping of my ideas to be attractive for the robotic community.

From out of IRIDIA, I would like to express my gratitude to one of my first collaborators at the university of Paderborn, *Dr. Emi Mathews*. Additionally, I would like also to thank Professor *Friedhelm Meyer auf der Heide*, Professor *Heiko Hamann* and *Dr. Michele Pace* for serving on my dissertation committee. I am grateful to the international graduate school (IGS), for supporting this work with a 3 year Fellowship.
This work was not easy to finish without the enjoyable times I have shared with my friends. Thus many thanks to them specially:
From IRIDIA: *Carlo*, *Michele*, *Franco*, *Lorenzo*, *Andreagiovanni*, *Manuele*, *Touraj*, *Eliseo*, *Gianpiero*, *Dhananjay*, *Leslie*, *Giovanni*, *Gabriele*, *Marie-Éléonore*.
From Out of IRIDIA: *Elisa*, *Rachael*, *Eva*, *Elisabetta*, *Chelly* and *Shirwan*. A special thanks goes to *Christine* my friend who was always ready to support me. Last but not least, My stay in Brussels would not be as wonderful as it was, without the special lady received my in her warm house and made me feel home, to whom I owe many thanks *Mrs. Régine Van Den Abeele*.
Starting from this line, no words may describe my gratitude to the following persons:
My mother, for much more than my words can say, for a life she filled with endless love and boundless support. For all what she gave me and what she taught me.
My Grandmother, for being my second (sometime my first) mother, who grew me up and still supporting me in each and every step.
My sister, my soul mate with whom I have shared the largest part of my life, the happy and the bad moments. She was always my closest friend and my big support.
Melissa, the unique person, who has joined my journey to make it "colorful". I would like to express her my deep gratitude for standing always next to me, for making my life much more beautiful and for all what we shared together.

Abstract

Robotic systems offer auspicious solutions for many real life applications, where robots can replace humans in many dangerous and critical tasks. A promising branch of robotic systems is swarm robotics. This branch has stimulated a significant interest by researchers belonging to different fields such as biology, mathematics, computer science and robotics.

A swarm robotics system consists of a large number of simple robots, which can vary between hundreds and thousands. These systems are able to execute complex tasks, which are beyond the capability of a single robot system. They offer a range of advantages including: fault tolerance, flexibility and scalability. The high level of fault tolerance offered by swarm robotics is due to their large size, as functioning robots can substitute malfunctioning or even broken robots. Swarm robotics is a flexible system which is evinced by its ability to be involved in a large spectrum of applications without the need to modify the robots' hardware. Furthermore, it is a scalable system, where changing the number of robots does not affect the algorithm of the solution. Tasks which require massive parallelism, redundancy or to be executed in complex environments, are good examples where swarm robotics can provide appropriate solutions.

This thesis focuses on a special category of tasks, referred to as *time-constrained tasks*. In time-constrained tasks a specific amount of work should be accomplished on the task within a given deadline. Since tasks with time constraints represent a large category of the real-world applications, swarm robotics will have the need to participate often on executing

such tasks. Our work concentrates on developing task allocation strategies which are supposed to assign robots swarms to time-constrained tasks under the condition of the task deadlines.

Designing swarm robotics systems can be performed following a bottom-up paradigm, which is referred to as a *microscopic-macroscopic* design. It starts from designing the single robot behavior and aims to achieve a desired global behavior. Another paradigm which can be used in designing swarm robotics systems is the top-down one, which is referred to as the *macroscopic-microscopic* design. It starts from designing the swarm behavior and ends with designing the behavior of the single robots. Selecting between the two paradigms depends on the goal of the study in addition to the focus of the swarm application.

In this thesis, global constraints, namely the task deadlines, are required to be fulfilled. Therefore, designing the system starts from the global level and aims to achieve individual robots design that satisfies the global requirements. Hence, the design paradigm used in this thesis, is the macroscopic-microscopic one.

Contents

List of Figures

Chapter 1

Introduction

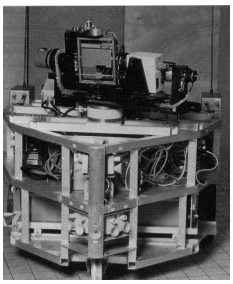
Start by doing what's necessary; then do what's possible; and suddenly you are doing the impossible.

- Francis of Assisi

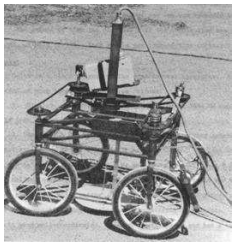
In this chapter, we introduce the work presented in this thesis, formulate the problem of interest, list the contributions of the thesis and present the organization of the work.

Robots have been developed and introduced in the last decades to replace humans in many dangerous, critical or routine tasks. A robot is a physical or virtual agent, which is programmed to perform specific operations or tasks. Mobile robotics systems appeared in the $1960's$ with the first mobile robot *Shakey* (Nilsson, 1984) Figure 1.1 (a). After that, in the $1970's$ other mobile robots came up such as: *Hilare* (Meystel, 1991) Figure 1.1 (b) and *Stanford Cart* (Moravec, 1990) Figure 1.1 (c). Mobile robots were used initially in the form of single-robot systems, where one robot was responsible to participate in the task. Later on, *multi-robot systems* showed up where *Cooperation* or *Competition* among robots were introduced.
The use of multi-robot systems offers significant advantages, among which are overcoming the single point of failure problem existing in single robot systems and having the ability to execute tasks which are beyond the capabilities of a single robot. In addition, multi-robot systems are able to deal with tasks which need to be executed in parallel.
G. Beni and J. Wang have introduced swarm robotics systems in Beni (1988); Beni & Wang (1990a), first under the term of *Cellular Robotic Systems*. It changed later by a comment of Alex Meystel during one of the Ciocco conferences to the term of *Swarm Robotics* (Beni & Wang, 1990b).

(a) Shakey ©*Computer History Museum.*



(b) Hilare ©2003-2005 *CNRS-LAAS.*



(c) Stanford Cart ©*By Les Earnest.*

Figure 1.1 – *Early mobile robots.*

A swarm robotics system is a subset of mobile robotics systems. It is a multi-robot

system with a high density of robots, which have generally limited sensing and communication capabilities. Swarm robotics systems are inspired by social insect colonies and animal societies, therefore, it consists of a large number of robots which may vary between hundreds to thousands of participating robots. The large number of robots allows swarm robotics to offer several advantages including fault-tolerance, which is considered as one of the main advantages of these systems. Moreover, the low cost of individual robots allows for the construction of large swarms and makes swarm robotics a preferable system to be involved in several applications. Another advantage is flexibility, which expresses the possibility to integrate swarm robotics in many kinds of applications without having the need to modify the robots hardware. In addition, swarm robotics is a scalable system, where the size of the swarm can be increased or decreased without to affect the algorithm of the solution. These represent some of the features which make swarm robotics an attractive approach for research and a promising solution for many practical applications.

Tasks which swarm robotics executes can be categorized, based on the interrelation among robots, into cooperative and competitive tasks. Cooperative tasks are tasks where robots work together to achieve a common goal, whereas competitive tasks are tasks where robots work against each other for individual profit. A cooperative relation is always defined among individual robots. However, a competitive relation can be constructed among individual robots or among robot groups as we can see in robot teams playing soccer. An example is the world-wide competition Robocup[1], where our university (university of Paderborn) used to participate since 2001[2]. In this thesis, we focus only on cooperative tasks. Another tasks category represents tasks with temporal constraints, which swarm robotics will need to confront as soon as they are integrated in real-world applications. The correctness of these tasks does not depend only on the logical correctness of the results but also on the time at which the results are delivered. These tasks are referred to as *real-time tasks* or *time-constrained tasks*.

The goal of this thesis is to design self-organized and autonomous task allocation strategies which are able to assign swarms of homogeneous robots to execute a set of time-constrained tasks. Swarm robotics system could be designed following a

[1] http://www.robocup.org

[2] http://paderkicker.upb.de

microscopic-macroscopic paradigm where local mechanisms are identified to produce a desired global behavior, or a macroscopic-microscopic paradigm, where the design start from the swarm level down to reach the design of the single robot behavior. In this thesis, a global behavior is required to be guaranteed, namely the execution of the task within its deadline. Since it is particularly difficult to start from the level of designing and analyzing the individual behaviors to end up with a desired global behavior, thus, a macroscopic-microscopic paradigm is followed to achieve the goal of this thesis. We start by designing the global behavior of the swarm based on the time constraints of the tasks, before being able to derive the design of individual robot behaviors.
The rest of this chapter is organized as follows, Section 1.1 defines the problem of interest. Section 1.2 lists the scientific contributions of the thesis and Section 1.3 illustrates the organization of the work.

1.1 Problem Statement

In this thesis, we focus on the problem of assigning a swarm of homogeneous robots to a set of tasks under the constraint that they have to meet some deadlines. The problem represents a particular case of the general *task allocation* problem, where the task deadline represents the time point up to which the task should be executed. The tasks, we are considering in this thesis, consist of discrete sub-tasks referred to as *parts*, which can be executed in parallel. A single robot is enough to execute an individual part of any of the considered tasks. However, the number of parts required to be accomplished is much larger than what a single robot can accomplish within the task deadline. Therefore, cooperation among robots is necessary. Each time a part is executed, a new part is generated. The task is characterized by its deadline and by its size, which represents the number of parts that should be accomplished up to its deadline. Accomplishing a task is achieved when the number of executed parts within the task deadline is equal to or greater than the task size. In other words, when the time of executing the task is smaller than or equal to the task deadline. An example is the foraging task where robots are required to retrieve a specific number of parts back to the nest up to a specific deadline.

The task deadlines can be categorized mainly into *hard-deadlines* where missing the deadline affects the correctness of the results and *soft-deadlines*, where finishing the task after its deadline is associated with specific costs. The problem we focus on in this thesis, is defined for both categories of deadlines:

- **Problem description for hard-deadline tasks:** Since missing a hard-deadline is counted as a failed execution of the task, the main focus, here, is to find out the set of tasks that can be executed within their deadlines and which is referred to as the set of executable tasks. The set of executable tasks is a subset of the task set. In addition to the set of executable tasks, the related assignment of the robots to the different executable tasks is designed. For a stochastic system such as swarm robotics the categorization of the task as executable cannot performed in a deterministic manner. Therefore, it is verified probabilistically using the *execution probability* of the task. The execution probability of any task is the probability of executing the task within its deadline.

- **Problem description for soft-deadline tasks:** Since missing a soft-deadline is associated with specific costs, the main focus, here, is to minimize the costs resulting from the task part which remains unprocessed at the deadline. The costs are calculated in terms of the difference between the task size and the executed part of the task at the task deadline. This represents the number of parts remain unprocessed at the deadline. The proper assignment of robots to the soft-deadline tasks is the assignment which minimizes the described costs.

When the tasks are located on different physical sites, the robots will need to travel between the different sites when task switching is applied. The time required by a robot to travel between its current task and its next task is referred to as the *switching cost*. This time is the time between stopping the execution of the current task and starting the execution of the next task. Since the execution of the tasks we consider in this thesis is evaluated under their time constraints, the time lost in switching among the tasks plays a main role in planning the successful execution of these tasks. In this thesis, we differentiate between tasks with negligible switching costs, such as tasks located on the same physical site and tasks with high switching costs. The problem of interest is described for both kinds of tasks:

- **Problem description for static allocation:** Static allocation is the technique used when the switching costs among tasks are high. It is based on preventing the robots to switch their tasks which they select at the beginning of the execution. The main focus, here, is on finding out the proper assignment of the robots to the different time-constrained tasks. This assignment is used by the robots only once at the beginning of the execution.

- **Problem description for dynamic allocation:** Dynamic allocation is the technique used when the switching costs among tasks are negligible. It allows the robots to switch among tasks during their execution time. The main focus, here, is to find out the proper switching probabilities, which can be used by the robots to switch among the tasks with respect to the task deadlines.

1.2 Scientific Contribution

The main contribution of this thesis can be summarized as follows: The development of autonomous non-communicative allocation strategies that assign robot swarms to parallel time-constrained tasks with respect to their deadlines (hard and soft). The tasks can have negligible as well as high switching costs.
This main contributions are:

- Mathematical modeling of swarm performance: This contribution includes the definition of the swarm performance for static and dynamic allocations in general and under the consideration of task deadline. After that, different mathematical techniques were investigated to model the swarm performance as defined for both static and dynamic allocations. The applied models are used later to design the robots' assignments to execute the tasks under their time constraints.

- Task allocation strategies: This contribution is associated with the development of different allocation strategies to be applied autonomously by the robots of the swarm in order to execute time-constrained tasks. The allocation strategies are developed to be used with both hard and soft deadlines and under

both static and dynamic allocation techniques, see Section 1.1. The allocation strategies developed in this thesis require no communication among the robots. Consequently, possible conflicts which may arise among the robots on performing the same task parts are not within the scope of this work. However, in case the robot-to-robot communication is exploited, such conflicts can be solved easily by implementing simple local rules such as “the nearest robot to the part executes it" or “the robot with the smallest ID executes the task part”.

- Verification of task allocation strategies: The other important contribution of this thesis is the verification of the developed allocation strategies. This verification is performed mainly by comparing the theoretical results out of the allocation strategies with the simulated results. For the simulations two techniques were used: Monte-Carlo simulations using Matlab software[1] and physics-based simulations using the ARGoS robotic simulator (Pinciroli *et al.*, 2012).

1.3 Organization of the Thesis

This thesis is organized as follows:

- Chapter 2 is divided into swarm background and mathematical background. Swarm background describes the *swarm intelligence* term starting from its origin in the nature moving to the kind of intelligence inspired by biological swarms and then to the swarm robotics systems. Mathematical background is dedicated to introduce and define, in summary, the mathematical models used throughout this thesis.

- Chapter 3 introduces the related work conducted mainly in the fields of mathematical modeling of swarm robotics systems and task allocation in multi-robot systems. The mathematical modeling of swarm robotics systems is presented as the thesis introduces novel methods of modeling the swarm performance, mathematically. Since, the main problem considered in this thesis is about the

[1] http://www.mathworks.com

development of efficient task allocation strategies for time-constrained tasks, an overview of the literature in task allocation for multi-robot systems is presented.

- Chapter 4 presents the mathematical framework to model the swarm performance for both hard and soft deadlines under both static and dynamic allocations. In this chapter, we derive the execution probability of the task based on the mathematical modeling of the swarm performance. This is used later in designing the allocation of hard-deadline tasks. In addition, we derive the expected swarm performance at the task deadlines, which is used in designing the allocation for soft-deadline tasks.

- Chapter 5 presents the task allocation strategies developed for both hard and soft deadlines under both static and dynamic allocation techniques. The chapter describes the mechanisms used by each task allocation strategy to handle the particular case it is developed for. In addition, it includes a set of Monte-Carlo simulations implemented in Matlab to verify the performance of the developed allocation strategies.

- Chapter 6 presents a simple version of the bio-inspired foraging task for swarm robotics, where different scenarios of foraging are introduced under the different kinds of tasks and deadlines discussed in Section 1.1. Physics-based simulations are performed in this chapter to evaluate the performance of the allocation strategies developed in chapter 5.

- Chapter 7 concludes the thesis presenting a summary of the main contributions. In addition, it proposes future directions for improving and expanding of the conducted work.

Chapter 2

Background

> Everything has been said before, but since nobody listens we have to keep going back and beginning all over again.
>
> - Andre´ Gide

In this chapter we present a brief background concerning the swarm intelligence *and the* mathematical *modeling techniques used in this thesis. For swarm intelligence, we go through its origin starting from the nature. After that, we present how the biological intelligence was imported by artificial systems and investigated in two main fields, namely, optimization and robotics.*
Under the mathematical background we briefly present several techniques, which are used in this thesis, including: a general background of useful probability distributions and Markov processes especially the Poisson process.

2.1 Swarm Background

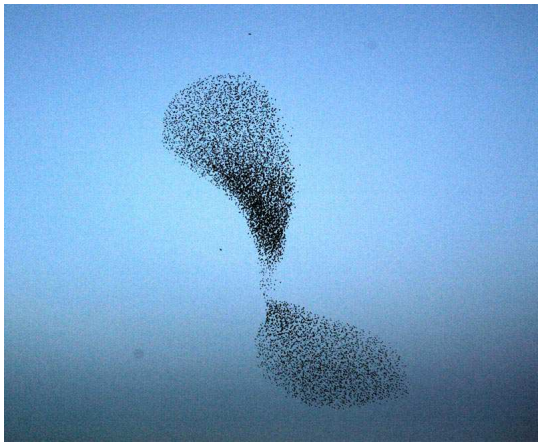
2.1.1 Swarm Intelligence in Nature

Swarm intelligence is a form of collective intelligence that can be observed in nature. A remarkable example of swarm intelligence is the *foraging* behavior observed in social insect colonies such as bees and ants. This colony behavior is performed with a high level of intelligence that allows to select the richest food source or the shortest path to the food. Ants, for example, use a communication mechanism through the chemical pheromone trails in order to find out the shortest path to the food source. On the other hand, in bee colonies, a bee becomes a forager only when being 3 weeks old and it becomes able to fly up to 3 miles searching for food sources and amazingly coming back to the same hive. In case it has found some food sources during its search, it communicates with the other bees conveying two pieces of information, namely, the direction of the food source and the distance to it. This information is announced through a special kind of dance performed by the bee, referred to as "waggle dance". The information is encoded in the movements and the sounds of the bee.

Another example of swarm intelligence is *moving in groups*. This example presents not only the way insects or fish move together, but the way many other kinds of intelligent species do including humans. Determining the movement direction represents the main question, specially in large groups where collision probabilities are considerably high. The movement direction can be determined normally in two ways: Either all individuals participate on the decision (Grünbaum, 1998) ,(Berthold *et al.*, 1992) or some called "leaders" guide the majority (Swaney *et al.*, 2001), (Reebs, 2000). The question remains how the guiding information is transferred in case of a leader strategy and how a consensus is achieved when all individuals participate. There are two theories presented: the first one states that the leaders are faster than other individuals, thus they steer the group when others try to align with their neighborhood. The second theory states that it is enough for the informed individuals to have preferable directions of movement to steer the group. On the other hand, when no leaders exist and all group members should contribute to reach

a consensus, there should be a minimum (threshold) number of members which are aligned initially together.

(a) *Locusts in Mauritania - Photograph by Jean-François Hellio and Nicolas Van Ingen, ©National Geographic Society.*



(b) *Flock of starlings - Photograph by the La Sapienza team - ©StarFlag.*

Figure 2.1 *– Swarm in nature.*

An additional example of swarm intelligence which can be observed in nature is *nest building*, where sophisticated architectures are achieved in a self-organized manner with no control. Bees build their combs consisting of almost exact hexagonal cells, which hold nectar, breed and pollen. Thompson *et al.* (1942) has explained this phenomena as a pattern formed by the physical forces that are applied to all layers of bubbles. In case of a bee's comb, the soft wax causes the bubbles to be formed in cells.

Division of labour or task allocation is another significant example of natural swarm intelligence. It describes the methodology applied by the species to allocate themselves to the different tasks they should execute. They perform this division in a self-organized way using a "threshold-response" mechanism. Based on this mechanism, an increasing stimuli increases the probability of the individual to participate in the task.

Other examples of nature swarm intelligence, are: brood sorting, cooperative transport and clustering.

2.1.2 Swarm Intelligence in Artificial Systems

Swarm intelligence in its different forms that can be witnessed in nature, has fascinated a large number of researchers to inspire them for many artificial systems. Many mechanisms that have been revealed recently and which describe the complex behaviors which could be seen in biologically systems, have been used as inspiration. The main goal was to be able to build intelligent large distributed and self-organized systems starting from a large set of simple individuals. A global collective behavior should emerge from local rules and interactions.
Swarm intelligence was first inspired by Beni (Beni & Wang, 1993), under the concept of *cellular robotics systems*. After that, swarm intelligence started to find its way successfully in the area of *optimization*. In the following we present a brief background concerning swarm intelligence inspiration in both fields of optimization and robotics.

2.1.2.1 Swarm Intelligence in Optimization

Optimization techniques inspired by biologically swarm intelligence have gained a significant interest and have become popular in the recent years. Their advantages of being self-organized, distributed, flexible and robust allowed them to outperform a large set of traditional techniques.
We present in this section two of the most successful and most applied techniques:

A) **Ant colony optimization ACO:** this technique was introduced by Marco Dorigo and his colleagues in the early 1990. It is inspired by the biological foraging within ant colonies. The core of this technique is associated with the indirect communication among the ants, namely the "pheromone trails". In the foraging task, ants start exploring the area by random search and they leave pheromone along the paths to the food sources. The pheromone represents a chemical material that can be smelled by other ants. They will follow the paths marked with the more intensive amount of pheromone.

   ACO was applied to a large range of problems including "traveling salesman problem" Dorigo & Gambardella (1997); Dorigo *et al.* (1991, 1996), scheduling problems Besten *et al.* (2000); Blum (2005); Blum & Sampels (2004);

Gagné *et al.* (2002); Merkle *et al.* (2002); Stützle *et al.* (1998), vehicle routing problems Gambardella *et al.* (1999); Reimann *et al.* (2004), bio-informatics problems Karpenko *et al.* (2005); Korb *et al.* (2006); Moss & Johnson (2003); Reyes-Sierra & Coello (2006); Shmygelska *et al.* (2002), in addition to industrial problems Bautista & Pereira (2007); Blum *et al.* (2006); Corry & Kozan (2004); Gottlieb *et al.* (2003); Silva *et al.* (2002). An overview of ACO can be found in Dorigo (2007); Dorigo & Stützle (2004)

B) **Particle swarm optimization PSO:** This optimization technique was first proposed by James Kennedy and Russell Eberhart, Kennedy & Eberhart (2001). It was inspired by the social behavior of animal societies and insect colonies for the purpose of self-protecting. It describes the way the group members move together as a unit while avoiding potential collisions.

In PSO, a set of individuals, which is referred to as particles, move initially in the whole search space of the problem, seeking the optimal position. They broadcast their positions to the neighborhood. The position of each individual solution (particle) is then adjusted according to the best position found by the neighborhood and its best found position. As the process goes on, the particles focus more and more on the best solution area heading towards the optimal one.

PSO was applied to different optimization problems including: "traveling salesman problem" Onwubolu & Clerc (2004), neural networks Gudise & Venayagamoorthy (2003); Kennedy & Eberhart (2001); Mendes *et al.* (2002); Settles *et al.* (2003), bio-informatics Correa *et al.* (2006); Georgiou *et al.* (2004), multi-objective problems Coello & Lechuga (2002); Hu & Eberhart (2002b); Li (2003); Moore & Chapman (1999); Parsopoulos & Vrahatis (2002), in addition to dynamic problems Blackwell & Branke (2004, 2006); Carlisle & Dozier (2000); Carlisle & Dozler (2002); Eberhart & Shi (2001); Hu & Eberhart (2002a); Janson & Middendorf (2006); Li *et al.* (2006); Parrott & Li (2006). An overview of PSO can be found in Dorigo *et al.* (2008); Kennedy & Eberhart (2001)

2.1.2.2 Swarm Robotics

Swarm robotics represents the second major field, after optimization, to be based on swarm intelligence inspired from insect colonies and animal societies. A significant effort was paid to build robotic systems which follow the same logic. Beni (Beni & Wang, 1993) was the first who denoted a class of cellular robots as "swarm robotics". Swarm robotics systems provide a range of advantages starting from fault-tolerance, where the large number of robots allows the system to operate appropriately even in the case of malfunctioning individuals. Flexibility is another advantage of swarm robotics that represents the ability of the system to tackle different kinds of tasks, starting from nanoscopic tasks inside humans bodies to outer-space tasks without having the need to apply significant changes in the robots hardware. In addition to the advantages mentioned above, scalability belongs to the remarkable advantages of swarm robotics. Scalability expresses the system ability of operating under scalable size of the swarm without a considerable impact on the behavior and the functionality of the system. Swarm robotics have been studied as homogeneous systems where all robots are identical as well as heterogeneous systems, where several types of robots build up the swarm.
Şahin *et al.* (2008) has categorized the swarm robotic studies into 4 directions: design, modeling and analysis, robots, and problems. Under design, the authors have focused on the goal related to the "engineering of self-organization". The paper highlights the importance of modeling and analyzing of swarm robotics under two objectives: first when guarantees are needed for the performance on the system level, second to design the optimal values of the individual behaviors. The models of swarm robotics were categorized mainly into microscopic and macroscopic models. The microscopic models are performed on the level of the individuals, Ijspeert *et al.* (2001), whereas the macropscopic models are performed on the level of the swarm system, Lerman *et al.* (2001).
This thesis focuses on a macroscopic modeling of swarm robotics system, where the swarm performance should obey specific time constraints of the tasks.
Developing robots for swarm robotics systems is a non-trivial process. Different concepts need to be taken into account during this design, including: sensing abilities, communications, cost, power, size, and others. Designing robots under all those

concepts is a challenging task. Thus, the design choices are made mostly under one requirement.

From the other side, a large spectrum of problems, which are inspired mainly from biological swarm behaviors were studied using swarm robotics systems. The examples include aggregation which refers to the self-organized aggregation of robots in clusters (Dorigo *et al.*, 2004), (Soysal & Şahin, 2007). Foraging, which can be seen as mentioned above in many ant and bee colonies, where the individuals search for food items to return them to the nest (Hamann & Wörn, 2007). Foraging is the task used to verify the task allocation strategies developed in this thesis. Self-assembly deals with problems where individuals are meant to build chains through connecting to each other like bridges or structures (Christensen *et al.*, 2007), (O'Grady *et al.*, 2007). Other problems like connected movement (Trianni & Dorigo, 2005), (Trianni *et al.*, 2006) and cooperative transport (Groß *et al.*, 2005) were also studied. An overview of swarm robotics research can be found in Brambilla *et al.* (2013)

2.2 Mathematical Background

This section is dedicated to present a brief mathematical background about techniques and models used throughout this thesis.

2.2.1 Probability Distribution

Probability is the measurement which describes how likely an event is to occur. The probability distribution is the procedure of assigning a specific probability to each measurable subset of the potential outcomes of the considered experiment. For experiments whose sample space is discrete, their probability distribution can be specified by a *probability mass function* (PMF). Let us suppose a discrete random variable: $X : S \to A$ ($A \subseteq \mathbb{R}$) which is defined on a discrete sample space S. The probability mass function $f_X : A \to [0, 1]$ for X can be defined as in the following:

$$f_X(x) = \Pr(X = x) \qquad \text{and} \qquad \sum_{x \in A} f_X(x) = 1 \tag{2.1}$$

For random variables which are defined on continuous sample spaces, the probability distribution is specified by a *probability density function* (PDF). These two functions belong to the methods that could be used to characterize the probability distribution. Another method of characterizing the probability distribution is by using the *cumulative distribution function* (CDF). The cumulative distribution function describes the probability that a random variable X which is sampled from a given probability distribution, takes a value less than or equal to x:

$$F_X(x) = \Pr(X \leqslant x) \qquad \text{and} \qquad \Pr(a < X \leqslant b) = F_X(b) - F_X(a) \tag{2.2}$$

In case of a continuous random variable, the cumulative distribution function can be calculated as the integral of the probability density function (PDF):

$$F_X(x) = \int_{-\infty}^{x} f_x(t)dt \tag{2.3}$$

Some of the well-known probability distributions which are used in this thesis are listed in the following:

- **Exponential Distribution:** it is a wide-used distribution, which is investigated in modeling a large range of natural and physical phenomena. Examples can be found in queuing theory, reliability theory and physics. It is a continuous probability distribution with the probability density function given by:

$$f(x, \lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geqslant 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.4}$$

  where λ is the distribution rate.

  The cumulative distribution function of the exponential distribution is given by:

$$F(x, \lambda) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geqslant 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.5}$$

  where λ is the distribution rate.

- **Truncated Exponential Distribution:** this distribution is a special case of the exponential distribution, where the random variable takes its value over

a defined range $[a, b]$. It is probability density function is given as in the following:

$$f(x, \lambda, a, b) \begin{cases} \frac{\lambda e^{-\lambda x}}{e^{-\lambda a} - e^{-\lambda b}} & a \leqslant x \leqslant b \\ 0 & \text{Otherwise} \end{cases} \tag{2.6}$$

where λ is the distribution rate.

- **Normal (Gaussian) Distribution:** This distribution counts also as one of the wide-used distributions especially for modeling real-valued quantities that grow linearly, like errors or offsets. Its probability density function is given by:

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{2.7}$$

where μ is the mean and σ is the standard deviation.

The cumulative distribution function of the normal distribution is given by:

$$F(X, \mu, \sigma) = \frac{1}{2}\left[1 + erf(\frac{x-\mu}{\sqrt{2\sigma^2}})\right] \tag{2.8}$$

where μ is the mean and σ is the standard deviation.

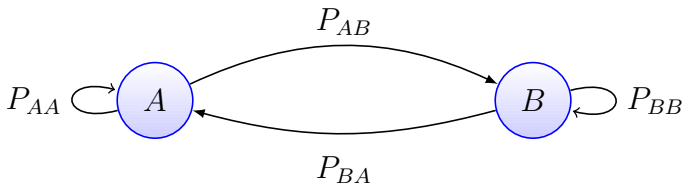
2.2.2 Markov Chains

A Markov process is a kind of stochastic processes, which is named after the Russian mathematician Andrey Markov. A stochastic process, sometimes called a random process, is a set of random variables which represent the evolution of some random value over time. A Markov process is a stochastic process which has the property, referred to as the *Markov property* or the *memoryless property*. It states that the future state of the process depends only on the current state and not on any other previous states.

$$\Pr(X_n = x_n | X_{n-1} = x_{n-1} \ldots X_0 = x_0) = \Pr(X_n = x_n | X_{n-1} = x_{n-1}) \tag{2.9}$$

where X_n is the $n - th$ value of the random process.
Markov chains are categorized into *discrete time Markov chains* (DTMC) and *continuous time Markov chains* (CTMC). The discrete time Markov chain is a stochastic process which visits a countable set of states over a discrete set of times. The

changes of the system state are called transitions and the probabilities to change among the states are referred to as the transition probabilities, see Figure 2.2 (a). The transition matrix, Figure 2.2 (b) is the matrix which holds the transition probabilities used to switch from any state to any other state. This kind of Markov chains describes a system which is at a specific state at each time step.



(a) *Markov chain.*

$$P = \begin{bmatrix} P_{AA} & P_{AB} \\ P_{BA} & P_{BB} \end{bmatrix}$$

(b) *Transition matrix.*

Figure 2.2 – *Markov chain and transition matrix.*

On the other hand, a continuous time Markov chain (CTMC) is a stochastic process which evolves over continuous time. It has the Markov property mentioned above however over the continuous time t. Let $t_0, t_1, \ldots, t_n$ represent the n time points at which the continuous time process visits the following states: $i_0, i_1, \ldots, i_n$, then the Markov property can be written as in the following:

$$\Pr(X_{t_{n+1}} = x_{t_{n+1}} | X_{t_n} = x_{t_n} \ldots X_{t_0} = x_{t_0}) = \Pr(X_{t_{n+1}} = x_{t_{n+1}} | X_{t_n} = x_{t_n}) \quad (2.10)$$

The continuous time Markov chain is characterized by its transition rate matrix called *Q-matrix*. The Q-matrix is a matrix which holds the rate at which the Markov chain is moving among the different states. Let I be a countable set of states, the Q-matrix on I is a matrix $Q = (q_{ij} : i, j \in I)$ and it satisfies the following conditions:

- $0 \leqslant -q_{ii} < \infty \quad \forall i.$
- $q_{ij} \to 0 \quad \forall i \neq j.$
- $\sum_{j \in I} q_{ij} = 0 \quad \forall i.$

Markov chains represent a large field of study where intensive research is performed. For more about Markov chains please refer to Norris (1998) and Ross (1992).
Markov chains were applied in several fields including: biology, queuing systems, computer systems, telecommunication systems, economic systems, and others. In

swarm robotics studies as a new field of research, Markov chains represent a promising technique to develop useful models for such large-scale stochastic systems. In Brambilla *et al.* (2012), the authors have applied discrete time Markov chains (DTMC) in developing a top-bottom method for verifying properties in swarm robotic design. Lerman *et al.* (2005) investigated discrete time Markov chains to develop macroscopic models for swarm robotics. Markov chains were exploited also to model robotic systems on the microscopic level, where a robot was considered as the system unit and its actions were the states of the Markov chain, such as in Arkin (1998), Lerman *et al.* (2001) and in Goldberg & Mataric (2003). Berman *et al.* (2009) has presented a decentralized strategy of dynamic task allocation of a swarm of robots to a set of tasks which are running in parallel. The states of the Markov chain are the distribution of the robots among the different tasks and a desired distribution of the robots was required to be achieved.

2.2.2.1 Poisson Process

A Poisson process is a time continuous Markov chain. It is a stochastic process which counts random events that occur over the continuous time. The process is named after the French mathematician Siméon Denis Poisson. It was investigated to model different systems including: radioactive decay, telephone calls, requests for a particular document on a web server, queuing systems and others.
The inter-arrival times Δt_i between the consecutive events, which are counted by a Poisson process, are assumed to be independent from each other and to have an exponential distribution with the parameter λ, see Figure 2.3.

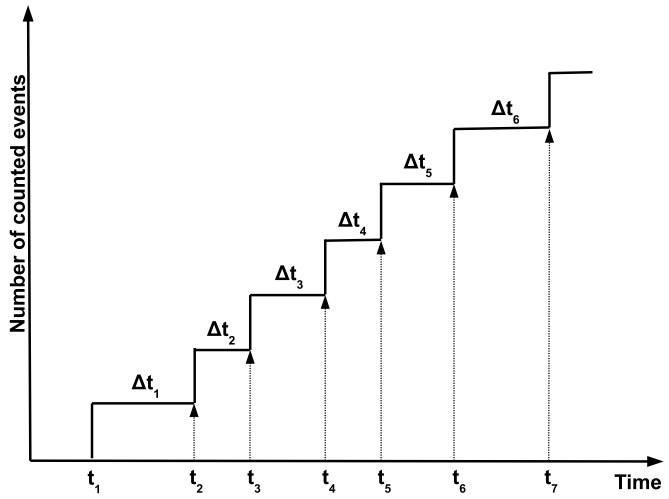


Figure 2.3 – *A Poisson process with its inter-arrival times.*

A Poisson process is defined in Ross (1992) as in the following:
A Poisson process $N(t)$ is a time-continuous stochastic process that counts the number of events and that has the following properties:

- $N(0) = 0$
- $\{N(t), t \geqslant 0\}$ has independent increments
- The number of events in any interval of the length t has a Poisson distribution with the mean $t\lambda$, for all $s, t \geqslant 0$:

$$P\{N(t+s) - N(s) = n\} = e^{-\lambda t}\frac{(\lambda t)^n}{n!}$$

Poisson processes can be categorized into homogeneous processes and non-homogeneous processes. A homogeneous Poisson process is characterized by its rate λ, which is the expected number of events to occur per one time unit. The number of events within an interval of length τ: $[t, t+\tau]$ has a Poisson distribution with rate $\lambda\tau$:

$$P[N(t+\tau) - N(t) = k] = \frac{e^{-\lambda\tau}(\lambda\tau)^k}{k!} \qquad k = 1, 2, \ldots$$

where $N(t+\tau) - N(t) = k$ is the number of events occurred in the time interval $[t, t+\tau]$.

In cases where the rate of the Poisson process may change over time, $\lambda(t)$, the process is referred to as a non-homogeneous Poisson process. The expected number of events, in this case, between the time $t = a$ and $t = b$ is calculated as in the following:

$$\lambda_{a,b} = \int_a^b \lambda(t)dt$$

For more about Poisson processes please refer to Norris (1998) and Ross (1992). Since, mathematical modeling was not intensively investigated in studying and designing swarm robotics systems so far, there exist only a few studies which apply Poisson processes models in designing swarm robotics. Some examples are as in Galstyan & Lerman (2005), where the authors have presented a simple stochastic model for adaptive task allocation in a team of robots. The considered task was a foraging of two distinct types of pucks, Red and Green scattered around the arena. The process of encountering a puck was modeled as a Poisson process with a specific rate for each kind of the pucks. Another example is a biologically inspired redistribution of swarm robots among multiple sites in Hsieh *et al.* (2008). The authors have investigated the Poisson process to model the transition of agents between the different sites in order to perform a top-down design of the system. In Hsieh *et al.* (2009) the authors have studied the role of specialization while executing a set of collaborative tasks by a swarm of robots, specially the well-known "stick-pulling" task. They have shown how specialization is useful when external conditions change. The individual robot dynamics were described by a set of states where all transitions were governed by Poisson processes. The process of the roaming robot to encounter a stick was modeled as a Poisson process whose rate represents the probability per time unit to discover a stick. Another example is the foraging task considered in Guerrero & Oliver (2011), where an auction strategy was discussed for performing a task allocation of swarm robotics in real time scenarios. The arrival of new objects to the environment was modeled as a Poisson process.

Chapter 3

Related Work

Research is to see what everybody else has seen, and to think what nobody else has thought.

- Albert Szent-Gyorgyi

This chapter presents the state of the art concerning the mathematical modeling of swarm robotics systems on its two levels: the macroscopic modeling and the microscopic modeling. In addition, the chapter goes through the different studies performed on the task allocation in multi-robot systems, swarm robotics systems and special approaches considering task allocation in swarm robotics under time constraints.

3.1 Mathematical Modeling of Swarm Robotics

One of the main challenges when designing swarm robotics systems is to understand the effect of the behavior of individual robots on the global behavior of the swarm. Designing a large-scale, stochastic system such as swarm robotics with the goal of obtaining a desired global behavior, is a non-trivial process. Design methodologies include: Real experiments, Simulations and Mathematical modeling. Real experiments represent an expensive methodology for designing swarm robotics, as large numbers of real robots should be used. Simulations, on the other hand, are faster and much more reliable. However, they are mostly time consuming and generalizing their results requires an exhaustive scan of the complete parameter space. Mathematical modeling forms an alternative method which is fast and accurate to predict the long-term behavior of the swarm. Moreover, it can be used to select the design parameters which optimize the system performance. This kind of modeling can be applied on two levels: Macroscopic modeling and Microscopic modeling. Macroscopic models are those which consider the whole system as a unit and focus on modeling its long-term behavior, where microscopic models treat the individual robots as the fundamental units.

In Lerman *et al.* (2005), the authors have described the design difficulties associated with real experiments and simulations, before they highlight the importance of mathematical modeling for designing systems such as swarm robotics. They have provided a recipe for constructing macroscopic models based on rate equations. The recipe can be summarized in two steps: First, to capture the system by a finite state automaton (FSA). Second, to translate the model into rate equations, where each state of the FSA becomes a dynamic variable with its own rate equation. One important remark done in this study was concerning the advantage of using Markov processes in modeling swarm robotics systems, as the robots are assumed to have a reactive behavior and their future actions are based only on what is sensed currently. Martinoli & Easton (2003a) is another study where the authors have described the microscopic and macroscopic modeling for several kinds of experiments. They focused on the experiment of pulling sticks from the ground, where two robots are needed to pull a stick.

Hamann *et al.* (2008) has introduced two macroscopic models for spatial distribution of robots over a known area. The authors have presented, on one hand a rough distribution of the area in sub-areas, and on the other hand a continuous representation of the space. In Lerman *et al.* (2001), the authors have presented a macroscopic model of the cooperative task of pulling sticks out of ground holes. It was assumed that a stick cannot be pulled by a single robot, thus the task represents a cooperative one. The proposed model assumed no explicit communications among the robots. The states of the macroscopic model represented the increasing size of the robots' group. Pulling a stick required "g_r" robots. The system starts by exploring the sticks. When a robot finds one stick, it moves to the state called "g_1". It waits for a random time and if no other robot arrives, the robot makes a transition back to the searching state, otherwise to the state "g_2" and so on. When the system reaches state "g_r", the task can be executed successfully. The model assumed a *static environment*, where the number of sticks remains constant in the area. Another macroscopic model is to be found in Soysal & Şahin (2006). This paper has focused on the aggregation task, which is a well-known behavior in insect colonies where the individuals aggregate on specific locations. The three behavior states defined for the task were: Random walk, Aggregate and Waiting. In the random walk, the robot roams randomly in the arena for a specified period of time. At the end of this period, the robot switches to the aggregate state in case it finds an aggregation group. Otherwise, the robot switches to waiting state where it forms a one-robot aggregating group. The robot leaves the wait state based on a specific transition probability of the model. The proposed macroscopic model defined the aggregation of m robots as *m-aggregation* or C_m. The states of the model were represented as the different possible aggregations: $C_1 \dots C_N$, where N is the number of robots. The transition probabilities were referred to by *shrinking probability* which is the probability of one robot to move out of the aggregation and *the probability of growth* which is the probability of a robot to join the aggregation. In Agassounon & Martinoli (2002b), another macroscopic model for analyzing the aggregation behavior in swarm robotics was presented. Lerman & Galstyan (2002) has presented a macroscopic mathematical model of the well-studied foraging task with a special focus on physical interferences effect on the swarm performance. The model consisted of states related to the robots' behavior. It represented a series of coupled

differential equations, one for each state to describe the average number of robots at that state. The authors were able at the end to extract several factors which affect the swarm performance, mainly the size of the group, in addition to the complexity of the environment and the spatial distribution of the task.

Macroscopic mathematical models were also applied for optimization purposes as in Martinoli & Easton (2003b). In this study, the authors have presented a time-discrete macroscopic model to derive quantitatively correct predictions about the collaboration dynamics of a specific distributed manipulation experiment. This model can be used as a tool to estimate optimal parameters of the robotic system as a function of the environment or the task conditions.

Microscopic modeling is the other modeling level of swarm robotics systems (Galstyan *et al.*, 2005). As mentioned above, the main difficulty in designing such systems is to understand the relation between the individual behaviors and the overall global behavior of the system. The authors highlighted the goal of evaluating the design trade-off(s) for robots in poorly characterized environments like for nano-robotic systems. They modeled the system on the agent level, where they assumed that agents interact with the environment through stigmergy. The transition between the states didn't depend only on the states but on the spatial coordination and concentration of the chemical field used for the stigmergy communications. The robots' behaviors were described by transition rates. In Ijspeert *et al.* (2001), the authors were able to predict the collaboration dynamics, namely, the rate of successful robots' collaboration in a stick pulling experiment using a microscopic model of the system. The designed microscopic model described the finite state automata (FSA) of the robot's controller. Another study is Jeanson *et al.* (2005), where the behavior of the cockroaches was studied and the authors tried to prove that the global aggregation emerges from the local interactions. Experiments were carried out to measure important parameters like the probability to stop in an aggregation or the probability to move. After that, the measured probabilities were exploited in the developed model. Although the real experiments didn't agree with the numerical model, the authors claimed that it provides an evidence that aggregation can be represented in terms of local interactions among individuals. A time-discrete, incremental methodology was presented for modeling manipulation experiments in Martinoli *et al.* (2004). First, a specific time unit was defined for the model, after

that the transition probabilities between the states were calculated per time unit using real-robots experiments.
As noticed from the above listed works, most of the mathematical modeling applied in swarm robotics systems focus on the robot actions as a stochastic process on the microscopic level and on the average number of robots (swarm) in the different system states on the macroscopic level. To our best knowledge, no mathematical modeling was applied to designing a swarm robotics system in order to satisfy a global performance property or constraint.

3.2 Task Allocation in Multi-robot Systems

3.2.1 General Task Allocation

Task allocation represents a main focus in multi-robot systems, which gained a lot of interest in the last years. Nonetheless, most of the efforts have concentrated on finding efficient heuristics and allocation strategies such as in Chevaleyre *et al.* (2006) and in Endriss *et al.* (2005), while theoretical and probabilistic studies were rare in comparison to the experimental studies. One probabilistic analysis has been presented in Lerman *et al.* (2006).
Task allocation can be observed in nature, as by ant and bee colonies (Bonabeau *et al.*, 1998). In such colonies, we can notice how the members allocate themselves to different tasks. For example a part of the colony can perform foraging while another part looks after the larvae and they get specialized over time on their tasks, which improves the performance.
Solutions proposed in the literature for task allocation in multi-robotic systems in general, can be classified into three broad categories: *centralized*, *negotiation-based* and *self-organized*. Centralized techniques assume the presence of a central coordinator responsible for the allocation of the agents (robots) to the tasks. The single point of control causes problems related to the robustness of the solution and to the availability of such a central control over any application environment. Self-organized systems, on the contrary, are constituted by peers that take decisions autonomously, through limited negotiations with other peers and without any central point of control. This kind of systems are generally less prone to catastrophic failures

and are considered as a better approach, when rapid adaptation to changes in the environment, is required. Most of these studies tackle simple scenarios without task interdependencies (Dahl *et al.*, 2009). Negotiation-based approaches, generally based on auction-based strategies, are the compromise solution between centralized and self-organized systems (Dias *et al.*, 2005; Gerkey & Mataric, 2002; Zheng *et al.*, 2006). In auction-based strategies, the robots bid on the announced tasks according to the task characteristics and to their relative capabilities. Robots with the highest bid win the allocation to the announced task. One of the criticizing points of this approach is the assumption that robots have a fully connected network among each other to negotiate and that robots possess complete communication abilities. In many real-world applications, individual robots are not aware of the whole system and can only communicate with their neighborhood.

Classical robotic systems handle task allocation as an optimization problem, where one robot or more come up with an optimal plan for allocating robots to tasks (Gerkey & Mataric, 2003; Goldberg *et al.*, 2003). A comparison between the auction-based approaches and the self-organized ones based on the threshold mechanism can be found in Kalra & Martinoli (2006). This study has shown that the auction-based approaches outperform the self-organized ones when accurate information about the tasks and the self-state is available. On the other hand, if such information is not available accurately, the self-organized approaches perform almost as good as the auction-based ones.

A general taxonomy of task allocation strategies in robotics systems has been presented in Gerkey & Mataric (2004) along three main comparisons: *single-task robots (ST) vs. multi-task robots (MT)*, where single and multiple robots are able to perform only one task at a time; *single-robot tasks (SR) vs. multi-robot tasks(MR)*, where each task needs one robot or more, and the *instantaneous assignment (IA) vs. time-extended assignment (TA)* where it was assumed that the robots and the environment allow only for an instantaneous task allocation with no future planning.

In swarm robotics, *response-threshold* mechanisms are relatively common (Ducatelle *et al.*, 2009a,b; Nouyan *et al.*, 2005). Following the response-threshold approach, each robot is programmed to react to the stimuli associated with the different tasks, and to start the execution of the task when the level of a stimulus exceeds a predefined threshold. Similar to the task allocation studied in Krieger & Billeter (2000),

Agassounon & Martinoli (2002a) has introduced a task allocation for the swarm task foraging. The only difference was that the probability to switch between foraging and resting was a function of the success associated with the last foraging trial, also of the frequency with which other robots were encountered while foraging, or of the perceived density of prey. Another threshold-based algorithm for allocating workers to a given task whose demand evolves dynamically over time was presented in Agassounon *et al.* (2001). Sequentially interdependent tasks which are common in nature were considered in swarm robotics systems, where a self-organised task allocation was developed in Brutschy *et al.* (2012). In Liu *et al.* (2007) a mathematical model for a similar task allocation behavior was introduced. Some works combined the common swarm response-threshold approach with a kind of communication protocol to avoid the need for a central unit as we can find in Zhang *et al.* (2007). Very few works, to our best knowledge, have assumed a target distribution for the robots over the considered tasks to be reached as in McLurkin & Yamins (2005). Here, the authors have investigated different strategies to achieve the target distribution. The main focus of this study, was on evaluating the communication density that results from each of the applied strategies.
In Ferreira-Jr *et al.* (2008), the problem of distributed task allocation was addressed using the distributed constraint optimization problem (DCOP) formalism. Approximate solutions were designed based on theoretical models of division of labour in social insect colonies and an algorithm called Swarm-GAP was proposed. In this context, however, no deadlines were considered and the design of the algorithm was based on the empirical determination of stimulus thresholds associated to the possible decisions of the robots. Results were validated experimentally but no probabilistic considerations were derived.

3.2.2 Task Allocation under Deadlines and Priority Constraints

Only few authors did concentrate on task allocation in multi-robot systems including swarm robotics systems under temporal constraints. *Jose Guerrero* and *Gabriel Oliver* are two of the authors who worked intensively on that problem. They have introduced a series of articles related to task allocation in multi-robot system based on the auction strategy, which was designed to respect the time constraints and

the priorities of the considered tasks. In Guerrero & Oliver (2011), the authors have introduced three approaches. The first approach was called, *SBPA*, in which an auction process was launched as soon as a new task appears or when a robot becomes free of work. The second approach was referred to as *EDFA*, in which the well-known scheduling algorithm *EDF* was exploited to order the tasks based on their deadlines. Thus, when new tasks arrive or a robot becomes free, a new auction round was launched after ordering the tasks based on EDF. The third presented approach was referred to as *SUA*, where the tasks were ordered according to the rule first in, first out FIFO. Furthermore, they have discussed a swarm approach, where robots select their tasks based on their distances to the different tasks without applying any kind of negotiations among them.

In Guerrero & Oliver (2007), the same authors have discussed the task allocation in heterogeneous swarm robotics system for tasks labelled with priorities. The work differentiated between pre-emptive and non-pre-emptive allocation. First, it was assumed that each task is controlled by a leader. The leader estimates the work required to accomplish the task and lets the robots bit with their capabilities to participate on the task. The leader of the task, decides on the size of the group should work on the task based on the task priority, on the estimated size of the task and on the capacity of the robots. After that, the leader launches an auction process to select the required group. In the pre-emptive version of the approach, the leaders of the different tasks were allowed to negotiate, which can lead to exchanging robots between them. The leader is allowed to negotiate with leaders of other tasks, which have a similar or lower priority, in order to obtain additional robots. Exchanging robots can also have the benefit of reducing the density of robots in the overcrowded tasks, which reduces in turn the potential physical interferences between the working robots. Guerrero & Oliver (2010) is a work based on the previous one in Guerrero & Oliver (2007), where the priorities of the tasks were translated in terms of deadlines and the goal was to find the optimal number of robots to allocate to the considered tasks.

From a completely different perspective, tasks with time constraints were discussed in Acebo & de-la Rosa (2008), where a new heuristic, different from the traditional auction process, was introduced. The system was developed to transport containers by a robotic system from a set of starting points to a set of destination points. The

system simulated the waitress work in a bar where they pay more attention to the clients that shout lauder and more intensive. Tasks in this scenario represented the clients and robots represented the waitresses. This approach applied an attraction linear function, in which the tasks become more attractive when their deadlines become nearer or when their distances to the destination points are larger.

In Schneider *et al.* (2005) and Jones *et al.* (2007), market-based task allocation strategies were introduced, where time was the critical constraint. This was considered together with a reward mechanism associated to tasks which were successfully completed.

Chapter 4

Swarm Performance Modeling

> As far as the laws of mathematics refer to reality, they are not certain, and as far as they are certain, they do not refer to reality.
>
> - Albert Einstein

In this chapter we present a significant part of the main contribution of this thesis. As mentioned in the previous chapters, the goal of the thesis is to design task allocation strategies for swarm robotics systems which allow these systems to execute tasks with respect to their temporal constraints. In other words, the robots should be assigned to the time-constrained tasks such that a specific required performance is achieved on each task up to its deadline.

The concept of swarm performance is defined and introduced under the influence of the spatial interferences among robots. After that, and based on the switching costs between tasks, two kinds of task allocation techniques are presented: Static allocation *and* Dynamic allocation.

The static allocation technique is proposed for tasks which are characterized by their high switching costs. The core idea of this technique is to avoid the costs associated with switching tasks by assigning the robots once at the beginning of the execution. The robots allocate themselves to the different tasks autonomously and independently,

where each robot dedicates a random working time to its selected task, referred to as the robot's activity time. The minimum threshold of the total time that should be dedicated to the task up to its deadline represents the constraint of the swarm performance. On the other hand, when switching costs among the tasks can be considered as negligible, a dynamic allocation technique is proposed. The dynamic allocation allows the robots to switch among the tasks during their execution time, which relaxes the constraint of continuing to work on the same task. The minimum amount of work that should be accomplished up to the task deadline, represents the constraint associated with the swarm performance.

For both static and dynamic allocation techniques, a probabilistic analysis for the constrained swarm performance is carried out, in addition to an estimation of the variance that may occur between the required performance and the expected one to be achieved at the task deadline. These are applied later in designing the task allocation strategies.

Figure 4.1 illustrates the main terms associated with the swarm performance, when designing the robots allocation is required to be performed under the time constraints of the tasks.

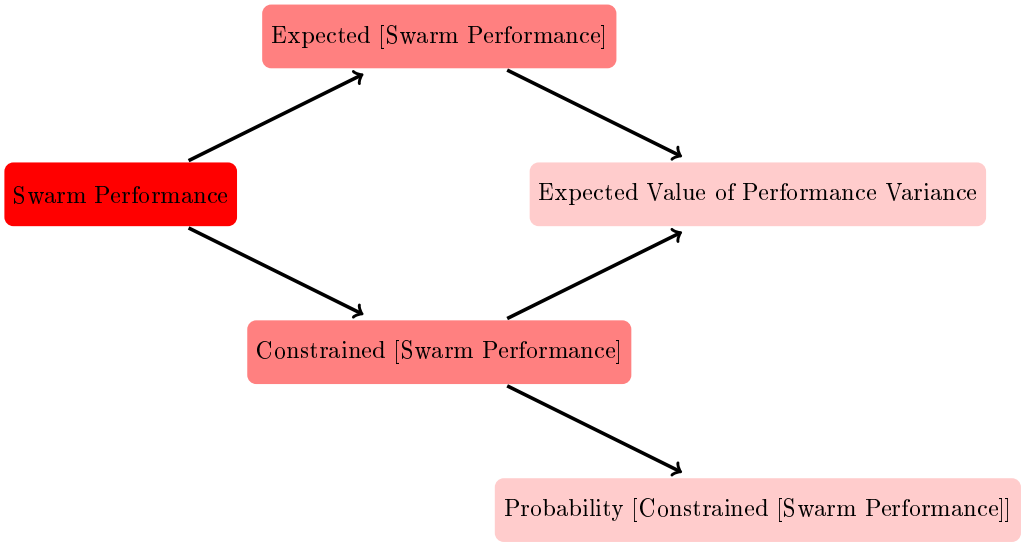


Figure 4.1 *– The swarm performance terms.*

4.1 Chapter Notations

- N: the number of robots used in the swarm (the swarm size).
- N_i: the number of robots assigned to task T_i.
- T_i: the i-th task of the task set which is required to be executed by the swarm.
- D_i: the deadline of task T_i.
- S_i: the size of task T_i. It represents the number of parts that are required to be executed up to the task deadline.
- R_j: the j-th robot in the swarm.
- AT_{ij}: the activity time dedicated by robot R_j to task T_i.
- $AT_{ij}(D_i)$: the part of robot R_j's activity time, which is dedicated to task T_i up to its deadline D_i.
- α_i: It is used under static allocation technique and refers to the minimum threshold of the total time that should be dedicated to task T_i by the robots.
- τ_i: the total time dedicated by a swarm of N_i robots to task T_i.
- $\tau_i(t)$: the total time dedicated by a swarm of N_i robots to task T_i during the time period $[0, t]$.
- $\tau_i(D_i)$: the total time dedicated by a swarm of N_i robots to task T_i up to the deadline, D_i.
- $\tau_i(S_i)$: the time required to accomplish S_i parts of the task T_i.
- μ_i: the mean time required by the swarm to accomplished one part of the task T_i.
- $a_i(t)$: the number of robots which are still active working on task T_i at the time point t.
- tp_k: the time point at which the k-th robot has left task T_i.
- $\theta_i(D_i)$: the total amount of work accomplished on task T_i up to the deadline D_i. It represents the number of parts executed up to the deadline D_i.
- γ_{ik}: the k-th contribution on task T_i that represents the performing of the k-th part on task T_i.

- ψ_i: The variance in the swarm performance that represents the difference between the swarm performance required to be achieved at the deadline of task T_i and the performance expected to be achieved at that deadline.
- $\hat{\mu_i}$: the mean time required by a single robot to accomplish one part of task T_i.
- $\hat{\sigma_i}$: the standard deviation of the random time required by a single robot to accomplish one part of task T_i.

4.2 Swarm Performance

The Performance in swarm robotics systems is defined based on the criteria and goals associated with the tasks on which the performance is measured. For example in foraging, if the task is to *"collect as many objects as possible during* 10 *minutes"*, the performance is measured in terms of the number of objects that will be collected during the 10 minutes deadline. On the other hand, if the task is to *"collect* 5 *objects as quickly as possible"*, the performance is measured in terms of the time that will be required to retrieve the 5 objects (Balch, 1998). Generally, we can introduce two kinds of metrics in which the swarm performance can be expressed, the *quantitative* metric and the *temporal* metric.

In coverage tasks for example, the goal is normally to cover the largest possible area. The quantitative metric used to measure the swarm performance is related to the area covered by the robots during a given time (Fazli *et al.*, 2012; Wong *et al.*, 2002). In the literature, two classes of coverage problems have been discussed: The single coverage, where the goal is to cover all the points of interest by one visit at least, while minimizing the required time, the distance of travel, and the number of visits to the points. The other class is known as the repeated coverage, where the goal is to cover all points of interest repeatedly over the time while maximizing the frequency of visiting points and minimizing the sum of the length of the robot tours. Here, we can consider two metrics to measure the swarm performance: A quantitative metric which is related to the number of visited points and the distance traveled by the robots, and a temporal metric which can be defined as the time required to cover a specific area.

Another example is given by navigation tasks which aim to control the robots' travel paths between start and destination points, exploiting the robots' sensing system to compute the next motion. In Ceballos *et al.* (2010) the authors have listed the performance criteria related to the navigation tasks including: the number of successful missions, the path length, the number of collisions per mission, per distance or per time, the mean distance to obstacle, the number of narrow passages which are passed successfully, the smoothness of the trajectory, and the time taken to accomplish the task. As we can notice, all the mentioned criteria represent

quantitative metrics to measure the robots' performance except the time needed to accomplish the task which represents a temporal metric.

Foraging is one of the intensively studied tasks in swarm robotics systems. As explained in Chapter 2, Section 2.1.1, foraging is inspired by social insects, where the individuals search for food resources to retrieve food back to an area referred to as the home. Foraging can be mapped to several real-world applications among them are: mining operations, explosive ordnance disposal, foraging for recycling of materials, and specimen collection in hazardous environments (Balch, 1999a). Swarm performance on foraging tasks can be measured, using quantitative metrics as well as temporal ones. The quantitative metric is associated with the number of items retrieved per time unit, whereas the temporal metric is associated with the time required to retrieve a specific number of items. In Balch (1999a), the authors have studied different strategies of multi-foraging scenarios to evaluate the swarm performance associated with each strategy. Multi-foraging is defined as the task in which robots are responsible to retrieve different types of items. They applied first the strategy of simple foraging, where all robots are allowed to retrieve from any type of the available items. After that, they employed the specialization strategy, where different groups of robots specialize to retrieve from specific types of items. Finally, they tried a strategy referred to as territorial foraging, where all robots work in searching and retrieving tasks except for one robot, which waits at the home region to pick the retrieved items which are dropped on the border of the home and transports them inside the home. This technique aims to reduce the influence of physical interferences among robots in the home region. The paper studied the performance under diversity (size of the robot group) and the performance expressed in its quantitative term, which represents the number of retrieved items. In other foraging studies, the performance was measured using other metrics such as the energy cost of foraging. However, these metrics represented indirect measurement of the number of retrieved objects as it is calculated in Winfield (2009).

In Balch (1998), the authors have introduced two taxonomies, one to characterize the multi-robot tasks and another to characterize the multi-robot reward. They stated that the reward function can be defined as *performance* if and only if the maximum reward implies optimal performance. They discussed the possible relation between reward and performance such as the task of docking a boat. In such a task

the performance was defined as $P = -elapsed\_time$ and it was considered as an appropriate performance-based reward task because the reward was maximized if and only if the performance was maximized. On the other hand, for tasks such as foraging where the reward does not represent a direct result of the performance, they introduced the concept of *heuristic* rewards. Heuristic rewards are based on the intuition of the value of a robot action in particular states.
In this thesis, we focus on the performance achieved by a swarm of robots under specific time constraints. This perspective of the swarm performance combines the two metrics defined above for measuring the performance, namely the quantitative and the temporal. We don't aim to maximize the quantity within a particular time period, such as the number of points to visit or the number of objects to retrieve. Neither do we want to minimize the time required by the swarm to achieve a specific quantity of performance. But rather our goal is to achieve a specific quantity of the performance within a defined time period referred to as the *deadline*. This kind of performance is referred to, here, as the *time-constrained* performance of the swarm.

4.3 Swarm Performance under Spatial Interferences

Spatial interferences are the interferences caused by the competition among robots to occupy a shared physical area or space where they are operating. A high density of spacial interferences can be observed in swarm robotics systems according to the large number of robots which typically build up these systems. Several types of interferences in multi-robotic systems have been defined in Goldberg (2001). The work presented the interactions among robots working together in a common area as the main type of robots interactions. The authors have proposed two techniques to arbitrate the impact of interactions. First, by making sure that robots are working in different areas and second, by scheduling the access to the shared areas. The first proposal was further investigated under the term *bucket-brigade* examples can be found in Shell & Mataric (2006), Lein & Vaughan (2008a), and Østergaard *et al.* (2001). In addition to Lein & Vaughan (2008b), where the approach was extended to consider adaptive working areas.
The influence of the spacial interferences in swarm robotics systems is observed on both single robot performance and global swarm performance. In Lerman &

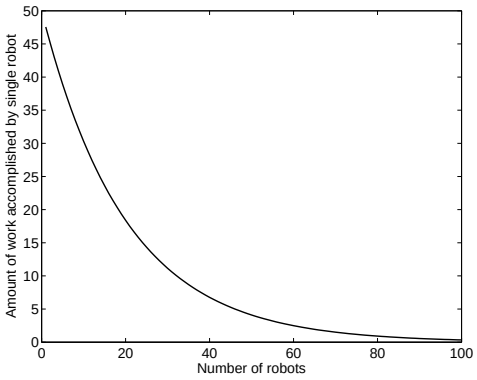
Galstyan (2002), the authors have presented a mathematical model of the foraging task in swarm robotics through which they have characterized the performance of the single robot and the swarm under spacial interferences.
In cases where the swarm performance is defined in terms of the amount of work accomplished during a specific time unit, increasing the number of robots, decreases the amount of work accomplished by the single robot. However, the amount of work accomplished by a swarm increases up to a maximum amount before it starts to decrease again by adding more robots. In case of the single robot, adding robots increases the time the single robot spends in avoiding (interfering with) other robots. This decreases, in turn, the time during which the robot is working. Consequently, the amount of work accomplished by the robot during a specific time unit decreases. From the swarm point of view, the amount of work accomplished during a specified time unit keeps increasing as long as the gain of parallelizing the work among many robots, is greater than the time cost paid in escaping from the interferences. This amount starts to decrease as soon as the time lost by robots in avoiding each other becomes greater than the time gained by parallelizing the work. This result was reported, experimentally, in several studies as in Lerman & Galstyan (2002) and Østergaard *et al.* (2001). Based on the reported results, the single robot performance under the influence of spatial interferences can be modeled mathematically using the *exponential functions*. The parameters of mathematical model can be determined through short-term experiments on the considered tasks.
Let consider an example where the single robot performance is modeled using the following exponential function:

$$f_{single}(x) = ae^{-bx} \qquad \text{where } x \text{ is the swarm size}$$

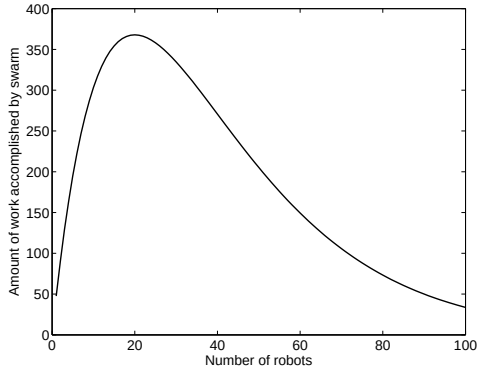
we set $a = 50$ and $b = 0.05$.
Swarm performance is the amount of work accomplished by the swarm during the time unit. This performance is calculated using the single robots performance as in the following:

$$\begin{aligned} f_{swarm}(x) &= x f_{single}(x) \\ &= x \times ae^{-bx} \qquad \text{where } x \text{ is the swarm size} \end{aligned} \tag{4.1}$$

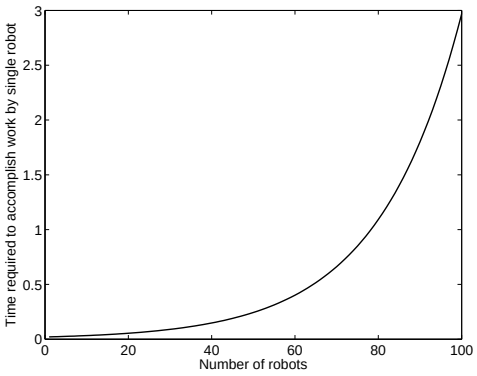
Figure 4.2 (a) shows the single robot performance when the size of swarm is increased over the range $[1, 100]$ robot and 4.2 (b) illustrates the calculated swarm performance using Equation (4.1).
When the swarm performance is defined in terms of the time required to accomplish a specific work amount, increasing the number of robots increases the time required by a single robot to accomplish the specified amount of work. For the same reason mentioned above, increasing the number of robots will decrease the time during which the robot is in working mode and consequently will increase the time it requires to accomplish the required amount of work. In the case of a swarm, the time required to accomplish a specific amount of work decreases by increasing the robots due to parallelizing the work until a minimum time is reached. After that, it starts to increase as soon as the time dedicated by robots to avoid interferences becomes greater than the time gained by parallelizing the work. Figures 4.2 (c) and 4.2 (d) show the single robot performance and the swarm performance in terms of the time required to accomplish one part of the work, calculated for the above example.
Since, the goal for time-constrained tasks is to achieve a specific swarm performance at a particular deadline and as the swarm performance is directly influenced by spatial interferences among the robots, a successful planning of the tasks execution should take this influence into account.



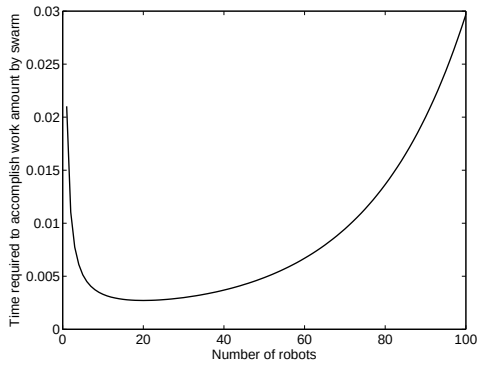
(a) *Amount of work accomplished by a single robot within time unit.*



(b) *Amount of work accomplished by the swarm of robots within time unit.*



(c) *Time required to accomplish a specific amount of work by a single robot.*



(d) *Time required to accomplish a specific amount of work by the swarm.*

Figure 4.2 *– The single robot performance and the swarm performance under the influence of spatial interferences.*

4.4 Static Task Allocation

Reassigning robots among tasks during their execution times require the robots to travel among the different tasks, when these are located in different physical sites. Such tasks were introduced in Chapter 1, Section 1.1 to have high switching costs. For time-constrained tasks, since the time at which the task is finished plays the main role in the correctness (quality) of the results, spending the time in switching between tasks does not provide an efficient solution. Therefore, switching costs should be eliminated through preventing the reassignment of the robots among the tasks during there execution time.

The allocation technique we propose, here, is referred to as *static allocation*. The core idea of static allocation is to assign the robots to the different tasks at the beginning of the execution and to prevent them from switching among the tasks during the execution time. The problem to solve here, is two folds: First, finding out the proper number of robots to assign to each of the considered tasks. Second, specifying the time duration which each robot should dedicate to the task it is working on. These two parameters should be designed with respect to the constraint of accomplishing the required size of each task up to its deadline. The time dedicated by each individual robot to its selected task is referred to as the activity time of the robot. This activity time represents a random time duration which is sampled independently by the robots without any central control. It is not necessary that each robot works on the task up to its deadline, but rather, the total time dedicated to the task should respect the size of the task to execute up to its deadline. The robot's activity time includes the actual working time in addition to the time spent by the robot in a non-working mode as the time spent in performing obstacle avoidance or experiencing physical interferences with other robots. AT_{ij} is used to denote the activity time sampled by robot R_j on task T_i. As soon as the robot samples its activity time it starts to work on the task and becomes an *active* robot. When the activity time of this robot expires the robot stops to work and becomes an *inactive* robot. Figure 4.3 illustrates an example of the static allocation where each robot samples its own activity time. At the beginning of the activity time the robot becomes active on the selected task, it works until the end of its activity time at which it becomes inactive.

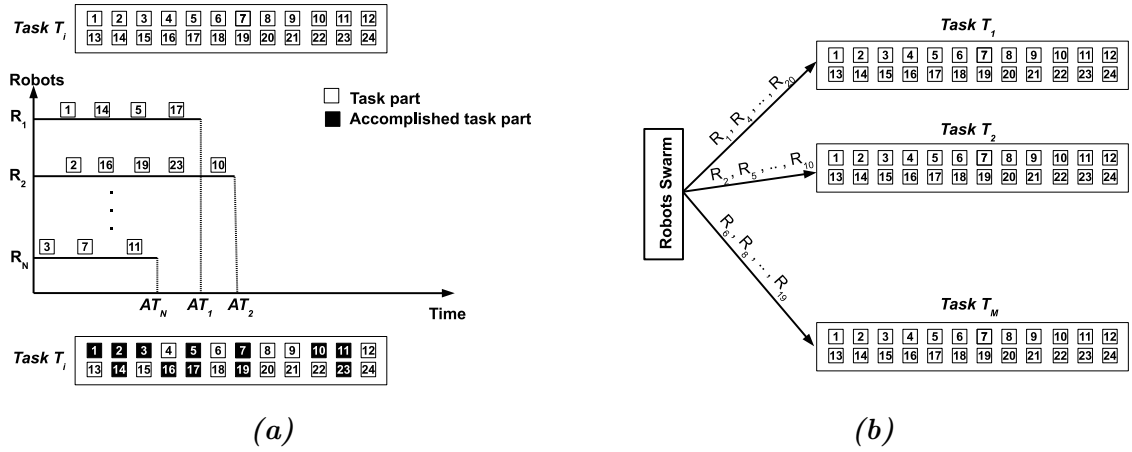


Figure 4.3 – *Figure (a) illustrates the static allocation where the robots select their tasks from M considered tasks. Figure (b) illustrates the work of the robots during their activity times on task T_i.*

Robots' activity times are continuous random variables which are sampled independently by the robots under the constraint of dedicating enough time to the task up to its deadline. On the other hand, the time required by a robot to accomplish one part of its selected task is a continuous random variable which is influenced by several factors including the physical interferences among robots, the distribution of the task parts and the environment complexity. This time is averaged over short-term robotic experiments or simulations. The mean of this random time is used later to calculate the minimum threshold of the total time required to be dedicated to the task up to its deadline.

In the following, we assign the proper definitions of the swarm performance terms in Figure 4.1 for the case of static allocation:

- Swarm performance: In the context of static allocation, swarm performance is defined as the total (cumulative) time assigned by the robots to their selected task up to its deadline. This time is the sum of the activity times of all robots which have selected this task to work on. Let us denote the total time dedicated to task T_i by τ_i, then τ_i can be calculated as follows:

$$\tau_i = \sum_{j=1}^{N_i} AT_{ij}$$

where N_i is the number of the robots assigned to task T_i.

The performance of interest in the case of time-constrained tasks is the performance achieved by the swarm at the task deadline. This performance is the sum of all time intervals of the robots' activity times, which are included in the task deadline.

The total time dedicated to task T_i up to the deadline D_i is calculated as:

$$\begin{aligned} \tau_i(D_i) &= \sum_{j=1}^{N_i} AT_{ij}(D_i) \\ &= \sum_{j=1}^{N_i} min\{AT_{ij}, D_i\} \end{aligned} \tag{4.2}$$

- Constrained swarm performance: Under static allocation, the constraint of swarm performance is associated with the total time dedicated to the task up to its deadline. Let us use α_i to denote the minimum threshold of the total time that should be dedicated to task T_i by the robots working on it. α_i represents the total time required to accomplish at least S_i parts of task T_i at the deadline D_i. The constraint associated with the swarm performance on task T_i can be written as:

$$\tau_i(D_i) > \alpha_i \Longleftrightarrow \sum_{j=1}^{N_i} AT_{ij}(D_i) > \alpha_i \tag{4.3}$$

In the following, we explain how we calculate the minimum threshold α_i of the total time needed to be dedicated to task T_i. Let us denote the number of parts accomplished by robot R_j on task T_i during the robot's activity time AT_{ij} by S_{ij} and the random time required by a robot to accomplish one part of T_i by φ_i, then we have:

$$AT_{ij} \geqslant S_{ij}\varphi_i$$

Now we can calculate the sum of the robots' activity times as in the following:

$$\begin{aligned} \sum_{j=1}^{N_i} AT_{ij} &\geqslant \sum_{j=1}^{N_i} S_{ij}\varphi_i \\ &\geqslant \varphi_i \sum_{j=1}^{N_i} S_{ij} \end{aligned} \tag{4.4}$$

The number of parts we aim to accomplish on task T_i up to its deadline D_i should be equal to or greater than the size S_i, therefore we have:

$$\sum_{j=1}^{N_i} S_{ij} \geqslant S_i \tag{4.5}$$

By substituting (4.5) in (4.4) we have:

$$\sum_{j=1}^{N_i} AT_{ij} \geqslant S_i \varphi_i \tag{4.6}$$

Therefore, the allocation strategy aims to dedicate a total time with the minimum threshold $S_i\varphi_i$ to task T_i. By using the mean $\hat{\mu}_i$ of the random time φ_i required by the robot to accomplish one part of task T_i when N_i robots are assigned to the task, we can approximate the minimum threshold as in the following:

$$\alpha_i = S_i \hat{\mu}_i(N_i) \tag{4.7}$$

- Expected swarm performance: this calculates the expected value of the swarm performance that will be achieved at the deadline D_i. Since swarm performance under static allocation is defined as the total time dedicated to the task up to its deadline, the expected value of this performance is the expected value of the random variable associated with the sum of the robots' activity times:

$$\mathbb{E}(\sum_{j=1}^{N_i} AT_{ij}) = \sum_{j=1}^{N_i} \mathbb{E}(AT_{ij}) \tag{4.8}$$

- Probability of the constrained swarm performance: this represents the probability of the constraint in Equation (4.3):

$$\Pr(\tau_i(D_i) > \alpha_i) = \Pr(\sum_{j=1}^{N_i} AT_{ij}(D_i) > \alpha_i) \tag{4.9}$$

This probability will be characterized by its density function in addition to its cumulative distribution function, see Chapter 2, Section 2.2.1 for for more details about these two probability functions.

- Expected variance in swarm performance: this estimates the difference between the minimum threshold of the performance required to be achieved at the deadline D_i and the expected performance to obtain at D_i.

$$\begin{aligned} \mathbb{E}(\psi_i) &= \alpha_i - \mathbb{E}(\sum_{j=1}^{N_i} AT_{ij}) \\ &= \alpha_i - \sum_{j=1}^{N_i} \mathbb{E}(AT_{ij}) \end{aligned} \tag{4.10}$$

Figure 4.4 illustrates the substitution of the terms definitions, which were originally depicted in Figure 4.1 for the case of static allocation.

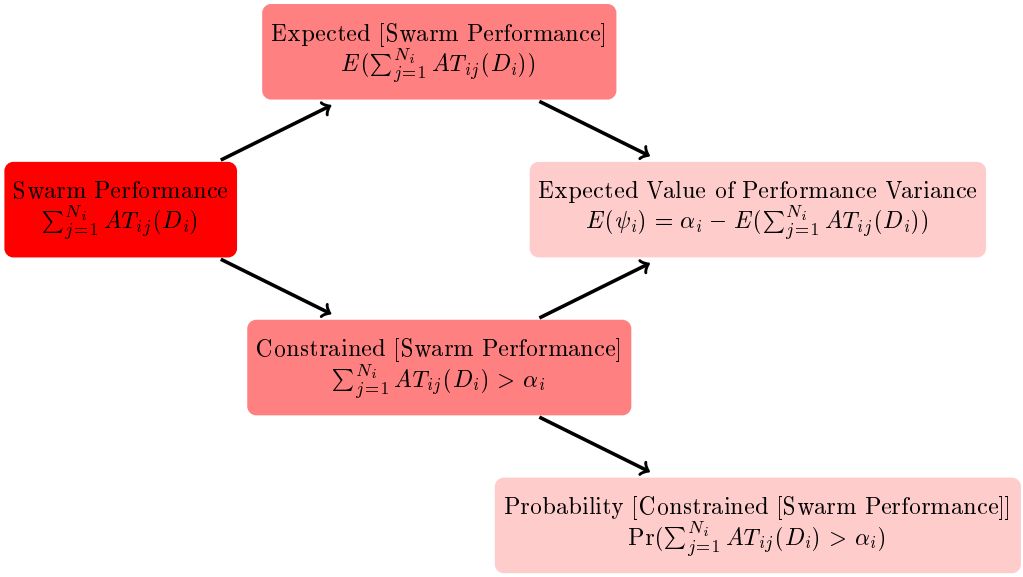


Figure 4.4 – *Assign the proper definitions to the swarm performance terms under static allocation.*

4.4.1 Swarm Performance Modeling under Static Allocation

The robots' activity times are, as mentioned above, random time durations, which are sampled independently as a set of continuous independent random variables. There are several probability distributions which can be used to model these activity times. In the following, we list some of the suitable probability distributions used in

this thesis to model the robot's activity times. After that, we introduce a probability analysis of the swarm performance under static allocation. Finally an estimation of the performance variance as the difference between the required swarm performance and the expected one to achieve, is performed.

A) **Exponential Distribution:**

The exponential distribution is a well-known distribution for modeling continuous random variables such as the length of the events and the inter-arrival times. It is widely used to model many real-world phenomena. Queuing systems provide an important example where the exponential distribution is investigated intensively. In queuing systems, the customer's service times are modeled typically as exponential random variables.

The exponential distribution can be characterized by its probability density function which was given in Chapter 2, Section 2.2.1 by:

$$f_{exp}(t) = \begin{cases} \lambda e^{-\lambda t} & \text{if } t \geqslant 0 \\ 0 & \text{if } t < 0 \end{cases}$$

where λ is the rate parameter of the exponential distribution.

The problem of executing a set of time-constrained tasks by a swarm robotics system, can be mapped to a queuing system, where the tasks represent the customers who need to be served and the respective servers are the robots. The service time which the customer spends in the server represents in our case the time dedicated by the robot to the task, namely, the robot's activity time. Similarly to the distribution used to model the service time in queuing systems, the exponential distribution counts as an appropriate distribution to model the robots' activity times. The rate parameter λ of the exponential distribution used to model the robot's activity times is a task-specific parameter, which differs among the considered tasks:

$$AT_{ij} \sim \mathcal{E}xp(\lambda_i) \qquad i \in \{1, \ldots, M\}$$

The sum of n independent exponential variables follows the Gamma distribution (Ross, 2006) and as n is an integer, the Gamma distribution is substituted

with the Erlang distribution, whose probability density function is given by:

$$f_{E(\lambda,k)}(x) = \frac{\lambda e^{-\lambda x}(\lambda x)^{k-1}}{(k-1)!} \tag{4.11}$$

The cumulative distribution function of the Erlang distribution, is given by:

$$F_{E(\lambda,k)}(x) = 1 - \sum_{n=0}^{k-1} \frac{1}{n!} e^{-\lambda x}(\lambda x)^n \tag{4.12}$$

When the robots' activity times dedicated to task T_i are modeled as exponentially distributed variables, the sum of these activity times is distributed following the Erlang distribution:

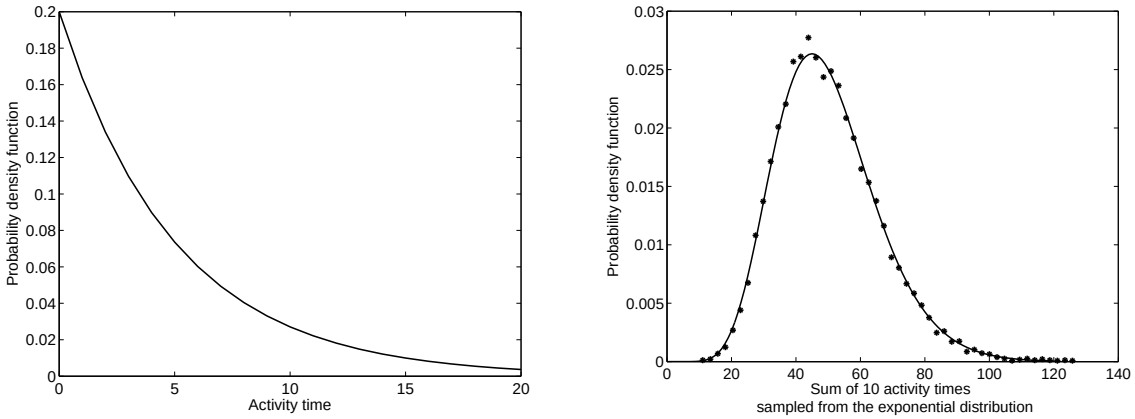
$$\sum_{j=1}^{N_i} AT_{ij} \sim \mathcal{E}rlang(\lambda_i, N_i)$$

λ_i is the rate parameter of the Erlang distribution and N_i is its shape.

The probability of dedicating a total time to task T_i, that is greater than the threshold α_i when the robots sample their activity times from an exponential distribution with rate λ_i, is calculated using the cumulative distribution function of the Erlang distribution as in the following:

$$\begin{aligned}
\Pr(\sum_{j=1}^{N_i} AT_{ij} > \alpha_i) &= 1 - \Pr(\sum_{j=1}^{N_i} AT_{ij} \leqslant \alpha_i) \\
&= 1 - F_{E(\lambda_i,N_i)}(x) \\
&= 1 - [1 - \sum_{n=0}^{N_i-1} \frac{1}{n!} e^{-\lambda_i\alpha_i}(\lambda_i\alpha_i)^n] \\
&= \sum_{n=0}^{N_i-1} \frac{1}{n!} e^{-\lambda_i\alpha_i}(\lambda_i\alpha_i)^n
\end{aligned} \tag{4.13}$$

Equation (4.13) calculates the probability of having the total time τ_i dedicated by N_i robots to task T_i, being greater than the threshold α_i. Later on we will calculate the probability of interest in Equation (4.9), namely, the probability of having the total time $\tau_i(D_i)$ dedicated to T_i up to the deadline D_i, being greater than the threshold α_i.



(a) *(b)*

Figure 4.5 – *Figure (a) illustrates the probability density function of an exponential activity time with the mean* $1/\lambda_i = 5$ *time units. Figure (b) shows the probability density function of the sum of* $N_i = 10$ *exponential r.v. with mean* $1/\lambda_i = 5$ *time unit and the simulated sum with* 10000 *runs of Monte-Carlo simulation.*

Figure 4.5 (a) shows the probability density function of the random variable associated with the robot's activity time when it is sampled from an exponential distribution with the mean of $1/\lambda_i = 5$ time units. Figure 4.5 (b) illustrates the probability density function of the sum of 10 activity times, calculated with Eq (4.12), as well as the distribution obtained with a Monte-Carlo simulation of 10000 runs.

B) **Truncated Exponential Distribution:**

The tasks we are focusing on in this thesis, are time-constrained tasks which should be accomplished up to particular deadlines. The temporal constraints associated with the considered tasks can be divided, as mentioned in Chapter 1, Section 1.1 into hard-deadline and soft-deadline. In the case of soft-deadlines where robots can continue executing the task till it is finished even after its deadline is exceeded, sampling activity times which are longer than the task deadline is feasible. However, in the case of hard-deadlines where there is no profit of continue executing the task after its deadline, sampling activity times which are longer than the deadline is not feasible. Considering such cases where the robots should stop working on the task by reaching its deadline, the

distribution of the robots' activity times should be able to restrict the length of the sampled activity times to the task deadlines.

The truncated exponential distribution is a special case of the exponential distribution where the continuous random variable is defined over a restricted range $[a, b]$. This distribution is appropriate to sample the robots' activity times when the robots should stop to work on the task as soon as its deadline is reached. The robot's activity time which is sampled from the truncated exponential distribution defined over the range $[0, D_i]$, is restricted between 0 and D_i.

The truncated exponential distribution can be characterized by its probability density function which was given in Chapter 2, Section 2.2.1 by:

$$f_{T(a_1,b_1)}(t) = \begin{cases} \dfrac{\lambda e^{-\lambda t}}{e^{-\lambda a_1} - e^{-\lambda b_1}} & a_1 \leqslant t \leqslant b_1 \\ 0 & \text{Otherwise} \end{cases} \tag{4.14}$$

By substitution $x = \lambda t$, Equation (4.14) can be written as:

$$\hat{f}_{T(a,b)}(x) = \begin{cases} \dfrac{\lambda e^{-x}}{e^{-a} - e^{-b}} & a \leqslant x \leqslant b \\ 0 & \text{Otherwise} \end{cases} \tag{4.15}$$

where $a = \lambda a_1, b = \lambda b_1$.

When the activity time of individual robots is modeled as a truncated exponential variable, the probability density function of their sum is provided in Lavender (1967) by using the properties of the characteristic function of the sum of random variables. The main steps of the derivation are resumed here, however, a detailed derivation is to be found in the appendix A. The characteristic function of the probability density function $f(x)$ of the random variable x is defined as:

$$\varphi(t) = \int_{-\infty}^{\infty} e^{itx} f(x) dx \tag{4.16}$$

If $\varphi(t)$ is the characteristic function of $f(x)$, then $\varphi(t+i\lambda)$ is the characteristic function of $e^{-\lambda x}f(x)$ since:

$$\begin{aligned}\varphi(t+i\lambda) &= \int_{-\infty}^{\infty} e^{i(t+i\lambda)x} f(x)dx \\ &= \int_{-\infty}^{\infty} e^{itx}(e^{-\lambda x}) f(x)dx\end{aligned} \tag{4.17}$$

In addition, the characteristic function associated with the distribution of the sum of n independent variables, each with the characteristic function $\varphi(t)$ is: $[\varphi(t)]^n$ (Kendall & Stuart, 1977). This property allows to establish the form of the probability distribution function of the sum of n truncated exponential variables from the expression related to the characteristic function of the sum of n rectangular random variables:

$$\varphi_{n,U(a,b)}(t) = \frac{(e^{itb} - e^{ita})^n}{(b-a)^n (it)^n} \tag{4.18}$$

The expression of the anti-transform of Equation (4.18) is calculated in Cramer (1946):

$$g(s) = (b-a)^{-n} \sum_{k=0}^{m} (-1)^k \binom{n}{k} \frac{[s - na - k(b-a)]^{n-1}}{(n-1)!} \tag{4.19}$$

where $na + m(b-a) < s < na + (m+1)(b-a)$ and $m = 0, 1, .., n-1$.

The probability density function of the sum of n independent random variables, each is distributed with the probability density function $f_U(x)$, is thus given by Equation (4.19).

The characteristic function of a truncated exponential random variable as well as that of the sum of n independent truncated exponential random variables are easily calculated by applying the definition:

$$\varphi_{T(a,b)}(t) = \frac{e^{(it-1)b} - e^{(it-1)a}}{(e^{-a} - e^{-b})(it-1)} \tag{4.20}$$

$$\varphi_{n,T(a,b)}(t) = \frac{(e^{(it-1)b} - e^{(it-1)a})^n}{(e^{-a} - e^{-b})^n (it-1)^n} \tag{4.21}$$

The close resemblance between Equation (4.18) and Equation (4.21), allows for the derivation of the probability density function of the sum of n independent

truncated exponential random variables. In particular, by using the fact that $i(t+i) = it - 1$ we have:

$$\varphi_{n,U(a,b)}(t+i) = \frac{(e^{i(t+i)b} - e^{i(t+i)a})^n}{(b-a)^n(i(t+i))^n} = \frac{(e^{(it-1)b} - e^{(it-1)a})^n}{(b-a)^n(it-1)^n} \tag{4.22}$$

Which is, by applying Equation (4.17), the characteristic function of $g(S)e^{-S}$. Equation (4.22) can be obtained by multiplying Equation (4.21) by a constant factor. It follows that the anti-transform of the characteristic function of the sum of n truncated exponential random variables that we are looking for is:

$$f_{n,T(a,b)}(S) = \frac{(b-a)^n}{(e^{-a} - e^{-b})^n} g(S)e^{-S} \tag{4.23}$$

$$= \frac{1}{(e^{-a} - e^{-b})^n} \sum_{k=0}^{m} (-1)^k \binom{n}{k} \frac{[s - na - k(b-a)]^{n-1}}{(n-1)!} e^{-s} \tag{4.24}$$

where $na + m(b-a) < s < na + (m+1)(b-a)$ and $m = 0, 1, .., n-1$.

Therefore, the sum of robots' activity times, when these are sampled from a truncated exponential distribution with the parameter λ and the bounds a and b, is distributed according to the probability density function in Equation (4.23):

$$\sum_{j=1}^{N_i} AT_{ij} \sim f_{n,T(a,b)}(S) \tag{4.25}$$

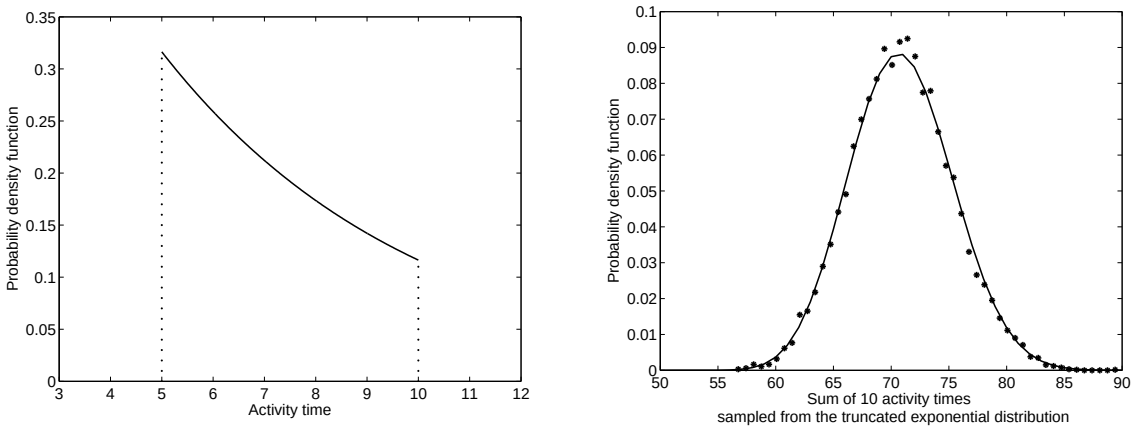
Additional calculations lead to the expression of the cumulative distribution function of the sum of n independent truncated random variables (Lavender, 1967):

$$F_{n,T(a,b)}(s) = \sum_{k=0}^{m0} a_k G(y, n, \lambda) \tag{4.26}$$

Where $G(y, n, \lambda)$ denotes the cumulative distribution function of a Gamma distribution, (Papoulis, 1984) with shape n and parameter λ.

$a_k = (-1)^k \binom{n}{k} [e^{a-b}]^k a_0$ with $a_0 = (\lambda^n)/((1 - e^{a-b})^n)$.

The cumulative distribution function in Equation (4.26) can be used to calculate the probability of having the sum of the robots' activity times greater than the threshold α_i, when the activity times are sampled from the truncated exponential distribution.



(a) *(b)*

Figure 4.6 – *Figure (a) illustrates the probability density function of a truncated exponential r.v. with mean* $1/\lambda = 5$ *and bounds* $a = 5$ *and* $b = 10$ *time units. Figure (b) shows the probability density function of the sum of* $N = 10$ *truncated exponential r.v. with the same parameters and the simulated sum with* 10000 *runs of Monte-Carlo simulation.*

Figure 4.6 (a) shows the probability density function of the random variable associated with the robot's activity time that is sampled from a truncated exponential distribution with $\lambda = 0.2$ and the bounds $a = 5$ and $b = 10$ time units. Figure 4.6 (b) illustrates the probability density function of 10 activity times, which are calculated with Equation (4.23), as well as the distribution obtained with a Monte-Carlo simulation of 10000 runs.

It is evident from Figures 4.5 (b) and 4.6 (b) that when the swarm is composed by a large number of robots, the distribution of the sum of their activity times can be approximated by a normal (Gaussian) distribution. More specifically, when the individual activity times are statistically independent and have an identical distribution (this condition can easily be released) with finite mean μ and standard deviation σ, the distribution of the sum converges to $\mathcal{N}(t; N\mu, N\sigma^2)$ whose probability density function is given by:

$$f_{n,G}(t) = \frac{1}{\sqrt{2\pi N}\sigma} e^{-\frac{(t-N\mu)}{2N\sigma^2}} \tag{4.27}$$

as assured by the central limit theorem (CLT), see for example Durrett (2004).

This allows many of the calculations to be simplified in the general case, as

$N \to \infty$. Figure 4.7 shows the error of the approximation of the closed form probability density function derived for the truncated exponential distribution in Equation (4.23) with the Gaussian distribution, as defined by:

$$\epsilon = \int_0^{\infty} |f_{n,T}(t) - f_{n,G}(t)| dt \tag{4.28}$$

When the number of robots is moderately large, which is the assumption in swarm robotics systems, the Gaussian distribution constitutes a viable approximation for many practical scenarios.

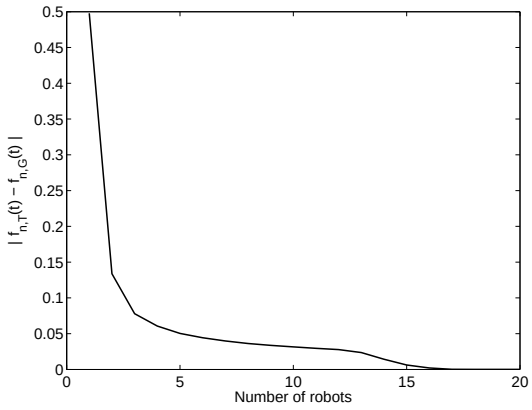


Figure 4.7 – *Error in the Gaussian approximation of Equation (4.23). As assured by the Central Limit Theorem the distribution of the sum converges to a Gaussian distribution.*

C) Arbitrary Distribution:

[*The work in this section was done in collaboration with Michele Pace.*]

The exponential distribution and the truncated exponential distribution, presented above, belong to the most proper distributions for modeling the random variables associated with the robot's activity times. However, probabilistic estimators can be built for cases where the robots' activity times may follow an arbitrary distribution with the probability density function $\pi(t)$. In the case when N_i robots are assigned to task T_i, the total time dedicated to the task will be the sum of the N_i activity times.

The probability density function of the sum of n random variables following any distribution is, by definition, the n-fold convolution of the probability

density function of the used distribution for the individuals. Consequently, the probability density function of the sum of N_i activity times will be the N_i-fold convolution of the density function $\pi(t)$ (Grinstead & Snell, 1998).

On the other hand, the central limit theorem states that in case of an arbitrary distribution with finite mean μ and standard deviation σ, the distribution of the sum of n variables can be well approximated with a Gaussian distribution of the mean $\mu_n = n\mu$ and the standard deviation the $\sigma_n = \sqrt{n}\sigma$ when n is large enough. As an example, Figure 4.10 illustrates the error in the approximation with the probability density function of Gaussian distribution of the sum of n variables, each is distributed according to the non-standard bi-model probability density function shown in Figure 4.8.

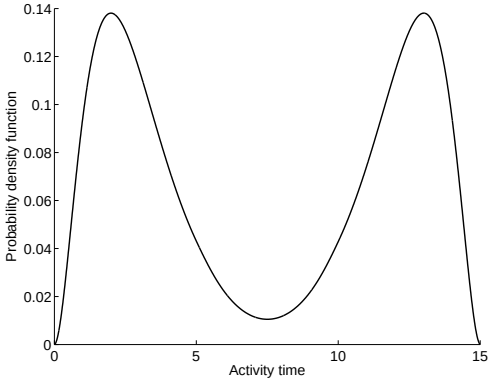


Figure 4.8 – *Non-standard bi-model probability density function obtained as a normalized sum of an Erlang distribution with its reflected and translated image.*

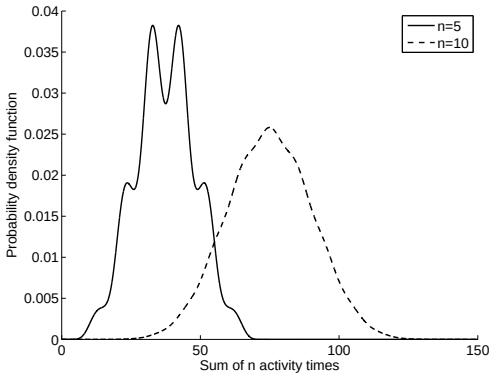


Figure 4.9 – *Convergence to Gaussian distribution of the sum of n random variable distributed according to the PDF in Fig.4.8.*

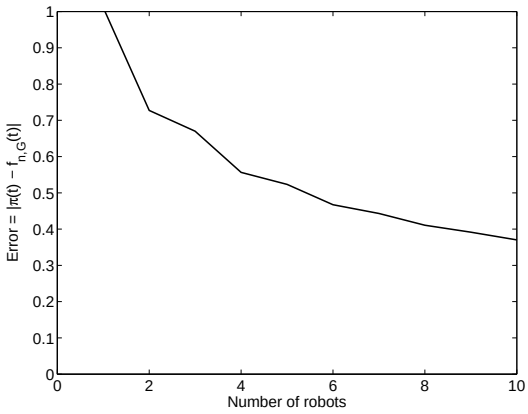


Figure 4.10 – *Error in Gaussian approximation of the distribution of the sum of n random variable distributed as in Fig.4.8.*

4.4.2 Probability Analysis of the Constrained Swarm Performance under Static Allocation

[*The work in this section was done in collaboration with Michele Pace.*]

The results presented in section 4.4.1 characterize, probabilistically, the total time a swarm dedicates to the task when the robots choose their activity times by sampling them from particular probability distributions. After that, each robot works on the selected task for the duration of its activity time before it becomes inactive. Under the time constraints of the task, useful estimators should provide a characterization of the probability that the total time assigned to the task up to its deadline is above a minimum threshold.

Robots make their decisions concerning their activity times without any kind of coordination or central control. For the purpose of probability analysis, we assume to have N_i robots assigned to task T_i. They sample their activity times at the beginning of the execution $t = 0$. The probability of interest is the probability of having the total time dedicated by the N_i robots up to the deadline D_i being greater than the minimum threshold α_i.

Let $a_i(t)$ denotes the number of robots which are still active at time $t \geqslant 0$. $N_i - a_i(t)$ refers to the number of robots which have left at different time points before t:

$tp_k < t$. The total time spent by the robots on task T_i up to time t is given by:

$$\tau_i(t) = a_i(t)t + \sum_{k=1}^{N_i - a_i(t)} tp_k \tag{4.29}$$

The probability we are looking for, can be written as follows:

$$\begin{aligned} \Pr(\tau_i(D_i) > \alpha_i) &= 1 - \Pr(\tau_i(D_i) \leq \alpha_i) \\ &= 1 - \sum_{j=0}^{N_i} \Pr(a_i(D_i) = j) \Pr(\sum_{k=0}^{N_i - j} tp_k \leq \alpha_i - jD_i) \end{aligned} \tag{4.30}$$

We perform the calculation of Equation (4.30) in parts. $\Pr(a_i(D_i) = j)$ is the probability that there are exactly j active robots at time $t = D_i$, where j can take any value over the discrete range $[0, N_i]$. The probability that exactly j robots are active at time D_i can be calculated using the *binomial distribution*[1]. The probability of having exactly k successes in n trials is given by the following probability mass function:

$$f(k, n, p) = \binom{n}{k} p^k (1-p)^{n-k} \tag{4.31}$$

Our success probability p here, is the probability of having the robot's activity time longer than the deadline D_i. Let us assume that the robots' activity times are sampled using the *exponential distribution*, the probability that robot R_j has sampled an activity time AT_{ij} longer than the deadline D_i, can be calculated using the cumulative distribution function of the exponential distribution as in the following:

$$\begin{aligned} \Pr(AT_{ij} > D_i) &= 1 - \Pr(AT_{ij} \leqslant D_i) \\ &= 1 - (1 - e^{-\lambda_i D_i}) \\ &= e^{-\lambda_i D_i} \end{aligned}$$

Consequently, the probability that j robots sample each an activity time longer than the deadline D_i is calculated by using Equation (4.31) after substituting n by N_i, k by j and p by $e^{-\lambda_i D_i}$. based on the binomial distribution as in the following:

$$\Pr(a_i(D_i) = j) = \binom{N_i}{j} e^{-j\lambda_i D_i} (1 - e^{-\lambda_i D_i})^{N_i - j} \tag{4.32}$$

[1] The binomial distribution is the discrete distribution of the number of successes in a sequence of n independent experiments with a binary output. Each of these experiments is a *Bernoulli experiment* with a success probability p and a failure probability $q = 1 - p$.

The second term $\Pr(\sum_{k=1}^{N_i-j} tp_k \leq \alpha_i - jD_i)$ expresses the probability of having the total time dedicated to task T_i by the $N_i - j$ robots which have left before the deadline D_i, being smaller than $\alpha_i - jD_i$. This probability can be calculated using the cumulative distribution function of the random variable associated with the sum of the robots' activity times. Hence, when the activity times are sampled from the exponential distribution, we can use Equation (4.12) to calculate the second probability term of Equation (4.30) as in the following:

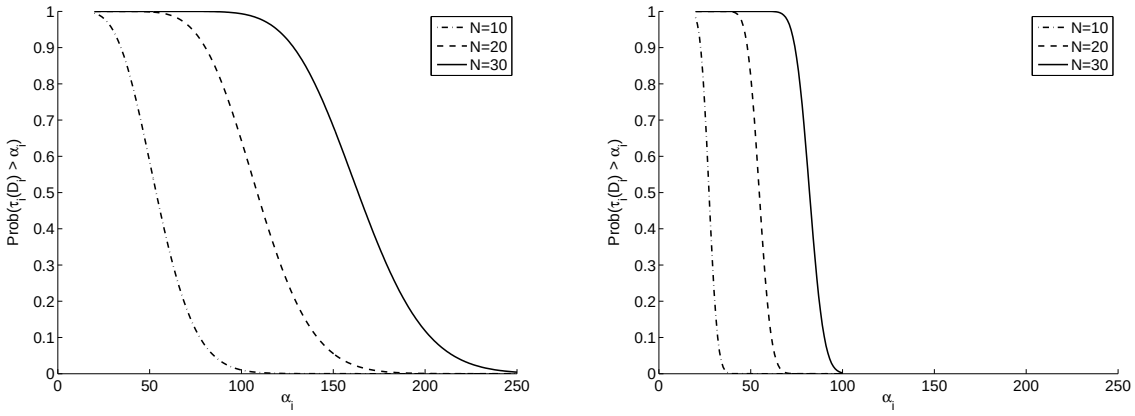
$$\begin{aligned}\Pr(\tau_i(D_i) > \alpha_i) &= 1 - \Pr(\tau_i(D_i) \leq \alpha_i) \\ &= 1 - \sum_{j=0}^{N_i} \binom{N_i}{j} e^{-j\lambda_i D_i} (1 - e^{-\lambda_j D_i})^{N_i - j} F_{E(\lambda_i, N_i)}(\alpha_i - jD_i) \\ &= 1 - \sum_{j=0}^{N_i} [\binom{N_i}{j} e^{-j\lambda_i D_i} (1 - e^{-\lambda_i D_i})^{N_i - j}][1 - \sum_{k=0}^{N_i - 1} \frac{1}{k!} e^{-\lambda_i(\alpha_i - jD_i)} (\lambda_i(\alpha_i - jD_i))^k]\end{aligned} \tag{4.33}$$

Similar considerations allow for the derivation of the probability in case the robots' activity times are sampled from the truncated exponential distribution. The probability that a robot is still active at time $a_1 \leqslant t \leqslant b_1$ is easily calculated from Equation (4.14) and the probability of having the total time dedicated to a task T_i up to its deadline, being greater than the threshold α_i is given by:

$$\Pr(\tau_i(D_i) > \alpha_i) = 1 - \sum_{j=0}^{N_i} \binom{N_i}{j} \frac{(e^{-\lambda_i a_1} - e^{-\lambda_i D_i})^j (e^{-\lambda_i D_i} - e^{-\lambda_i b_1})^{N_i - j}}{(e^{-\lambda_i a_1} - e^{-\lambda_i b_1})_i^N} F_{N_i, T(a_1, b_1)}(\alpha_i - jD_i) \tag{4.34}$$

which can be approximated as well by using the result obtained when applying the central limit theorem and substituting $F_{N_i,T(a_1,b_1)}(\alpha_i - jD_i)$ with the Gaussian distribution, in case N_i is large enough.

Figure 4.11 (a) and 4.11 (b) illustrate the probability of dedicating a total time equal to or greater than the increasing threshold α_i by swarms of different sizes. The robots choose their individual activity times sampled once from an exponential distribution with parameter $\lambda_i = 0.2$ and again from the truncated exponential distribution with parameter $\lambda_i = 0.2$ and the range limits $[a, b]$, where $a = 1$ and $b = 5$ time units. Monte-Carlo simulations are performed and the results are compared with Equation (4.33) and Equation (4.34). Figure 4.12 compares the probability of having the total time dedicated by swarms of different sizes being greater than the increasing



(a) *(b)*

Figure 4.11 – *Probability* $\Pr(\tau_i(D_i) > \alpha_i)$*, for* $D_i = 15$*,* $\alpha_i \in [20, 250]$ *and different number of robots* $N_i = \{10, 20, 30\}$*. Robots sample their activity times in figure (a) from an exponential distribution with parameter* $\lambda_i = 0.2$*. Model (Equation (4.33)) and in figure(b) from a truncated exponential distribution with parameter* $\lambda_i = 0.2$ *and bounds* $a = 1$*,* $b = 5$*. Model (Equation (4.34)). Monte-Carlo simulation is performed with* 100000 *runs.*

thresholds α_i, when robots' activity times are sampled once from the exponential and again from the truncated exponential distribution.

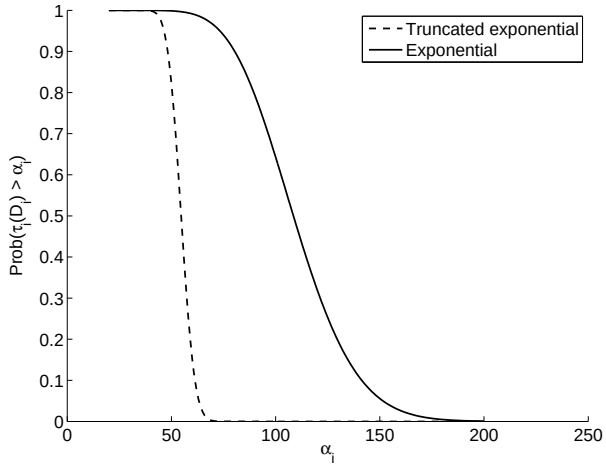


Figure 4.12 – *Probability* $\Pr(\tau_i(D_i) > \alpha_i)$*, for* $D_i = 15$*,* $\alpha_i \in [20, 250]$ *and* $N = 20$ *robots are participating on the task. Robots sample their activity times from: exponential and truncated exponential distributions with parameter* $\lambda_i = 0.2$ *and bounds* $a = 1$*,* $b = 5$*.*

At the end of this section, we are able to calculate the probability associated with the constrained swarm performance by using Equation (4.30). The first part of this equation is calculated using the Bernoulli distribution and its second part is related to the distribution which is used in sampling the robots' activity times. More precisely, the second part of the equation is calculated using the cumulative distribution function of the random variable resulted by summing up the robots' activity times. It is necessary to have the closed form of the cumulative distribution function of the applied distribution in order to derive the probability analysis of the constrained swarm performance.
This probability analysis will be applied later in Chapter 5 to design the task allocation for time-constrained tasks under static allocation.

4.4.3 Performance Variance Estimation under Static Allocation

In this Section, we estimate the variance between the swarm performance required to be achieve at the task deadline and the swarm performance expected to be achieved at the task deadline. The performance variance on task T_i is denoted by ψ_i and it represents the difference between the minimum time that should be dedicated to

task T_i and the total time dedicated. As mentioned above, the total time $\tau_i(D_i)$, dedicated by N_i robots up to the deadline D_i, is the sum of the intervals of the robots' activity times located within the start of the execution and the task deadline, see Equation (4.2). Therefore, the expected value of the performance variance can be calculated using the expected value of the continuous random variable associated with $\tau_i(D_i)$ as in the following:

$$\mathbb{E}(\psi_i) = \begin{cases} \alpha_i - \mathbb{E}(\tau_i(D_i)) & \mathbb{E}(\tau_i(D_i)) < \alpha_i \\ 0 & \mathbb{E}(\tau_i(D_i)) \geqslant \alpha_i \end{cases}$$

The distribution of the random variable $\tau_i(D_i)$ is determined by the distribution of the individual activity times. By substituting the total time dedicated to the task $\tau_i(D_i)$ by the sum of the activity time intervals located within the start of the execution and the deadline D_i, the expected value of the performance variance can be written as follows:

$$\mathbb{E}(\psi_i) = \begin{cases} \alpha_i - \mathbb{E}(\sum_{j=1}^{N_i} AT_{ij}(D_i)) & \mathbb{E}(\sum_{j=1}^{N_i} AT_{ij}(D_i)) < \alpha_i \\ 0 & \mathbb{E}(\sum_{j=1}^{N_i} AT_{ij}(D_i)) \geqslant \alpha_i \end{cases}$$

Let us consider the case, where the robots' activity times are exponentially distributed with the rate parameter λ_i. In this case, the expected value of the robot's activity time is the expected value of an exponentially distributed variable and this is given by:

$$\mathbb{E}(AT_{ij}) = \frac{1}{\lambda_i} \tag{4.35}$$

Therefore, the interval of the expected robot's activity time which is located within the start of the execution and the deadline D_i is calculated as in the following:

$$\mathbb{E}(AT_{ij}(D_i)) = \begin{cases} \frac{1}{\lambda_i} & \frac{1}{\lambda_i} < D_i \\ D_i & \frac{1}{\lambda_i} \geqslant D_i \end{cases}$$

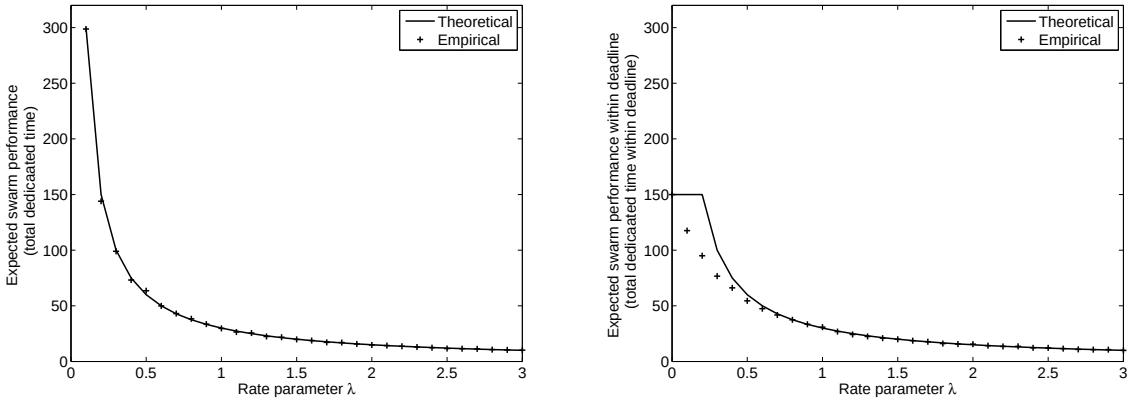
The expected value of the time dedicated by N_i robots to task T_i up to its deadline D_i can be written as:

$$\mathbb{E}(\sum_{j=1}^{N_i} AT_{ij}(D_i)) = \begin{cases} \frac{N_i}{\lambda_i} & \frac{1}{\lambda_i} < D_i \\ N_i D_i & \frac{1}{\lambda_i} \geqslant D_i \end{cases} \tag{4.36}$$

The expected value of the performance variance is the difference between the performance which is required to be achieved at the deadline and the expected performance to obtain at the deadline. It is calculated using Equation (4.10):

$$\mathbb{E}(\psi_i) = \begin{cases} max(\alpha_i - \frac{N_i}{\lambda_i}, 0) & \frac{1}{\lambda_i} < D_i \\ max(\alpha_i - N_i D_i, 0) & \frac{1}{\lambda_i} \geqslant D_i \end{cases} \tag{4.37}$$

Figure 4.13 (a) shows the expected value of the sum of 30 activity times which are sampled each from an exponential distribution with a rate parameter that varies over the range $[0, 3]$. The expected value of the robot's activity time is calculated theoretically using Equation (4.35). After that, it is multiplied by the number of robots, $N_i = 30$, to obtain the expected value of the time dedicated by the robots to the task. In the same figure, the total time dedicated by the robots is simulated over 100 runs of Monte-Carlo simulation. Figure 4.13 (b) illustrates the expected value of the swarm performance, which represents the sum of the activity times intervals located within the start of the execution and the task deadline, here $D_i = 5$ time units. The activity times are also sampled from an exponential distribution with a rate parameter which varies over the range $[0, 3]$. The expected value of the swarm performance is calculated theoretically using Equation (4.36) and is compared to the simulated swarm performance over 100 runs of Monte-Carlo simulation.
Since the longest interval which a robot can dedicated to the task up to its deadline is the deadline itself, the upper-bound of the sum of 30 activity times in our example is $30 \times 5 = 150$ time unit. That what we see in Figure 4.13 (b) that illustrates the total time dedicated to the task up to its deadline.



(a) *(b)*

Figure 4.13 – *Figure(a) the expected value of the sum of* 30 *robots' activity times calculated using Equation* (4.35). *Figure(b) the expected value of the sum of* 30 *robots' activity times intervals located within the start of the execution and the deadline* $D_i = 5$, *calculated using Equation* (4.36). *The robots' activity times are sampled from an exponential distribution with the rate parameter* $\lambda_i \in [0, 3]$. *The results are simulated over* 100 *runs of Monte-Carlo simulation.*

4.5 Dynamic Task Allocation

Dynamic task allocation is an allocation technique developed for the tasks which can be characterized by their negligible switching costs, see Chapter 1, Section 1.1. It allows the robots to switch among the tasks during their execution time. They can change dynamically and independently their current task and move to work on any other. For the kind of tasks we are considering in this thesis, where the task is executed in discrete parts, the robot has the possibility to change its task each time it finishes working on the current part of its task. Following dynamic allocation technique relaxes the constraint for the robot to keep working on the same task. This represents an efficient solution, for example, in cases where the robot may encounter parts of different tasks on its working area, so it has the chance to switch. Dynamic allocation provides the system with a high level of flexibility while assigning the robots to the different tasks. However, it counts as an efficient technique under the constraint of having negligible switching costs among the tasks. In the following are two cases where the switching costs among tasks can be considered as negligible.

- Tasks sharing the same physical area: robots can switch among the different

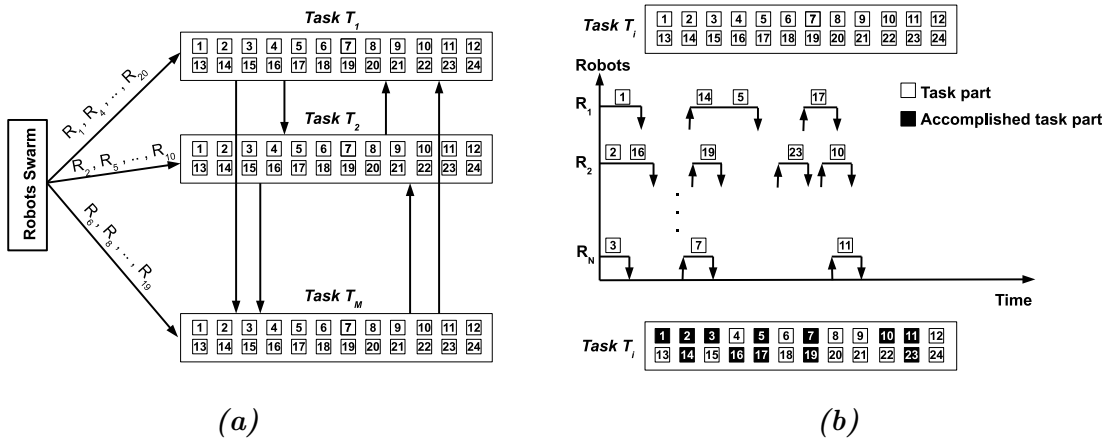


Figure 4.14 *– Figure (a) illustrates the dynamic allocation where the robots are allowed to switch among the tasks during their execution times. Figure (b) illustrates the work the robots accomplish on task T_i during their recursive visits to the task.*

tasks with negligible costs when they are occupying the same physical area or space since no additional traveling is needed. An example where this condition is feasible is a multi-foraging task where several types of items need to be retrieved by a swarm of robots to some home or nest. In case the items are scattered on the same area, the robots can switch to work on any type of these items without any extra switching costs.

- Switching times are negligible in comparison to the task execution times: In cases where the execution times of the considered tasks are significantly longer than the average time required to travel among the tasks, the switching costs can be considered as negligible.

Figure 4.14 illustrates an example of the dynamic allocation, where the robots are free to change their current task to any other task as soon as they finish accomplishing the current part of their task.
In the following, we assign the proper definitions of the swarm performance terms in Figure 4.1 for the case of dynamic allocation:

- Swarm performance: In the context of dynamic allocation, swarm performance is defined as the amount of work accomplished on the task during a specified time unit. For our time-constrained tasks, the swarm performance is amount

of work accomplished up to the task deadline and it is denoted by $\theta_i(D_i)$ for task T_i. Let us use γ_{ik} to denote the k-th contribution on task T_i. γ_{ik} represents the execution of the k-th part on task T_i. In case Q contributions have been occurred on task T_i up to its deadline D_i, the swarm performance on task T_i can be calculated as the sum of all the contributions occurred up to the deadline D_i:

$$\theta_i(D_i) = \sum_{k=1}^{Q} \gamma_{ik} \tag{4.38}$$

- Constrained swarm performance: Under dynamic allocation, the constraint associated with the swarm performance is associated with the amount of work accomplished by the swarm up to the task deadline. The size of the task represents the minimum amount of work that should be accomplished on the task up to its deadline. This size is not necessary equal to the parts available on the task to execute. Moreover, concerning the tasks we are considering in this thesis, the parts of the task are regenerated continuously. The constraint associated with the swarm performance on task T_i can be written as:

$$\begin{aligned} &\theta_i(D_i) \geqslant S_i \\ &\sum_{k=1}^{Q} \gamma_{ik} \geqslant S_i \end{aligned} \tag{4.39}$$

- Expected swarm performance: this calculates the expected value of the swarm performance that will be achieved at the deadline D_i. Since swarm performance under dynamic allocation is defined as the amount of work accomplished by the robots up to the task deadline. The expected value of this performance is the expected value of the discrete random variable which represents the number of parts accomplished by the swarm up to the task deadline:

$$\mathbb{E}(\theta_i(D_i)) = \mathbb{E}(\sum_{k=1}^{Q} \gamma_{ik}) \tag{4.40}$$

- Probability of the constrained swarm performance: this represents the probability of the constraint in Equation (4.39):

$$\Pr(\theta_i(D_i) \geqslant S_i) = \Pr(\sum_{k=1}^{Q} \gamma_{ik} \geqslant S_i) \tag{4.41}$$

This probability will be characterized by its probability density function and its cumulative distribution function.

- Expected variance of swarm performance: this estimates the difference between the minimum performance required to be achieved at the deadline D_i, namely the task size S_i and the expected performance to obtain at D_i.

$$\begin{aligned} \mathbb{E}(\psi_i) &= S_i - \mathbb{E}(\theta_i(D_i)) \\ &= S_i - \mathbb{E}(\sum_{k=1}^{Q} \gamma_{ik}) \end{aligned} \tag{4.42}$$

Figure 4.15 illustrates the substitution of the terms definitions, which were originally depicted in Figure 4.1 for the case of dynamic allocation.

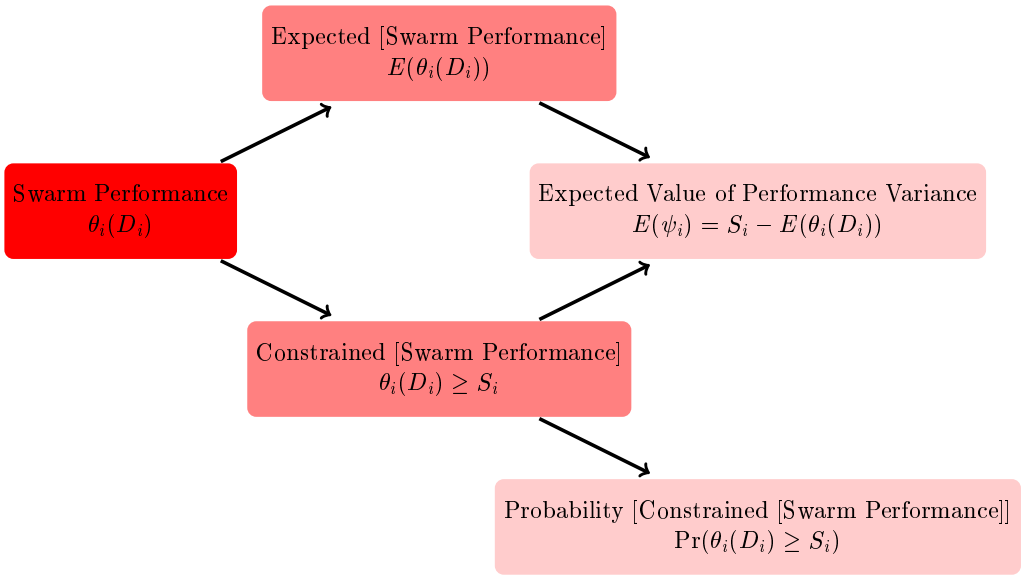


Figure 4.15 – *Assign the proper definitions to the swarm performance terms under dynamic allocation.*

4.5.1 Swarm Performance Modeling under Dynamic Allocation

[*The work in this section was done in collaboration with Michele Pace.*]

Swarm performance under dynamic allocation expresses the total amount of work

accomplished by the swarm on the task up to its deadline. This amount of work represents a random value which may belong to the continuous as well as to the discrete space based on the kind of the considered task. For example, in the task of pushing a box between two points by using a swarm of simple robots, the swarm performance can be measured as the distance which the box has been traveled during a specific period of time. The random value associated with the swarm performance belongs, here, to the continuous space. However in a foraging task where items should be retrieved by the robots, the swarm performance can be measured as the number of retrieved items during a specific time unit. The random variable associated with the swarm performance belongs, here, to the discrete space.

The tasks we are considering in this thesis consist of discrete parts where the size S_i represents the number of parts required to be accomplished on task T_i up to its deadline D_i. The cumulative work accomplished on any task up to its deadline will be the sum of the individual contributions of the robots which worked on the task up to its deadline, as defined in Section 4.5, Equation (4.38).

The time required by a single robot to accomplish one part on task T_i is a continuous random variable, which can be characterized by a task-specific mean $\hat{\mu}_i$ and a task-specific standard deviation $\hat{\sigma}_i$. Figure 4.16 shows an example where three robots R_1, R_2 and R_3 are working on task T_i. The random times required by each robot to accomplish individual parts are depicted using horizontal lines associated with the robots' colors. Each time a robot succeeds in finishing one part, the total number of executed parts increased by one. The increment in the number of accomplished parts over time is depicted in the lower part of Figure 4.16.

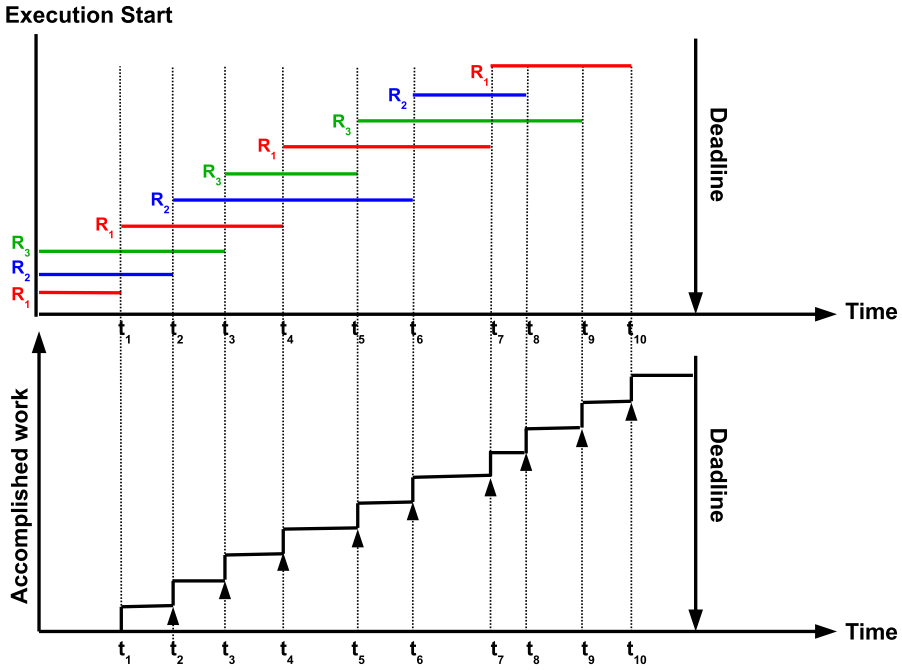


Figure 4.16 – *The evolution of the work on task T_i.*

The evolution process of the amount of work accomplished on task T_i up to the deadline D_i represents a time-continuous stochastic process $X(t)$. At the beginning of the task execution, the process has the value zero, $X(0) = 0$, since no part is accomplished yet. After that, it starts to have discrete increments of the total number of parts accomplished on T_i up to its deadline. This process is a counting process of the discrete random number of parts accomplished during a specific time period (the deadline). In addition, the increments of the number of parts accomplished in disjoint intervals are independent from each other. If we have a look at the definition of the Poisson process in Chapter 2.1, Section 2.2.2.1, we can conclude that the Poisson process represents an appropriate model for the work progress over time on the tasks we are considering in this thesis.

4.5.2 Probability Analysis of the Constrained Swarm Performance under Dynamic Allocation

The probability of interest, here, is the probability of completing the task up to its deadline. This probability was given above in Section 4.5, Equation (4.41) as:

$$\Pr(\theta_i(D_i) \geqslant S_i)$$

where $\theta_i(D_i)$ refers to the total number of parts (work amount) accomplished on task T_i up to its deadline D_i.
The increment of the number of parts accomplished over time is modeled using a Poisson process with a task-specific rate as described in Section 4.5.1. It is well known that the number of events counted by a Poisson process with the rate λ within a time interval of the length t follows a Poisson distribution with the parameter λt. Therefore the Probability of interest in Equation (4.41) can be calculated using the cumulative distribution function of the Poisson distribution with the parameter $\lambda_i D_i$ for task T_i as in the following :

$$\begin{aligned} \Pr(\theta_i(D_i) \geqslant S_i) &= 1 - \Pr(\theta_i(D_i) < S_i) \\ &= 1 - e^{-\lambda_i D_i} \sum_{k=0}^{S_i} \frac{(\lambda_i D_i)^k}{k!} \end{aligned} \tag{4.43}$$

On the other hand, the probability in Equation (4.41) is equivalent to the probability of having the time required to accomplish S_i parts being shorter than or equal to the deadline D_i. We use $\tau_i(S_i)$ to denote the time required to accomplish S_i parts of task T_i. Hence, we can write the probability of Equation (4.41) as follows:

$$\Pr(\tau_i(S_i) \leqslant D_i) \Longleftrightarrow \Pr(\theta_i(D_i) \geqslant S_i) \tag{4.44}$$

The Poisson process used to model the work evolution on task T_i is characterized with its specific rate λ_i. Let us denote the time between two consecutive events $i-1$ and i by t_i. The occurrence time τ_n of the n-th event in a Poisson process is given by (Ross, 2006):

$$\tau_n = \sum_{k=1}^{n} t_k \tag{4.45}$$

It is well known that the waiting times between consecutive events (inter-arrival times) in a Poisson process with the rate λ, are exponentially distributed with the same rate parameter λ. In addition, the sum of n random variables distributed exponentially with the parameter λ, is a random variable that follows the Gamma distribution with the shape parameter n and the rate λ. The random variable τ_n in Equation (4.45) is the sum of n exponentially distributed random variables with the parameter λ. Therefore, it will be distributed following the Gamma distribution with the parameters n and λ and because n is discrete, the Gamma distribution can be substituted with the Erlang distribution for the same parameters.

$$\tau_n \sim \mathbb{E}rlang(n, \lambda)$$

The time we are interested to analyze probabilistically, in Equation (4.44), is the occurrence time of the S_i-th event which is the time of achieving S_i parts accomplished on task T_i. Based on Equation (4.45), this time can be written as:

$$\tau_i(S_i) = \sum_{k=1}^{S_i} t_k \tag{4.46}$$

This time follows the Erlang distribution with the parameters S_i and λ_i, where S_i of task T_i and λ_i is the rate of the Poisson process which models the work evolution on task T_i:

$$\tau(S_i) \sim \mathcal{E}rlang(S_i, \lambda_i) \tag{4.47}$$

Thus, the probability in Equation (4.44) represents the cumulative distribution function of the Erlang distribution and is given by:

$$\Pr(\tau_i(S_i) \leqslant D_i) = 1 - \sum_{k=0}^{S_i} \frac{(\lambda_i D_i)^k}{k!} e^{-\lambda_i D_i} \tag{4.48}$$

We have presented two methods of calculating the probability in Equation (4.41) based on the equivalence of the two probabilities in Eqaution (4.44). Both methods have led to the same result in Equations (4.48) and (4.43).
Figure 4.17 illustrates the probability calculated by Equation (4.48) for $\lambda_i = 5$, $D_i = 25$ and $S_i \in [100, 150]$.

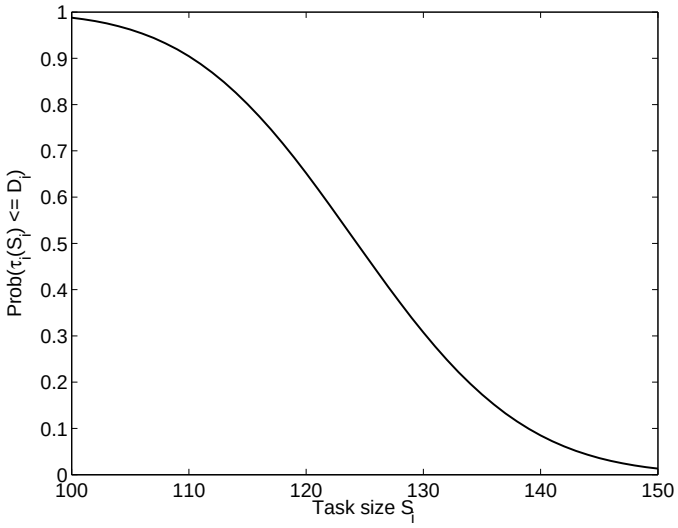


Figure 4.17 – *Probability* $\Pr(\tau_i(S_i) \leqslant D_i)$*, for* $\lambda_i = 5$*,* $D_i = 25$ *and* $S_i \in [100, 150]$*.*

4.5.3 Performance Variance Estimation under Dynamic Allocation

In this section, we estimate the variance between the swarm performance required to be achieve at the task deadline and the swarm performance expected to be achieved at the task deadline. The performance variance on task T_i is denoted by ψ_i and it represents the difference between the task size and the number of parts expected to be accomplished up to the task deadline. As mentioned in Section 4.5.1, the work progress on the tasks is modeled using Poisson processes. It is well known that the number of events counted by a Poisson process with a rate λ is distributed during any time interval t, according to a Poisson distribution with the parameter λt. Consequently, the number of parts accomplished up to the deadline will be distributed as in the following:

$$\theta_i(D_i) \sim \mathcal{P}oisson(\lambda_i D_i) \tag{4.49}$$

The expected value of a random variable which follows a Poisson distribution with the parameter λ is equal to λ. Therefore, the expected value of the random variable associated with the swarm performance, $\theta_i(D_i)$, is given by:

$$\mathbb{E}(\theta_i(D_i)) = \lambda_i D_i \tag{4.50}$$

After calculating the expected value of the swarm performance using Equation (4.50), the expected variance can be calculated as the difference between the number of parts required to be accomplished S_i and the expected number of parts to execute. Equation (4.42) is used to calculate the expected value of the performance variance as in the following:

$$\mathbb{E}(\psi_i) = \begin{cases} S_i - \lambda_i \; D_i & \lambda_i \; D_i < S_i \\ 0 & \lambda_i \; D_i \geqslant S_i \end{cases} \tag{4.51}$$

Figure 4.18 shows the expected swarm performance at the deadline $D_i = 10$ when the Poisson process which models the evolution of the work on the task has a varying rate $\lambda \in [1, 10]$. In addition, the figure shows the averaged performance over 100 runs of Monte-Carlo simulation.

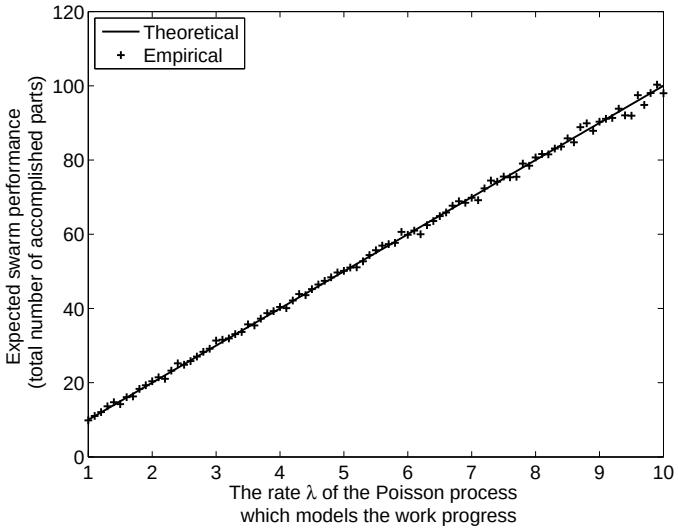


Figure 4.18 – *Comparison between the expected swarm performance and the simulated one over* 100 *runs of Monte-Carlo simulation, the deadline is* $D_i = 10$. *The Poisson process that models the evolution of the work on the task has its rate within the range* $[1, 10]$.

4.6 Conclusions

In this chapter, we have introduced the concept of *swarm performance* and the influence of spatial interferences on it. After that, two kinds of allocations were introduced: Static allocation and Dynamic allocation.
The static allocation is proposed to deal with tasks which are characterized by their high switching costs. In this technique, the swarm performance on the task is defined as the total time dedicated by the robots up to the deadline of the task. The robots assign themselves to the different tasks at the beginning of the execution and each robot determines, independently, the time it will dedicate to the selected task. The constraint associated with the swarm performance is related to the minimum threshold of the total time that should be dedicated to the task up to its deadline.
Dynamic allocation, on the other hand, allows the robots to switch among the tasks during their execution time, as this technique is proposed for the tasks with negligible switching costs. It provides a high level of flexibility in assigning and reassigning the robots among the different tasks, which may improve the global performance of the system. The swarm performance associated with the dynamic allocation is defined as the amount of work accomplished on the task up to its deadline. Therefore, the constraint of the performance is related to the amount of work that should be accomplished on the task up to its deadline.
The chapter has provided a probabilistic analysis of the constrained swarm performance in both cases of static and dynamic allocations. In addition, it has presented an estimation of the expected variance between the swarm performance required to achieve and the expected performance to achieve both at the task deadline.

Chapter 5

Task Allocation Design

Better never than late.

- George Bernard Shaw

In this chapter we present a set of task allocation strategies, developed for executing time-constrained tasks by swarm robotics. The task deadlines are categorized into hard deadlines, where executing the task after its deadline has no benefit, and soft deadlines, where continuing the execution of the task after its deadline is associated with specific costs in terms of deteriorating performance quality.
In the previous chapter, two kinds of allocation techniques were introduced: Static allocation for tasks with high switching costs and Dynamic allocation for tasks with negligible switching costs. The task allocation strategies in this chapter are developed in the context of both allocation techniques: Static and Dynamic.
The allocation strategies provide the robots of the swarm with the necessary input at the beginning of the execution. After that, the robots assign themselves independently and start to work fully autonomously on their selected tasks under the temporal constraints of these tasks.

5.1 Chapter Notations

- N: the number of robots used in the swarm (the swarm size).
- N_i: the number of robots assigned to task T_i.
- T_i: the i-th task of the task set which is required to be executed by the swarm.
- D_i: the deadline of task T_i.
- S_i: the size of task T_i. It represents the number of the task parts that are required to be executed up to the task deadline.
- α_i: It is used under static allocation technique and refers to the minimum threshold of the total time that should be dedicated to task T_i by the robots.
- $\Pr(T_i)$: the execution probability of task T_i represents the probability of finish executing the task at its deadline.
- β_i: the priority of task T_i which reflects the tightness of its temporal constraint. It increases for tasks with larger sizes or/and with earlier deadlines.
- μ_i: the mean time required by the swarm to accomplished one part of the task T_i.
- $\hat{\mu_i}$: the mean time required by a single robot to accomplish one part of task T_i.
- $\hat{\sigma_i}$: the standard deviation of the random time required by a single robot to accomplish one part of task T_i.
- N_{opt}: the number of robots which is associated with the highest performance can the swarm achieve under the influence of spatial interferences on task T_i.
- $\theta_i(D_i)$: the total amount of work accomplished on task T_i.
- $\theta_i(D_i)$: the total amount of work accomplished on task T_i up to the deadline D_i. It represents the number of parts executed up to the deadline D_i.
- λ_i: The rate of the Poisson process that models the work progress on task T_i over time when dynamic allocation is considered.

- λ_{ij}: The rate of the Poisson process that models the work progress on task T_i over the j-th activation period.
- ηj: The length of the task activation period j.
- π_{ij}: The probability used by the robots to switch from task T_i to task T_j.
- ψ_i: The swarm performance variance, which represents the difference between the swarm performance required to be achieved and the expected performance to achieve at the task deadline.

5.2 Hard-deadline Tasks

Based on the definition of hard-deadline tasks, the successful execution of any task is achieved when the task is accomplished up to its deadline. In a stochastic system such as swarm robotics, the amount of work accomplished on any task represents a stochastic value that is influenced by a large set of factors including the number of robots working on the task, the task specification, the task environment, and other factors. Therefore, no deterministic answer is available for the question concerning the possibility of finish executing the task at its deadline. Hence, the answer to the question of finish executing the task within its deadline is performed in a probabilistic manner, where the probability of executing the task within its deadline is referred to as the *execution probability* of the task. It was introduced in Chapter 4, Equation (4.30) for static allocation and Equation (4.48) for dynamic allocation.

In traditional real-time systems, a hard-deadline task is considered as executable if it can be accomplished before or at its deadline. In this thesis, we categorize the hard-deadline task as *executable* when its has an acceptable execution probability that is near to one, otherwise it is categorized as *un-executable*. The allocation strategies developed in this chapter for hard-deadline tasks aim to perform an allocation that maximizes the number of executable tasks by maximizing the tasks with an acceptable execution probability. An acceptable execution probability of task T_i is defined as a probability near to one $\Pr(T_i) = 1 - \epsilon$, where ϵ is a design parameter that can be defined for each task. For simplicity, we use the same value of ϵ for all the considered tasks. The allocation strategy deals with tasks based on their priorities which reflect their relative importance. In this thesis, the task importance is based on the tightness of its time constraint, where this tightness increases for tasks with larger sizes or/and earlier deadlines. A simple way of defining such a task priority is given by:

$$\beta_i = \frac{S_i/D_i}{\sum_{k=1}^{M} S_k/D_k} \tag{5.1}$$

where M is the number of tasks to execute, S_i and D_i are the size and the deadline of task T_i, respectively.

In the following, we present the allocation strategies developed for hard-deadline tasks under both static and dynamic allocation techniques.

5.2.1 Task Allocation for Hard-deadline Tasks under Static Allocation

Static allocation was proposed in Chapter 4, Section 4.4 as an allocation technique for tasks characterized with their high switching costs. The robots' assignment is performed at the beginning of the execution and robots are prevented to switch among the tasks after their initial allocation. The core idea of this technique is to dedicate enough time in order to execute the task within its deadline. The total time which can be dedicated to the task is influenced by: First, the number of robots assigned to the task and Second, the parameters of the distribution associated with the robots' the activity times, see Chapter 4, Section 4.4.1. The constraint of the swarm performance is to have the total time dedicated to the task equal to or greater than a minimum threshold, defined in Chapter 4, Equation (4.7) as in the following:

$$\alpha_i = S_i \hat{\mu}_i(N_i)$$

where S_i is the size of task T_i. and $\hat{\mu}_i(N_i)$ is the mean time required by a single robot to accomplish one part of task T_i when N_i robots are working on T_i.
The mean time $\hat{\mu}_i$ required by a single robot to accomplish one part of task T_i is influenced by the spatial interferences among robots, as illustrated in Chapter 4, Section 4.3. Figure 4.2 (c) in the same section, shows an example of how the time required by a single robot to accomplish a specific amount of work varies by changing the number of robots. When the mean time $\hat{\mu}_i$ changes, the minimum threshold α_i of the total time required to be dedicated to task T_i changes respectively. The mean time $\hat{\mu}_i$ can be estimated easily by performing short-term experiments on the considered tasks.
The following inputs are available for the allocation strategy developed for hard-deadline tasks under static allocation:

- The task deadlines;
- The task sizes;
- The mean time μ_i required to accomplish one part on task T_i by the swarm under the spatial interferences. This mean is characterized over short-term experiments for different swarm sizes;

- The probability distribution of the robots' activity times: The probability distribution used, here, is the exponential distribution. However, other distributions including the truncated-exponential can substitute the exponential distribution by exchanging the proper probability density function and cumulative distribution function.

The output of the allocation strategy includes:

- A list of the M' executable tasks under the hard-deadline constraints;
- The number of robots that should be assigned to each of the M' tasks and consequently the assignment probability;
- The distribution parameters which will be used by each robot individually to sample its activity time on the selected task.

The allocation strategy aims to maximize the number of executable tasks according to the task priorities.
In the following, two allocation strategies are developed for hard-deadline tasks under static allocation. The strategies are developed under two different optimization criteria.

5.2.1.1 Energy-aware Allocation for Hard-deadline Tasks under Static Allocation

The limited life-time of robots' batteries represents a critical restriction for executing tasks with long-term deadlines. On the other hand, long-term deadline is a common characteristic of robotic tasks. Under static allocation, the robots are assigned to the tasks at the beginning of the execution and they keep working on their selected tasks till their activity times are exceeded. Therefore, recharging their batteries during the task execution is a non-trivial process. For this reason, energy-aware allocation attempts to maximize the number of robots assigned to the different tasks. This allows to reach the required threshold of the total time that should be dedicated to the task with shorter activity times. This strategy offers two benefits: First, it increases the probability of performing the task during the life-time of the robots' batteries. Second, it makes the swarm faster ready for further task executions.

As we have seen in Chapter 4, Section 4.3, by increasing the number of robots working on the task, the swarm performance increases till a specific swarm size is reached. After that, the swarm performance starts to decrease influenced of the spatial interferences among robots. The swarm size associated with the maximum swarm performance is referred to as the *optimal swarm size*. Based on the swarm performance behavior under the influence of spatial interferences, the energy-aware strategy cannot assign a larger number of robots than the optimal number which produces the maximum swarm performance on the considered task. Figure 5.2 shows the direction of optimizing the number of assigned robots following the energy-aware strategy.

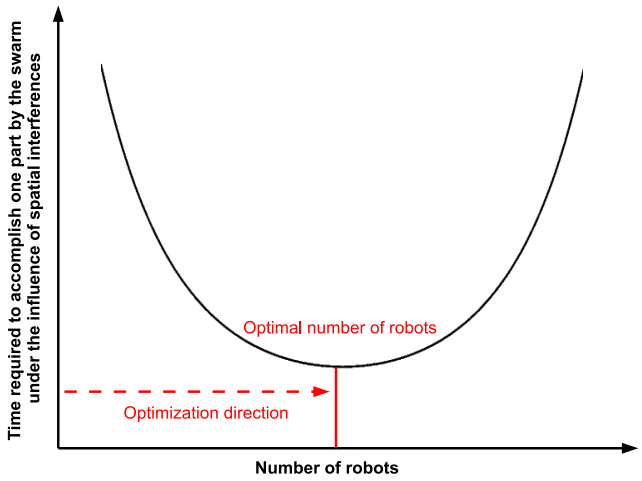


Figure 5.1 – *The optimal number of robots is depicted under the influence of spatial interference. In addition to the optimization direction in case of applying the energy-aware strategy.*

The following steps of the energy-aware strategy are performed *off-line* in order to obtain the necessary parameters required for the task allocation.

- ***Priority assignment:*** The tasks are assigned their priorities, which are calculated using Equation (5.1).
- ***Minimizing the task execution time:*** In this step, the average time required to accomplish individual parts of the task is minimized. This is done by maximizing the number of robots assigned to the task, under two constraints:

First, to not exceed the optimal number of robots that generates the maximum swarm performance under the influence of spatial interferences. Second, to limit the number of assigned robots such that the sum of the robots assigned to the different tasks doesn't exceed the swarm size.

We assume that the average time required to perform individual parts under spatial interferences is estimated a priori by performing short-term experiments with different swarm sizes. The average time μ_i required to accomplish an individual part on task T_i is a function of the number of assigned robots N_i:

$$\mu_i = f(N_i) \tag{5.2}$$

The goal is to minimize μ_i for the M considered tasks by increasing N_i over the range $N_i \in [0, min(N_{opt}, N)]$.

The problem represents a multi-objective optimization of M objective functions and as N_i represents a positive integer, this optimization belongs to the class of *non-linear integer programming*. Additionally, the optimization should be performed under the constraint of not exceeding the swarm size N.

$$\begin{aligned} \underset{N_i}{\text{minimize}} \quad & \begin{cases} \mu_1 = f(N_1) \\ \mu_2 = f(N_2) \\ \vdots \\ \mu_M = f(N_M) \end{cases} \\ \text{subject to:} \quad & \sum_{i=1}^{M} N_i \leqslant N \end{aligned}$$

The above M multi-objective optimization problem could be transferred into a cumulative weighted objective function[1]. The weight which reflects the relative importance of each of the objective functions, here, is the priority of the task associated with that objective function. Consequently, the optimization

[1] A cumulative weighted objective function is constructed by summing up the objectives after assigning appropriate weights to each of them based on their relative importance.

problem can be expressed as in the following:

$$\underset{N_i}{\text{minimize}} \quad \sum_{i=1}^{M} \beta_i \mu_i$$

$$\text{subject to:} \quad \sum_{i=1}^{M} N_i \leqslant N$$

The output of this optimization for task T_i is as in the following:

- The maximum number of robots N_i that can be assigned to task T_i;
- The average time $\mu_i(N_i)$ required to accomplish one part on task T_i when N_i robots are assigned to it. It is calculated by using Equation (5.2);
- The average time $\hat{\mu}_i(N_i)$ required by a single robot to accomplish one part on task T_i when N_i robots are assigned to it;
- The minimum threshold α_i of the total time required to be dedicated to task T_i. It is calculated by substituting $\hat{\mu}_i(N_i)$ in Equation (4.7).

- ***Calculating the parameters of the activity time distribution:*** After the number of robots which should be assigned to each of the tasks is optimized and the associated time threshold α_i is calculated, the next step is to find out the parameters related to the probability distribution of the robots' activity times.

  This step includes two contradicting goals: First is to design the distribution parameters in order to minimize the length of the robots' activity times, and Second is to design the distribution parameters in order to maximize the execution probabilities of the tasks. However, maximizing the execution probability of a task is achieved by increasing the total time dedicated to the task up to its deadline, which is achieved by maximizing the robots' activity times. This contradicts the goal of the energy-aware strategy, namely, to minimize the robots' activity times for energy purposes.

  First, the allocation strategy finds the parameters of the activity time distribution that maximize the execution probability of the tasks by solving the following optimization problem:

$$\underset{\lambda_i}{\text{maximize}} \, \Pr(T_i) = f(\lambda_i)$$

The output of the optimization is:

- The set of executable tasks M', where each task belonging to this set has an acceptable execution probability;
- The upper/lower bound of the parameter λ_i. This bound is applied to tune λ_i in order to minimize the expected value of the robots' activity times on each of the executable tasks while preserving an acceptable execution probability.

Second, the allocation strategy tunes the parameters of the activity time distribution to minimize the expected value of the robots' activity times under the constraint to preserving an acceptable execution probability of the tasks. The optimization problem to solve is the following:

$$\begin{aligned} &\underset{\lambda_i}{\text{minimize}} \quad \mathbb{E}(AT_i) = f(\lambda_i) \quad \forall i \in M' \\ &\text{subject to:} \quad \Pr(T_i) \geqslant 1 - \epsilon \end{aligned}$$

The output of this step is the optimal parameters of the activity times distribution.

- ***Tasks categorization:*** As the energy-aware strategy attempts to maximize the number of robots assigned to each task, some tasks may receive more robots than they need. Consequently, other tasks may suffer from a potential lack in the number of robots assigned to them according to their priorities. In order to handle this assignment problem, the tasks are categorized, based on their execution probabilities, into: *Granter* tasks and *Withholder* tasks. The granter tasks are tasks with an acceptable execution probability ($\Pr(T_i) \geqslant 1-\epsilon$), whereas the withholder tasks are the un-executable tasks with execution probability ($\Pr(T_i) < 1 - \epsilon$). Since the granter tasks may have received more robots than they need, they may be able to offer robots to other un-executable tasks without becoming un-executable themselves.
- ***Granting process:*** During this process, each granter task checks the number of robots it can offer, without becoming un-executable. This number of robots is assigned to the withholder tasks based on their priorities, in order to increase

their execution probabilities. After the granting process finishes, only tasks with acceptable execution probabilities ($\Pr(T_i) \geqslant 1-\epsilon$) are executed since the considered deadlines are hard.

- ***Assignment Probability:*** After finding out the number of robots that should be assigned to each of the executable tasks, the probability used by the robots to assign themselves to the executable tasks, which is referred to as the *assignment probability*, is calculated as in the following:

$$P_i = \frac{N_i}{N} \tag{5.3}$$

The output of the energy-aware strategy includes:

- The set of executable tasks;
- The assignment probability associated with each of the executable tasks;
- The parameters of the activity times distribution;
- The execution probability of each executable task.

After obtaining the above output, the robots can start to assign themselves autonomously to the tasks and each robot samples its own activity time under the consideration of the task deadline. The robot's activity time is a continuous random variable sampled independently by each robot. For robot R_j working on task T_i, the activity time dedicated under the task time constraint belongs to the range $AT_{ij} \in \{0, min\{D_i, \kappa_j\}\}$, where D_i is the task deadline and κ_j is the time through which robot R_j can still operate and it is computed based on its current battery level. Figure 5.2 illustrates the different steps of the energy-aware strategy, with the inputs marked in red and the outputs marked in blue. The calculated steps are performed off-line and the necessary part of the output to use by the robots is marked with blue arrows.

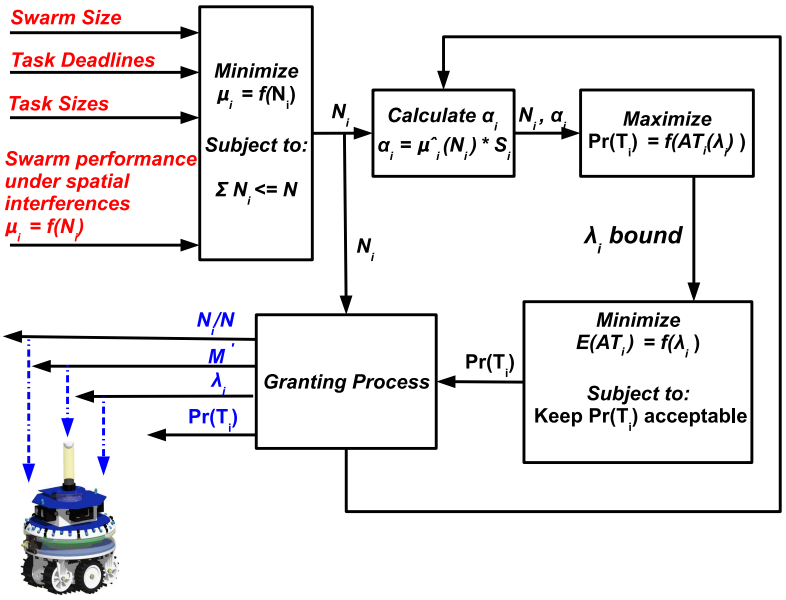


Figure 5.2 – *Energy-aware strategy for hard-deadline tasks under static allocation.*

The following is the pseudo-code of the energy-aware strategy:

Algorithm 1 Energy-aware Strategy.

Input: S_i, D_i ,N, $\mu_i = f(N_i)$, ϵ
Output: solution N_i and solution λ_i

1: $M \leftarrow$ tasks
2: $Acceptable\_Prob \leftarrow 1 - \epsilon$ ▷ % ϵ is a design parameter%
3: $\beta_i = \dfrac{S_i/D_i}{\sum_{j=1}^{M} S_j/D_j}$ ▷ % calculate the task priorities%
4: $\hat{\mu}_i(N_i) = N_i\mu_i(N_i)$
5: **for** $i = 1$ to M **do**
6: $\quad N_{opt}(i) = \underset{N_i}{minimum}\ \mu_i$ ▷ % calculate the optimal robots number%
7: **end for**
8: $lb(N_i) \leftarrow 0$
9: $ub(N_i) \leftarrow N_{opt}(i)$
10: $\underset{N_i}{minimize}\ \sum_{i=1}^{M} \beta_i\mu_i$ ▷ % non-linear integer programming%

subject to: $\sum_{i=1}^{M} N_i \leqslant N$
12: $\alpha_i = \hat{\mu}_i(N_i)S_i$ ▷ % the time threshold for task T_i%
13: **for** $i = 1$ to M **do**
14: $\quad \underset{\lambda_i}{maximize}\ (\Pr(T_i) = f(N_i, \alpha_i, D_i, \lambda_i))$
15: $\quad$ **while** $\Pr(T_i) \geqslant$ Acceptable_Prob **do**
16: $\quad\quad \underset{\lambda_i}{minimize}\ \mathbb{E}\ (AT_i)$

$\quad\quad$ subject to: $\Pr(T_i) \geqslant$ Acceptable_Prob
18: $\quad$ **end while**
19: **end for**
20: **for** $i = 1$ to M **do** ▷ % start granting process%
21: $\quad$ **if** $\Pr(T_i) <$ Acceptable_Prob **then**
22: $\quad\quad$ add T_i to withholder task set W
23: $\quad$ **else**
24: $\quad\quad$ add T_i to the granter task set G
25: $\quad$ **end if**
26: **end for**

27: $w \leftarrow$ size of W
28: $g \leftarrow$ size of G
29: **if** $((w > 0)\ \&\ (g > 0))$ **then**
30: order the withholder task set based on the task priorities
31: order the granter task set based on the task priorities
32: **for** $j = 1$ to w **do**
33: **for** $k = 1$ to g **do**
34: **while** $(\Pr(T_k) \geqslant$ Acceptable_Prob)
35: $\&\ (\Pr(T_j) <$ Acceptable_Prob)
36: $\&\ (N_j < N_{opt}(T_j))$ **do**
37: $N_j = N_j - 1$
38: $N_k = N_k + 1$
39: update $\mu_j(N_j)$, $\mu_k(N_k)$, α_j, α_k
40: $\underset{\lambda_j}{maximize}\ (\Pr(T_j) = f(N_j, \alpha_j, D_j, \lambda_j))$
41: $\underset{\lambda_k}{maximize}\ (\Pr(T_k) = f(N_k, \alpha_k, D_k, \lambda_k))$
42: **end while**
43: **end for**
44: **end for**
45: **end if**
46: **for** $i = 1$ to M **do**
47: **if** $\Pr(T_i) \geqslant$ Acceptable_Prob **then**
48: add T_i to the set of executable tasks
49: **else**
50: add T_i to the set of un-executable tasks
51: **end if**
52: **end for**

5.2.1.2 Robots-aware Allocation for Hard-deadline Tasks under Static Allocation

There are scenarios where the arrival times of the tasks are unknown a priori, hence maximizing the number of robots assigned to the known tasks does not provide an efficient solution. In other scenarios, maximizing the number of robots assigned to the task may lead to a potential miss in one deadline or more because of the limited size of the swarm. Since the goal is to finish the task within its deadline and not to speed up its execution, robots-aware strategy provides an efficient mechanism to assign the only required number of robots to the task under its time constraint. The required number of robots can be defined as the number of robots that allows to obtain an acceptable execution probability of the task. This strategy results in maximizing the number of free robots available in the swarm to participate on newly arriving tasks. Robots-aware strategy can be applied smoothly when the lifetime of the robot's battery is comparable to the execution time of the task or when recharging robots during the tasks execution is possible.
The core idea of this strategy is to assign the minimum number of robots N_i required by task T_i to be executed up to its deadline D_i. N_i is an integer which takes its value in the range of $[0, min(N_{opt}, N)]$, where N_{opt} denotes the number of robots associated with the maximum swarm performance under spatial interferences. Figure 5.3 shows the direction of optimizing the number of robots under the robots-aware strategy.

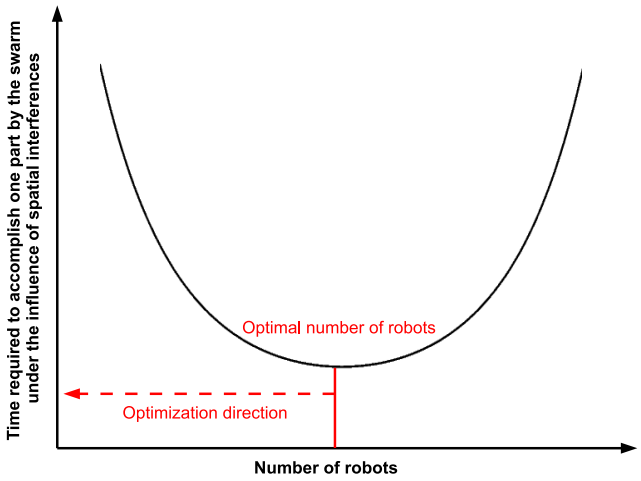


Figure 5.3 – *The optimal number of robots is depicted under the influence of spatial interference. In addition to the optimization direction in case of robots-aware strategy.*

The following steps of the robots-aware strategy are performed *off-line* in order to obtain the necessary parameters required for the task allocation.

- ***Priority assignment:*** The tasks are assigned their priorities, which are calculated using Equation (5.1).

- ***Maximize the task execution probability:*** The allocation strategy starts with fulfilling the task needs of robots based on their priorities. Let $N_{current}$ be the current number of robots available in the swarm and let T_i be the task with the highest priority. The number of robots N_i assigned to task T_i is set to $min(N_{current}, N_{opt})$, where N_{opt} is the optimal number of robots defined above. As soon as N_i is calculated, the mean time $\mu_i(N_i)$ required to accomplish an individual part on task T_i when N_i robots are assigned, in addition to the threshold α_i are calculated. Finally, the execution probability $\Pr(T_i)$ on task T_i is maximized as a function of λ_i, where λ_i is the parameter of the activity times distribution.

$$\underset{\lambda_i}{\text{maximize}} \quad \Pr(T_i) = f(\lambda_i)$$

  In case the optimized execution probability is an acceptable one ($\Pr(T_i) \geqslant 1 - \epsilon$), the task is categorized as executable, otherwise it is un-executable.

- ***Minimize the number of assigned robots:*** In case of robots-aware strategy, the number of robots that is assigned to an *executable* task is minimized while maintaining an acceptable execution probability. Finding the required number of robots represents an optimization problem which belongs to the class of the *integer linear programming (ILP)* (Papadimitriou & Steiglitz, 1982; Williams, 2009) and can be expressed for the executable task T_i as in the following:

$$\begin{aligned} &\text{minimize} \quad N_i \\ &\text{subject to:} \quad \Pr(T_i) \geqslant 1 - \epsilon \end{aligned}$$

  The mean time required by N_i robots to accomplish an individual part of task T_i is updated for each tested N_i, in addition to the time threshold α_i using Equation (4.7). after that, the execution probability of task T_i is calculated for the specified values of N_i and α_i.

  The output of this optimization is as in the following:

  - The set of executable tasks M', where each task belonging to this set has an acceptable execution probability;
  - The minimized number of robots N_i to be assigned;
  - The mean time $\mu_i(N_i)$ which is estimated under the influence of the spatial interferences among the N_i robots assigned to task T_i;
  - The average time $\hat{\mu}_i(N_i)$ required by a single robot to accomplish one part on task T_i when N_i robots are assigned to it;
  - The minimum threshold α_i of the total time required to be dedicated to task T_i. It is calculated by substituting $\hat{\mu}_i(N_i)$ in Equation (4.7).

- ***Calculating the parameter of the activity times distribution:*** the parameters of the activity times distribution are designed in order to maximize the expected value of the robots' activity times on each of the executable tasks, while maintaining an acceptable execution probability. The optimization problem is written as in the following:

$$\begin{aligned} &\underset{\lambda_i}{\text{maximize}} \quad \mathbb{E}(AT_i) = f(\lambda_i) \quad \forall i \in M' \\ &\text{subject to:} \quad \Pr(T_i) \geqslant 1 - \epsilon \end{aligned}$$

The output of this step is the optimal parameters of the activity times distribution.

- ***Assignment Probability:*** After finding out the number of robots that should be assigned to each of the executable tasks, the probability used by the robots to assign themselves to the executable tasks, which is referred to as the *assignment probability*, is calculated as in the following:

$$P_i = \frac{N_i}{N} \tag{5.4}$$

The output of the robots-aware strategy includes:

- The set of executable tasks;
- The assignment probability associated with each of the executable tasks;
- The parameters of the activity times distribution;
- The execution probability of each executable task.

After obtaining the above output, the robots can start to assign themselves autonomously to the tasks and each robot samples its own activity time under the consideration of the task deadline. The robot's activity time is a continuous random variable sampled independently by each robot. For robot R_j working on task T_i, the activity time dedicated under the task time constraint belongs to the range $AT_{ij} \in \{0, min\{D_i, \kappa_j\}\}$, where D_i is the task deadline and κ_j is the time through which robot R_j can still operate and it is computed based on its current battery level. Figure 5.4 illustrates the steps of the robots-aware strategy, with the inputs marked in red and the outputs marked in blue. The calculated steps are performed off-line and the necessary part of the output to use by the robots is marked with blue arrows.

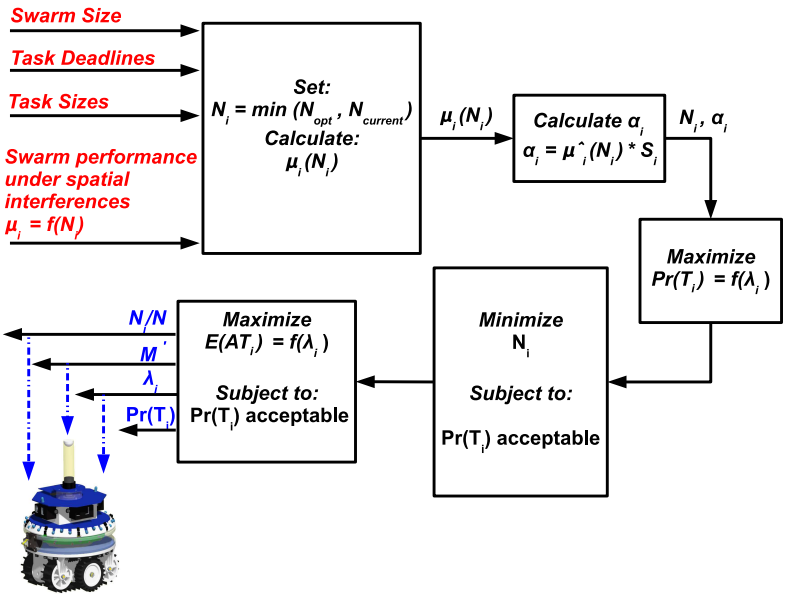


Figure 5.4 – *Robots-aware strategy for hard-deadline tasks under static allocation.*

The following is the pseudo-code of the robots-aware strategy:

Algorithm 2 Robots-aware Strategy.

Input: S_i, D_i ,N, $\mu_i = f(N_i)$, ϵ
Output: solution N_i and solution λ_i

$M \leftarrow$ tasks
$Acceptable\_Prob \leftarrow 1 - \epsilon$ ▷ % ϵ is a design parameter%
$\beta_i = \dfrac{S_i/D_i}{\sum_{j=1}^{M} S_j/D_j}$ ▷ % calculate the task priorities%
$\hat{\mu}_i(N_i) = N_i \mu_i(N_i)$
for $i = 1$ to M **do**
    $N_{opt}(i) = \underset{N_i}{minimum}\ \mu_i$ ▷ % calculate the optimal robots number%
end for
order the tasks set based on their priorities
for $i = 1$ to M **do**
    $N_i = min(N_{opt}(i), N_{current})$
    if $N_i > 0$ **then**
        $\mu_i(N_i) = f(N_i)$
        $\alpha_i = S_i \hat{\mu}_i(N_i)$
        $\Pr_{max}(T_i) \leftarrow \underset{\lambda_i}{maximize}\ \Pr(T_i)$
        if $\Pr_{max}(T_i) \geqslant$ Acceptable_Prob **then**
            add T_i to the set of executable tasks
            minimize N_i

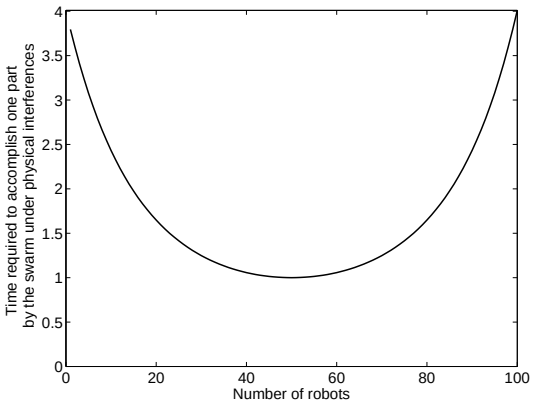
            subject to: $\Pr(T_i) \geqslant$ Acceptable_Prob
            update $\hat{\mu}_i(N_i)$, α_i, $\Pr(T_i)$
            maximize $\mathbb{E}\ (AT_i)$

            subject to: $\Pr(T_i) \geqslant$ Acceptable_Prob
        else
            add T_i to the set of un-executable tasks
        end if
    end if
end for

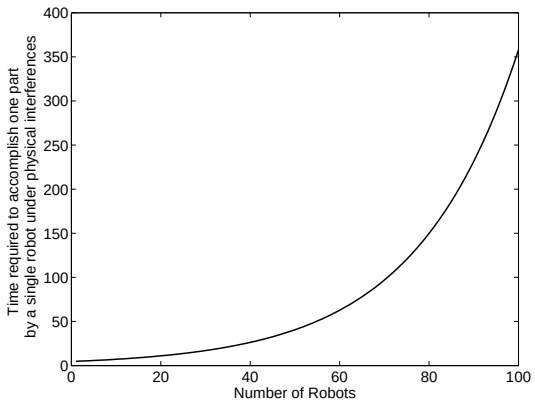
5.2.1.3 Scenario and Evaluation

Let us consider three tasks which are characterized by their hard deadlines $D = \{100, 150, 300\}$ time units. The switching costs among the tasks are considered to be high, hence, the static allocation technique is applied. A homogeneous swarm of $N = 100$ robots is used to execute the tasks under both: energy-aware and robots-aware strategies.

The time required to accomplish one part by the swarm is assumed to be the same for the three tasks. Figure 5.5 (a) shows how the mean of this time changes with increasing the number of robots working on the task. On the other hand, the mean of the time required by a single robot to accomplish one part is calculated and depicted 5.5 (b). It shows how the mean time increases by increasing the number of robots working on the task.



(a) *Time required by a swarm to accomplish one part.*



(b) *Time required by a single robot to accomplish one part.*

Figure 5.5 *– Swarm and single robot performance under spatial interferences.*

Different variants of the task sizes are examined as listed in the below table. In the first set, the task sizes are selected so that the three tasks have equal priorities. After that, we start to increase the size of the first task and consequently its priority will increase while keeping the sizes of the other two tasks constant. We keep increasing the size of the first task until it becomes un-executable (with unacceptable execution probability). The same is applied on the second and the third task, where we increase

the size of each of them separately until it becomes un-executable, while keeping the sizes of the other tasks constant.

$$\begin{array}{lccc} \text{Sizes} & T_1 & T_2 & T_3 \\ S1: & 10 & 15 & 30 \\ S2: & 20 & 10 & 10 \\ S3: & 50 & 10 & 10 \\ S4: & 100 & 10 & 10 \\ S5: & 10 & 30 & 10 \\ S6: & 10 & 90 & 10 \\ S7: & 10 & 120 & 10 \\ S8: & 10 & 150 & 10 \\ S9: & 10 & 10 & 50 \\ S10: & 10 & 10 & 150 \\ S11: & 10 & 10 & 200 \\ S12: & 10 & 10 & 300 \end{array}$$

The tasks priorities are calculated using Equation (5.1) and depicted in Figure 5.6. The task priorities sum up always to 1.

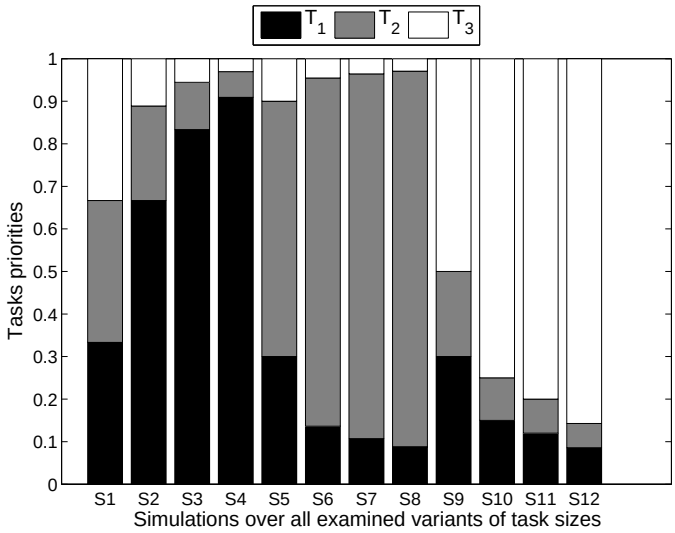
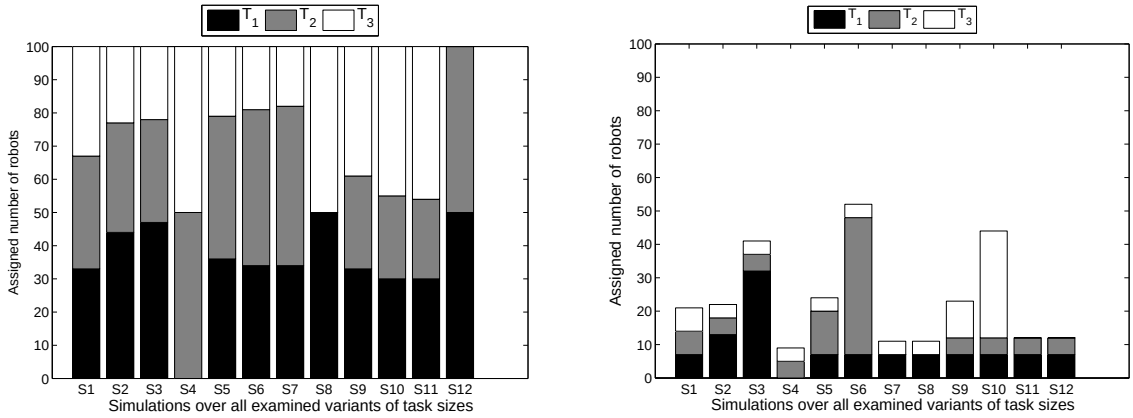


Figure 5.6 – *Tasks priorities calculated based on Equation (5.1) for the different variants of task sizes.*

The optimal number of robots which is associated to the maximum swarm performance is $N_{opt} = 50$ for the three tasks as we can see in Figure 5.5 (a). N_{opt} is applied as an upper-bound for the number of robots which could be assigned to the tasks. Since the mean time required to accomplish one part on any of the considered tasks, varies while changing the number of robots, hence it is calculated after the number of robots to assign is determined. This mean is used later to obtain the threshold of the total time that should be dedicated to the task up to its deadline.
Figure 5.7 shows number of robots assigned to the different tasks. Figure 5.7 (a) according to the energy-aware strategy and Figure 5.7 (b) according to the robots-aware strategy.

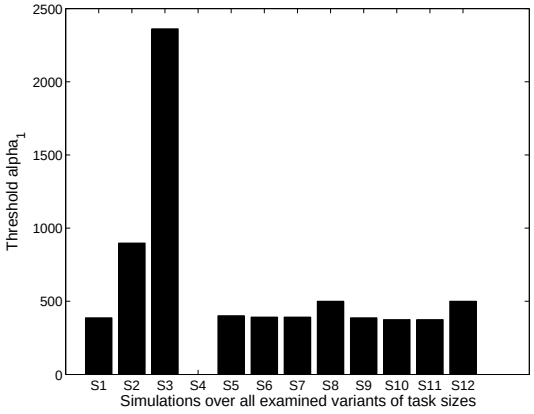


(a) *Under energy-aware strategy.* ***(b)*** *Under robots-aware strategy.*

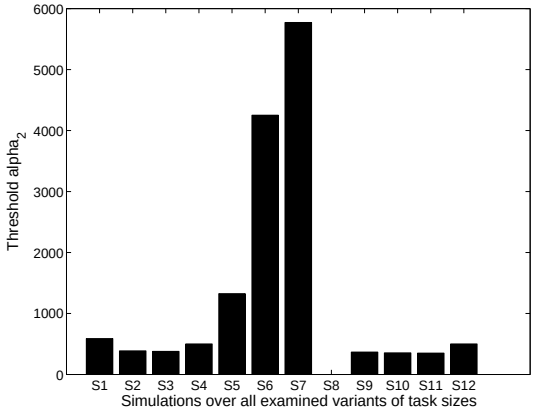
Figure 5.7 – *The number of robots assigned to each of the* 3 *tasks when a swarm of* $N = 100$ *robot is used to execute the different variants of the task sizes.*

For energy-aware strategy, the number of robots assigned to the task is maximized under the upper-bound $N_{opt} = 50$. In cases where the sum of the activity times dedicated by N_{opt} robots to task T_i, is shorter than the threshold α_i and consequently is not enough to execute the task within its deadline, the task is marked as unexecutable. We can see this in Figure 5.7 when no robot is assigned to the task. For robots-aware strategy, the number of robots assigned to the task is minimized.
Figures 5.8 and 5.9 show the threshold of the total time that should be dedicated to each of the three tasks, under both energy-aware and robots-aware strategies, respectively. They show how this threshold varies over the examined variants of

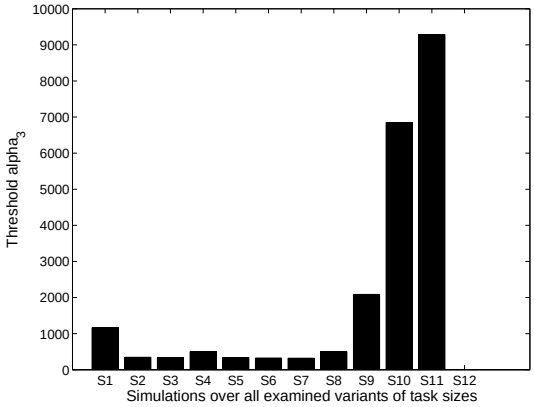
task sizes as the number of assigned robots N_i in addition to the size of the task S_i both change. The threshold is not depicted in the figures for un-executable tasks.



(a) On task T_1.

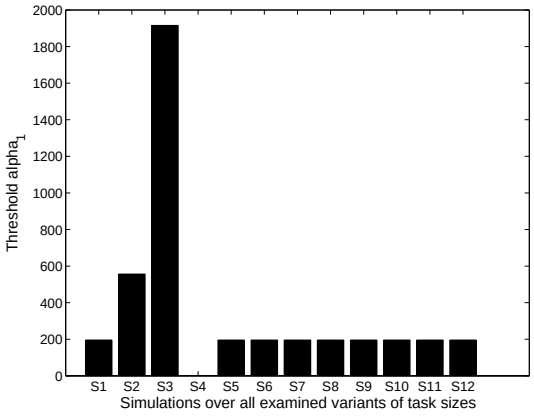


(b) On task T_2.

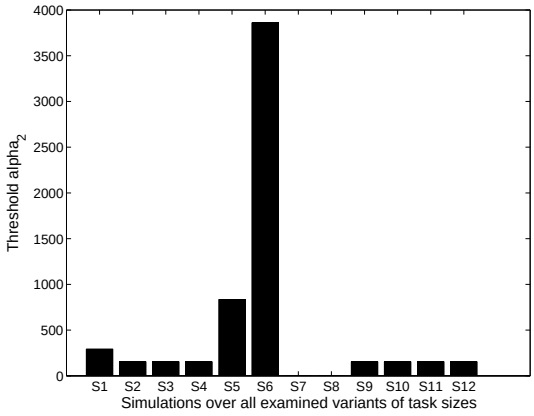


(c) On task T_3.

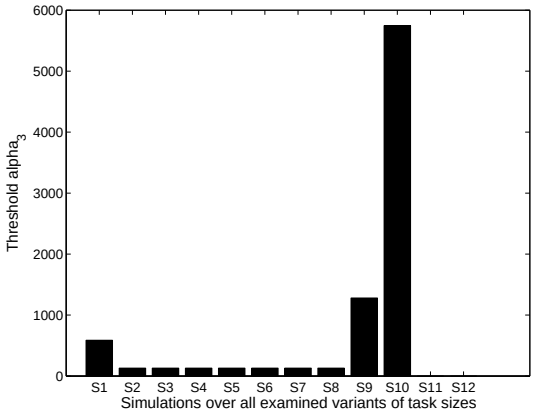
Figure 5.8 *– The threshold α_i associated with the total time required to be dedicated to the task. It is calculated over the different variants of the task sizes under* energy-aware *strategy.*



(a) On task T_1.



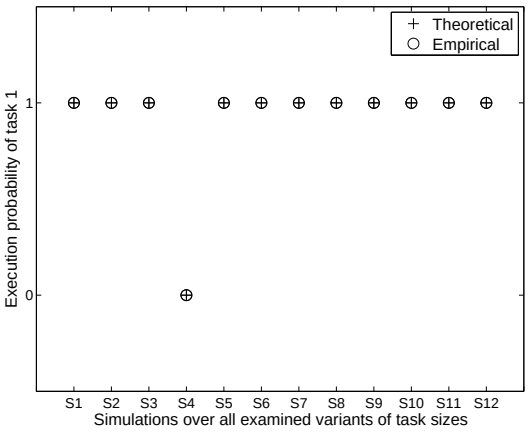
(b) On task T_2.



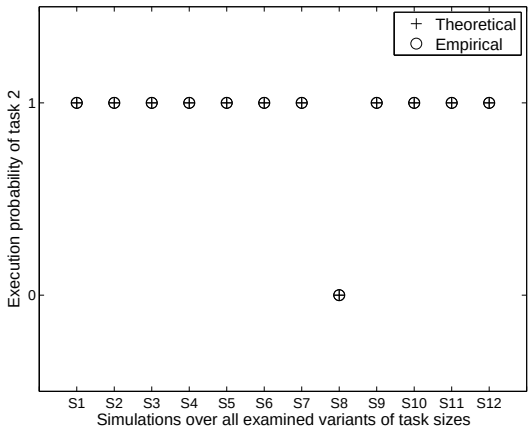
(c) On task T_3.

Figure 5.9 – *The threshold α_i associated with the total time required to be dedicated to the task. It is calculated over the different variants of the task sizes under* robots-aware *strategy.*

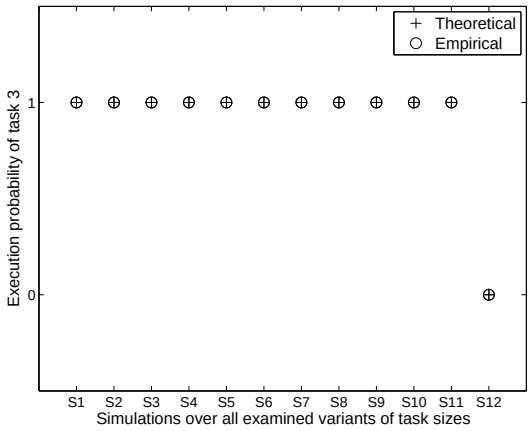
Figures 5.10 and 5.11 show the execution probability associated with each of the three tasks over the examined variants of task sizes. The execution probability is equal to 1 when the total time dedicated to the task up to its deadline is equal to or greater than the minimum threshold. The execution probability is 0 otherwise. The execution probabilities of the tasks are averaged over 1000 runs of Monte-Carlo simulation and depicted in addition to the calculated execution probabilities in the figures 5.10 and 5.11 for both energy-aware and robots-aware strategies respectively.



(a) *On task* T_1.

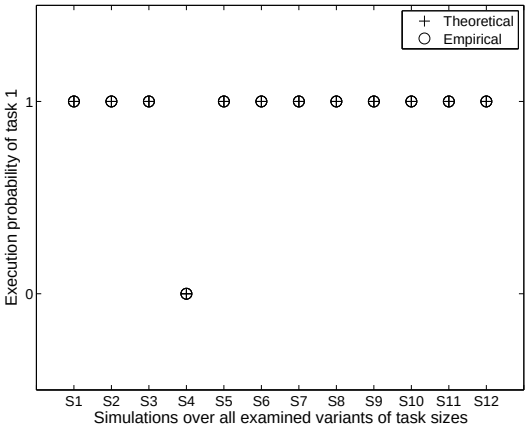


(b) *On task* T_2.

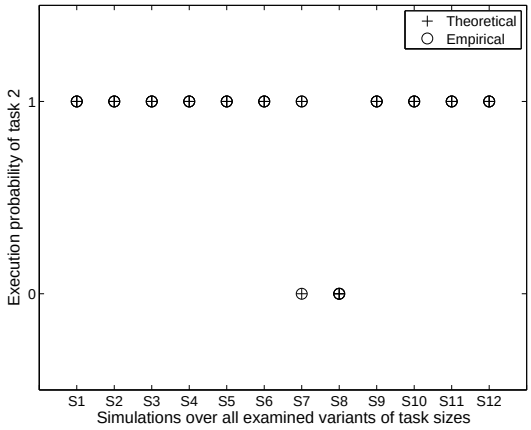


(c) *On task* T_3.

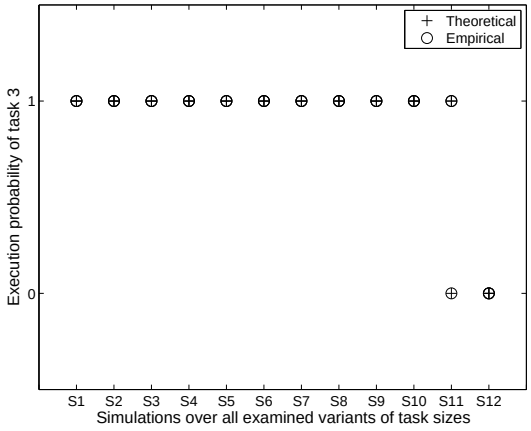
Figure 5.10 – *The execution probability* $\Pr(T_i)$ *of the three tasks for the of different variants of the task sizes under energy-aware strategy.*



(a) On task T_1.



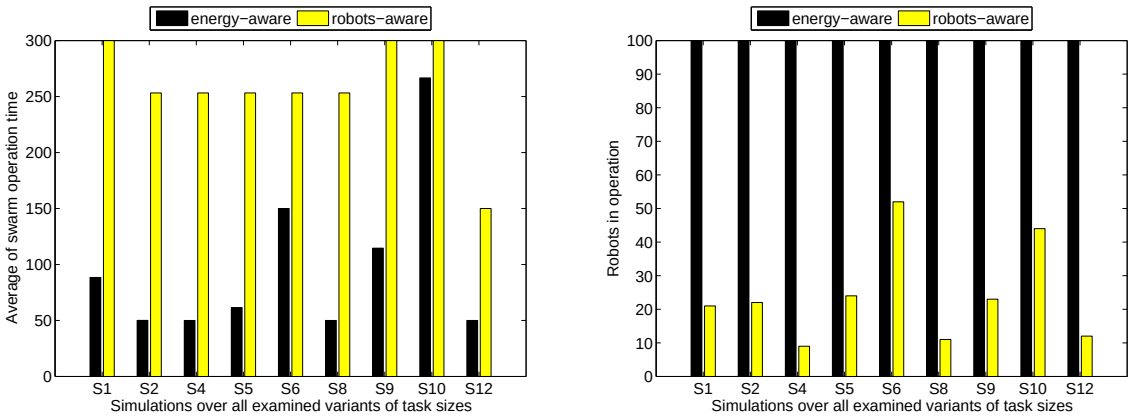
(b) On task T_2.



(c) On task T_3.

Figure 5.11 *– The execution probability* $\Pr(T_i)$ *of the three tasks for the of different variants of the task sizes under robots-aware strategy.*

Finally we perform a comparison between the two strategies concerning the two following criteria: First, the average time spent by the swarm operating on the tasks before the whole swarm is free and available again. Second, the total number of robots in operation, which refers to the sum of robots working on the different tasks. For the comparison to be fair, we select the task sets where both strategies agree on the execution probabilities of their tasks. From Figures 5.10 and 5.11, we can see that the two strategies agree on the execution probabilities associated with the task sets: $\{S1, S2, S4, S5, S6, S8, S9, S10, S12\}$. Therefore, the comparison between the two strategies is applied over the task sets: $\{S1, S2, S4, S5, S6, S8, S9, S10, S12\}$. Figure 5.12 (a) shows how the energy-aware strategy outperforms the robots-aware strategy, concerning the average time required to finish the complete set of executable tasks and to have the whole swarm free again. This expresses the motivation behind developing the energy-aware strategy, namely, to cope with the limited life-time of robots' batteries and to make the swarm available for further tasks as soon as possible. Figure 5.12 (b), on the other hand, shows how the robots-aware strategy outperforms the energy-aware one, concerning the number of free robots available continuously and ready for any new demand.



(a) *The average time to finish the* 3 *tasks.* ***(b)*** *The number of robots in operation.*

Figure 5.12 *– A comparison between energy-aware and robots-aware allocation strategies.*

5.2.2 Task Allocation for Hard-deadline Tasks under Dynamic Allocation

As mentioned in Chapter 4, Section 4.5, dynamic allocation is the technique developed for time-constrained tasks with negligible switching costs. It provides the system with a high level of flexibility in assigning and reassigning the robots among the tasks during their execution time. In addition it relaxes the constraint of working on the same task, therefore, in cases where the robot may encounter parts of different tasks it can execute them. Swarm performance under dynamic allocation, is defined as the amount of work (the number of parts) accomplished by the swarm during a specific time period. The progress of this performance over time is modeled using the well-known Poisson process with a task-specific rate, see Chapter 4 Section 4.5.1.

The allocation strategy developed under dynamic allocation should govern the average number of robots working on the task in order to obtain a particular progress rate that respects the time constraint of that task (the deadline).

We assume that the average time required to accomplish individual parts on the considered tasks can be estimated easily via short-term experiments or computer simulations. The inputs for the allocation strategy are:

- The task deadlines;
- The task sizes;
- The mean time μ_i required to accomplish one part on task T_i by the swarm under the spatial interferences. This mean is characterized over short-term experiments for different swarm sizes.

The output of the allocation strategy will be a probability matrix of the following form:

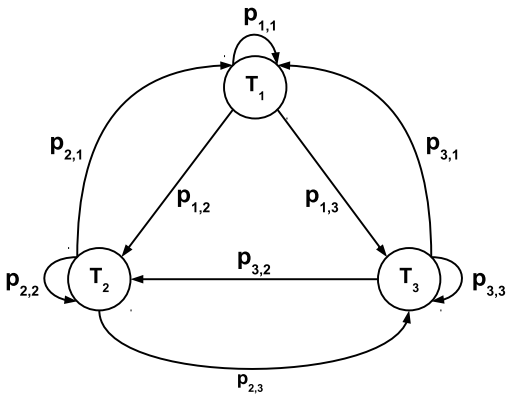
$$P_{m,m} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,m} \end{pmatrix}$$

where $p_{i,j}$ is the probability of switching from task T_i to task T_j.

The robots use this matrix to allocate themselves to the different tasks, independently without any central control. Following the given allocation matrix, the progress rate on each of the considered tasks, respects the time constraint of that task.
The allocation strategy aims to maximize the number of executable tasks according to the task priorities.
Before explaining the details of the allocation strategy for hard-deadline tasks under dynamic allocation, we present how the individual robot's behavior and the global swarm performance are modeled.

5.2.2.1 Semi-Markov Model for Individual Robots' Behavior

The behavior of individual robots under dynamic allocation can be described as follows: Each robot selects one of the tasks to work on. The robots starts to accomplish parts of their selected tasks and each time the robot finishes executing one part, it has the possibility to switch to another task or to continue on the same task. The average time required by the robot to accomplish one part of task T_i, is denoted by $\hat{\mu_i}$, and is assumed to be known a priori.
The robot with the above-described behavior can be modeled as an individual process over M states (tasks). Each of the defined processes can visit the different states continuously during the execution time of the tasks. It stays in the visited task for a random time, namely, the time required to accomplish one part of that task whose mean is known a priori. After executing one part of the current task, the robot chooses the next task (state) to visit and the choice is made among the M tasks including its current one. The described process associated with each robot represents a *Semi-Markov* process (Ross, 2006). The robots choose the next task based on a probability matrix referred to as the *decision matrix*. The decision matrix describes the limiting probabilities or also known as the invariant measure, which will be used by the individual robots to switch between any pair (T_i,T_j) of tasks as illustrated in Figure 5.13.



$$P_{m,m} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,m} \end{pmatrix}$$

Figure 5.13 – *The Markov chain associated with the tasks and its decision matrix.*

The semi-Markov process associated with the behavior of individual robots has an invariant (limiting) probability measure, π_i, which is obtained by solving the following system:

$$\pi_i = \sum_{j=1}^{M} \pi_j p_{j,i} \qquad \text{where} \sum_{i=1}^{M} \pi_i = 1 \tag{5.5}$$

π_i represents the proportion of transitions that take the robots into task T_i. Therefore, the proportional time the robot spends at task T_i, when $\hat{\mu}_i$ is the mean time the robot spends at task T_i at each visit to this task, is given by:

$$\tau_i = \frac{\pi_i \hat{\mu}_i}{\sum_{j=1}^{M} \pi_j \hat{\mu}_j} \tag{5.6}$$

However, for time-constrained tasks, we are interested in the time spent by the robot on task T_i up to the deadline D_i. This time is denoted by $\tau_i(D_i)$ and can be obtained by using Equation (5.6) as in the following:

$$\tau_i(D_i) = \frac{\pi_i \hat{\mu}_i}{\sum_{j=1}^{M} \pi_j \hat{\mu}_j} D_i \tag{5.7}$$

When a swarm of N robots is used to execute the M tasks and each single robot is modeled as a semi-Markov process with the above-described behavior, the total time $\tau_i(N, D_i)$ spent on task T_i up to its deadline D_i can be calculated as follows:

$$\tau_i(N, D_i) = \frac{\pi_i \hat{\mu}_i}{\sum_{j=1}^{M} \pi_j \hat{\mu}_j} D_i N \tag{5.8}$$

Consequently, the number of times $n_i(N, D_i)$ at which task T_i is expected to be visited up to its deadline D_i by a swarm of N robot, is given by:

$$n_i(N, D_i) = \frac{\pi_i}{\sum_{j=1}^{M} \pi_j \hat{\mu_j}} D_i N \tag{5.9}$$

The rate of visits λ_i to task T_i within the deadline D_i by the swarm of N robots which represents at the same time the number of parts expected to be accomplished on T_i up to its deadline, is given by dividing Equation (5.9) by the duration of the deadline:

$$\lambda_i = \frac{\pi_i}{\sum_{j=1}^{M} \pi_j \hat{\mu_j}} N \tag{5.10}$$

5.2.2.2 Poisson Model for Swarm Performance

The evolution of swarm performance under dynamic allocation over time is modeled using the Poisson process, see Chapter 4 Section 4.5.1. The rate of the Poisson process should be designed to respect the deadline of the task on which it models the performance progress.

On the other hand, time-constrained tasks are marked as *inactive* when one of the two following conditions is valid: when the task is executed even if the task deadline has not been reached yet, or, when the deadline of the task is reached. Otherwise, the task is marked as *active*. In case of hard deadlines, robots work on the task as long as the task is active and they stop when it becomes inactive. Therefore, the number of robots available to be assigned varies based on the number of current active tasks. Hence, we divide the time between the start of the tasks execution and the largest deadline into periods, where the length η_i of the ith period is given by:

$$\eta_i = D_i - D_{i-1} \qquad \forall i \in \{2, \ldots, M\} \tag{5.11}$$

where $\eta_1 = D_1$. In general, the number of active tasks decreases over consecutive periods and consequently the number of robots available to be reassigned increases. Therefore, the rates of the Poisson processes, which are associated with the active tasks, change over the periods as we can see in Figure 5.14.

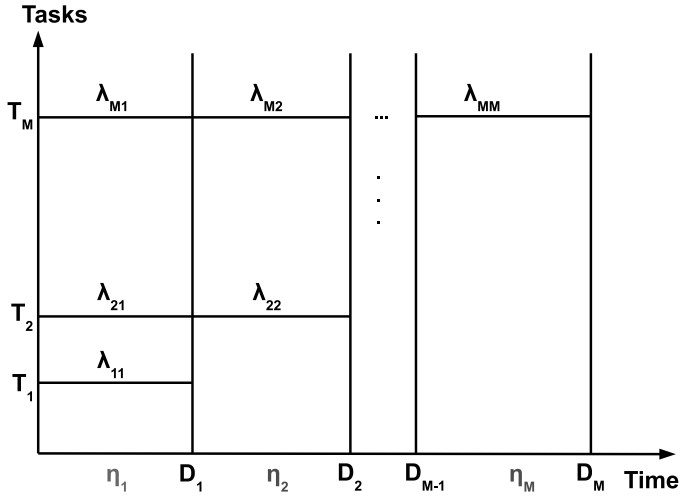


Figure 5.14 *– The Poisson rates over the activation periods of the tasks.*

We link each rate λ_{ij} on task T_i to an individual Poisson process, which models the progress of the swarm performance θ_{ij} over the jth period of task T_i. Thus, we have:

$$\theta_{ij} \sim \mathcal{P}oisson(\lambda_{ij}\ \eta_j)$$

It is well-known that the sum of Poisson processes is a Poisson process with the rate equal to the sum of the rates associated with the summed up processes. Consequently, the progress of the total amount of work accomplished on task T_i is modeled as a Poisson process with the rate equal to the sum of the rates of the individual Poisson processes modeling the work progress θ_i over the different periods of task T_i:

$$\theta_i \sim \mathcal{P}oisson(\sum_{j=1}^{i} \lambda_{ij}\ \eta_j) \qquad \forall i \in \{1, \ldots, M\} \tag{5.12}$$

5.2.2.3 Task Allocation Strategy for Hard-deadlines under Dynamic Allocation

Since hard-deadline tasks are tasks where the correctness of the results is associated with accomplishing the tasks within their deadlines. The developed allocation strategy aims to perform the robots' assignment such that the number of met deadlines, is increased. For a stochastic system such as swarm robotics, the task execution is analyzed in terms of the execution probability, which is the probability of finish

executing the task before or at its deadline. Consequently, the allocation strategy aims to maximize the number of tasks with acceptable execution probability.
The allocation strategy generates a decision matrix which holds the probabilities used by the robots to assign themselves independently among the different active tasks in the current period. As mentioned in Section 5.2.2.2, the Poisson process which models the swarm performance on task T_i has a rate equal to the sum of the rates associated with the Possion processes which model the work progress on the task over all its activation periods, Equation (5.12). Based on the definition of the Poisson process with the rate λ, the random number of events generated within the time period, δt, follows the Poisson distribution with the parameter $\lambda\delta t$. Consequently, the probability of accomplishing an amount of parts equal to or greater than S_i up to the deadline D_i, can be calculated using the cumulative distribution function of the Poisson distribution:

$$\begin{aligned}\Pr(\theta_i \geqslant S_i) &= 1 - \Pr(\theta_i < S_i) \\ &= 1 - \Pr(\theta_i \leqslant (S_i - 1)) \\ &= 1 - \left[e^{z_i} \sum_{j=0}^{S_i - 1} \frac{z_i^j}{j!} \right]\end{aligned} \tag{5.13}$$

where z_i is the rate of the Poisson process which models the work progress on task T_i and is calculated using Equation (5.12):

$$z_i = \sum_{k=1}^{i} \lambda_{ik}\ \eta_k \qquad \forall i \in \{1, \ldots, M\} \tag{5.14}$$

On the other hand, the rate at which the work is progressing on a specific task was calculated in Section 5.2.2.1 using Equation (5.10). This equation is used to calculate the rate of work progress on task T_i over any period j with the length η_j as in the following:

$$\lambda_{ij} = \frac{\pi_{ij}}{\sum_{c=j}^{M} \pi_{cj} \hat{\mu}_c} N \qquad \forall i \in \{1, \ldots, M\}, \forall j \in \{1, \ldots, M\} \tag{5.15}$$

By substituting Equation (5.15) in Equation (5.14) we have:

$$z_i = \sum_{k=1}^{i} \left(\frac{\pi_{ik}}{\sum_{c=k}^{M} \pi_{ck} \hat{\mu}_c}\ \eta_k \right)\ N \qquad \forall i \in \{1, \ldots, M\} \tag{5.16}$$

Now we can substitute Equation (5.16) in Equation (5.13) to calculate the probability of accomplishing at least S_i parts of task T_i up to the deadline D_i:

$$\Pr(\theta_i \geqslant S_i) = 1 - \left[e^{\sum_{k=1}^{i} \left(\frac{\pi_{ik}}{\sum_{c=k}^{M} \pi_{ck} \hat{\mu}_c} \eta_k \right)} \sum_{j=0}^{N S_i - 1} \frac{\left(\sum_{k=1}^{i} \left(\frac{\pi_{ik}}{\sum_{c=k}^{M} \pi_{ck} \hat{\mu}_c} \eta_k \right) N \right)^j}{j!} \right] \tag{5.17}$$

The following steps of the developed strategy for hard-deadline tasks under dynamic allocation, are performed *off-line* in order to obtain the necessary parameters required for the task allocation.

- ***Priority assignment:*** The tasks are assigned their priorities, which are calculated using Equation (5.1).
- ***Calculate the decision matrices:*** In this step the set of executable tasks (with execution probabilities $\Pr(T_i) \geqslant 1 - \epsilon$) is calculated, in addition to the decision matrix (transition probability matrix for each activation period). The allocation strategy exploits the available swarm to maximize the execution probability of each task based on its priority. The execution probability of task T_i is calculated as in Equation (5.17). Therefore, this equation represents the objective function we need to maximize for each of the M tasks. The optimization process searches for the limiting probabilities π_{ij} (transition probability) associated with task T_i over all the periods where this task is active. The optimization problem can be written as in the following:

$$\underset{\pi_{ij}}{\text{Maximize}} \quad \begin{cases} \Pr(T_1) = f(\pi_{11}) \\ \Pr(T_2) = f(\pi_{21}, \pi_{22}) \\ \vdots \\ \Pr(T_M) = f(\pi_{M1}, \pi_{M2}, \ldots, \pi_{MM}) \end{cases}$$

This represents a multi-objective optimization problem which can be solved by introducing the cumulative weighted objective function. The weights reflect the tightness of the time constraints of the different tasks. Hence, the

task priorities are used as the weights associated with the different objective functions:

$$\underset{\pi_{ij}}{\text{Maximize}} \quad \sum_{i=1}^{M} \beta_i \Pr(T_i) \qquad j \in [1, i]$$

After performing the optimization process, the tasks are categorized into: executable tasks with execution probability near to one and un-executable tasks with the execution probability out of the acceptable range.

The allocation strategy starts by eliminating the un-executable task with the lowest execution probability and repeats the step for the resulting task set. The step is repeated until the task set includes only executable tasks or it becomes empty.

- ***Minimize the number of assigned robots:*** After the set of executable tasks has been found, the allocation strategy aims to minimize the number of robots assigned to the executable tasks. This minimization has two advantages: First, it reduces the spatial interferences among robots by reducing the robots assigned to the task. Second, since the swarm size may be considerably larger than needed, minimizing the number of operating robots allows to preserve robotic resources for other purposes such as fault-tolerance or executing new arrived tasks. Minimizing the number of robots is performed under the constraint of maintaining an acceptable execution probability for each of the executable tasks. This optimization problem belongs to the class of *Integer linear programming (ILP)* problems. It can be expressed as in the following:

$$\begin{aligned} &\text{Minimize} && N \\ &\text{subject to:} && \begin{cases} \Pr(T_1) \geqslant 1 - \epsilon \\ \Pr(T_2) \geqslant 1 - \epsilon \\ \vdots \\ \Pr(T_{M'}) \geqslant 1 - \epsilon \end{cases} \end{aligned}$$

M' denotes the number of the executable tasks calculated in the previous step.

The output of the allocation strategy developed for hard-deadline tasks under dynamic allocation includes:

- The set of executable tasks;
- The decision matrices which hold the transition probabilities among the tasks during all their activation periods;
- The execution probability of each executable task.

After obtaining the above output, the robots can start to assign themselves autonomously to the tasks and switch among them each time they finish executing an individual part of their current task. The initial robots' assignment to the executable tasks and the later decision of their next task is performed by each robot, independently, using the decision matrix. Figure 5.15 illustrates the steps of the strategy, with the inputs marked in red and the outputs marked in blue. The calculated steps are performed off-line and the necessary part of the output to use by the robots is marked with blue arrows.

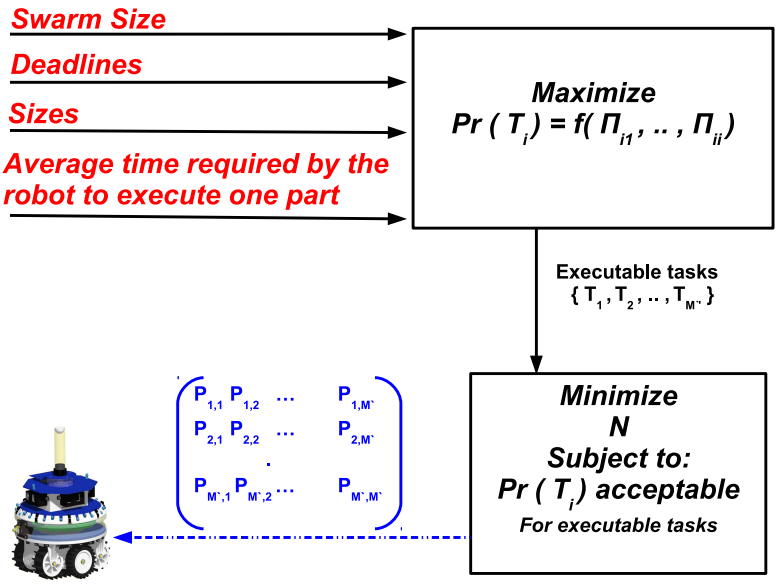


Figure 5.15 – *The task allocation strategy for hard deadlines under dynamic allocation.*

The following is the pseudo-code of the allocation strategy developed for hard-deadline tasks under dynamic allocation:

Algorithm 3 Strategy developed for hard-deadline tasks under dynamic allocation.

Input: S_i, D_i ,N, $\hat{\mu}_i = f(N_i)$
Output: solution $P_{M',M'}$

$M \leftarrow$ tasks
$Acceptable\_Prob \leftarrow 1 - \epsilon$ ▷ % ϵ is a design parameter%
$\beta_i = \dfrac{S_i/D_i}{\sum_{j=1}^{M} S_j/D_j}$ ▷ % calculate the task priorities%
$\eta_i = D_i - D_{i-1} \quad \forall i \in \{2, \ldots, M\}$ ▷ % task activation periods%
for $i = 1$ to M **do** ▷ % the rate of the Poisson process using Eq. (5.16)%
    $z_i \leftarrow \sum_{k=1}^{i} \left(\dfrac{\pi_{ik}}{\sum_{c=k}^{M} \pi_{ck}\mu_c} \eta_k \right) N \quad \forall i \in \{1, \ldots, M\}$
    $\Pr(\theta_i \geqslant S_i) = 1 - \left[e^{z_i} \sum_{j=0}^{S_i - 1} \dfrac{z_i^j}{j!} \right]$ ▷ % Eq. (5.13)%
end for
while M is not Empty **do**
    $\underset{\pi_{ij}}{maximize} \ \sum_{i=1}^{M} \beta_i \Pr(\theta_i \geqslant S_i)$
    for $i = 1$ to M **do**
        if $\Pr(T_i) \geqslant$ Acceptable_Prob **then**
            add T_i to the set of executable tasks
        else
            add T_i to the set of un-executable tasks
        end if
    end for
    if M includes un-executable tasks **then**
        Update M by removing the un-executable task with the lowest execution probability
    else
        Break
    end if
end while
$M' = M$
Maximize N

5.2.2.4 Scenario and Evaluation

Let us consider three tasks which are characterized by their hard deadlines $D = \{500, 1000, 1500\}$ time units and their negligible switching costs. Therefore, the dynamic allocation is the proper technique to be applied. A homogeneous swarm of $N = 100$ robots is used to execute the tasks. Different variants of the task sizes are examined as listed in the below table, where each line holds a new variant of the task sizes.

$$\begin{array}{lccc} \text{Sizes} & T_1 & T_2 & T_3 \\ S1: & 300 & 600 & 900 \\ S2: & 450 & 450 & 750 \\ S3: & 200 & 1000 & 700 \\ S4: & 250 & 450 & 1400 \\ S5: & 750 & 240 & 350 \\ S6: & 750 & 240 & 500 \\ S7: & 1000 & 240 & 500 \\ S8: & 1000 & 500 & 500 \\ S9: & 1000 & 1000 & 500 \\ S10: & 1000 & 1000 & 1000 \end{array}$$

The tasks priorities are calculated for each variant of the task sizes using Equation (5.1) and are depicted in Figure 5.16. The task priorities sum up to always to 1.

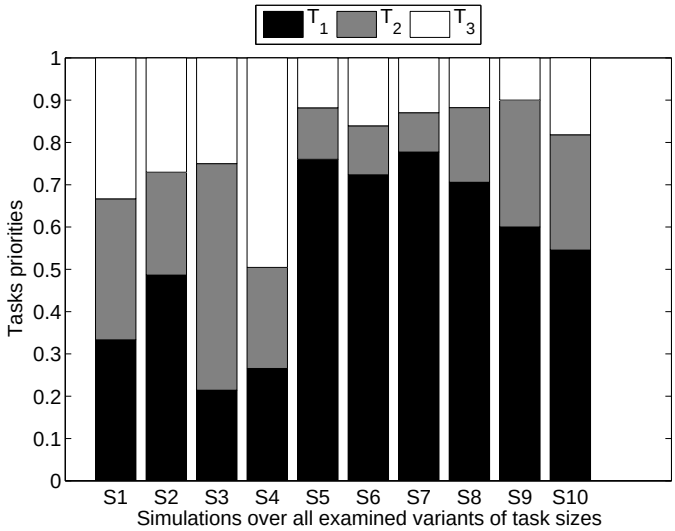
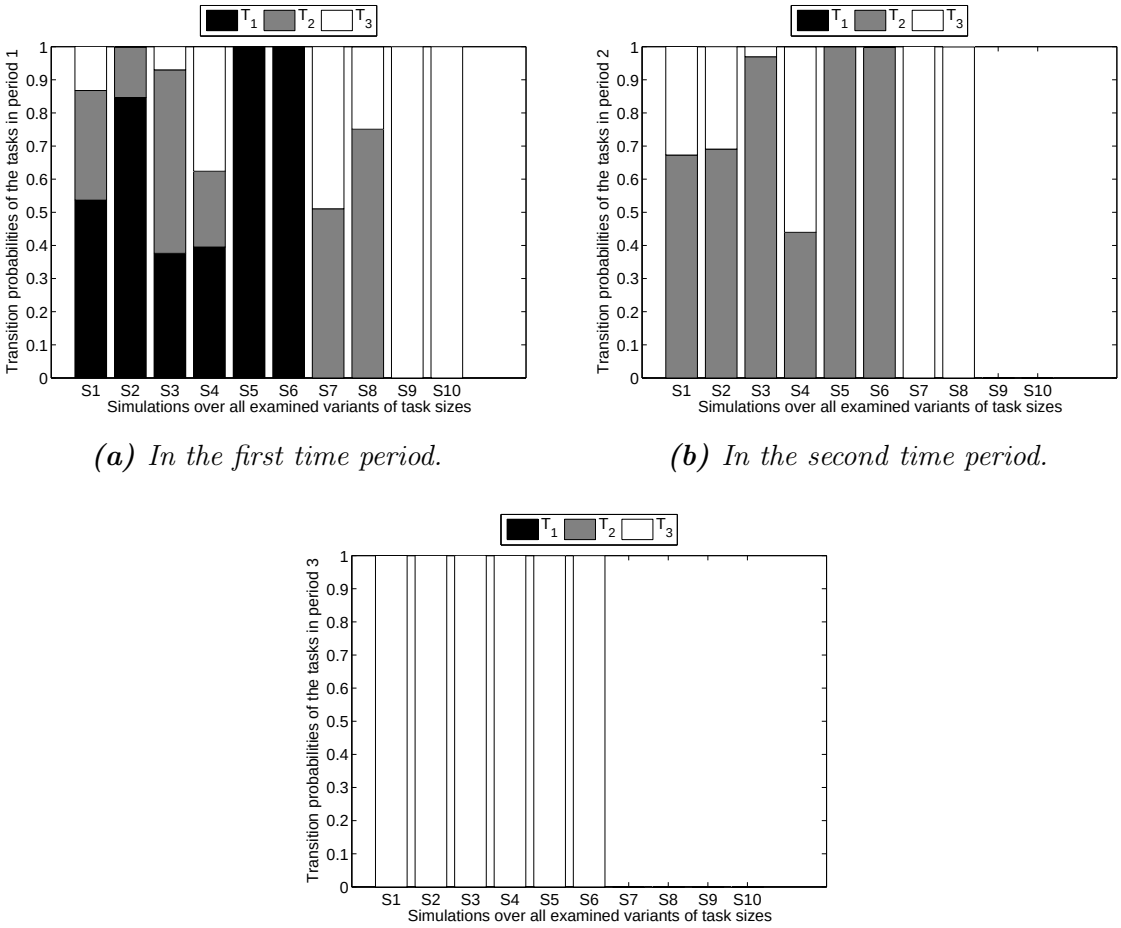


Figure 5.16 – *Tasks priorities calculated based on Equation* (5.1) *for the different variants of sizes.*

Figure 5.17 depicts the calculated transition probabilities which are used by the robots, independently: Figure 5.17 (a) in the first activation period $[0 - 500]$, in Figure 5.17 (b) in the second activation period $[500 - 1000]$, and in Figure 5.17 (c) in the third activation period $[1000 - 1500]$.



(a) *In the first time period.*

(b) *In the second time period.*

(c) *In the third time period.*

Figure 5.17 – *The transition probabilities of the three tasks over their activation periods.*

As we can notice in Figure 5.17, task T_1 is inactive over the second and the third activation periods, therefore, the transition probability of this task in the second and the third activation periods are zeros for all the examined variants of the task sizes. The same applies for task T_2 over the third activation period, where all robots are assigned to task T_3 and the probability to select task T_3 becomes equal to 1.
The execution probabilities of the tasks are maximized by the allocation strategy as explained in Section 5.2.2.3. The tasks with acceptable execution probabilities (near to one), are categorized as executable tasks whereas the others as un-executable. After that, the swarm size is minimized to reduce the robots' interferences and to

increase the size of free robots. The minimization is performed under the constraint of preserving an acceptable execution probability of the executable tasks. Figure 5.18 shows the number of robots N, which are used to execute each variant of the task sizes.

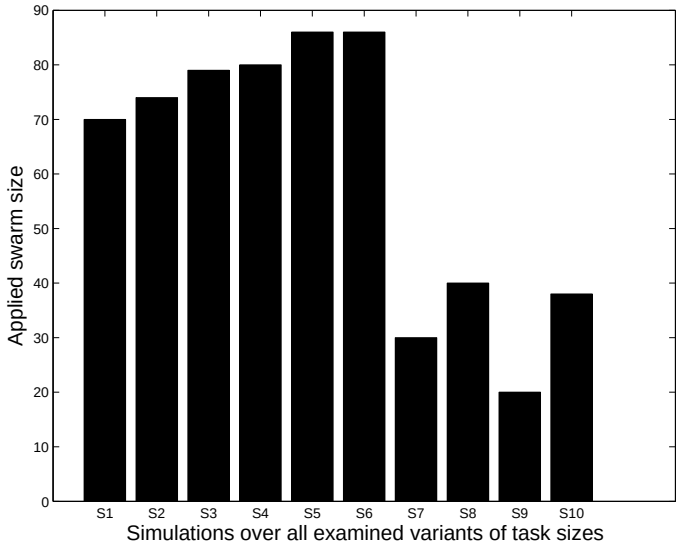
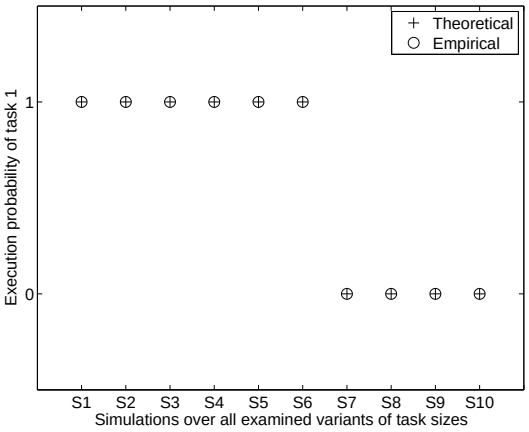
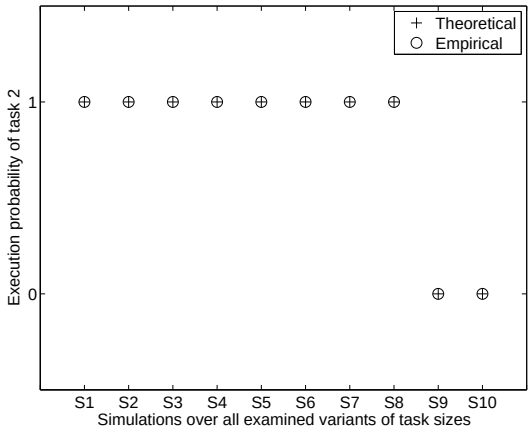


Figure 5.18 – *The swarm size optimized to be used for each set of the examined task sizes.*

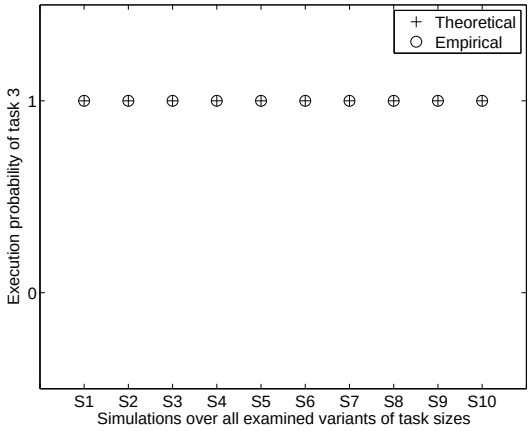
Finally, Monte-Carlo simulations are used to verify the robots' allocation results, where the calculated decision matrices of the different activation periods are used by the robots to allocate themselves to the different tasks. After selecting task T_i, each robot requires a time with the mean $\hat{\mu}_i$ to perform one part of task T_i before it selects its next task based on the decision matrix of the current activation period. The time required to accomplish an individual part is assumed to have the mean $\hat{\mu}_i = 50$ and the standard deviation $\hat{\sigma}_i = 0.0001$. The empirical execution probabilities are averaged over 100 runs of Monte-Carlo simulation and are compared to the calculated probabilities in Figure 5.19 for the three considered tasks.



(a) On task T_1.



(b) On task T_2.



(c) On task T_3.

Figure 5.19 *– The execution probability* $\Pr(T_i)$ *of the three tasks for the different variants of task sizes under dynamic allocation.*

5.3 Soft-deadline Tasks

Soft-deadline tasks are such tasks where missing the deadline is not related to the correctness of the results but to their quality. In other words, missing a soft deadline of a specific task reduces the quality of the system performance and this is associated normally with particular costs. The costs increase when the unprocessed part of the task at its deadline increases.

Planning the execution of time-constrained tasks on a stochastic system such as swarm robotics is a critical process, especially for hard deadlines as missing them violates the correctness of the results. Soft deadlines represent a more tolerant kind of deadlines to be executed by swarm robotics. In the case of soft deadlines, the robots can continue working on the task even after its deadline is expired, however, particular costs will be encountered. These costs are calculated based on the size of task, which remains unprocessed at the task deadline.

Since swarm performance was defined differently under static and dynamic allocations, see Chapter 4, Section 4.4 and Section 4.5, therefore the costs associated with missing a soft deadline are defined differently for static and dynamic allocations. For static allocation, the costs are associated with the time missed to be dedicated to the task before its deadline is reached. The expected value of this time represents the difference between the minimum threshold of the total time that should be dedicated to the task up to its deadline and the expected time that will be dedicated to the task up to its deadline D_i. It was calculated in Chapter 4, Equation (4.10) for task T_i as:

$$\mathbb{E}(\psi_i) = \alpha_i - \mathbb{E}(\tau_i(D_i))$$

For dynamic allocation, the costs are associated with the number of parts which remains unprocessed at the task deadline. The expected value of this work amount represents the difference between the task size and the expected amount of parts which is accomplished at the task deadline. It was calculated in Chapter 4, Equation (4.42) for task T_i as:

$$\mathbb{E}(\psi_i) = S_i - \mathbb{E}(\theta_i(D_i))$$

The expected values calculated above are referred to as the *performance variance* and the quality of the system performance increases by decreasing this performance variance. Consequently, the goal of the allocation strategy for soft-deadline tasks, is

to assign the robots such that, the performance variance on all the tasks is minimized based to their priorities. The optimal performance is achieved when no costs are present, i.e. when the performance variance on each task is equal to zero.
In the following, we present the allocation strategies developed to allocate a swarm of robots to a set of soft-deadline tasks under both techniques of static and dynamic allocation.

5.3.1 Task Allocation Strategy for Soft-deadline Tasks under Static Allocation

Under static allocation, the robots sample individual activity times and dedicate them to the selected tasks. They work during their activity times and as soon as these are expired, the robots become inactive. As we have seen in Chapter 4 Section 4.4.1, there are different probability distributions which can be used for the robots' activity times. In this section, we consider the exponential distribution, however, using any other distribution is possible by substituting with the proper probability functions.
In the same Chapter 4 in Section 4.4.3, the expected variance in the swarm performance was presented associated with the exponential distribution of the robots' activity times and was calculated as in the following:

$$\mathbb{E}(\psi_i) = \begin{cases} \alpha_i - \dfrac{N_i}{\lambda_i} & \text{for } \dfrac{N_i}{\lambda_i} < \alpha_i \\ 0 & \text{for } \dfrac{N_i}{\lambda_i} \geqslant \alpha_i \end{cases}$$

where α_i is the minimum threshold of total time that should be dedicated to task T_i up to its deadline. N_i is the number of robots assigned to task T_i, and λ_i is the rate parameter of the exponential distribution used to sample the robots' activity times.
As mentioned above for soft deadlines, the robots can continue to execute the tasks even after their deadlines have been exceeded. This occurs when the total time which should be dedicated to the task is not reached within the deadline, hence, the robots continue to work and stop only when the required time is dedicated. Executing the task after its deadline being exceeded is associated with specific costs which are higher for tasks with higher priorities.

The allocation strategy, developed here, aims to assign the robots to the tasks such that the part of time which is missed to be dedicated up to the deadline, is minimized for all tasks based on their priorities. In other words, the goal is to minimize the expected variance of the swarm performance which is defined in the equation above as:

$$\begin{aligned}\mathbb{E}(\psi_i) &= \alpha_i - \frac{N_i}{\lambda_i} \\ &= S_i\hat{\mu}_i(N_i) - \frac{N_i}{\lambda_i}\end{aligned}$$

As we can notice in the above-equation, increasing the number of assigned robots N_i will increase obviously the second term of the right-side, namely $\frac{N_i}{\lambda_i}$. However, it will also increase the first term, α_i, which is calculated as $\alpha_i = S_i\hat{\mu}_i(N_i)$, where $\hat{\mu}_i(N_i)$ represents an exponentially increasing function of N_i. In order to eliminate this contradiction while searching the solution state-space to minimize $\mathbb{E}(\psi_i)$, we could express the variance of the swarm performance as the difference between the time required to accomplish the S_i parts *by the swarm* and the time which the *swarm* is expected to dedicate to task T_i. The time required by the swarm to accomplish S_i part of task T_i can be calculated as $S_i\mu_i(N_i)$, where $\mu_i(N_i)$ is the mean time required by the swarm of size N_i to accomplish one part of task T_i. On the other hand, when the robots sample their activity time from an exponential distribution with the rate λ_i, the time which the swarm is expected to dedicate to task T_i is given by $\frac{1}{\lambda_i}$. Consequently, the objective function associated with task T_i which needs to be minimize can be written as in the following:

$$\mathbb{E}(\psi_i') = S_i\mu_i(N_i) - \frac{1}{\lambda_i} \tag{5.18}$$

The final formula of the objective function will be as in the following:

$$\mathbb{E}(\psi_i') = \begin{cases} S_i\mu_i(N_i) - \frac{1}{\lambda_i} & \text{for } \frac{1}{\lambda_i} < S_i\mu_i(N_i) \\ 0 & \text{for } \frac{1}{\lambda_i} \geqslant S_i\mu_i(N_i) \end{cases}$$

This represents a multi-objective optimization problem with M objective functions. Each objective function is associated with one of the tasks and aims to minimize

the costs associated with executing that task after its deadline is exceeded. This optimization is performed under the constraint of the limited swarm size.

$$\underset{N_i,\lambda_i}{\text{minimize}} \quad \begin{cases} \mathbb{E}(\psi'_1) = f(N_1, \lambda_1) \\ \mathbb{E}(\psi'_2) = f(N_2, \lambda_2) \\ \vdots \\ \mathbb{E}(\psi'_M) = f(N_M, \lambda_M) \end{cases}$$

$$\text{subject to:} \quad \sum_{i=1}^{M} N_i \leqslant N$$

Each of the above optimization problems represents a mixed-integer non-linear optimization one where λ_i is the rate parameter of the exponential distribution of the robots' activity times and the integer number N_i is the number of robots which should be assigned to task T_i. N_i belongs to the range $[0, min(N_{opt}, N)]$, where N_{opt} represents the optimal number of robots that produces the maximum performance on task T_i under the influence of spatial interferences. For simplicity, N_{opt} is assumed to be the same over all tasks. N_i and the parameter λ_i should be selected such that the time missed to be dedicated to the task within its deadline, is minimized. The output of this optimization is the assignment probability associated to the different tasks and which is calculated as in the following:

$$P_i = \frac{N_i}{N}$$

and the parameters λ_i, of robots' activity time distribution.

The above multi-objective optimization problem is solved *off-line*. It is solved through building a single optimization problem by constructing the cumulative weighted objective function. We sum the M objective functions, where each is weighted by the priority of its task as in the following:

$$\underset{N_i,\lambda_i}{\text{minimize}} \quad \sum_{i=1}^{M} \beta_i \mathbb{E}(\psi'_i)$$

$$\text{subject to:} \quad \sum_{i=1}^{M} N_i \leqslant N$$

After obtaining the above output, the robots can start to assign themselves autonomously to the tasks and each robot samples its own activity time under the

consideration of the task deadline. Figure 5.20 illustrates the steps of the allocation strategy developed for soft-deadline tasks under static allocation with the inputs marked in red and the outputs marked in blue. The calculated steps are performed, as mentioned above, off-line and the output to use by the robots is marked with blue arrows.

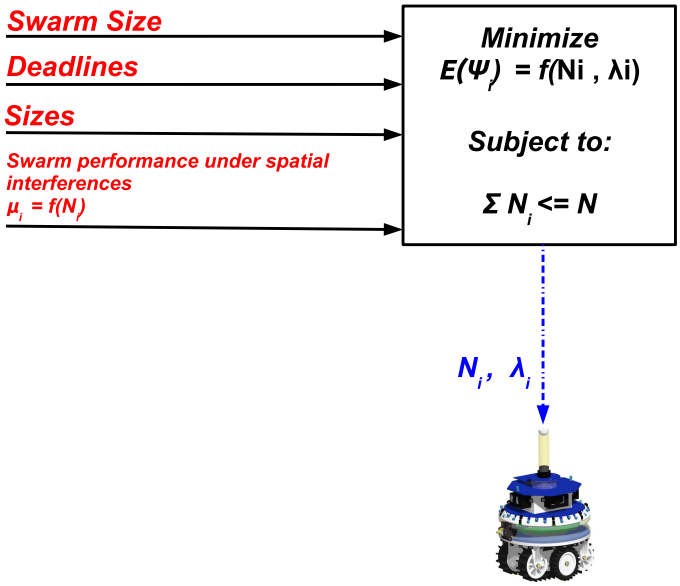


Figure 5.20 – *The task allocation strategy for hard deadlines under static allocation.*

The following is the pseudo-code of the allocation strategy for soft-deadline tasks under static allocation:

Algorithm 4 The allocation strategy for soft-deadline tasks under static allocation.

Input: S_i, D_i ,N, $\mu_i = f(N_i)$
Output: solution N_i and solution λ_i

1: $M \leftarrow$ tasks
2: $\beta_i = \dfrac{S_i/D_i}{\sum_{j=1}^{M} S_j/D_j}$ ▷ % calculate the task priorities%
3: **for** $i = 1$ to M **do**
4: $N_{opt}(i) = \underset{N_i}{minimum}\ \mu_i$ ▷ % calculate the optimal robots number%
5: $\mathbb{E}(\psi_i') = max(0, S_i\mu_i(N_i) - \dfrac{1}{\lambda_i})$
6: **end for**
7: minimize $\sum_{i=1}^{M} \beta_i \mathbb{E}(\psi_i')$

subject to: $\sum_{i=1}^{M} N_i \leqslant N$

5.3.1.1 Scenario and Evaluation

Let us consider an example of three tasks which can be characterized with their sizes $\{500, 700, 1000\}$ parts and their soft deadlines $D_i = \{50, 200, 600\}$ time units. We use a homogeneous swarm of an increasing size to execute this set of tasks. The swarm size varies over the range $[5 - 150]$ robots, with an increment-step of 10 robots.

The time required to accomplish one part by the swarm is assumed to be the same for the three tasks. Figure 5.5 (a) (page 93) shows how the mean of this time changes with increasing the number of robots working on the task. On the other hand, the mean of the time required by a single robot to accomplish one part is calculated and depicted 5.5 (b). It shows how the mean time increases by increasing the number of robots working on the task.

Figure 5.21 depicts the task priorities calculated using Equation (5.1).

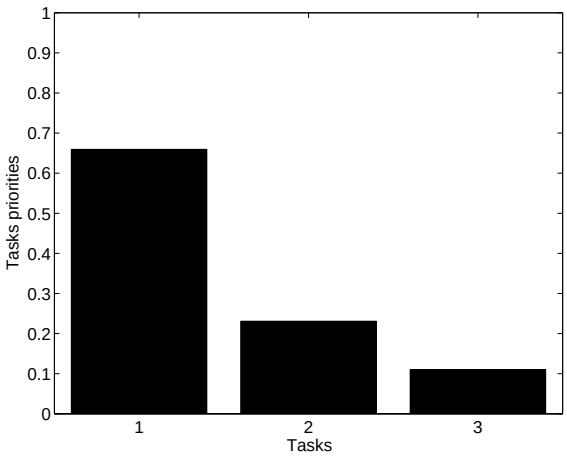


Figure 5.21 – *The priorities of the three tasks.*

The optimal number of robots which is associated with the maximum swarm performance is $N_{opt} = 50$ for the three tasks as we can see in Figure 5.5 (a). N_{opt} is applied as an upper-bound for the number of robots which could be assigned to any of the tasks. Since the mean time required to accomplish one part on any of the considered tasks, varies while changing the number of robots, hence it is calculated after the number of robots to assign is determined. This mean is used later to obtain the threshold of the total time that should be dedicated to the task up to its deadline. The number of robots which should be assigned to each of the considered tasks in addition to the rate parameter of the activity times distribution are calculated. Figures 5.22 (a-c-e) show how the number of robots assigned to the different tasks increases by increasing the size of the swarm. The increment on the different tasks occurs according to their priorities. This is why the number of the robots starts to increase first on task T_1 and then on tasks T_2 and T_3 sequentially. In Figures 5.22 (b-d-f), we can see how the time missed to be dedicated to the tasks within their deadlines, is decreasing with increasing the number of robots assigned to that task. The number of robots increases on each task up to the optimal number of robots $N_{opt} = 50$ specified under spatial interferences. The same figure shows a comparison between the expected value of the missed time to be dedicated calculated and averaged over 100 runs of Monte-Carlo simulation for all the examined swarm sizes.

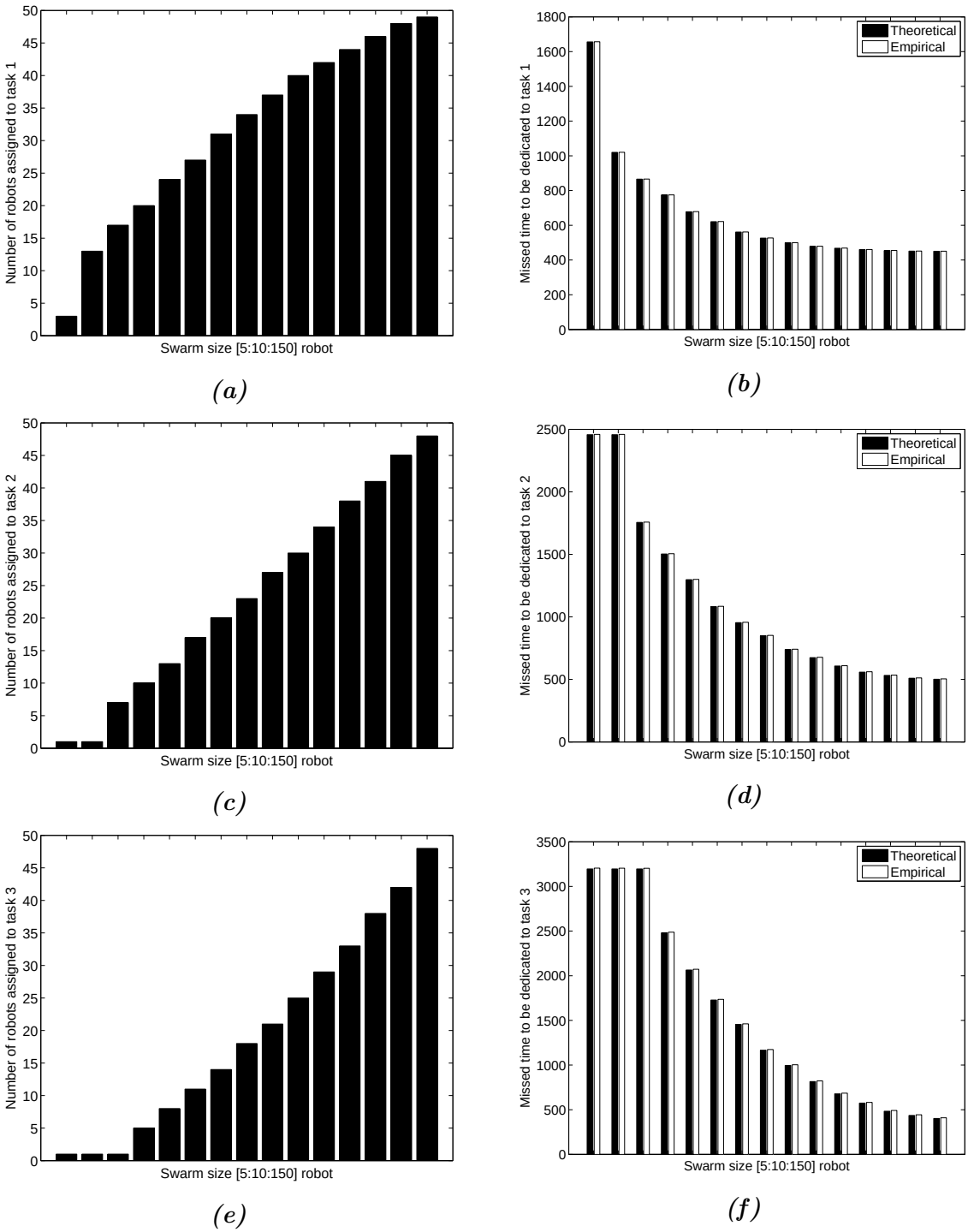


Figure 5.22 – *Evaluation of the task allocation strategy developed for the soft deadlines under the technique of static allocation. Figures (a), (c) and (e) show the number of robots assigned to each of the considered tasks by increasing the swarm size over the range* $[5 : 10 : 150]$*. Figures (b), (d) and (f) show the performance variance associated with the used size of the swarm. A comparison is performed between the calculated and the simulated performance variance over* 100 *runs of Monte-Carlo simulation.*

5.3.2 Task Allocation Strategy for Soft-deadline Tasks under Dynamic Allocation

Under dynamic allocation, the time between the start of the tasks execution and the largest deadlines is divided into activation periods, as illustrated in Section 5.2.2.2, Figure 5.14. In the case of hard deadlines, the robots are stopping the work on the task as soon as it becomes inactive. i.e. either when the required size is accomplished on the task or when its deadline is reached. However, when the deadlines are soft, the robots will consider the task as active until the planned amount of work is accomplished on the task.
The expected value of the swarm performance whose progress is modeled as a Poisson process, under dynamic allocation, is given as in Chapter 4, Section 4.5.3, Equation (4.50). The expected variance in the swarm performance $\mathbb{E}(\psi_i)$, which is defined as the difference between the size of the task and the expected amount of work to achieve at its deadline, is given by Equation (4.51):

$$\mathbb{E}(\psi_i) = \begin{cases} S_i - \lambda_i \ D_i & \text{for } \lambda_i \ D_i < S_i \\ 0 & \text{for } \lambda_i \ D_i \geqslant S_i \end{cases}$$

where S_i is the size of task T_i, D_i is its deadline, and λ_i is the rate associated with the Poisson process that models the progress of the swarm performance on task T_i. Based on the definition of the activation periods and on Equation (5.14) in Section 5.2.2.3, the rate λ_i of the progress of the swarm performance on task T_i is calculated as in the following:

$$\lambda_i \ D_i = \sum_{k=1}^{i} \lambda_{ik} \eta_k \qquad \forall i \in \{1, \ldots, M\}$$

By substituting it in Equation (4.51) for calculating the expected value of the swarm performance variance, we have:

$$\mathbb{E}(\psi_i) = \begin{cases} S_i - \sum_{k=1}^{i} \lambda_{ik} \eta_k & \text{for } \sum_{k=1}^{i} \lambda_{ik} \eta_k < S_i \\ 0 & \text{for } \sum_{k=1}^{i} \lambda_{ik} \eta_k \geqslant S_i \end{cases}$$

The rate λ_{ij} of the Poisson process during the j-th activation period of task T_i, can be written in terms of the transition probabilities based on Equation (5.15), Section

5.2.2.3:

$$\lambda_{ij} = \frac{\pi_{ij}}{\sum_{c=j}^{M} \pi_{cj}\hat{\mu}_c} N \qquad \forall i \in \{1,\ldots,M\}, \forall j \in \{1,\ldots,M\}$$

So, the expected value of the swam performance variance can be written in terms of the transition probabilities, as in the following:

$$\mathbb{E}(\psi_i) = \begin{cases} S_i - \sum_{k=1}^{i} \frac{\pi_{ij}}{\sum_{c=j}^{M} \pi_{cj}\hat{\mu}_c} N\eta_k & \text{for } \sum_{k=1}^{i} \frac{\pi_{ij}}{\sum_{c=j}^{M} \pi_{cj}\hat{\mu}_c} N\eta_k < S_i \\ 0 & \text{for } \sum_{k=1}^{i} \frac{\pi_{ij}}{\sum_{c=j}^{M} \pi_{cj}\hat{\mu}_c} N\eta_k \geqslant S_i \end{cases}$$

where $i \in \{1,\ldots,M\}$.

The costs associated with the potential part of the task which will be unprocessed at the task deadline, are generally higher for tasks with tighter time constraints. The allocation strategy aims to design a decision matrix which minimizes the costs associated with missing the deadlines of all the tasks based on their priorities. This is achieved by minimizing the expected value of the swarm performance variance, given above. This represents a multi-objective optimization problem with M objective functions. Each objective function is associated with the cost of one of the M considered tasks.

$$\underset{\pi_{ij}}{\text{minimize}} \quad \begin{cases} \mathbb{E}(\psi_1) = f(\pi_{11}) \\ \mathbb{E}(\psi_2) = f(\pi_{21}, \pi_{22}) \\ \vdots \\ \mathbb{E}(\psi_M) = f(\pi_{M1}, \pi_{M2}, \ldots, \pi_{MM}) \end{cases}$$

The above multi-objective optimization is solved *off-line*. It is solved through building a single optimization problem by constructing the cumulative weighted objective function. We sum the M objective functions, where each is weighted by the priority of its task as in the following:

$$\underset{\pi_{ij}}{\text{minimize}} \quad \sum_{i=1}^{M} \beta_i \mathbb{E}(\psi_i)$$

After obtaining the above output, the robots can start to assign themselves autonomously to the tasks and switch among them each time they finish executing

an individual part of their current task. The initial robots' assignment to the executable tasks and the later decision of their next task is performed by each robot, independently, using the decision matrix (one matrix per activation period). Figure 5.23 illustrates the steps of the strategy, with the inputs marked in red and the outputs marked in blue. The calculated steps are performed off-line and the necessary part of the output to use by the robots is marked with blue arrows.

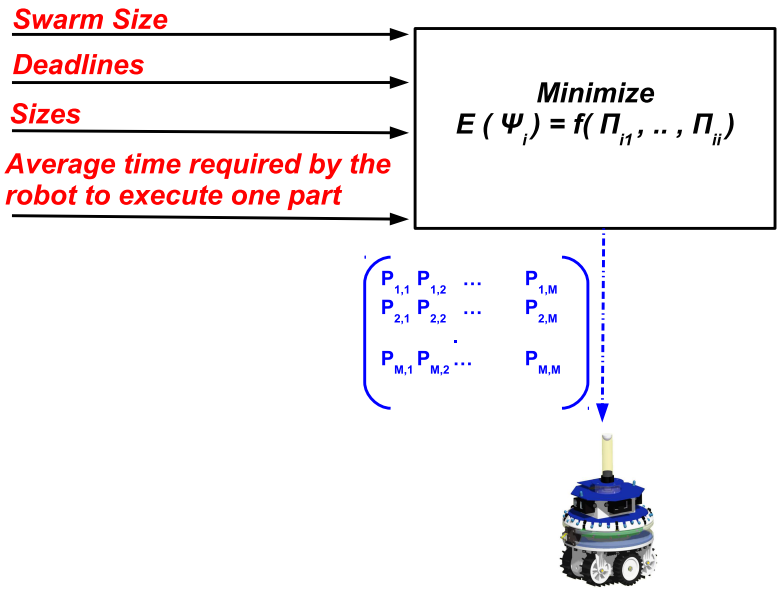


Figure 5.23 – *The task allocation strategy for soft deadlines under dynamic allocation.*

The following is the pseudo-code of the allocation strategy developed for soft-deadline tasks under dynamic allocation:

Algorithm 5 Strategy developed for soft-deadline tasks under dynamic allocation.

Input: S_i, D_i ,N, $\hat{\mu}_i = f(N_i)$
Output: solution $P_{M,M}$

1: $M \leftarrow$ tasks
2: $\beta_i = \dfrac{S_i/D_i}{\sum_{j=1}^{M} S_j/D_j}$ ▷ % calculate the task priorities%
3: $\eta_i = D_i - D_{i-1} \quad \forall i \in \{2, \ldots, M\}$ ▷ % task activation periods%
4: **for** $i = 1$ to M **do**
5: $\quad \mathbb{E}(\psi_i) = max(0, S_i - \dfrac{\pi_{ij}}{\sum_{c=j}^{M} \pi_{cj}\mu_c} N\eta_j)$
6: **end for**
7: minimize $\sum_{i=1}^{M} \beta_i \mathbb{E}(\psi_i)$

5.3.2.1 Scenario and Evaluation

Let us consider an example of three tasks which are characterized with their soft deadlines $\{500, 1000, 1500\}$ time units and their sizes $\{1500, 1000, 500\}$ parts. A homogeneous swarms with different sizes is used to execute these tasks. The swarm size varies over the range $[10 - 100]$ robots, with an increment step of 5 robots. Figure 5.24 shows the tasks priorities calculated using Equation (5.1).

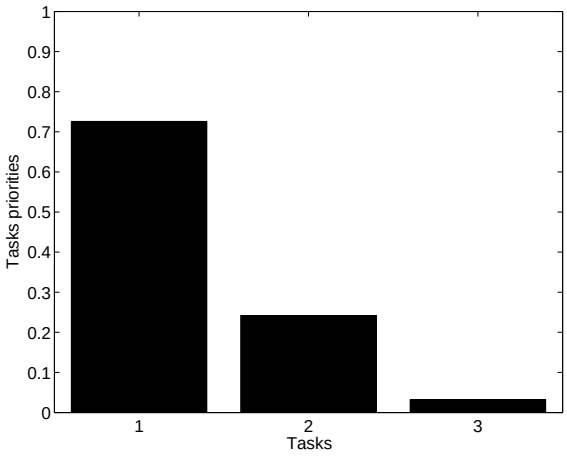


Figure 5.24 – *The priorities of the* 3 *tasks.*

Figure 5.25 shows the expected amount of work (parts) left unprocessed on each task

at its deadline, calculated as explained in Chapter 4, Section 4.5.3. In addition, in the same figure the performance variance is averaged over 100 runs of Monte-Carlo simulation.

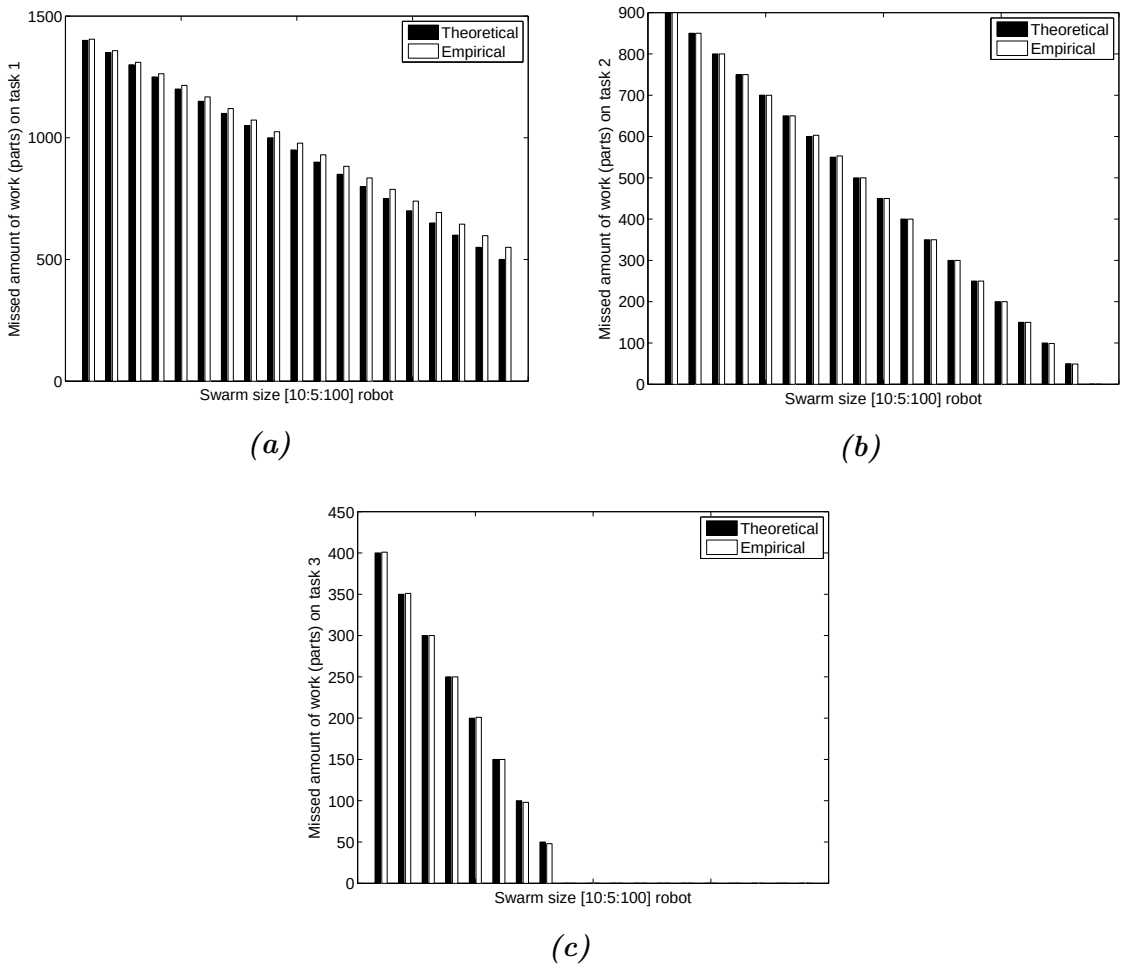


Figure 5.25 – *Evaluation of the task allocation strategy developed for the soft deadlines under the technique of dynamic allocation. The figure shows the expected difference between the amount of work accomplished (in parts) and the task size. This difference is calculated and simulated over* 100 *runs of Monte-Carlo simulation.*

As we can notice, both calculated and simulated swarm performance variances decrease with increasing swarm size. In Figure 5.25 (a), we see how the performance variance on task T_1 decreases slower as the task size is larger than the sizes of tasks T_2 and T_3. The same remark applies for task T_2 in Figure 5.25 (b) in comparison to

task T_3 in Figure 5.25 (c).

5.4 Conclusions

This chapter was dedicated to present the task allocation strategies developed for swarm robotics systems to execute a set of time-constrained tasks.
The deadlines were categorized into: Hard deadlines where continuing to execute the task after its deadline has no benefit and Soft deadlines where continuing to execute the task after its deadline is associated with specific costs. The allocation strategies were developed under the two techniques of static allocation and dynamic allocation, presented in Chapter 4. They were proposed to handle both kinds of scenarios, namely, when the switching costs among tasks are high or when they are negligible. For hard deadlines, the focus was on studying the possibility of the system to execute the task within its deadline. This possibility was defined in terms of the task execution probability, which is the probability of finishing the task within its deadline. For hard deadlines, the goal was to allow the robots to perform an independent allocation, that maximizes the number of executable tasks. The task is categorized as executable when its execution probability is acceptable (near to one). On the other hand, for soft-deadline tasks, the focus was on studying quantitively the difference between the required swarm performance and the expected one to achieve at the task deadline, this difference represents the cost associated with missing the task deadline. The goal of the allocation strategies under both static and dynamic allocation was to design a self-organized allocation that minimizes the costs associated with missing the task deadlines.
All allocation strategies were designed so that the robots can allocate themselves statically or dynamically to the tasks independently and under the respect of the task deadlines.

Chapter 6

Applications: Bio-Inspired Foraging

> Swarming robots could have important roles to play in the future of micro-medicine, as 'nanobots' are developed for non-invasive treatment of humans. On a larger scale, they could play a part in military, or search and rescue operations, acting together in areas where it would be too dangerous or impractical for humans to go. In industry too, robot swarms could be put to use, improving manufacturing processes and workplace safety.
>
> - Roderich Gross

In this chapter, we present a biologically inspired task which is well-studied in swarm robotics systems, namely, foraging. We consider a special case of this task referred to as multi-foraging*, where different types of items should be retrieved by robots back to the home area. In our multi-foraging scenario, each type of the considered items represents a specific task where a specific number of parts should be retrieved within a specified deadline. Hence, the multi-foraging tasks represent a set of time-constraint*

tasks.

In this chapter, we aim to verify the task allocation strategies developed in Chapter 5 for hard and soft deadlines under both static and dynamic allocations through a set of physics-based simulations.

6.1 Foraging

Foraging is a task which can be observed in a large number of social insect colonies and animal societies. It plays a main role in the survival and the reproduction of the group members (Danchin *et al.*, 2008). Foraging in its simple form consists of a set of sub-tasks including the exploration of the environment of interest to search for food items and the transportation of the food items found in addition to navigating with the food items back to some safe area referred to as the nest or the home.
Foraging can be categorized as solitary foraging where the individuals perform the searching, finding, capturing and consuming the prey alone (Reidman, 1990). This biological foraging does not represent an interesting source of inspiration by systems such as swarm robotics which are characterized by their large number of robots. The other kind of biological foraging is known as the group foraging (Stephens *et al.*, 2007). It is a cooperative task whose success depends on the collective behavior of the individuals which are performing the foraging for the benefit of all.
Foraging was inspired from nature and studied intensively in the context of swarm robotics systems under different concepts, including:

- A complete robot-foraging study: starting from the solitary foraging with one robot and scaling to the group foraging with multi-robot systems such as swarm robotics. A full taxonomy is provided (Winfield, 2009).

- Mathematical analysis: A few mathematical studies were conducted in the domain of swarm robotics. Some of them were focusing on foraging as a verification task (Lerman & Galstyan, 2002; Liu *et al.*, 2009).

- Spatial interferences influence on foraging of large scale multi-robot systems: spatial interference among robots play a significant role on the performance of single robot and swarm. Some studies have focused on this concept for foraging tasks as in Østergaard *et al.* (2001); Shell & Mataric (2006).

- Learning: explore different learning methodologies such as the reward functions for improving the foraging task in multi-robot systems (Balch, 1999b), or considering the learning in challenging noisy and dynamic environments for multi-robot foraging tasks (Matarić, 1997)

Foraging can be mapped to a large spectrum of real applications where swarm robotics provide a promising solution. Some examples of these applications are listed below:

- Data mules: this task is a relatively new approach applied in large-scale sensors or spare networks for collecting nodes' storage and transferring it to some base station. This technique was first proposed in the context of wireless sensor networks by Shah *et al.* (2003). It aims mainly to extend the life-time of the sensor nodes by preserving their energy through applying the data transfer using an one-hop approach between the mobile node and the sensor node instead of the multi-hop routing to transfer the data to the base station (Jain *et al.*, 2006; Jea *et al.*, 2005). Some works focus on even determining the movements of the mobile node and the times it sojourns at certain network nodes so that network lifetime is maximized (Wang *et al.*, 2005). Other works, as in Somasundara *et al.* (2004), concentrate on providing a schedule for the visits to the sensor nodes based on the sizes of their buffers and consequently on the length of the time periods through which they can keep the data before losing it.

  Let us consider a large-scale network where a swarm of robots is employed in collecting and transferring data to the base station. If we divide the network in areas based on the time intervals during which the nodes can hold the measured data before to lose it, then each area can be characterized by its specific deadline.

- A recycling system in a factory: Let us consider a factory which consists of several sections and each section produces periodically a particular amount of items for recycling. The recycling task can be executed using a swarm of robots which travel among the different sections to retrieve the materials for recycling. In case of, the different sections of the factory are generating the materials to recycle at different rates, each section will represent a task with its deadline. Otherwise, all the tasks will share a common deadline.

- Pollution cleaning or de-mining: Let us consider the scenario of having different pollution spots, which needs to be cleaned within a specific deadline or a field

of mines that needs to be removed (deactivated) also within a specific deadline. Being generally considered as dangerous for humans, they can be performed using a swarm of robots. Such tasks can be mapped to biological foraging, where the robots need to explore the operation area searching for particular items to remove them (retrieving stage is not included, in this case). These represent, normally, time-constrained tasks, based on the criticality of the time within which these items should be removed.

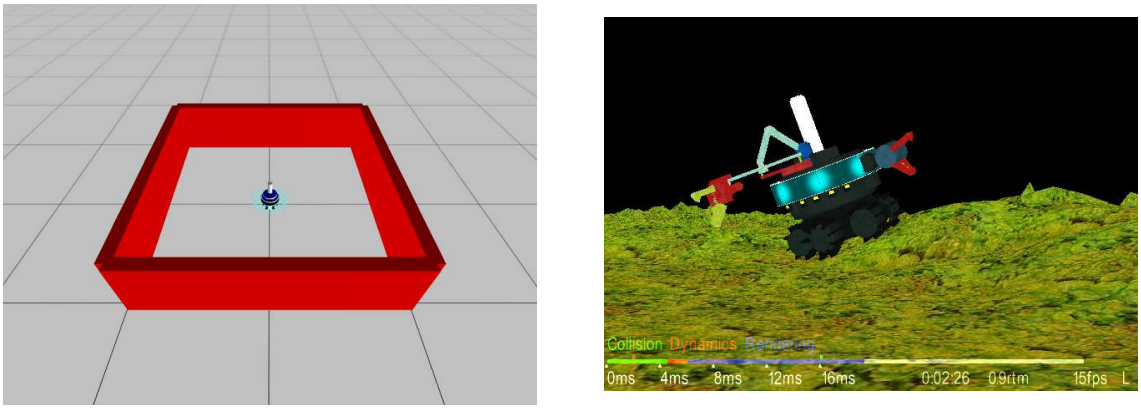
There are other real-world applications whose execution can be mapped to the biological foraging behavior, where swarm robotics systems provide an appropriate solution for them.
The task allocation developed in this thesis is a non-communicative approach. Therefore, no communications is applied among the robots while performing the foraging tasks presented in this chapter. Consequently, no optimization concerning the announcement of the richest food source or the shortest path to the food source, is practiced. Such kind of optimization plays no role in the performance of our system as we assume a *uniform* distribution of the task parts over the arena.

6.2 Simulations

6.2.1 The ARGoS Simulator

ARGoS is a state-of-the-art, open source $3D$ robot simulator. Its design allows the simulation of large heterogeneous swarms of robots (Pinciroli *et al.*, 2012). ARGoS was developed initially within the context of the *Swarmanoid* project. It is currently the main robot simulation tool used in the following European projects: ASCENS[1], H2SWARM[2], E-SWARM[3] and Swarmix[4]. In addition, it is used by several universities and research laboratories across the world. Figure 6.1 shows two snapshots from simulations by ARGoS. It is considered as the first multi-robot simulator that offers both flexibility and efficiency. Flexibility refers to the ability of simulating

[1] http://ascens-ist.eu/
[2] http://www.esf.org/
[3] http://www.e-swarm.org/
[4] http://www.swarmix.org/



(a) *(b)*

Figure 6.1 – *Snapshots taken from the multi-robot simulator* ARGoS.

different kinds of experiments and to add new features. Efficiency refers to the ability of running experiments with thousands of robots in the shortest time possible. These two features, namely the flexibility and the efficiency are at odds. ARGoS has solved this trade-off by introducing a set of novel techniques. One of the main design features provided by ARGoS and which plays a significant role in flexibility, is *modularity*. Modularity allows the user to modify any aspect in the simulation or even add new features: such as robots, sensors, actuators and others. Another feature which is behind the efficiency provided by ARGoS, is *space partitioning*. Space partitioning is considered as the most distinctive feature of ARGoS. It represents the possibility of partitioning the simulated $3D$ space into non-overlapping portions of arbitrary size and managing those portions separately in parallel. This feature plays the main role in producing short time simulations. In addition, ARGoS architecture is designed to exploit modern CPU architectures by using *multiple threads*, which in turn maximizes the simulation speed.

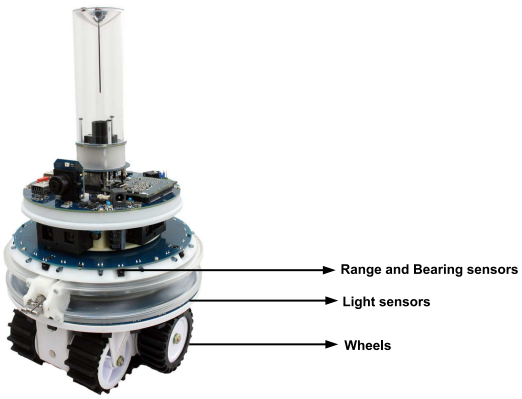
The robot's control code is written using *C++*. In addition, *XML* files are used to define and configure the different modules of the environment which is referred to generally as the world. A set of robots' models are already supported by ARGoS including: the epuck, the Foot-bot[1], the Eye-bot[1] and the Hand-bot[1]. However, the user can model any other required robot. Additionally, a large set of sensors and

[1] www.swarmanoid.org/swarmanoid_hardware.php

actuators are available in ARGoS, again with the ability of modeling any new one required. ARGoS can be used mainly on the operating systems *Linux* and *MacOS* and has been tested on several of their versions.
The simulator defines multiple kinds of physics engines to perform the calculations of the physical kinematics in addition to the collisions happening among the physical entities in the simulation arena. The presence of multiple physics engines allows to model the different robots and their physical interaction according to the required level of precision and to available computational resources. A single simulation can include one or more physics engines. Each subspace managed by a specific physics engine, can be a $1D$, $2D$, or $3D$ subspace. All simulated entities have access to the full $3D$ representation of the environment through a special geometrical transformation system. This allows to communicate with and sense other robots and entities in the environment.
Using multiple physics engines is a unique feature of ARGoS that allows selection among different levels of physical precision. On the other hand, it allows to optimize the usage of the computational resources. All physical interactions on the ground can be managed by a physics engine based on a $2D$ geometry that considers only the simulation of the kinematics.

6.2.2 Setup Foraging Experiment

We use homogeneous swarms to perform our foraging tasks. The robot used in our foraging experiments is the *Foot-bot* robot, see Figure 6.2.



(a) Foot-bot.

(b) Foot-bots using the color display to communicate.

Figure 6.2 *– Foot-bot robots re-printed from the* Swarmanoid project *web-page.*

Foot-bot is a wheeled robot which can be used only on the ground. It is 17 cm diameter × 29 cm hight and weights 1.8 kg. It possesses a powerful *treel* mobility system which is composed of wheels and tracks. A lithium polymer battery which is used by foot-bot provides it with the ability to participate in long-period experiments (up to 2.5 hours of continuous use without recharging). In addition, the feature of hot-swapping available in foot-bot(s), where a capacitor keeps the robot alive while the battery is swapped, allows theoretically for unlimited long experiments.
The foot-bot is equipped with a set of sensors and actuators. Its sensor set includes two 2.0 mega-pixels cameras with image on-board pre-processing. The first camera provides the foot-bot with an omni-directional vision while the second one is used for the eye-bot vision as it looks towards the ceiling. Obstacle avoidance is performed by the Foot-bot using a set of 24 short range (up to 10 cm) proximity sensors, in addition to long-range (up to 150 cm) infra-red scanner. A foot-bot is quipped with many other sensors. Robots are able to grip objects and to connect themselves to other robots using their grippers. Other actuators are laser beam actuator, rotor blades actuator, and the robot's wheels.
The foot-bot robot is supported by an on-board CPU with a high computational power including floating-point processing. It can self-assemble and can communicate

with other robots in several ways including a color display which is one of the most important communication methods.
In the considered foraging tasks, the working arena is divided into *items-arena*, where large numbers of parts belong to different types of items are uniformly scattered. In addition, the working arena includes a *nest (home)*, where the robots should retrieve the harvested parts. The nest is generally marked with a set of lights to be sensed by the working robots. Each time a part is retrieved another part is generated.
The foraging behavior we consider throughout this chapter can be described by the following stages: the robots are, initially, in a *resting* state at a special area referred to as the *robot-repository*. As soon as the experiment starts, the robots move to the *exploring* state. In the exploring state, the robots use the *diffusion* behavior to explore the items-arena while applying obstacle avoidance. The diffusion behavior offers the maximum coverage of the items-arena while avoiding other robots and obstacles. The diffusion behavior is governed by a *light-repulsion* behavior, in which the robot moves away from the light of the nest towards the items-arena. As soon as the robot finds some part of the item it is currently working on, it picks that part and switches to the *return to the nest* state. In the return to the nest, it starts to apply a combined behavior of *light attraction* and obstacle avoidance heading back to the nest. When the robot arrives to the nest, it tries to drop the harvested part before it switches directly back to the exploring state.

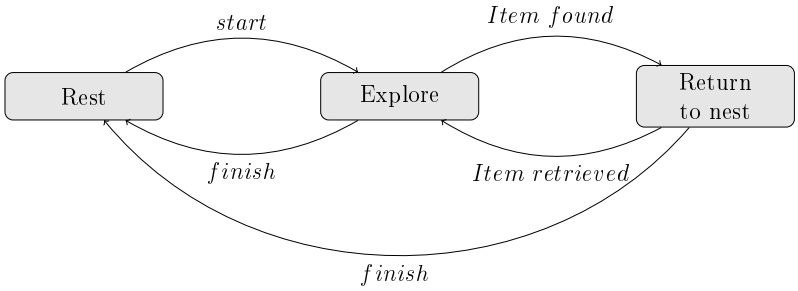


Figure 6.3 *– Robot's controller.*

6.3 Static Allocation

Based on the definition of static allocation in Chapter 4, Section 4.4, robots are assigned to the tasks at the beginning of the execution and no switching among the tasks, is performed later. This kind of allocation is applied when the switching costs among the tasks are high.

For static allocation, we consider three foraging tasks where the parts to retrieve are scattered over three separated arenas. Each arena holds one type of items and consists of two sub-areas: One 9×4 meters item-arena where the item parts are scattered and One 3×4 meters nest-arena to where the item parts should be retrieved. The robots are located first at a 3×14 meters robots-repository, see figure 6.4. As soon as the robots are assigned to the tasks, they start to move towards their working arenas. The arenas are separated from each other and the switching costs are high for the robots to travel from one to another arena.

The mean time required to retrieve one part is the same for the three items, as the same number of parts of each item is scattered uniformly at its item-arena and the three item-arenas have the same area. This mean is measured over short-term simulations using the following swarm sizes $\{10, 15, 20, 25, 30, 35, 40\}$. The results are averaged over 25 simulation runs. After that, a polynomial estimation of the degree 3 is performed to obtain the performance function. Figure 6.5(a) shows the average time required to retrieve one part of each of the items, when the swarm size varies. In this figure we see how the average time required to retrieve one part by a swarm of robots decreases by increasing the size of the swarm as a result of parallelizing the work on the task. However, as soon as the effect of the spatial interferences among the robots starts to overcome the benefit gained by parallelizing the work, the average time of retrieving a single part starts to increase again. Figure 6.5(b) shows the average time required to retrieve one part by a single robot. This time increases continuously by increasing the size of the swarm, influenced by spatial interferences.

The robots' activity times are sampled, independently, from the exponential distribution and the robots batteries are assumed to be full recharged at the beginning of the execution.

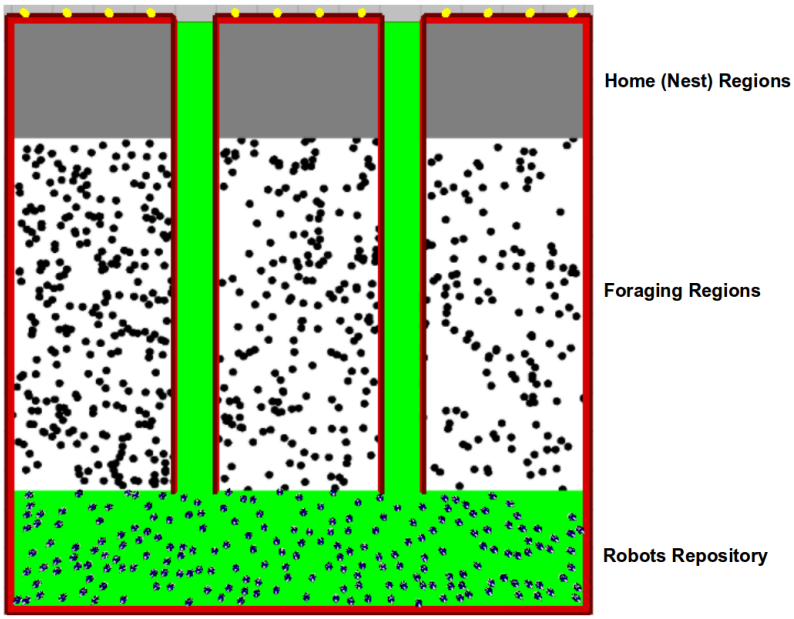
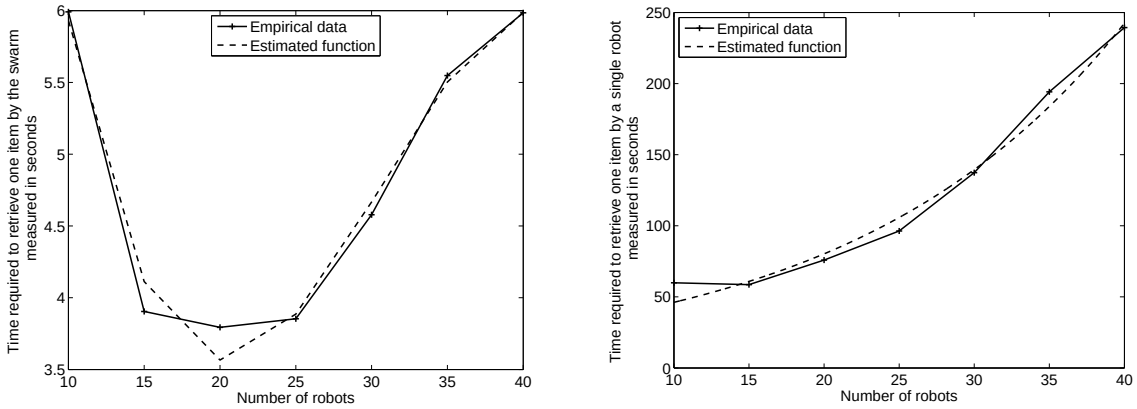


Figure 6.4 – *Foraging under static allocation.*



(a) *Swarm performance.* ***(b)*** *Single robot performance.*

Figure 6.5 – *Swarm and single robot performance under spatial interferences.*

6.3.1 Hard-deadline Foraging under Static Allocation

In this section, the deadlines of the three tasks are considered to be hard. Hence, missing the deadline of the task is critical. The deadlines are $\{720, 1080, 1800\}$ seconds and the foot-bot swarm used to execute the tasks consists of 50 robots. The probability of interest, in this study, is per task the probability of finish executing the task within its deadline, it is referred to as the task execution probability. The following sizes of the three tasks are examined:

$$\begin{array}{lccc} \text{Sizes} & T_1 & T_2 & T_3 \\ S1: & 30 & 30 & 30 \\ S2: & 200 & 30 & 30 \\ S3: & 30 & 400 & 30 \\ S4: & 30 & 30 & 500 \\ S5: & 300 & 100 & 100 \\ S6: & 100 & 400 & 100 \\ S7: & 100 & 100 & 500 \end{array}$$

where each line represents a variant of the task sizes.
First, we select the same size for the three tasks. So, the priority of the task depends only on the tightness of its deadline. This is what we see in the first column in Figure 6.6. The task with the highest priority is T_1 which has the shortest deadline, then task T_2 and the lowest priority is the one of task T_3. In the size variants $\{S_2, S_3, S_4\}$, we assign a larger size to the tasks T_1, T_2 and T_3 sequentially while keeping the size of the other two tasks equal to 30. In this step, the priority of the tasks is influenced not only by their deadlines but also by their sizes. This is why the priority of task T_1 is the highest in S_2 and in S_5, Figure 6.6. Whereas, the priority of task T_2 becomes the highest in S_3 and in S_6 and the priority of task T_3 is the highest in S_4 and in S_7.

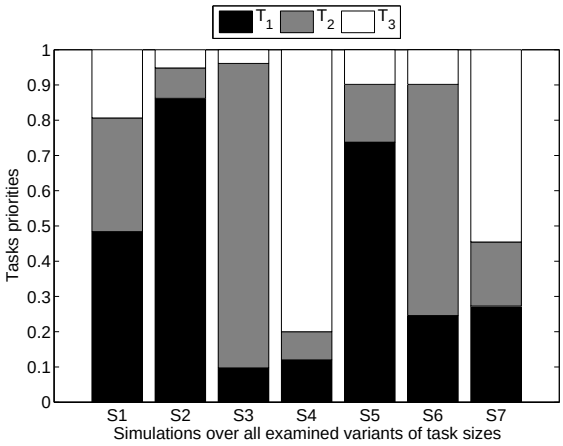
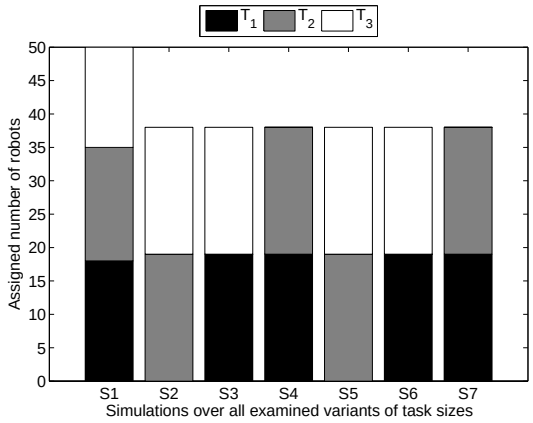
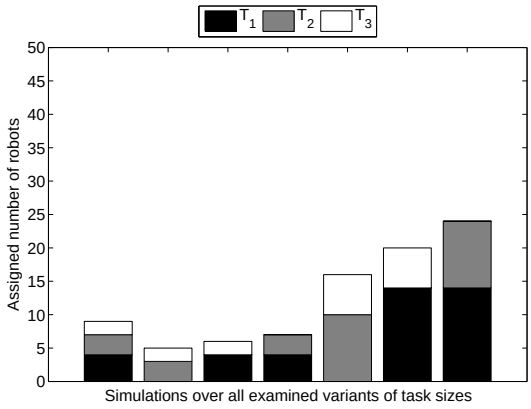


Figure 6.6 – *Tasks priorities calculated based on Equation* (5.1) *for the different variants of task sizes.*

In Chapter 5, two allocation strategies were developed for tasks with hard deadlines under static allocation: the Energy-aware strategy and the robots-aware strategy. The energy-aware strategy is developed to cope with the limited battery life of the robots. It cares, in addition to execute the task within its deadline, to speed up its execution. Therefore, it maximizes the number of robots assigned to the tasks. On the other hand, the robots-aware strategy attempts to preserve the robotic resources by assigning the necessary number of robots to each task under the consideration of its temporal constraint. Both allocation strategies assign no robots to the task when no acceptable (near to one) execution probability is possible to be obtained for that task. Figure 6.7 shows the number of robots assigned to each of the three tasks over all the examined variants of the task sizes under both energy-aware and robots-aware strategies. This figure shows how the number of robots assigned to the different tasks is smaller when the robots-aware strategy is applied.



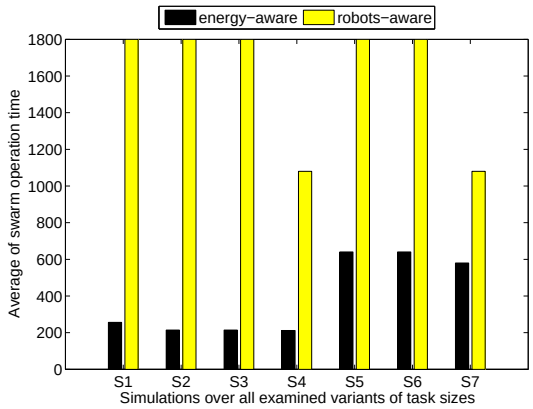
(a) Under the energy-aware strategy.



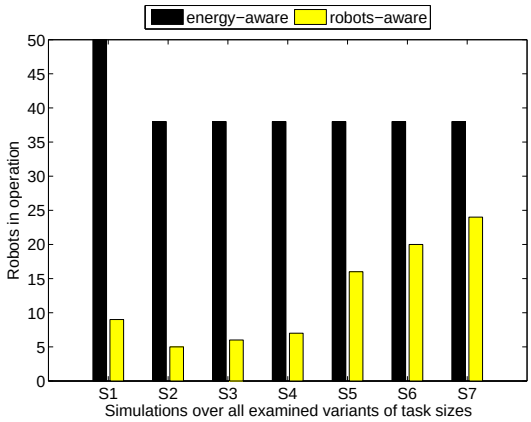
(b) Under the robots-aware strategy.

Figure 6.7 *– The number of robots assigned to each of the three tasks.*

Figure 6.8 (a) shows how the energy-aware strategy outperforms the robots-aware strategy, concerning the average time required to finish the complete set of executable tasks and to have the whole swarm again available. In addition, Figure 6.8 (b) shows how the robots-aware strategy outperforms the energy-aware one, concerning the number of free robots available continuously and ready for any new demand.



(a) The average time to finish the 3 *tasks.*

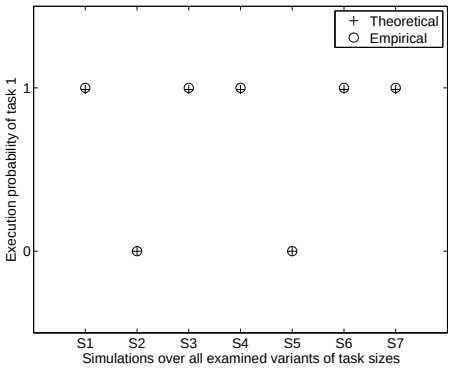


(b) The number of robots in operation.

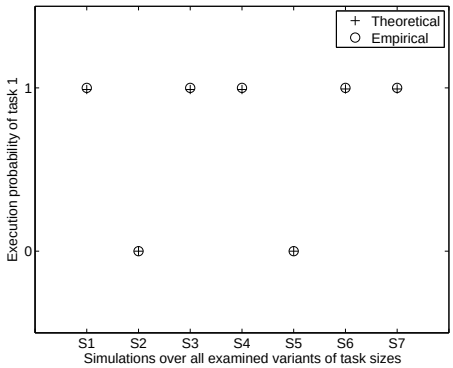
Figure 6.8 *– A comparison between energy-aware and robots-aware allocation strategies.*

Figure 6.9 illustrates the execution probability (the probability to finish the task within its deadline) of the three tasks over the different variants of the task sizes.

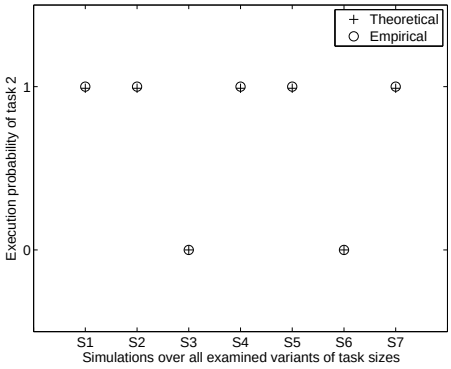
It depicts the probabilities calculated as explained in Chapter 5 Sections 5.2.1.1 and 5.2.1.2. In the same figure, the empirical probabilities obtained over repeated ARGoS simulations (10 runs for each variant of the task sizes) are also shown. Figures 6.9(a,c,e) show the execution probability of each of the three tasks, when the energy-aware strategy is followed, while Figures 6.9(b,d,f) show the same execution probabilities, however when the robots-aware strategy is applied.



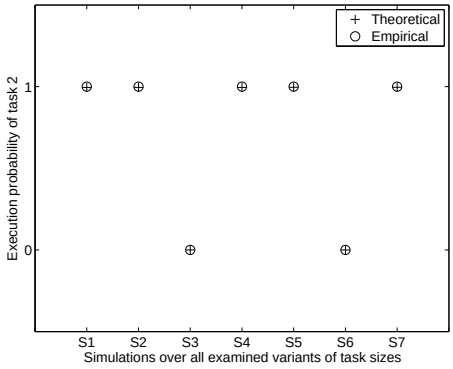
(a) *Execution probability of T_1 over all examined task sizes under energy-aware.*



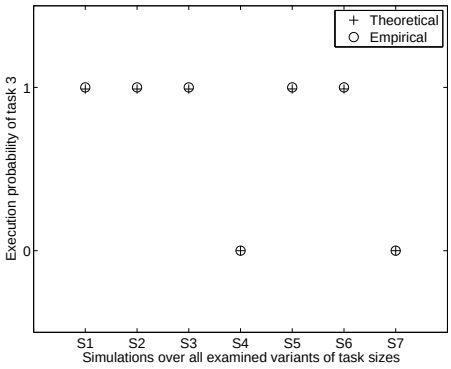
(b) *Execution probability of T_1 over all examined task sizes under robots-aware.*



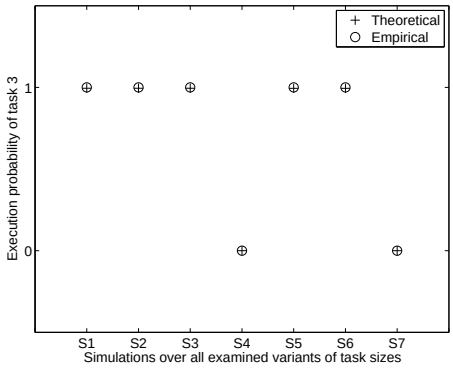
(c) *Execution probability of T_2 over all examined task sizes under energy-aware.*



(d) *Execution probability of T_2 over all examined task sizes under robots-aware.*



(e) *Execution probability of T_3 over all examined task sizes under energy-aware.*

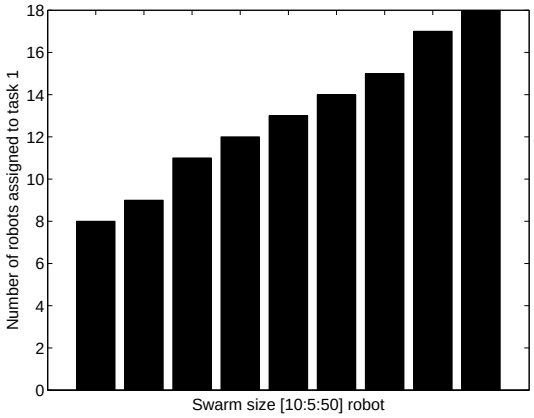


(f) *Execution probability of T_3 over all examined task sizes under robots-aware.*

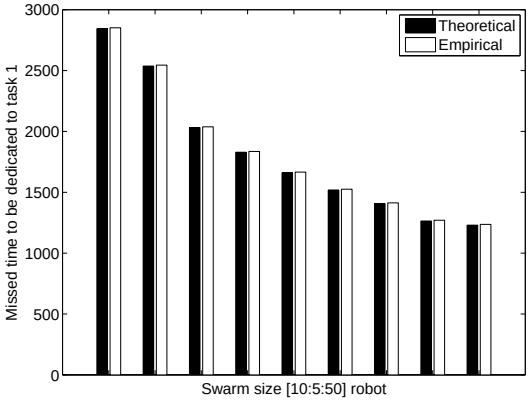
Figure 6.9 *– The theoretical and empirical execution probabilities of the three tasks under energy-aware and robots-aware strategies.*

6.3.2 Soft-deadline Foraging under Static Allocation

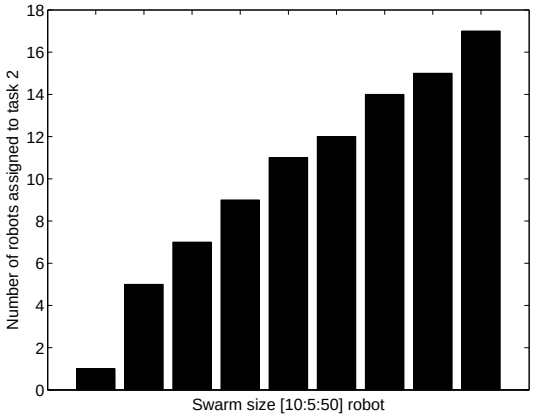
In this section, the three tasks are characterized by the following soft deadlines $\{720, 1080, 1800\}$ seconds. Missing any of the soft deadlines is associated with a specific cost, however, it is not considered to be catastrophic. The sizes of the tasks are set to be equal, where 500 part has to be retrieved from each item within its specified deadline. The expected value of the performance variance is defined, in Chapter 5 Section 5.3.1, as the difference between the total time that has to be dedicated to the task within its deadline and the expected time to be dedicated to the task at its deadline. The allocation technique is designed to minimize the costs associated with the time which was not dedicated the task within its deadline.
A homogeneous swarm of an increasing size over the range $[10 - 50]$ robots with an increment step of 5 robots, is used to execute the set of tasks. The expected performance variance is calculated first using Equation (4.37) in Section 4.4.3, Chapter 4. After that, the performance variance is averaged over repeated ARGoS simulations (10 runs for each of the examined swarm sizes), see Figure 6.10.
Figures 6.10 (a-c-e) show how the number of robots assigned to the different tasks increases by increasing the size of the swarm according to the task priorities. The number of assigned robots is upper-bounded with the optimal number of robots N_{opt} associated with the maximum swarm performance under spatial interferences, here $N_{opt} = 20$ robots for all tasks, see Figure 6.5. Figures 6.10 (b-d-f), show the missed time to be dedicated to the three tasks each within its deadline. This was referred to as the expected value of the performance variance. It decreases with increasing the number of robots assigned to the tasks. The same figure shows a comparison between the expected value of the performance variance calculated using Equation (4.37) and the averaged value over 10 ARGoS simulation for each of the examined swarm size.



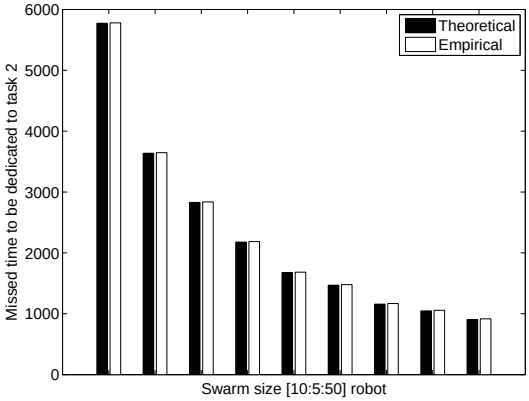
(a) Assigned robots on task T_1.



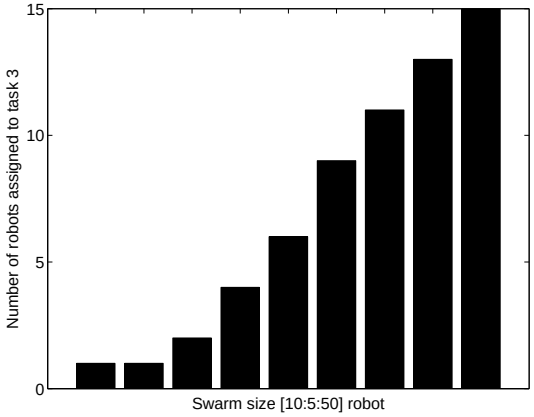
(b) Missed time to dedicate on task T_1.



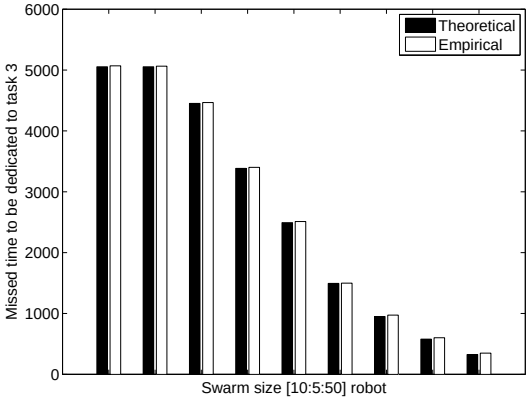
(c) Assigned robots on task T_2.



(d) Missed time to dedicate on task T_2.



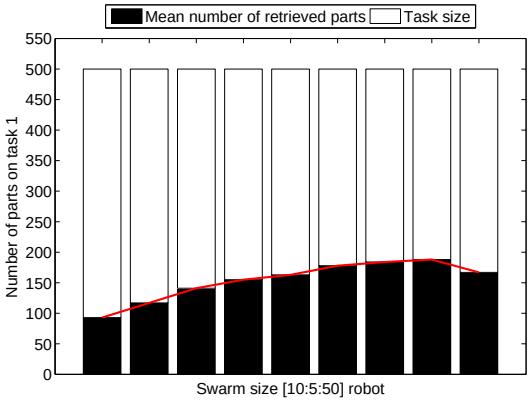
(e) Assigned robots on task T_3.



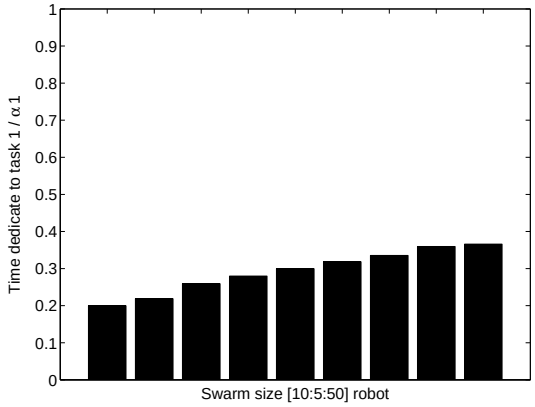
(f) Missed time to dedicate on task T_3.

Figure 6.10 *– Evaluation of the task allocation strategy developed for the soft-deadlines under the technique of static allocation. Figures (a), (c) and (e) show the number of robots assigned to each of the considered tasks by increasing the swarm size over the range* $[10:5:50]$*. Figures (b), (d) and (f) show the performance variance associated with the used size of the swarm. A comparison is performed between the calculated and the simulated performance variance over* 10 *simulations for each swarm size.*

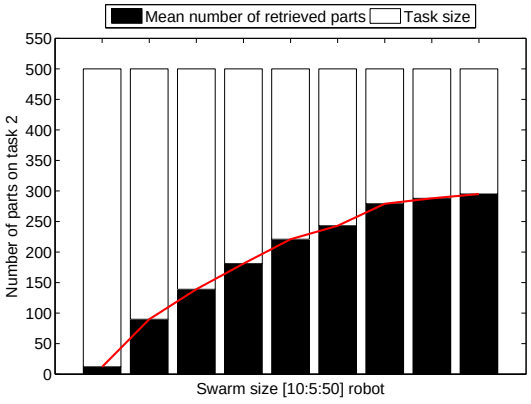
The number of retrieved parts on each of the considered tasks increases while increasing number of robots assigned to the task. Figures 6.11 (a-c-e) show how the number of retrieved parts increases towards the size required to be achieved on each of the tasks within its deadline (500 parts). Figures 6.11 (b-d-f) show the ratio of the total time dedicated to the task by the robots and the minimum threshold of time required to be dedicated to the task. By comparing the Figures 6.11 (a-c-e) with the Figures 6.11 (b-d-f), we can see that the ratios depicted in Figures 6.11 (b-d-f) can be used to approximate the mean of retrieved parts depicted in Figures 6.11 (a-c-e). This can be performed by multiplying the size of any of the three tasks (500 parts) with the corresponding ratio associated with the used swarm size. This rule gives a fair approximation only if the number of assigned robots to the task is large enough.



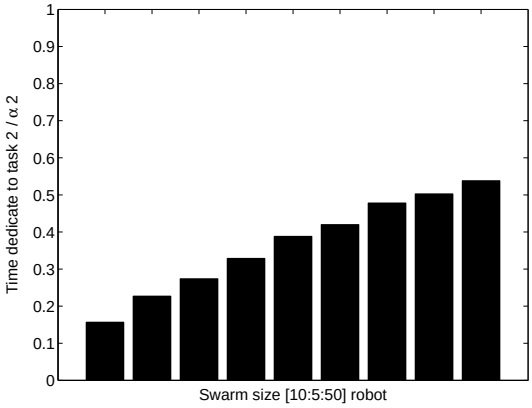
(a) Mean of parts retrieved on T_1.



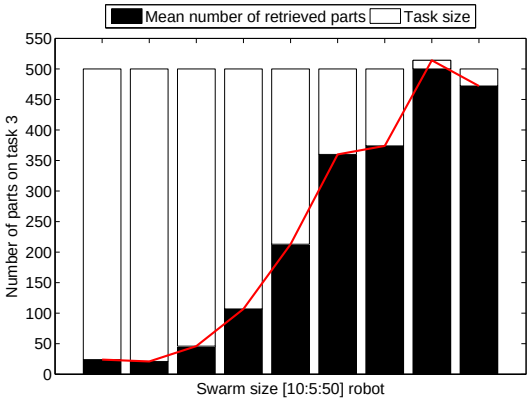
(b) Ratio: total dedicated time to T_1 / α_1.



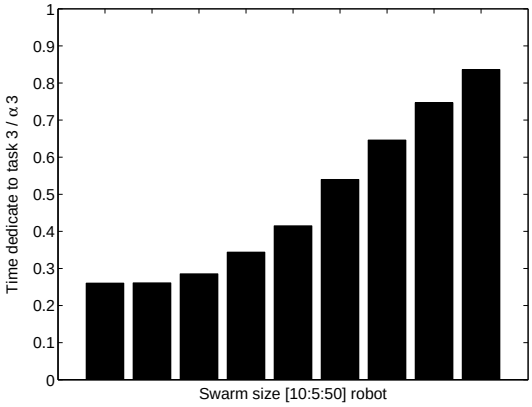
(c) Mean of parts retrieved on T_2.



(d) Ratio: total dedicated time to T_2 / α_2.



(e) Mean of parts retrieved on T_3.

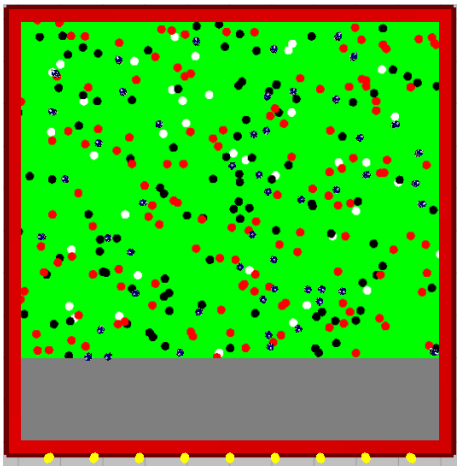


(f) Ratio: total dedicated time to T_3 / α_3.

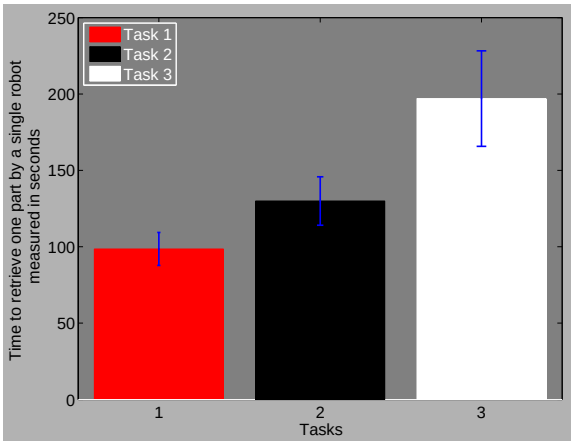
Figure 6.11 *– The number of retrieved parts on each of the tasks in Figures (a), (c) and (e). Figures (b), (d) and (f) show the ratio between the total time dedicated to the tasks and the minimum threshold of the time needs to be dedicated.*

6.4 Dynamic Allocation

Dynamic allocation is defined in Chapter 4, Section 4.5 as the allocation which is applied when the switching costs among tasks are negligible, therefore robots are allowed to change their tasks continuously during the execution time.
For dynamic allocation, we consider three foraging tasks where the parts to retrieve are scattered over the same arena. The arena consists of two sub-arenas: An item-arena of 7×10 meters, where three types of items are scattered and A nest-arena of 3×10 meters, to where the parts should be retrieved, see Figure 6.12 (a). Since the three types of items are located on the same arena, the switching costs among the tasks are negligible. A specific number of parts, referred to as the task size, should be retrieved on each of the three tasks within it deadline.
The mean time required to retrieve one part by a single robot, is measured for the three items (red, black and white) each over 25 runs of ARGoS simulation for the time period of 500 seconds and depicted in Figure 6.12 (b). The difference in the mean time between the three types is caused by the difference in the number of the parts scattered from each of the three items. The number of parts from the red item is 500 part, the number of parts from the black item is 300 part, and the number of parts from the white item is 150 part.
The estimation of the mean time required to retrieve a single part of each type is performed using a homogeneous swarm of 50 robots.



(a) Foraging under Dynamic Allocation.



(b) Mean time required by a single robot to retrieve one part of each item type.

Figure 6.12 *– Multi foraging under dynamic location technique.*

6.4.1 Hard-deadline Foraging under Dynamic Allocation

In this section, the deadlines of the three tasks are considered to be hard. Hence, missing the deadline is critical. The deadlines are $\{500, 1500, 3000\}$ seconds and the foot-bot swarm used to execute the tasks consists of 50 robots. The probability of interest, in this study, is per task the probability of finish executing the task within its deadline, it is referred to as the task execution probability. The following sizes of the three tasks are examined:

$$\begin{array}{lc} \text{Sizes} & \begin{array}{ccc} T_1 & T_2 & T_3 \end{array} \\ \begin{array}{l} S1: \\ S2: \\ S3: \\ S4: \\ S5: \end{array} & \left(\begin{array}{ccc} 10 & 100 & 150 \\ 30 & 150 & 300 \\ 50 & 200 & 400 \\ 70 & 250 & 500 \\ 100 & 300 & 600 \end{array}\right) \end{array}$$

where each line represents a variant of the task sizes.
Figure 6.13 shows the priorities of the three tasks over the examined variants of the task sizes.

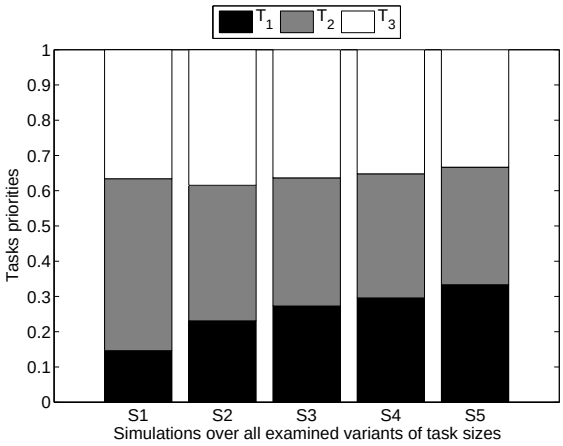
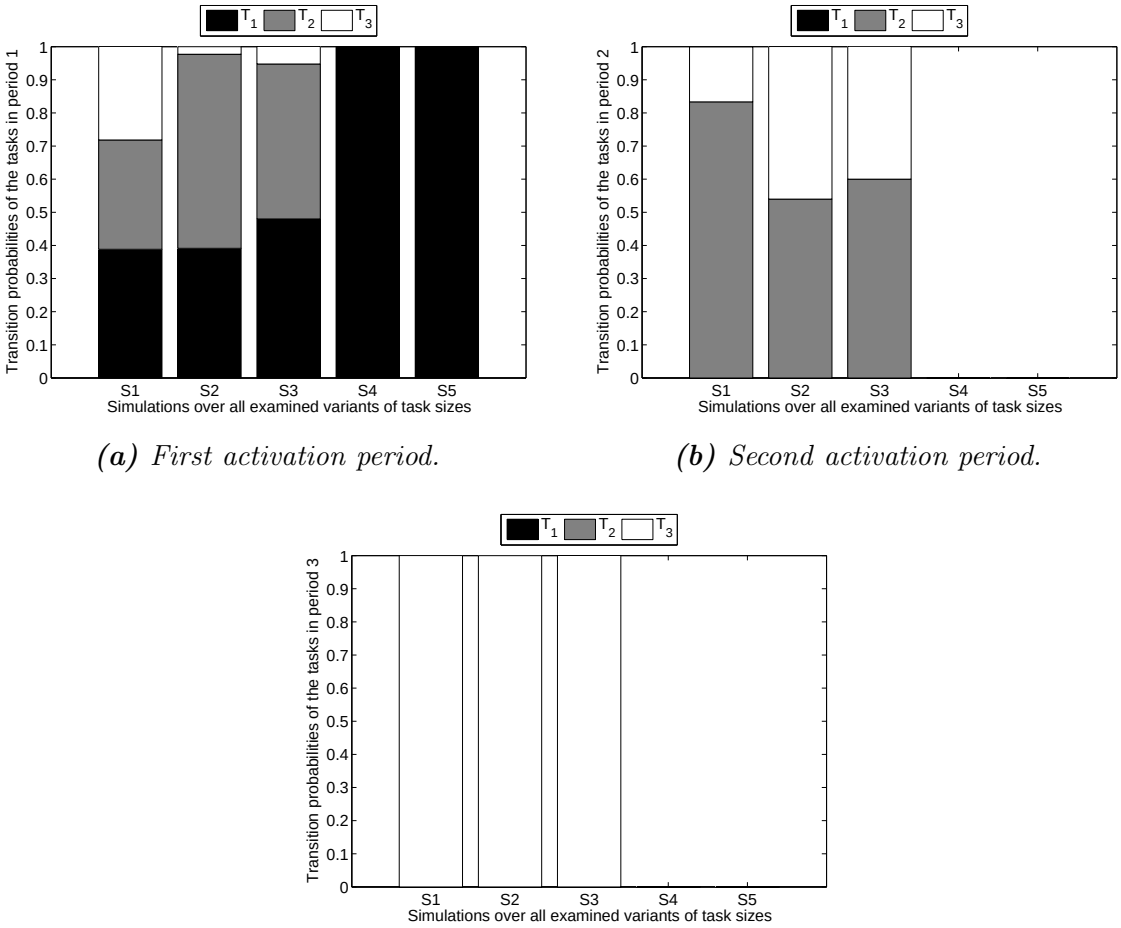


Figure 6.13 – *Tasks priorities calculated based on Equation (5.1) for the different variants of task sizes.*

In Chapter 5, Section 5.2.2, the allocation strategy developed for hard-deadline tasks under dynamic allocation defines first the set of executable tasks where tasks have an acceptable execution probability (near to one). After that, it finds the decision matrices which hold the transition probabilities and are used by the robots to assign themselves to the different tasks over the different activation periods. In our scenario, there are three activation periods defined using Equation (5.11) in Section 5.2.2.2, Chapter 5. During the first period, all tasks are active. During the second period, the first task which is associated with retrieving the red item parts becomes inactive and in the third period the second task which is associated with retrieving the black item parts becomes inactive. Figure 6.14 shows the decision matrices calculated as explained in Chapter 5 , Section 5.2.2. As we can see in the figure, during the first period, robots are assigned to the three tasks. In the second period, the first task with the shortest deadline becomes inactive, therefore, no robots are assigned to it. The same applies to the second task in the third period, where it becomes inactive. In addition, when the sizes of the tasks increase gradually, the whole swarm becomes required to work on the first task during the first period and no robot is left free to be assigned to the second or the third task. This case can be be observed when considering the variants S_4 and S_5 of the task sizes.



(a) First activation period.

(b) Second activation period.

(c) Third activation period.

Figure 6.14 *– The transition probabilities of the three tasks over their activation periods.*

After the decision matrices are calculated for the different activation periods, the number of robots is minimized on each of the executable tasks, under the constraint of keeping the execution probability within the acceptable limits. Figure 6.15 shows the number of robots requires to perform the set of executable tasks over the different variants of task sizes. In the figure, we can notice how the number of the required robots increases by increasing the task sizes up to the point where some of the tasks become un-executable, hence, less robots are required to execute the remaining tasks. This is reason why the number of required robots decreases after increasing.

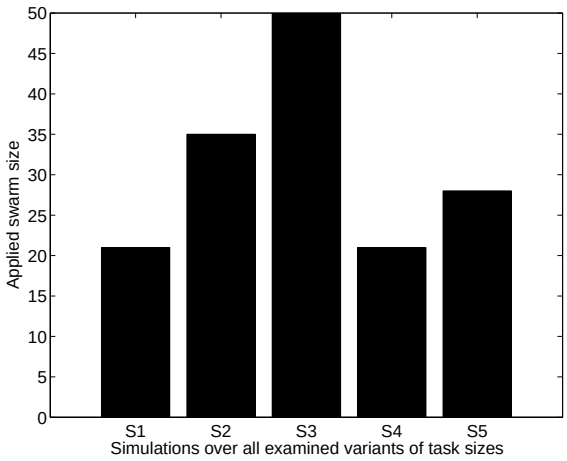
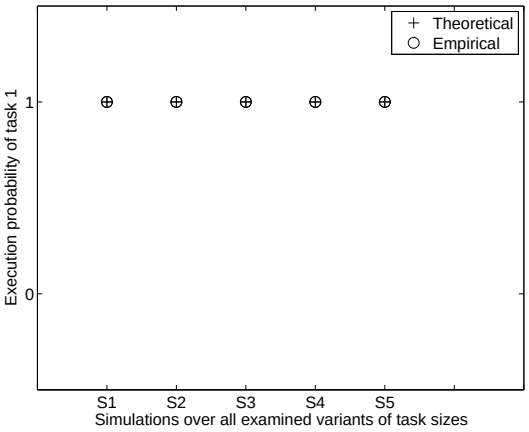
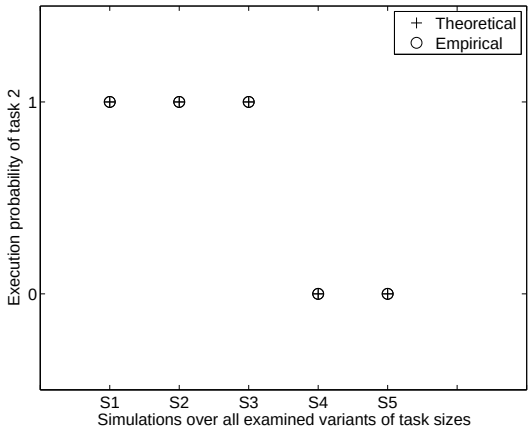


Figure 6.15 – *The swarm size optimized to be used for each set of the examined task sizes.*

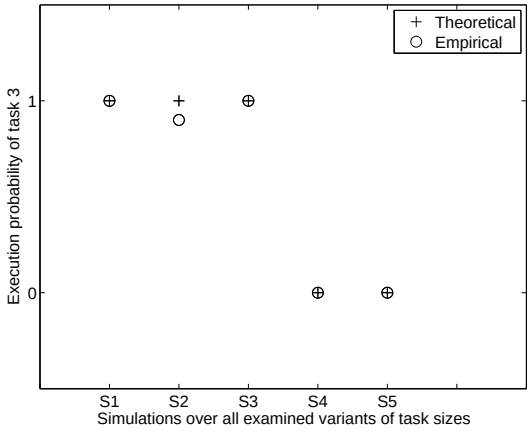
Figure 6.16 illustrates the execution probability (the probability to finish the task within its deadline) of the three tasks over the different variants of the task sizes. It depicts the probabilities calculated as explained in Chapter 5 Section 5.2.2. In the same figure, the empirical probabilities obtained over repeated ARGoS simulations (10 runs for each variant of the task sizes) are also shown, where we can see the consistency between the calculated and the simulated probabilities.



(a) Execution probability of task T_1.



(b) Execution probability of task T_2.



(c) Execution probability of task T_3.

Figure 6.16 *– The execution probabilities of the three tasks over the of different variants of task sizes under dynamic allocation.*

6.4.2 Soft-deadline Foraging under Dynamic Allocation

In this section, the three tasks are characterized by the following soft deadlines $\{500, 1500, 3000\}$ seconds. Missing any of these soft deadlines is associated with a specific cost. The task sizes are $\{150, 300, 500\}$ parts, i.e. 150 parts should be retrieved within 500 seconds on the first task, 300 parts within 1500 seconds on the second task and 500 parts within 3000 seconds on the third task. The expected value of the performance variance is defined in Chapter 5, Section 5.3.2, as the difference between the task size which should be executed within the task deadline and the expected number of parts to be performed up to the task deadline. The allocation technique is designed to minimize the costs associated with the number of unprocessed parts at the task deadline.

A homogeneous swarm of an increasing size over the range $[5 - 50]$ robots with an increment step of 5 robots, is used to execute the set of the three tasks. The expected performance variance is calculated first using Equation (4.42) in Section 4.5, Chapter 4, Section 5.3.2. After that, it is averaged over repeated ARGoS simulations (10 runs for each of the examined swarm sizes), see Figure 6.17.

In the figure, the difference between the task size and the calculated/simulated number of retrieved parts decreases while increasing the swarm size. The swarm performance measured in terms of the number of retrieved parts keeps improving while increasing the swarm size up to the maximum before it starts to decrease affected by the spatial interferences among the robots.

The difference between the calculated and the simulated performance variance, which can be noticed on tasks T_2 and T_3 for large numbers of assigned robots has the following reason: the mean time, $\hat{\mu}_i$, required by a single robot to retrieve one part on task T_i, is measured using a swarm of 50 robots on each of the three tasks, Figure 6.12 (b). As we have seen in Chapter 4, Section 4.3, the mean time, $\hat{\mu}_i$, increases by increasing the swarm size influenced by the spatial interferences between robots. Therefore, having 30 robots working on a task is associated with a smaller mean time $\hat{\mu}_i$, than in the case of having 50 robots. Consequently, the *simulated* performance variance is smaller than the *calculated* performance variance when less robots are working on the task than the swarm size used in estimating $\hat{\mu}_i$. This difference is always on the safe side, as the number of robots working on any task

will be smaller than the total swarm size used in estimating the mean time $\hat{\mu}_i$. As a part of the future work, the mean time, $\hat{\mu}_i$, can be estimated as a function of the number of robots working on the task.

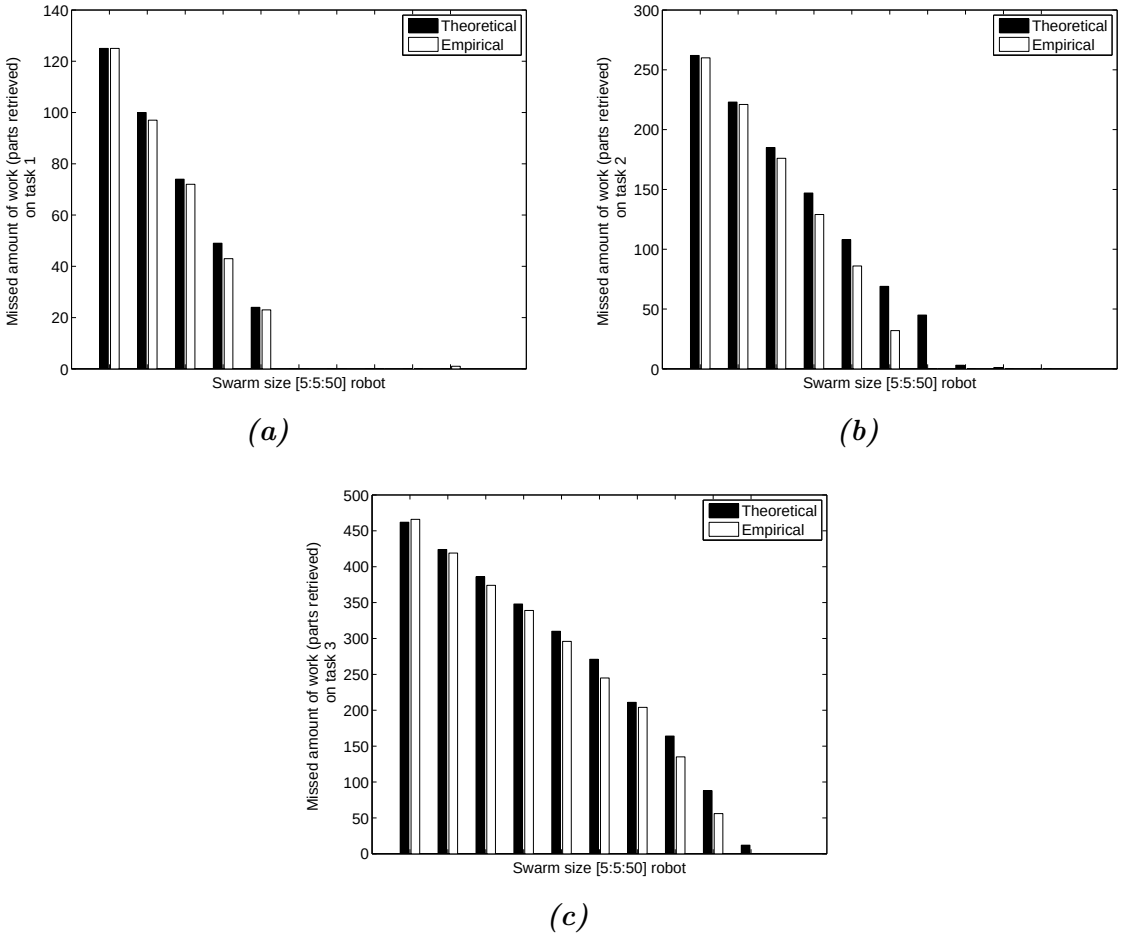


Figure 6.17 – *Evaluation of the task allocation strategy developed for soft-deadline tasks under the technique of dynamic allocation. The figure shows the expected difference between the task size and the accomplished amount of work (retrieved parts), theoretically and empirically averaged over ARGoS repeated simulations.*

6.5 Conclusions

This chapter was dedicated to verifying the task allocation strategies developed in Chapter 5 using physics-based simulations. The simulator ARGoS was used for

performing repetitive high-level simulations. ARGoS is an efficient fast simulator for heterogeneous robots. It possesses a set of unique features such as: The modular architecture where all system units including: robots, physics engines, visualizations, and controllers are plug-ins. In addition, ARGoS has the possibility to run multiple physics engines simultaneously. In our simulations, we have used the foot-bot robot to build up our swarms.

The simulation scenarios were presented under both static and dynamic allocation techniques and for both types of deadlines: Hard and Soft deadlines. For hard-deadline tasks, the main focus was on the execution probability of the tasks, where the calculated and simulated probabilities were consistent. For soft-deadline tasks, the main interest was related to reduce the costs associated with missing the task deadlines. These costs were expressed in terms of the difference between the required performance to obtain and the expected performance to achieve at the task deadline. The swarm performance was calculated using the corresponding equations from Chapter 5. After that, it was compared with the performance averaged over repeated ARGoS simulations. The results have shown the consistency between the calculated and the simulated performances.

Chapter 7

Conclusion

To succeed, jump as quickly at opportunities as you do at conclusions.

- Benjamin Franklin

This chapter concludes the thesis by giving a summary of the main contributions and the directions of possible future work.

7.1 Summary of Contributions

In this thesis, we have presented original research work to discuss the problem of assigning robot swarms to tasks with specific time constraints. Task allocation strategies have been developed to allow the robots to assign themselves to a set of tasks autonomously and independently under the temporal constraints of the tasks, namely, the deadlines.

Our approach focuses on a special kind of collective tasks, where each task is built up of discrete sub-tasks, referred to as parts. The part requires one robot to execute it and the task is characterized by its size and its deadline. The task size refers to the number of parts that should be accomplished on the task and the deadline represents the time point up to which the task should have been executed. Two main types of deadlines are discussed: *Hard deadlines* and *Soft deadlines*. Tasks where missing the deadline can lead to critical results or to a kind of an execution failure are characterized by their *hard deadlines*. Tasks where missing the deadline reduces the quality of the performance and this is associated with specific costs, are characterized by their *soft deadlines*.

For the development of efficient task allocation strategies, an important parameter was taken into account which plays a main role when temporal constraint are considered. This parameter is the switching costs between tasks which represent the time spent by the robot to stop the execution of its task and travel to start the execution of another task. These costs may be high or negligible based on the physical representation of the tasks and on the relativity between the switching time and the execution time of the task. For tasks with high switching costs *static allocation* was developed, where the robots select their tasks at the beginning of the execution and each robot decides independently the random time it will dedicate to the task. As soon as the robots select their tasks, they do not switch among the tasks during the execution time. On the other hand, for tasks with negligible switching costs, *dynamic allocation* was developed, where robots are allowed to switch between the tasks each time the robot finishes executing one part of its current task. A set of decision matrices which hold the switching probabilities between the tasks were designed to be exploited by the robots, independently, for the purposes of task switching.

This thesis includes the following main contributions:

- Mathematical modeling of swarm performance: swarm performance has been defined in Chapter 4, Section 4.4 under static allocation as the total time dedicated to the task, by the robots working on it, up to its deadline. On the other hand, swarm performance under dynamic allocation was defined in Chapter 4, Section 4.5, as the total number of parts accomplished on the task up to its deadline. Different mathematical techniques were proposed to model the swarm performance according to its particular definition.

  Under static allocation, the concept of robots' activity times was introduced as a set of continuous random durations sampled by the robots, independently, to be dedicated to the task. Different probability distributions were proposed for the random variable associated with the robot's activity time and the execution probability of the task was analyzed under the proposed distributions. The execution probability, here, was defined as the probability of dedicating the minimum threshold of time to the task (time required to finish executing the task) within its deadline.

  Under dynamic allocation, the evolution of the work on the task over time was modeled using the Poisson process, with different rates for each task over its activation periods. The behavior of the single robot was modeled as a Semi-Markov process. The execution probability was analyzed in association with the specific rate of the Poisson process that models the work progress on the task. The execution probability, here, was defined as the probability of executing a number of parts up to the task deadline which is equal to or greater than task size.

  In addition to the probabilistic analysis, the expected value of the variance in the swarm performance from the required performance was studied analytically and defined as the cost function associated with missing the soft deadlines, under both static and dynamic allocation techniques.

  The probability analysis performed under static and dynamic allocations was exploited in planning the execution of hard-deadline tasks by swarm robotics. Whereas the analytical study of the swarm performance variance was used to plan the execution of soft-deadline tasks.

- Task allocation strategies: developed under static and dynamic allocation techniques for tasks with hard and soft deadlines. The allocation strategies were designed based on the mathematical models developed for both static and dynamic allocations. They allow an autonomous allocation of the robots among the time-constrained tasks.

  For hard-deadline tasks, the developed strategies attempt to maximize the number of executable tasks with an acceptable execution probability (near to one). In the case of soft-deadline tasks, the goal is to minimize the costs associated with the missed deadlines.

- Additional contributions: including a set of simulated experiments and scenarios where the developed allocation strategies were verified successfully. Monte-Carlo simulations were performed using the Matlab platform. After that, the allocation strategies were verified through repetitive high-level physics-based simulations using the robots simulator *ARGoS*.

To conclude, the proposed strategies can be used for autonomous task allocation of a swarm of homogeneous robots to execute time-constrained tasks. The time constraints can be soft or hard and the tasks can have high or negligible switching costs.

7.2 Future Work

The research developed in this thesis is based on a set of assumptions related to the environment of operation and the kind of tasks we are considering. This is feasible since we are dealing with a relatively new field of research, namely swarm robotics, and we are performing a completely new kind of combination between this field and the field of real-time planning. The following are directions for possible future work:

- Methodology: the proposed allocation strategies are non-communicative strategies. Introducing robot-to-robot communication in the design of the allocation strategies may improve, considerably, the performance of the system and consequently the consideration of the task time constraints.

- Tasks: a set of assumptions were made about the category of tasks we are considering, including the uniform distribution of the task parts and the regeneration of the parts. Albeit these assumptions are verified in a set of real-world tasks, being able to relax some or all of these assumptions may allow the allocation approach to cover a larger set of tasks.

- Robots: the thesis assumes a homogeneous swarm of robots to execute the tasks. However, using a heterogeneous swarm of robots may offer a special kind of cooperation among the different types of robots which in turn may improve the performance of the system. In addition to the cooperation, specialization on the different tasks is another aspect which can be efficiently exploited in heterogeneous swarms and which may improve the system performance considerably.

Appendix A

The Probability Functions of n Truncated Exponentially Distributed Variables

- **From Probability Density Function (PDF) to Characteristic Function**

  We start with the characteristic function of a random variable (x) following the truncated exponential distribution and try to find the characteristic function of the sum of n random variables, each of them is following the truncated exponential distribution.

  In principle the characteristic function is the Fourier transformation of the probability density function (PDF) and is given by:

$$\varphi(t) = \int_{-\infty}^{\infty} e^{itx} f(x) dx \tag{A.1}$$

where $f(x)$ is the probability density function of the random variable x.

$$\varphi(t+i\lambda) = \int_{-\infty}^{\infty} e^{i(t+i\lambda)x} f(x)dx$$
$$= \int_{-\infty}^{\infty} e^{itx}(e^{-\lambda x}) f(x)dx$$

So if $\varphi(t)$ is the characteristic function of $f(x)$, then $\varphi(t+i\lambda)$ is the characteristic function of $e^{-\lambda x}f(x)$.
It is well known that the characteristic function of the sum of n independent variables, where each of them has the characteristic faction $\varphi(t)$ is given by: $[\varphi(t)]^n$ - Kendall (Kendall & Stuart, 1977).

Now back to the truncated exponential distribution, if z is a random variable which obeys the truncated exponential distribution with the parameter λ and the truncation points $a1$ and $b1$, it will have the following probability density function:

$$f_T(z) = \begin{cases} \dfrac{\lambda e^{-\lambda z}}{e^{-\lambda a_1} - e^{-\lambda b_1}} & a_1 \leqslant x \leqslant b_1 \\ 0 & \text{Otherwise} \end{cases}$$

Let us substitute $x = \lambda z$, $a = \lambda a1$ and $b = \lambda b1$, thus we can write it as in the following:

$$f_T(x) = \begin{cases} \dfrac{\lambda e^{-x}}{e^{-a} - e^{-b}} & a \leqslant x \leqslant b \\ 0 & \text{Otherwise} \end{cases}$$

The characteristic function of x in this case is defined based on the probability density function of x as in the following:

$$\varphi(t) = \frac{\lambda}{e^{-a} - e^{-b}} \int_a^b e^{itx} e^{-x} dx$$
$$\varphi(t) = \frac{\lambda}{e^{-a} - e^{-b}} \int_a^b e^{(it-1)x} dx$$

Thus, the characteristic function of a random variable follows the truncated exponential distribution is given by:

$$\varphi(t) = \frac{\lambda[e^{(it-1)b} - e^{(it-1)a}]}{(e^{-a} - e^{-b})(it-1)} \tag{A.2}$$

Consequently, the characteristic function of the sum of n independent random variables, each of them is following the truncated exponential distribution is, according to Kendall (Kendall & Stuart, 1977), given by:

$$\varphi(t) = \frac{[\lambda[(e^{(it-1)b} - e^{(it-1)a})]]^n}{(e^{-a} - e^{-b})^n (it-1)^n} \tag{A.3}$$

- **From Characteristic Function to Probability Density Function (PDF)**
  After finding out the characteristic function of the sum of n independent random variables which are following the truncated exponential distribution, we need now to find back the probability density function (PDF) of their distribution based on the obtained characteristic function.
  Let us consider the rectangular distribution of the random variable x. The probability density function is given by:

$$f_{REC}(x) = \begin{cases} \dfrac{1}{b-a} & a < x < b \\ 0 & \text{Otherwise} \end{cases} \tag{A.4}$$

  The characteristic function of the rectangular distribution is as in the following:

$$\varphi_{REC}(t) = \frac{e^{itb} - e^{ita}}{(b-a)it} \tag{A.5}$$

  If s is the sum of n independent variables, each has the probability density function $f_{REC}(x)$, then according to Cramer (Cramer, 1946), the probability density function of the sum s will be given by:

$$g(s) = (b-a)^{-n} \sum_{k=0}^{m} (-1)^k \binom{n}{k} \frac{[s - na - k(b-a)]^{n-1}}{(n-1)!} \tag{A.6}$$

  where $na + m(b-a) < s < na + (m+1)(b-a)$ and $m = 0, 1, .., n-1$.

As $i(t+i) = it - 1$, we can write:

$$\varphi_{REC}(t+i) = \frac{e^{i(t+i)b} - e^{i(t+i)a}}{(b-a)(i(t+i))}$$

$$\Rightarrow \varphi_{REC}(t+i) = \frac{e^{(it-1)b} - e^{(it-1)a}}{(b-a)(it-1)} \tag{A.7}$$

The characteristic function of the truncated exponential distribution is given by:

$$\varphi_{EXP}(t) = \frac{\lambda[e^{(it-1)b} - e^{(it-1)a}]}{(e^{-a} - e^{-b})(it-1)} \tag{A.8}$$

From (A.7) and (A.8) $\Rightarrow$

$$\varphi_{EXP}(t) = \frac{\lambda(b-a)}{(e^{-a} - e^{-b})}\varphi_{REC}(t+i) \tag{A.9}$$

And as : $\varphi(t+i) = \varphi(t)e^{-x} \Rightarrow$

$$\varphi_{EXP}(t) = \frac{\lambda(b-a)}{(e^{-a} - e^{-b})}\varphi_{REC}(t)e^{-x} \tag{A.10}$$

So for s the sum of n independent random variables obey the truncated exponential distribution and based on the calculated probability density function of the rectangular distribution, the probability density function of the sum of n random variables follow the truncated exponential distribution is given by:

$$f_T(s) = (\frac{\lambda(b-a)}{e^{-a} - e^{-b}})^n g(s)e^{-s} \tag{A.11}$$

where $g(s)$ is calculated in (A.6). Thus, the final form of the probability density function is as in the following:

$$f_T(s) = [\frac{\lambda}{(e^{-a} - e^{-b})}]^n \sum_{k=0}^{m}(-1)^k \binom{n}{k} \frac{[s - na - k(b-a)]^{n-1}}{(n-1)!}e^{-s} \tag{A.12}$$

where $na + m(b-a) < s < na + (m+1)(b-a)$ and $m = 0, 1, .., n-1$.
To find m_0 which is the upper bound of the sum s we have:

$$\begin{aligned} na + m_0(b-a) < s < na + (m_0+1)(b-a) \\ \Rightarrow m_0 < \frac{s-na}{(b-a)} < m_0 + 1 \end{aligned} \tag{A.13}$$

Which can be written as:

$$\begin{cases} m_0 < \dfrac{s-na}{(b-a)} \\ m_0 > \dfrac{s-na}{(b-a)} - 1 \end{cases}$$

So m_0 is the largest integer less than or equal to $\dfrac{s-na}{(b-a)}$
Now to find the cumulative density function (CDF) associated with the probability density function in (A.12), we must integrate $f_T(s)$ for all m values from 0 to m_0:

$$\begin{aligned}
\text{For } m=0 : \Rightarrow & \int_{na}^{na+(b-a)} f_T(s) \\
\text{For } m=1 : \Rightarrow & \int_{na+(b-a)}^{na+2(b-a)} f_T(s) \\
\text{For } m=2 : \Rightarrow & \int_{na+2(b-a)}^{na+3(b-a)} f_T(s) \\
& \vdots \\
\text{For } m=m_0 : \Rightarrow & \int_{na+m_0(b-a)}^{na+s(b-a)} f_T(s)
\end{aligned}$$

Since $f_T(s)$ is a sum of m_0+1 terms, the first term for $k=0$ to $k=m_0$ will occur in all segments, and hence must be integrated from na to s. The second term occurs for $k=1$ to $k=m_0$ and hence must be integrated from $na+(b-a)$ to s. Finally the last term occurs for $K=m_0$ only, and hence must be integrated from $na+m_0(b-a)$ to s. According to this the cumulative distribution function can be written as in the following:

$$F_T(s)=[\frac{\lambda}{(e^{-a}-e^{-b})}]^n\sum_{k=0}^{m_0}(-1)^k\binom{n}{k}\int_{na+k(b-a)}^{s}\frac{[s-na-k(b-a)]^{n-1}}{(n-1)!}e^{-s}ds \tag{A.14}$$

Let us have $y=s-na-k(b-a)\Rightarrow s=y+na+k(b-a)$ then:

$$F_T(s)=[\frac{\lambda}{(e^{-a}-e^{-b})}]^n\sum_{k=0}^{m_0}(-1)^k\binom{n}{k}\int_{0}^{y}\frac{y^{n-1}}{(n-1)!}e^{-y-na-k(b-a)}dy$$

$$F_T(s)=e^{-na}[\frac{\lambda}{(e^{-a}-e^{-b})}]^n\sum_{k=0}^{m_0}(-1)^k\binom{n}{k}\frac{e^{-k(b-a)}}{(n-1)!}\int_{0}^{y}y^{n-1}e^{-y}dy$$

$$F_T(s)=\underbrace{e^{-na}[\frac{\lambda}{(e^{-a}-e^{-b})}]^n}_{\text{a}}\sum_{k=0}^{m_0}(-1)^k\binom{n}{k}e^{-k(b-a)}\frac{1}{(n-1)!}\int_{0}^{y}y^{n-1}e^{-y}dy$$

a) can be written as:

$$e^{-na}[\frac{\lambda}{(e^{-a}-e^{-b})}]^n = \frac{\lambda^n e^{-na}(e^{na})}{(e^{-a}-e^{-b})^n(e^a)^n} = \frac{\lambda^n}{(1-e^{a-b})^n}$$

So $F_T(s)$ becomes:

$$F_T(s) = \sum_{k=0}^{m_0} \underbrace{\frac{\lambda^n}{(1-e^{a-b})^n}(-1)^k \binom{n}{k} e^{-k(b-a)}}_{\text{b}} \underbrace{\frac{1}{(n-1)!}\int_0^y y^{n-1}e^{-y}dy}_{\text{c}}$$

b) can be written as function of k:

$$a_k = (-1)^k \binom{n}{k} [e^{a-b}]^k a_0$$

where:

$$a_0 = \frac{\lambda^n}{(1-e^{a-b})^n}$$

c) is $\frac{1}{\Gamma(n)}\int_0^y y^{n-1}e^{-y}dy$ which is the cumulative distribution function of the *Gamma* Distribution with the shape n and parameter λ referred to as $G(y, n, \lambda)$. Consequently, The cumulative distribution function (CDF) of the sum of n independent random variables obey the truncated exponential distribution is given by:

$$F_T(s) = \sum_{k=0}^{m_0} a_k G(y, n, \lambda) \tag{A.15}$$

Where $a_k = (-1)^k \binom{n}{k} [e^{a-b}]^k a_0$ and $a_0 = \frac{\lambda^n}{(1-e^{a-b})^n}$
and $G(y, n, \lambda)$ is the cumulative distribution function of the *Gamma* distribution with shape n and parameter λ.

List of Publications

Khaluf, Y. & Rammig, F.J. (2013). Task allocation strategy for time-constrained tasks in robots swarms. In *ECAL 2013, 12th European Conference on Artificial Life*.

Khaluf, Y., Mathews, E. & Rammig, F.J. (2011). Self-organized cooperation in swarm robotics. In *14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW)*, 217–226, IEEE, IEEE Computer Society, Newport Beach, California, USA.

Khaluf, Y., Mathews, E. & Rammig, F.J. (2012a). Swarm robotic time synchronization for object tracking. In M.T. Higuera-Toledano, U. Brinkschulte & A. Rettberg, eds., *Self-Organization in Embedded Real-Time Systems*, 75 – 92, Springer, New York, Heidelberg, Dordrecht, London.

Khaluf, Y., Weiss, F. & Micus, S. (2012b). Master election for time synchronization in swarm robotic systems. In *The 10th IEEE International Symposium on Parallel and Distributed Processing with Applications*, Leganés, Madrid.

Khaluf, Y., Birattari, M. & Rammig, F.J., eds. (2013). *Probabilistic Analysis of Long-term Swarm Performance under Spatial Interferences*, Springer, 2nd International Conference on the Theory and Practice of Natural Computing, TPNC 2013, Cáceres, Spain.

References

Acebo, E.D. & de-la Rosa, J. (2008). Introducing bar systems: a class of swarm intelligence optimization algorithms. In *AISB Convention Communication, Interaction and Social Intelligence, Aberdeen, Scotland*, 18–23. 29

Agassounon, W. & Martinoli, A. (2002a). 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*, 1090–1097, ACM, New York, USA. 28

Agassounon, W. & Martinoli, A. (2002b). A macroscopic model of an aggregation experiment using embodied agents in groups of time-varying sizes. In *Proceedings of IEEE Conference on System, Man and Cybernetics*, 250–255. 24

Agassounon, W., Martinoli, A. & Goodman, R. (2001). A scalable, distributed algorithm for allocating workers in embedded systems. In *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, 3367–3373. 28

Arkin, R. (1998). *Behavior-based robotics*. MIT press, MA, USA. 19

Balch, T. (1998). Taxonomies of multirobot task and reward. Tech. rep., In. 35, 36

Balch, T. (1999a). The impact of diversity on performance in multi-robot foraging. In *Proceedings of the Third Annual Conference on Autonomous Agents*, 92–99, ACM, New York, USA. 36

Balch, T. (1999b). Reward and diversity in multirobot foraging. In *IJCAI-99 Workshop on Agents Learning About, From and With other Agents*. 132

Bautista, J. & Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research*, **177**, 2016–2032. 13

Beni, G. (1988). The concept of cellular robotic systems. In *The 3rd IEEE International Symposium on Intelligent Control*, 57–62. 2

Beni, G. & Wang, J. (1990a). Self-organizing sensory systems. In J. Tou & J. Balchen, eds., *Highly Redundant Sensing in Robotic Systems*, vol. 58, 251–262, Springer, Berlin Heidelberg. 2

Beni, G. & Wang, J. (1990b). Self-organizing sensory systems. In *Highly Redundant Sensing in Robotic Systems*, 251–262, Springer, Berlin Heidelberg. 2

Beni, G. & Wang, J. (1993). Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?*, 703–712, Springer, Berlin Heidelberg. 12, 14

Berman, S., Halász, A., Hsieh, M. & Kumar, V. (2009). Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, **25**, 927–937. 19

Berthold, P., Helbig, A., Mohr, G. & Querner, U. (1992). Rapid microevolution of migratory behaviour in a wild bird species. 10

Besten, M.D., Stützle, T. & Dorigo, M. (2000). Ant colony optimization for the total weighted tardiness problem. In *Parallel Problem Solving from Nature PPSN VI*, 611–620, Springer. 12

Blackwell, T. & Branke, J. (2004). Multi-swarm optimization in dynamic environments. In *Applications of Evolutionary Computing*, 489–500, Springer, Berlin Heidelberg. 13

Blackwell, T. & Branke, J. (2006). Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, **10**, 459–472. 13

Blum, C. (2005). Beam-aco-hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Computers & Operations Research*, **32**, 1565–1591. 12

Blum, C. & Sampels, M. (2004). An ant colony optimization algorithm for shop scheduling problems. *Journal of Mathematical Modelling and Algorithms*, **3**, 285–308. 12

Blum, C., Bautista, J. & Pereira, J. (2006). Beam-aco applied to assembly line balancing. In *Ant Colony Optimization and Swarm Intelligence*, 96–107, Springer, Berlin Heidelberg. 13

Bonabeau, E., Sobkowski, A., Theraulaz, G. & Deneubourg, J. (1998). Adaptive task allocation inspired by a model of division of labor in social insects. 26

Brambilla, M., Pinciroli, C., Birattari, M. & Dorigo, M. (2012). Property-driven design for swarm robotics. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, vol. 1, 139–146, International Foundation for Autonomous Agents and Multiagent Systems. 19

Brambilla, M., Ferrante, E., Birattari, M. & Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, **7**, 1–41. 15

Brutschy, A., Pini, G., Pinciroli, C., Birattari, M. & Dorigo, M. (2012). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, 1–25. 28

Carlisle, A. & Dozier, G. (2000). Adapting particle swarm optimization to dynamic environments. In *Proceedings of the International Conference on Artificial Intelligence*, vol. 1, 429–434. 13

Carlisle, A. & Dozler, G. (2002). Tracking changing extrema with adaptive particle swarm optimizer. In *Proceedings of the 5th Biannual World Automation Congress*, vol. 13, 265–270. 13

Ceballos, N., Valencia, J. & Ospina, N. (2010). Quantitative performance metrics for mobile robots navigation. In *Mobile Robots Navigation*, 485–500, In-Tech. 35

Chevaleyre, Y., Dunne, P., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodriguez-Aguilar, J. & Sousa, P. (2006). Issues in multiagent resource allocation. *Informatica (Slovenia)*, **30**, 3–31. 26

Christensen, A., O'Grady, R. & Dorigo, M. (2007). A mechanism to self-assemble patterns with autonomous robots. In *Advances in Artificial Life*, 716–725, Springer, Berlin Heidelberg. 15

Coello, C.C. & Lechuga, M. (2002). Mopso: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, 1051–1056. 13

Correa, E., Freitas, A. & Johnson, C. (2006). A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics data set. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 35–42. 13

Corry, P. & Kozan, E. (2004). Ant colony optimisation for machine layout problems. *Computational optimization and applications*, **28**, 287–310. 13

Cramer, H. (1946). *Mathematical methods of Statistics*. Princeton University Press, NJ, USA. 50, 166

Dahl, T., Matarić, M. & Sukhatme, G. (2009). Multi-robot task allocation through vacancy chain scheduling. *Robot. Auton. Syst.*, **57**, 674–687. 27

Danchin, E., Giraldeau, L., Cézilly, F. *et al.* (2008). *Behavioural ecology*. Oxford University Press, New York, USA. 132

Dias, M., Zlot, R., Kalra, N. & Stentz, A. (2005). Market-based multirobot coordination: A survey and analysis. Tech. Rep. CMU-RI-TR-05-13, Robotics Institute, Pittsburgh, PA. 27

Dorigo, M. (2007). Ant Colony Optimization. *Scholarpedia*, **2**, 1461. 13

Dorigo, M. & Gambardella, L. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, **1**, 53–66. 12

Dorigo, M. & Stützle, T. (2004). *Ant Colony Optimization*. MIT Press, Cambridge, MA. 13

Dorigo, M., Maniezzo, V. & Colorni, A. (1991). Positive feedback as a search strategy. Tech. rep., IRIDIA. 12

Dorigo, M., Maniezzo, V. & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, **26**, 29–41. 12

Dorigo, M., Trianni, V., Şahin, E., Gross, R., Labella, T., Baldassarre, G., Nolfi, S., Deneubourg, J., Mondada, F., Floreano, D. *et al.* (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, **17**, 223–245. 15

Dorigo, M., de Oca, M. & Engelbrecht, A. (2008). Particle swarm optimization. *Scholarpedia*, **3**, 1486. 13

Ducatelle, F., Förster, A., Caro, G.D. & Gambardella, L. (2009a). New task allocation methods for robotic swarms. In *9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions*. 27

Ducatelle, F., Förster, A., Caro, G.D. & Gambardella, L. (2009b). Task allocation in robotic swarms: new methods and comparisons. Tech. rep., Dalle Molle Institute for Artificial Intelligence. 27

Durrett, R. (2004). *Probability: theory and examples*. Cambridge University Press, Cambridge, UK. 52

Eberhart, R. & Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1, 94–100. 13

Endriss, U., Maudet, N., Sadri, F. & Toni, F. (2005). Negotiating socially optimal allocations of resources: An overview. Tech. rep., Imperial College London, Department of Computing. 26

Fazli, P., Davoodi, A. & Mackworth, A. (2012). Multi-robot repeated area coverage: Performance optimization under various visual ranges. In *Proceedings of the Ninth Conference on Computer and Robot Vision*, 298–305, IEEE Computer Society, Washington, DC, USA. 35

Ferreira-Jr, P., Boffo, F. & Bazzan, A. (2008). Using swarm-gap for distributed task allocation in complex scenarios. In *Massively Multi-Agent Technology*, 107–121, Springer, Berlin Heidelberg. 28

Gagné, C., Price, W. & Gravel, M. (2002). Comparing an aco algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. *Journal of the Operational Research Society*, **53**, 895–906. 13

Galstyan, A. & Lerman, K. (2005). Analysis of a stochastic model of adaptive task allocation in robots. In *Engineering Self-Organising Systems*, 167–179, Springer, Berlin Heidelberg. 21

Galstyan, A., Hogg, T. & Lerman, K. (2005). Modeling and mathematical analysis of swarms of microscopic robots. In *Proceedings of IEEE Swarm Intelligence Symposium*, 201–208. 25

Gambardella, L., Taillard, É. & Agazzi, G. (1999). Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In *New ideas in optimization*. 13

Georgiou, V., Pavlidis, N., Parsopoulos, K., Alevizos, P. & Vrahatis, M. (2004). Optimizing the performance of probabilistic neural networks in a bioinformatics task. In *Proceedings of the EUNITE conference*, 34–40. 13

Gerkey, B. & Mataric, M. (2002). Sold!: Auction methods for multirobot coordination. 27

Gerkey, B. & Mataric, M. (2003). Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *IEEE International Conference on Robotics and Automation.*, vol. 3, 3862–3868. 27

Gerkey, B. & Mataric, M. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, **23**, 939–954. 27

Goldberg, D. (2001). *Evaluating the dynamics of agent-environment interaction*. Ph.D. thesis, University of Southern California. 37

Goldberg, D. & Mataric, M. (2003). Maximizing reward in a non-stationary mobile robot environment. *Autonomous Agents and Multi-Agent Systems*, **6**, 287–316. 19

Goldberg, D., Cicirello, V., Dias, M., Simmons, R., Smith, S. & Stentz, A. (2003). Task allocation using a distributed market-based planning mechanism. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 996–997, ACM, New York, USA. 27

Gottlieb, J., Puchta, M. & Solnon, C. (2003). A study of greedy, local search, and ant colony optimization approaches for car sequencing problems. In *Applications of evolutionary computing*, 246–257, Springer, Berlin Heidelberg. 13

Grinstead, C. & Snell, J. (1998). *Introduction to probability*. American Mathematical Society, USA, 2nd edn. 54

Gross, R., Bonani, M., Mondada, F. & Dorigo, M. (2005). Autonomous self-assembly in mobile robotics. *IEEE Trans. Robot*. 15

Grünbaum, D. (1998). Schooling as a strategy for taxis in a noisy environment. *Evolutionary Ecology*, **12**, 503–522. 10

Gudise, V. & Venayagamoorthy, G. (2003). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, 110–117. 13

Guerrero, J. & Oliver, G. (2007). Multi-robot task allocation method for heterogeneous tasks with priorities. In *Distributed Autonomous Robotic Systems*, 181–190, Springer, Japan. 29

Guerrero, J. & Oliver, G. (2010). A multi-robot auction method to allocate tasks with deadlines. In *7th IFAC Symposium on Intelligent Autonomous Vehicles, Lecce, Italy*. 29

Guerrero, J. & Oliver, G. (2011). Auction and swarm multi-robot task allocation algorithms in real time scenarios. In T. Yasuda, ed., *Multi-Robot Systems, Trends and Development*, 437–456, InTech. 21, 29

Hamann, H. & Wörn, H. (2007). An analytical and spatial model of foraging in a swarm of robots. In *Swarm Robotics*, 43–55, Springer, Berlin Heidelberg. 15

Hamann, H., Wörn, H., Crailsheim, K. & Schmickl, T. (2008). Spatial macroscopic models of a bio-inspired robotic swarm algorithm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1415–1420. 23

Hsieh, M., Halász, A., Berman, S. & Kumar, V. (2008). Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence*, **2**, 121–141. 21

Hsieh, M., Halász, A., Cubuk, E., Schoenholz, S. & Martinoli, A. (2009). Specialization as an optimal strategy under varying external conditions. In *IEEE International Conference on Robotics and Automation*, 1941–1946. 21

Hu, X. & Eberhart, R. (2002a). Adaptive particle swarm optimization: detection and response to dynamic systems. In *Proceedings of the Congress on Evolutionary Computation.*, vol. 2, 1666–1670. 13

Hu, X. & Eberhart, R. (2002b). Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation.*, vol. 2, 1677–1681. 13

Ijspeert, A., Martinoli, A., Billard, A. & Gambardella, L. (2001). Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, **11**, 149–171. 14, 25

Jain, S., Shah, R., Brunette, W., Borriello, G. & Roy, S. (2006). Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mobile Networks and Applications*, **11**, 327–339. 133

Janson, S. & Middendorf, M. (2006). A hierarchical particle swarm optimizer for noisy and dynamic environments. *Genetic Programming and Evolvable Machines*, **7**, 329–354. 13

Jea, D., Somasundara, A. & Srivastava, M. (2005). Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *Distributed Computing in Sensor Systems*, 244–257, Springer, Berlin Heidelberg. 133

Jeanson, R., Rivault, C., Deneubourg, J., Blanco, S., Fournier, R., Jost, C. & Theraulaz, G. (2005). Self-organized aggregation in cockroaches. *Animal Behaviour*, **69**, 169–180. 25

Jones, E., Dias, M. & Stentz, A. (2007). Learning-enhanced market-based task allocation for oversubscribed domains. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2308–2313. 30

Kalra, N. & Martinoli, A. (2006). A comparative study of market-based and threshold-based task allocation. In *Proceedings of the 8th International Symposium on Distributed Autonomous Robotic Systems (DARS)*. 27

Karpenko, O., Shi, J. & Dai, Y. (2005). Prediction of mhc class ii binders using the ant colony search strategy. *Artificial Intelligence in Medicine*, **35**, 147–156. 13

Kendall, M. & Stuart, A. (1977). *The advanced theory of statistics*, vol. 1: Distribution theory. Macmillan, New York, USA, 4th edn. 50, 165, 166

Kennedy, J. & Eberhart, R. (2001). *Swarm intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 13

Korb, O., Stützle, T. & Exner, T. (2006). Plants: application of ant colony optimization to structure-based drug design. In *Ant Colony Optimization and Swarm Intelligence*, 247–258, Springer, Heidelberg Berlin, Germany. 13

Krieger, M. & Billeter, J. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. 27

Lavender, E. (1967). On the distribution of the sum of independent doubly truncated gamma variables. 49, 51

Lein, A. & Vaughan, R. (2008a). Adaptive multi-robot bucket brigade foraging. *Artificial Life*, **11**, 337. 37

Lein, A. & Vaughan, R. (2008b). Adaptive multirobot bucket brigade foraging. In *Proceedings of the Eleventh International Conference on Artificial Life (ALife XI)*, 337–342, MIT Press. 37

Lerman, K. & Galstyan, A. (2002). Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, **13**, 127–141. 24, 37, 38, 132

Lerman, K., Galstyan, A., Martinoli, A. & Ijspeert, A. (2001). A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, **7**, 375–393. 14, 19, 24

Lerman, K., Martinoli, A. & Galstyan, A. (2005). A review of probabilistic macroscopic models for swarm robotic systems. In *Swarm robotics*, 143–152, Springer, Heidelberg Berlin, Germany. 19, 23

Lerman, K., Jones, C., Galstyan, A. & Matarić, M. (2006). Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research*, **25**, 225–241. 26

Li, X. (2003). A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Genetic and Evolutionary Computation (GECCO)*, 37–48, Springer. 13

Li, X., Branke, J. & Blackwell, T. (2006). Particle swarm with speciation and adaptation in a dynamic environment. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 51–58, ACM. 13

Liu, W., Winfield, A., Sa, J., Chen, J. & Dou, L. (2007). Towards energy optimization: Emergent task allocation in a swarm of foraging robots. 28

Liu, W., Winfield, A. & Sa, J. (2009). A macroscopic probabilistic model of adaptive foraging in swarm robotics systems. 132

Martinoli, A. & Easton, K. (2003a). Modeling swarm robotic systems. In *Experimental Robotics VIII*, 297–306, Springer, Heidelberg Berlin, Germany. 23

Martinoli, A. & Easton, K. (2003b). Optimization of swarm robotic systems via macroscopic models. In *Proceedings of the Second International Workshop on Multi-Robots Systems*, 181–192. 25

Martinoli, A., Easton, K. & Agassounon, W. (2004). Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *The International Journal of Robotics Research*, **23**, 415–436. 25

Matarić, M. (1997). Reinforcement learning in the multi-robot domain. In *Robot colonies*, 73–83, Kluwer Academic Publishers, MA, USA. 132

McLurkin, J. & Yamins, D. (2005). Dynamic task assignment in robot swarms. *Proceedings of Robotics: Science and Systems*, **8**. 28

Mendes, R., Cortez, P., Rocha, M. & Neves, J. (2002). Particle swarms for feedforward neural network training. In *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, 1895–1899. 13

Merkle, D., Middendorf, M. & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on*, **6**, 333–346. 13

Meystel, A. (1991). *Autonomous Mobile Robots: Vehicles with Cognitive Control*. World Scientific, Singapore. 2

Moore, J. & Chapman, R. (1999). Application of particle swarm to multiobjective optimization. *Department of Computer Science and Software Engineering, Auburn University*. 13

Moravec, H. (1990). The stanford cart and the cmu rover. In I. Cox & G. Wilfong, eds., *Autonomous Robot Vehicles*, 407–419, Springer, Berlin Heidelberg. 2

Moss, J. & Johnson, C. (2003). An ant colony algorithm for multiple sequence alignment in bioinformatics. In *Artificial Neural Nets and Genetic Algorithms*, 182–186, Springer. 13

Nilsson, N. (1984). Shakey the robot. Tech. rep., DTIC Document. 2

Norris, J. (1998). *Markov chains*. Cambridge University Press, Cambridge, USA. 18, 21

Nouyan, S., Ghizzioli, R., Birattari, M. & Dorigo, M. (2005). An insect-based algorithm for the dynamic task allocation problem. *KI*, **19**, 25–31. 27

O'Grady, R., Gross, R., Christensen, A., Mondada, F., Bonani, M. & Dorigo, M. (2007). Performance benefits of self-assembly in a swarm-bot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2381–2387. 15

Onwubolu, G. & Clerc, M. (2004). Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization. *International Journal of Production Research*, **42**, 473–491. 13

Østergaard, E., Sukhatme, G. & Mataric, M. (2001). Emergent bucket brigading-a simple mechanism for improving performance in multi-robot constrained-space foraging tasks. In *Autonomous Agents*, 29–30, ACM, New York, USA. 37, 38, 132

Papadimitriou, C. & Steiglitz, K. (1982). *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., NJ, USA. 89

Papoulis, A. (1984). *Random Variables, and Stochastic Processes*. McGraw-Hill, New York, USA, 2nd edn. 51

Parrott, D. & Li, X. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *Evolutionary Computation, IEEE Transactions on*, **10**, 440–458. 13

Parsopoulos, K. & Vrahatis, M. (2002). Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM symposium on Applied computing*, 603–607, ACM. 13

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G., Ducatelle, F., Birattari, M., Gambardella, L. & Dorigo, M. (2012). Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, **6**, 271–295. 7, 134

Reebs, S. (2000). Can a minority of informed leaders determine the foraging movements of a fish shoal? *Animal Behaviour*, **59**, 403–409. 10

Reidman, M. (1990). The pinnipeds: seals, sea lions and walruses. 132

Reimann, M., Doerner, K. & Hartl, R. (2004). D-ants: savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, **31**, 563–591. 13

Reyes-Sierra, M. & Coello, C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, **2**, 287–308. 13

Ross, S. (1992). *Applied probability models with optimization applications*. Dover Publication, New York, USA. 18, 20, 21

Ross, S. (2006). *Introduction to probability models*. Academic Press, Inc., FL, USA. 46, 68, 102

Şahin, E., Girgin, S., Bayindir, L. & Turgut, A. (2008). Swarm robotics. In *Swarm Intelligence*, 87–100, Springer, Berlin Heidelberg. 14

Schneider, J., Apfelbaum, D., Bagnell, D. & Simmons, R. (2005). Learning opportunity costs in multi-robot market based planners. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1151–1156. 30

Settles, M., Rodebaugh, B. & Soule, T. (2003). Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network. In *Genetic and Evolutionary ComputationâĂŤGECCO 2003*, 148–149, Springer. 13

Shah, R., Roy, S., Jain, S. & Brunette, W. (2003). Data mules: Modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, 30–41. 133

Shell, D. & Mataric, M. (2006). On foraging strategies for large-scale multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2717–2723. 37, 132

Shmygelska, A., Aguirre-Hernandez, R. & Hoos, H. (2002). An ant colony optimization algorithm for the 2d hp protein folding problem. In *Ant Algorithms*, 40–52, Springer, Berlin Heidelberg, Germany. 13

Silva, C., Runkler, T., Sousa, J. & Palm, R. (2002). Ant colonies as logistic processes optimizers. In *Ant Algorithms*, 76–87, Springer, Berlin Heidelberg, Germany. 13

Somasundara, A., Ramamoorthy, A. & Srivastava, M. (2004). Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *Proceedings 25th IEEE International Real-Time Systems Symposium*, 296–305. 133

Soysal, O. & Şahin, E. (2006). A macroscopic model for probabilistic aggregation in swarm robotic systems. In *Proceedings of SAB06 Workshop on Swarm Robotics. Lecture Notes in Computer Science (LNCS).*, Springer. 24

Soysal, O. & Şahin, E. (2007). A macroscopic model for self-organized aggregation in swarm robotic systems. In *Swarm robotics*, 27–42, Springer, Berlin Heidelberg, Germany. 15

Stephens, D., Brown, J. & Ydenberg, R. (2007). *Foraging: behavior and ecology*. University of Chicago Press, Chicago, USA. 132

Stützle, T. *et al.* (1998). An ant approach to the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing*, vol. 3, 1560–1564. 13

Swaney, W., Kendal, J., Capon, H., Brown, C. & Laland, K. (2001). Familiarity facilitates social learning of foraging behaviour in the guppy. *Animal Behaviour*, **62**, 591–598. 10

Thompson, D. *et al.* (1942). *On growth and form*. Cambridge University Press, NY,USA. 11

Trianni, V. & Dorigo, M. (2005). Emergent collective decisions in a swarm of robots. In *Proceedings IEEE Swarm Intelligence Symposium*, 241–248. 15

Trianni, V., Nolfi, S. & Dorigo, M. (2006). Cooperative hole avoidance in a swarm-bot. *Robotics and Autonomous Systems*, **54**, 97–103. 15

Wang, M., Basagni, S., Melachrinoudis, E. & Petrioli, C. (2005). Exploiting sink mobility for maximizing sensor networks lifetime. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 287a. 133

Williams, H. (2009). *Logic and Integer Programming (International Series in Operations Research & Management Science)*. Springer, New York, USA, 1st edn. 89

Winfield, A. (2009). Foraging robots. In *Encyclopedia of Complexity and Systems Science*, Springer, New York, USA. 36, 132

Wong, S., Middleton, L. & MacDonald, B. (2002). Performance metrics for robot coverage tasks. In *Proceedings Australasian Conference on Robotics and Automation*, 7–12. 35

Zhang, D., Xie, G., Yu, J. & Wang, L. (2007). Adaptive task assignment for multiple mobile robots via swarm intelligence approach. *Robotics and Autonomous Systems*, **55**, 572–588. 28

Zheng, X., Koenig, S. & Tovey, C. (2006). Improving sequential single-item auctions. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*. 27