Université Libre de Bruxelles
Faculté des Sciences Appliquées
CODE - Computers and Decision Engineering
IRIDIA - Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Task Allocation in Swarm Robotics

Towards a method for self-organized allocation to complex tasks

Arne Brutschy

Supervisor:
Prof. Marco Dorigo

Co-Supervisor:
Dr. Mauro Birattari

Rapport d'avancement de recherche
Academic year: 2008/2009

Abstract

In this work, we propose a method for self-organized task partitioning and allocation. The current state of the art provides neither an abstract understanding of task allocation problems in self-organized swarm systems, nor tools for modeling and designing such systems. We present the first steps towards a unified method capable of exactly this.

We analyse the problem and provide a structured approach to complex task allocation problems. The method relies on partitioning complex tasks in smaller, more manageable subtasks which can be tackled by a swarm of robots in a self-organized way. Two types of subtasks are identified and discussed. We present experiments to study the properties of these subtasks, and propose algorithms to achieve self-organized allocation to these subtasks as well as test the algorithms in practical, real-world applications.

Acknowledgements

I would like to thank everyone in IRIDIA, my supervisors, my colleagues, my friends, for their support at various levels. Without you, IRIDIA would not be such an amazing place—a place where I actually manage to feel at home. I would like to thank my flatmates feeding me, and my Ultimate Frisbee team for supporting me.

I would like to thank Christine for everything.

*

* © 1995, Don Hertzfeldt, http://www.bitterfilms.com/

Contents

1 Introduction

In their life, most individuals constantly face complex tasks they have to master. Frequently, these tasks are time-consuming, difficult or even beyond the possibilities of a single individual. Tackling such tasks by a group of individuals, on the other hand, might surpass these problems, and is therefore a concept commonly found in nature. Be it carnivores hunting in packs or humans working in teams, cooperation allows to work more efficiently and tackle tasks which are beyond the capabilities of a single individual. Other reasons for teamwork might for example be increased robustness or specialization of individuals to certain tasks.

When creating artificial system like robots, designers face similar problems. One possibility is to create multi-purpose, complex and monolithic robots that are able to tackle the desired tasks alone. Although this solution seems to be the simplest way, it is limited when it comes to robustness, flexibility and scalability. An alternative approach is the so called *swarm intelligence*: drawing inspiration from natural systems like social insects, it tries to overcome the problems outlined above by creating a flexible swarm of simple individuals. Using methods such as decentralisation of control, limited communication abilities among individuals and the use of local information only, complex behaviour emerges at colony level. Swarm intelligence systems exhibit the desired characteristics like flexibility and robustness, while remaining manageable on a local level. *Swarm robotics* is the application of Swarm intelligence to robotics, using a swarm of relatively simple robots to tackle complex problems.

Of course, when trying to design groups of artificial individuals that should cooperate in order to solve a task, we are facing exactly the same problems as they occur in natural systems. The big problem is how to organize team work. Who is doing what? And when? If there are multiple ways of doing things, which is the most efficient? Humans, being faced with these questions, have developed complex social and operational rules in order to cooperate in a group—and still, massive failures can constantly be observed in our everyday life. Being much more limited than humans, robots (and therefore their human creators as well) seem to face a very complex problem. And still, taking inspiration from nature, observing existing groups and uncovering the underlying principles helps designing systems of robots that can, in a limited way, cooperate.

The problem of assigning tasks to individuals is commonly referred to as *task allocation*. In this work, we study the problem of task allocation in self-organized

swarms of robots. More specifically, we try to find a framework which allows us, the designers, to decompose a complex task into multiple subtasks which can be tackled by the swarm in a self-organized way.

1.1 Contribution and Goal

The goal of this work is to present the latest advancement in our research concerning self-organized task allocation, while giving the "overall picture" in terms of research direction and future work.

The main contribution is a unified method to represent and tackle complex task allocation in swarm robotic systems. The experiments presented in this work contribute to the general understanding of self-organized task allocation and help us to study the building blocks required for this method. Additionally, they serve as a testbed for a first version of a general framework about how to decompose complex tasks. Hence, the study ventures into finding solutions to self-organized task partitioning as well.

1.2 Outline of this Work

We will first present some fundamentals relevant to this work in Chapter 2, namely the field of swarm robotics and the *Swarmanoid* project, in which this work is embedded in. In the following Chapter 3 we review related works. In Chapter 4 we try to establish a general framework about how to decompose complex tasks into two types of interdependent subtasks. Chapters 5 and 6 are devoted on how to allocate individuals to these two types of subtasks in self-organized multi-robot systems. Chapter 5 reports current experiments and results, while Chapter 6 gives an outlook for future work and outlines possible experiments. Finally, Chapter 7 summarizes the presented studies and draws some conclusions.

2 Fundamentals

In this chapter, we introduce the underlying concepts of this work. First, we give an introduction to swarm intelligence and the related swarm robotics. Afterwards, we present the *Swarmanoid* project, in which this work is embedded in.

2.1 Swarm Intelligence

Swarm intelligence (commonly abbreviated with SI) is a branch of artificial intelligence based on the collective behavior of natural systems like social insects (Bonabeau et al., 2000; Garnier et al., 2007). It is commonly defined as:

> *[. . .] any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insects and other animal societies.*
>
> (Bonabeau et al., 1999)

The expression "swarm intelligence" was originally introduced by Beni and Wang (1989) for cellular robotics systems, but is nowadays widely used in the field of artificial intelligence. Swarm intelligence systems consist typically of a population of relatively simple individuals. Similar to social insects, the individuals follow simple behavioral rules and rely purely on local sensing and communication. Therefore, the individuals interact with each other and the environment on a local basis only. Because of the dynamic nature of swarm systems, these interactions are to a certain degree of a stochastic nature. Although there is no centralized control, such systems show the emergence of global behaviors that transcend the behavioral repertoire of the single individual: the swarm *self-organizes*. Natural examples of swarm intelligence include ant colonies (Detrain and Deneubourg, 2006), bird flocking (Reynolds, 1987), animal herding (Gautrais et al., 2007), colony of bacteria (Ben-Jacob et al., 2000), and fish schooling (Grünbaum et al., 2004) .

Swarm intelligence systems have a few invariant properties:

- the system is composed of many, relatively homogeneous individuals;
- interactions among the individuals are based on simple behavioral rules that exploit only local information;

- control is fully distributed among a number of individuals;
- communications among the individuals happen in a localized way;
- system-level behavior results from the interactions of individuals with each other and with their environment and appears to transcend the behavioral repertoire of the single individual;
- the overall response of the system is quite robust and adaptive with respect to changes in the environment.

Artificial systems that were developed following the swarm intelligence approach are usually flexible, robust, adaptive and scalable (Camazine et al., 2003; Cao et al., 1997). Nowadays, an increasing amount of human created algorithms fall into the domain of swarm intelligence, for example algorithms for optimization (Dorigo and Stützle, 2004), data analysis (Abraham et al., 2006) or network routing (Di Caro and Dorigo, 1998).

2.1.1 Swarm Robotics

Swarm robotics is a novel approach to robotics which tries to circumvent problems with classical, monolithic robots like inflexibility and individual complexity by applying the principles of swarm intelligence to the field of robotics (Dorigo and Şahin, 2004). Thus, it studies how a large number of physically embodied agents can be designed in such a way that the group self-organizes and a global, collective behaviour emerges. Swarm robotics emphasises aspects such as:

- decentralisation of control;
- limited communication abilities among robots;
- use of local information;
- self-organization of global behaviour.

Swarm robotic systems are, similar to their natural counterparts, made up of many simple robots. This allows for cheaper, less complex robots, which are more robust. Other goals of the swarm robotics approach are flexibility, adaptability and redundancy (see Bonabeau et al. (1999) or Beni (2005) for a review on swarm robotics).

Examples of known applications of swarm robotics include flocking (Turgut et al., 2008), morphogenesis and self-assembly (Christensen et al., 2007), intrinsic fault detection (Christensen et al., 2008), path formation and prey retrieval (Nouyan et al., 2008), coordinated behaviors (Sperati et al., 2008), and collective transport (Groß and Dorigo, 2008).

2.2 The *Swarmanoid* Project

This work is embedded in a project is called *Swarmanoid: Towards Humanoid Robotic Swarms. Swarmanoid* and is a a Future and Emerging Technologies project[1] funded by the European Commission. From the *Swarmanoid* project proposal:

> *The* Swarmanoid *project proposes a highly innovative way to build robots that can successfully and adaptively act in human made environments. The* Swarmanoid *project will be the first to study how to design, realise and control a heterogeneous swarm robotic system capable of operating in a fully 3-dimensional environment.*
>
> *The main scientific objective of the proposed research is the design, implementation and control of a novel distributed robotic system comprising heterogeneous, dynamically connected small autonomous robots so as to form what we call a* Swarmanoid*. The* Swarmanoid *that we intend to build will be comprised of numerous (about 60) autonomous robots of three types: eye-bots, hand-bots, and foot-bots.*

The *Swarmanoid* project is the successor of the *Swarm-bots* project, which consisted in study of the construction and the control techniques for a swarm of *s-bots*. The *s-bots* are able to self-assemble to a larger entity called *swarm-bot* in order to overcome the limitations of an individual robot. For example, connecting to each other in order to form a long line allowed the robots to pass over a trough in the environment. The *Swarmanoid* project will build on the results obtained during the *Swarm-bots* project.

2.2.1 Goals

As stated above, the goal of the *Swarmanoid* project is to build a swarm of robots which can operate in, and adapt to, a human-made environment. More specifically, the swarm consists of a heterogeneous group of robots, each with different capabilities, able to cooperate in order to solve complex tasks in 3-dimensional space. The project aims at constructing the following three types of robots:

Foot-bots are wheeled robots that drive on the ground. They have the ability to self-assemble and transport hand-bots by using a rigid gripper. They are specialized on moving on rough terrain and transportation tasks.

Eye-bots are flying robots, which can attach to the ceiling and observe the environment. They are specialized in sensing and analysing the environment, relaying relevant information to the other robots.

[1] FET-OPEN IST-022888, running from 1 October 2006 for 42 months.

Hand-bots are complex robots that can climb vertical surfaces and have two arms that allows them to manipulate objects. They are specialized in object manipulation and climbing. As they are not able to move on the ground by themselves, they need to self-assemble with foot-bots in order to move in the environment.

Additionally to the construction of the robotic hardware, important scientific contribution are:

- research and development for novel control algorithms for a self-organized swarm system;
- study and definition of distributed communication protocols.

These goals are required to form the *Swarmanoid* into a robotic entity that can act consistently and is distributed, robust, and scalable. In the following sections we give an overview over the robotic hardware and the simulation environment used to conduct the experiments presented in this work.

2.2.2 Robotic Hardware

In this section we give a short introduction to the *Swarmanoid* hardware. In the following, we explain common functionality shared by all robots. Afterwards, we give details about all three robots.

The base system of each of the three robots is the same. They are driven by an embedded Linux system, running on a low-energy processor (a Freescale i.MX31 with 533 MHz). The system can be accessed via USB, stores data on a 64 MB flash drive and uses 128 MB of main memory. All robots can be accessed by the operator using 802.11g wireless network, which can as well be used by the robots to communicate with each other.

Some sensors are common to two or all three of the robots. For example share all robots the same camera, although they are equipped with a different number of cameras and different optical devices (e.g., mirrors or lenses). Another important sensor is the *3D range and bearing sensor*[2], which is able to detect a robots range and bearing in 3D space using IR rays. Although this sensor is common to all robots, its capabilities might differ. For example, the foot-bot will be equipped with a full 3D system, while the eye-bot will carry only a 2.5D system because of weight limits.

[2] At the current stage of the project the range and bearing sensor is not finalized yet. It is therefore not known if there will be fully 3D or a reduced version (dubbed "2.5D") of the range and bearing sensor implemented on all robots. Nevertheless, we will refer to the sensor as the "3D range an bearing sensor" throughout the document.

Please note that the images shown below are schematics or photos of prototypes and might differ from the final robots[3]. More specifically, the scale of the three robots shown does not relate.

Foot-bots

Foot-bots are mobile robots driving on the ground, using a differential drive system. Their hardware is based on the *s-bot* from the *Swarm-bots* project (Mondada et al., 2004). Compared to their predecessor, foot-bots are larger, have more sensors, a modular design, and a hot-swappable battery. Figure 2.1 gives an overview over the hardware features of a foot-bot.

Apart from the common features listed above, foot-bots feature the following sensors and actuators:

- differential drive system (a combination of wheels and tracks);
- rotating turret;
- gripper, which allows to connect to other foot-bots or hand-bots;
- color LED ring for local communication;
- high-power light beacon for long-range signaling;
- upward facing camera for detecting eye-bots;
- omni-directional camera for detection of other ground robots in the environment;
- 24 IR proximity sensors around the robot;
- 8 IR proximity sensors for hole detection;
- 4 contact ground sensors;
- RFID ground reader;
- long-range distance scanner;
- torque and traction sensors in the wheels, gripper, and turret; to detect external forces the robot is subjected to.

Foot-bots are expected to run for 1 to 2 hours. A super-capacitor allows the exchange of the battery during the course of an experiment. At the current stage of the project, foot-bots are the only robots in the *Swarmanoid* which will feature a full 3D range and bearing sensor.

[3] More pictures of the robots and detailed pictures of specific subsystems, as well as videos of the prototypes in action, can be seen on the *Swarmanoid* website, `http://www.swarmanoid.org/`

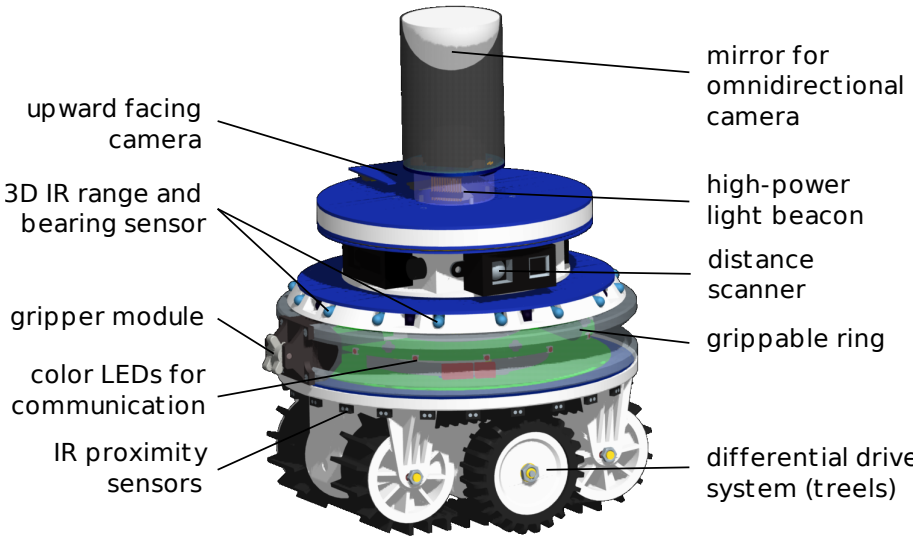


Figure 2.1: Foot-bot hardware design shown on the basis of a CAD-drawing of the current prototype

Eye-bots

Eye-bots are flying robots which have the ability to attach to a ferromagnetic ceiling. As they can stay passive while being attached, this ability extends the operation time of the robot. Because of their elevated position, eye-bot can have a aerial view on the environment. They are therefore equipped with several sensors that allow them to specialize in sensing tasks, for example a pan and tilt camera. Figure 2.2 gives an overview over the hardware features of an eye-bot.

Because of the nature of the robot, payload is extremely limited. Still, the robot features all of the common features listed above, although the range and bearing sensor is not fully 3D. Additionally, eye-bots feature the following sensors and actuators:

- main thrust rotor for lifting the robot;
- 4 steering fans in ducts used for maneuvering;
- color LED ring for local communication;
- upward facing camera for detecting other eye-bots;
- optical flow sensor;
- pan and tilt camera with a fish-eye lens for observing the environment below the eye-bot;
- rotating long-range distance scanner;
- altitude and air pressure sensor.

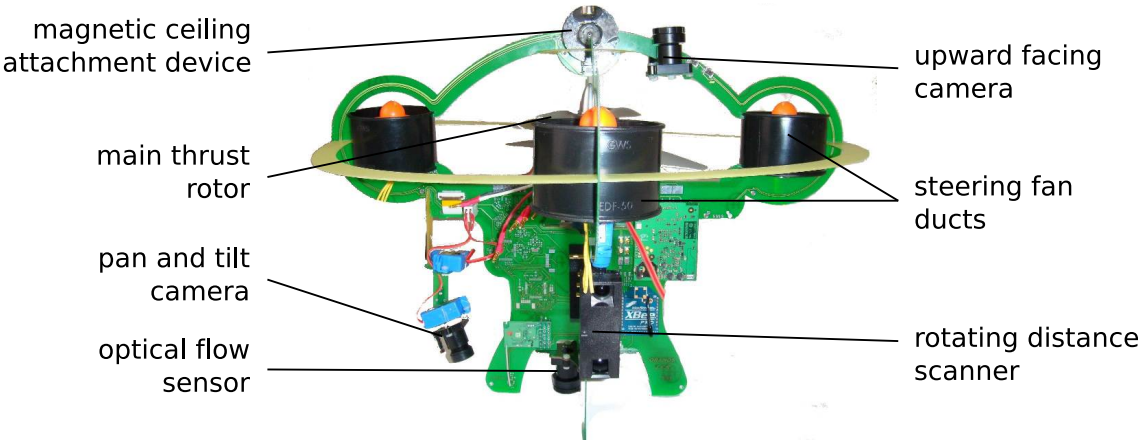


Figure 2.2: Eye-bot hardware design shown on the basis of a photo of the current prototype. Please note that the eye-bot shown is incomplete; missing parts are for example the 3D IR range and bearing sensor and the communication LEDs.

Eye-bots are expected to run for approximately 20 minutes of continuous flying. The operation time can be extended by attaching to the ceiling and operating in a power-saving mode without any mechanical devices active, which extends the run time to 1 to 2 hours. Eye-bots have interchangeable batteries, which allow for experiments that extend even beyond these limits.

Hand-bots

Hand-bots are the most unusual robots of the project. Hand-bots are able to move vertically on walls by launching a rope which attaches to a ferromagnetic ceiling. Subsequently to launching the rope, they can use their arms to climb a shelf or similar structure that supports them sufficiently. They are not able to move on the ground unaided, but have to be transported by foot-bots to their location of operation. Figure 2.3 gives an overview over the hardware features of a hand-bot.

At the time of writing, it was not yet finalized to which extend the range and bearing system is going to be integrated into the hand-bot. Nevertheless, a reduced 3D range and bearing system similar to the one used on the eye-bot is expected. Apart from this, the robot features all of the common features listed above. Additionally, hand-bots feature the following sensors and actuators:

- rope launcher with magnetic ceiling attachment device;
- arms that can move in 3D space (restricted to a cone in front of the robot);
- hands that can rotate;

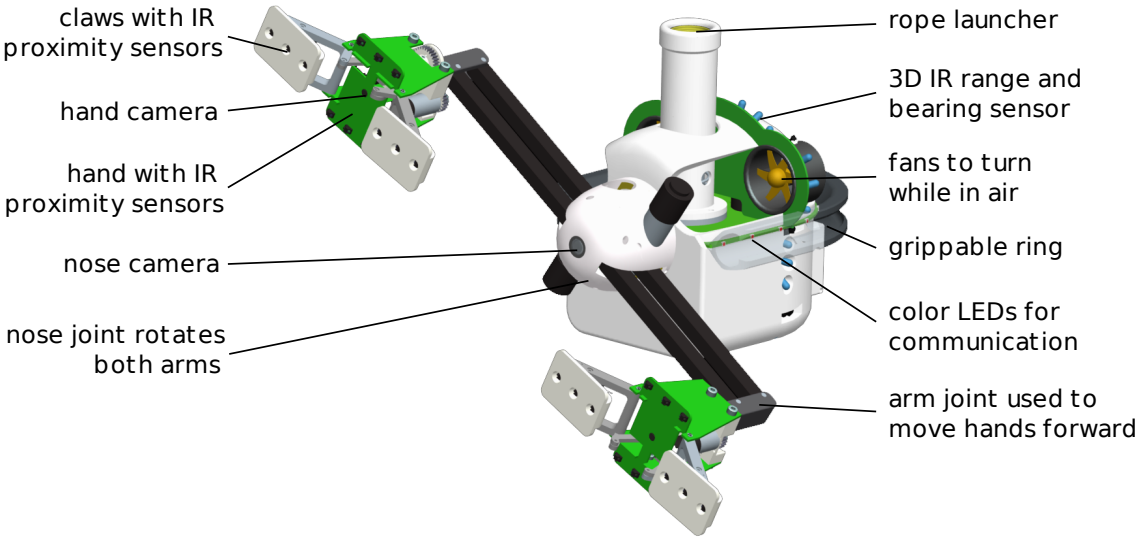


Figure 2.3: Hand-bot hardware design shown on the basis of a CAD-drawing of the current prototype

- claws that can grip objects;
- fans to turn the robot while suspended on the rope;
- color LED ring for local communication;
- a camera in the nose for observing the environment;
- cameras in the hands for object detection;
- IR proximity sensors around the robot;
- IR proximity sensors in the hands and claws for object detection;
- torque and traction sensors in all joints to detect external forces the robot is subjected to. to detect external forces the robot is subjected to.

The expected runtime of hand-bots is currently not known. Hand-bots feature an exchangeable battery.

2.2.3 Simulation Framework

Simulation-aided research and development is an important part to robotic research, as experiments on the real robots are very costly. Because of the characteristics and requirements of the *Swarmanoid* project, a special simulation environment has been developed. More specifically, the requirement of a simulation environment that allows a complex, heterogeneous swarm of robots to move in 3D space

while remaining computationally viable demanded a special design of the software used.

General Characteristics

The *Swarmanoid* simulation framework (dubbed "ARGoS", an acronym for *"Autonomous Robots Go Swarming"*) is a novel continuous-time, physics based simulation framework which is fully modular. Depending on the requirements of the researcher, different implementations of sensors, physics engines, and other parts can be chosen. Each implementation of an entity may simulate its physical counterpart on a different level of detail. This way, the simulation can be as accurate as required by the experiment while at the same time minimize computational overhead.

The simulation framework has been designed to provide the following core properties and functionalities (taken from the ARGoS documentation):

- Modeling of all the components of the *Swarmanoid* and of the environment according to multiple, selectable, levels of physical detail.
- Computational efficiency, to allow running simulations of complex indoor real-world scenarios including relatively large numbers of robots, and to ease the use of computationally-demanding algorithms for learning and control.
- Effective monitoring and visualization of the activities and performance of the single robots and of the entire *Swarmanoid*.
- Transparent migration of robot control code from the simulator to the real robots.
- High modularity, to facilitate reuse and integration of available software modules, permit independent code development from different contributors, and ease both the static and dynamic addition and removal of components and functionalities such as sensors, actuators, controllers, physics engines, and visualization modalities.

Simulation Experiments

As long as not stated otherwise, all the experiments presented in this work are carried out using the ARGoS simulation framework. For all the current experiments it is sufficient to use a purely kinematic model of the robots. Additionally, the current experiments only study robots acting in 2D-space. Therefore, the `2D kinematics` physics engine together with the specific sensors and actuators implementations are used.

3 Related Work

In the following chapter we will review some works related to our studies. In Section 3.1, we will review multi-robot foraging in general. Although this is a wide topic, it is somewhat important to our studies as it serves as our testbed for the methods proposed in the following chapters. Additionally, it is tightly intertwined with task allocation in general. Afterwards, we review the specific literature on task allocation, followed by a short review of the literature relevant to specific experiments. Because of the broad application of multi-robot foraging, the corresponding section partly overlaps with the following sections. This is desired, as each section reviews the publications from a different point of view.

3.1 Multi-Robot Foraging

The problem of foraging (or harvesting) objects or prey by a group of robots is one of the canonical testbeds for collective robotics. This is because it is relatively easy to model, has clear parallels in nature, and can be applied in many practical situations. It therefore provides us with a general method of comparison in the otherwise very diverse field of robotics. Especially in the case of swarm robotics, the foraging problem can provide a useful metric, as *"there are no generally accepted global criteria to evaluate a swarm system's performance"* (Rybski et al., 2008); although analytical models for specific systems exist (e.g., Lerman and Galstyan, 2002). Similar as with robotic systems themselves, the issue with foraging problems is that there exists a plethora of them (see Table 3.1 for a taxonomy). We will make a more simple distinction while discussing related works and distinguish only between two types of foraging problems: *simple foraging* versus *multi-foraging*.

In simple foraging, a group of robots have to collect objects of a single type, usually driven by an internal motivation like artificial hunger, energy balance or similar. Matarić (1997) proposed a behavior-based algorithm combined with a reinforcement learning technique to let robots learn how to collaborate in a "puck" foraging task. The robots were required to learn how to correlate appropriate conditions for each of the available behaviours in order to optimise the collective responses. Arkin et al. (1993) used a single-prey foraging task where the group collaborates through direct communication, similar to a social stimulus. They studied the impact of an optimal group size on task performance. Krieger and Billeter (2000) used a single-prey

→ **increasing complexity** →		
single robot	**vs.**	multiple robots
single storage locations	**vs.**	multiple storage locations
single source	**vs.**	multiple sources
open field environment	**vs.**	constrained environment
single prey object type	**vs.**	multiple prey object types
sparse sources	**vs.**	dense sources
homogeneous robots	**vs.**	heterogeneous robots
without communication	**vs.**	with communication

Table 3.1: A foraging problem taxonomy, inspired by Østergaard et al. (2001).

real-robot experiment, where harvesting is motivated by energy gain. In a similar experiment, Agassounon and Martinoli (2002) compared different threshold models in the same harvesting scenario. Labella et al. (2006) used a similar foraging task, motivated by energy gain, for studying a task allocation problem (see Section 3.2). Using a single-task foraging problem, Lerman and Galstyan (2002) studied the effect of interference on group performance (see Section 3.4). Several studies concerned the problem of interference reduction while using a simple foraging task (Lein and Vaughan, 2008; Liu et al., 2007; Shell and Matarić, 2006; Fontán and Matarić, 1996).

In multi-foraging, two or more types of objects are used to model more complex problems. In this case, comparisons are more problematic, as usually differing constraints (i.e., harvest a certain ratio of each object type) are imposed on the task. Jones and Matarić (2003) used a 2-task foraging problem to study different kinds (i.e., deterministic versus probabilistic) of non-adaptive threshold models. In their work, they imposed a ratio-constraint on the foraging task. Lerman et al. (2006) used a multi-foraging task for an extensive analysis of the underlying stochastic process and propose a framework for similar analysis. Similarly to works using simple foraging, Campo and Dorigo (2007) used a notion of the group's internal energy to allocate individuals to a multi-foraging task. Rosenfeld et al. (2005) used a 2-task foraging problem to develop and study an adaptive division of labour method in a large scale multi-robot system.

3.2 Task Allocation

Task allocation for multi-robot systems is a wide field, which can be divided in intentional and self-organized task allocation. Intentional task allocation relies on negotiation and explicit communication to create global allocations, whereas in self-organized task allocation global allocations result from local, stochastic decisions. A formal analysis and taxonomy that covers intentional task allocation has been proposed by Gerkey and Matarić (2004). Kalra and Martinoli (2006) recently compared the two best-known approaches of intentional and self-organized task allocation.

3.2.1 Intentional Task Allocation

Intentional task allocation is a common approach to the task allocation problem, as it follows a classical "engineering" philosophy. Thus, it is one of the most popular approaches. ALLIANCE is a prominent behavior-based algorithm of this class (Parker, 1998). It uses motivations inherent to the individual robot such as impatience and acquiescence to achieve adaptive task allocation. The method relies on a global state which has to be communicated to every individual in the group. Therefore, ALLIANCE works only for spatially dense groups and does not scale.

Another big group of approaches to the task allocation problem by using intentional cooperation is market or auction based approaches. In these approaches robots bid on tasks which are then assigned by an centralized auctioneer. Market based approaches have been thoroughly studied; we therefore limit ourselves to the best-known works in the field. TraderBots is a market-based approach proposed by Dias and Stentz (2003). TraderBots is inherently decentralized but uses centralized, informed sub-groups to improve efficiency. Another well-known market-based approach is MURDOCH, proposed by Gerkey and Matarić (2000). In MURDOCH, robots use a subscribe-publish architecture to assign tasks. The system monitors progress in short time intervals to detect and respond to faults. Kalra et al. (2005) give an extensive survey over existing market-based approaches.

Mclurkin and Yamins (2005) studied a subproblem of intentional task-allocation: the dynamic assignment of subgroups to subtasks. In their work, they assume that a global partition the problem is provided a-priori. They proposed three communication-based algorithms for assigning groups of individuals to these partitions.

3.2.2 Self-Organized Task Allocation

The amount of studies in self-organized task-allocation in considerably lower. The field can be regarded as being in its early stages, as most studies tackle simple problems without task interdependencies. Studies in self-organized task allocation are mostly based on threshold-based approaches, taking inspiration from division of labor in social insects. Krieger and Billeter (2000) as well as Agassounon and Martinoli (2002) where among the first to propose threshold-based approaches in multi-robot task-allocation. Labella et al. (2006) used threshold-based task allocation in a multi-foraging task. Similarly, Campo and Dorigo (2007) used a notion of the group's internal energy to allocate individuals to a multi-foraging task. Liu et al. (2007) studied a multi-foraging task while focusing on the influence of the utilisation of different social cues on the overall group performance. Recently, Magg and Te Boekhorst (2008) extended Labella's work to heterogeneous teams of robots, in which the (virtual) heterogeneity of the robots changes as a function of their specialization.

3.3 Task Partitioning and Division of Labour

Although the concepts of task partitioning, specialization, and division of labour are a key to the success of insect societies (e.g., see Jeanne, 1986; Ratnieks and Anderson, 1999), their use in robotic systems is less studied. To complicate matters, these concepts are usually intermingled with different problems and rarely studied individually. Labella et al. (2006) is one of the examples where division of labour through specialization can be observed in a multi-robot system. As they found out in their work, specialization occurs even in robots that were thought to be homogeneous—most probably through little differences in built and hardware quality. Adaptive division of labour is also studied by Jones and Matarić (2003), although the specialization among individuals seems to be arbitrary. Division of labour has also been applied for other problems like team formation (White and Helferty, 2005) and is usually inherent to heterogeneous groups of robots.

Spatial task partitioning (i.e., the partition of a large task in multiple smaller tasks which are essentially the same) has been extensively used for interference reduction (see the works on bucket brigading (e.g., Fontán and Matarić, 1996; Lein and Vaughan, 2008; Shell and Matarić, 2006). Nevertheless, to the best of our knowledge, adaptive task partitioning in terms self-organized task decomposition, as envisioned in this work, has up to now never been achieved.

3.4 Interference in Multi-Robot Systems

Interference has long been acknowledged as being one of the key issues in multi-robot cooperation (Goldberg and Matarić, 2003). Lerman and Galstyan (2002) devised a mathematical model that allows a quantification of the interference and its effect on group performance. Probably, the most thorough study was published by Goldberg (2001), who identified several types of multi-robot interactions. Goldberg notes that one of the most common types of interference is physical interference in a central area, for example the nest. This kind of interference results from resource conflicts, in this case physical space, and can be arbitrated by either making sure that robots stay in different areas all the time or by employing a scheduling mechanism to ensure that robots use the same space only at different times.

A simple method for reducing interference by using the first arbitration method mentioned is the so-called bucket-brigade: robots are forced to stay in exclusive working areas and to pass objects to the following robot as soon as they cross the boundaries of their area (Fontán and Matarić, 1996; Shell and Matarić, 2006). Recently, this has been extended to work with adaptive working areas by Lein and Vaughan (2008). Østergaard et al. (2001) compared the bucket-brigading approach in different environments and studied the influence of the group size on task performance. To the best of our knowledge, current works concerned with bucket brigading only studied the influence of interference due to obstacle avoidance. Other sources of interference (e.g., object manipulation) were never studied, although they might have a critical impact on the performance of any task partitioning approach. To quote Shell and Matarić (2006): *"If the cost of picking up or dropping pucks is significant [...], then bucket brigading may not be suitable."*

4 Task Allocation and Partitioning

In this chapter we first introduce the terms task allocation and task partitioning, as used in the context of this work. Afterwards, we will discuss the challenges in designing a swarm system capable of self-organized task allocation, as well as the advantages and disadvantages over classical approaches. Section 4.2 outlines a method that allows a swarm system to tackle complex tasks, followed by two example scenarios used to demonstrate the practical application of such a method.

4.1 Definition

The problem of assigning individuals to a task and its subtasks is a known problem, most commonly referred to as *task allocation*. Although the term task allocation is often used to describe the whole field concerned with task allocation problems in general, it actually only refers to the allocation of individuals to (sub)tasks. The equally important problem of *task partitioning* is referred to by different terms in the literature. We therefore define both task allocation and partitioning, as well as related terms, in the following.

4.1.1 Task Allocation

The term task allocation describes the problem of how to allocate available individuals to available tasks. A good allocation should pay attention to, among others, constraints like:

- task-interdependencies;
- limits on the number of individuals working on a task;
- specialization of individuals;
- location of tasks and individuals.

Additional complexity arises from the fact that the number and the nature of tasks might be unknown or change during execution time, requiring a flexible allocation method. Another big factor is the need for robustness and fault tolerance—for example the ability to handle failure of a specialized individual working on a key subtask. Thus, an optimal task allocation method should have, among others, the following characteristics:

- take into account characteristics of the individuals;
- allocate dynamically, and be able to reallocate, individuals;
- robustness and fault tolerance;
- generate optimal allocations;
- be decentralized;
- do not rely on global communication;
- scale with increasing number of individuals and tasks;
- produce exact, repeatable solutions.

It is obvious that designing a system that has all these characteristics is very complex, if not impossible. Thus, different approaches focus on different characteristics.

Intentional task allocation usually focuses on producing optimal allocations or close approximations thereof, which are exact and repeatable. Other characteristics like centrality and fault tolerance are usually second to these main features. Therefore, it is usually assumed that tasks and allocations are communicated by the means of a non-local communication channel. This makes the resulting task allocation problem a combinatorial optimization problem (Gerkey and Matarić, 2003). Most task allocations can be reduced to an instance of a well known problem, for example the Optimal Assignment Problem (OAP) from operations research. In practice, the complexity of finding an optimal allocation depends on the type of problem and can well be $\mathcal{NP}$-hard. Gerkey and Matarić (2004) defined a taxonomy for intentional task allocation problems, classifying them by type, specifying their complexity and to which type of known optimization problem they relate.

In systems that rely on self-organized behaviour, global behaviour emerges from the interaction of local sensing and communication with simple, often probabilistic rules (Bonabeau et al., 1999). In these systems neither a collective notion of a "goal", nor a centralized controller exists. Allocation of individuals to tasks has to be achieved by other means. Inspiration for designing such systems is often taken from social insects, as for example in Theraulaz et al. (1998). As these systems consist of many individuals that self-organize, they inherently posses the characteristics of decentralization and robustness. However, they provide, because of their stochastic nature, only an "average" behaviour and an approximate allocation. It is therefore

not feasible to treat the task allocation problem in these systems as a classical optimization problem.

In the context of this work we define task allocation as the problem of *finding the global allocation of individuals or groups of individuals to subtasks*, whereas the subtasks are not explicitly known to the individuals. It is implicit that the global allocation is more an average of the time-series of allocations, as the actual allocations might fluctuate because of the stochastic nature of the systems employed.

4.1.2 Task Partitioning

Task partitioning is a somewhat lesser known concept, although it can be found in most insect societies. It is sometimes defined as *"the division of a discrete task among workers"* (Ratnieks and Anderson, 1999). It is called task partitioning, as multiple individuals partition a large task among them, and basically allows to allocate groups to tasks rather than just single individuals.

We extend this notion for the purpose of this work to the following. Task partitioning is defined as the problem of *decomposing a global task into smaller (atomic) subtasks*, which might then be tackled by an individual or a group of individuals. The goal is to find methods for accomplishing this task partitioning (at least partly) in an automated, self-organized way. Task partitioning in a self-organized system might be dynamic and change constantly. Because of the dynamic nature of the problem and its heavy dependency on the characteristics of the global task, it is not clear yet to which extent a swarm can self-organize task partition, and to which extent partitions have to be pre-assigned by the designer.

4.1.3 Division of Labour

Division of labour describes the division of the workforce among the range of tasks encountered by a group. It is a requirement for parallel task processing and the basis for the formation of specialized individuals. Due to this specialization, division of labour might increase efficiency and facilitate the creation of specialists that are unable to do tasks different than the one they are specialized in. Thus, division of labour might lead to behaviourally heterogeneous populations. Division of labour and task partitioning are not mutually exclusive alternatives in the organisation of work (Jeanne, 1991). In fact, task partitioning can help to facilitate division of labour to a greater extend by partitioning larger tasks in smaller subtasks, which then can be tackled by specialized workers.

4.2 Self-Organized Task Allocation

How can effective task allocation be obtained in a self-organizing system? How can a swarm of individuals choose tasks and work on them, without having a notion of a "goal" or purpose? These questions are yet unanswered in the relatively unexplored field of self-organized task allocation. Present studies limit themselves to simple cases of task allocation, usually studying a two-task foraging problem (e.g., Labella et al., 2006; Campo and Dorigo, 2007). They usually employ threshold-based mechanisms, inspired by the intelligence of social insects, in order to solve relatively simple tasks. Although swarms that tackle complex tasks of different type, interdependencies, and locations, can be observed in nature, building artificial swarms with similar capabilities has yet not been accomplished. In the following, we propose a method for handling complex tasks in a self-organized swarm system—mainly a method for decomposing complex tasks in smaller and more simple subtasks. Afterwards, we discuss possibilities for self-organized decomposition and allocation, as well as give application examples for the proposed method.

4.2.1 Decomposing Complex Tasks

Most of the time, we can, upon closer inspection, identify smaller chunks of a task that seems initially to be very complex. These chunks are smaller pieces of work which can be tackled separately, but usually depend in some way on the global tasks and other chunks (e.g., order of execution). We refer to these chunks as subtasks, whereas identifying them is referred to as task partitioning.

We assume that the majority of the complex tasks can be broken down into two kinds of subtasks:

1. subtasks with sequential interdependencies;

2. subtasks with parallel interdependencies.

The mentioned interdependencies can be either spacial of temporal. Subtasks of the first type are any task that require in-order execution, i.e., any set of subtasks that have temporal or spatial dependencies that force them to be executed one after the other. An example is the transport of an object using two different kind of transportation methods. Subtasks of the second type are any task that allows parallel execution, but where the subtasks are somehow interlinked. An example is the retrieval of objects of two types, with the additional constraint of retrieving a specific ratio.

Task Dependency Graphs

By analysing a complex task and the dependencies among its subtasks, we can create *task dependency graphs*. Task dependency graphs represent a complex task in terms of smaller, atomic subtasks and are basically a flow-chart of task execution. By limiting subtasks to a small set of known types (i.e., the two types of subtasks mentioned above), and tackling these in a self-organized way, we hope to find a general method for accomplishing task allocation in complex tasks.

Let us define a *task* $\mathcal{T}$ as a directed acyclic graph (DAG) with a single root node and a single leaf node. Thus, a task can be written as $\mathcal{T}(S, D)$, with $S = \{\mathcal{T}_1, \dots \mathcal{T}_n\}$ being the set of *subtasks* and D being *dependencies* between these subtasks. Subtasks form the nodes of the graph, whereas dependencies form directed edges between the nodes.

As a task is defined as a directed acyclic graph whose nodes might be itself directed acyclic graphs, we can define a hierarchy of tasks, where the *rank* of a task $\mathcal{T}$ is equal to the number of its subgraphs. The task with the highest rank is called *global task* and represents the problem or mission. In the following, we explain the three possible types of subtasks we consider in this work. Figure 4.1 gives a graphical representation of them.

- A task is called *atomic* if the sets of subtasks and dependencies are empty, i.e., $\mathcal{T}(\emptyset, \emptyset)$. It can therefore not be decomposed further, and must be tackled directly.
- A pair of tasks $\mathcal{T}_i$ and $\mathcal{T}_j$ is called *sequentially dependent* when there exists a spatial or temporal dependency that requires sequential execution. This kind of dependency is denoted by $\mathcal{T}_\oplus = \mathcal{T}_i \oplus \mathcal{T}_j$.
- A pair of tasks $\mathcal{T}_i$ and $\mathcal{T}_j$ is called *parallel dependent* when the tasks can be executed in parallel, but their start and end node are the same. This kind of dependency is denoted by $\mathcal{T}_\ominus = \mathcal{T}_i \ominus \mathcal{T}_j$. Additionally to having the dependencies mentioned, tasks may have other constraints imposed upon them.

All tasks are in the powerset of all atomic tasks and all possible dependencies: $\mathcal{T} \in \mathcal{P}(\mathcal{T}'(S, D))$ with $S = \{\mathcal{T} \mid \forall \mathcal{T} = \mathcal{T}'(\emptyset, \emptyset)\}$ and $D = \{\oplus, \ominus\}$.

4.3 Main Idea of the Proposed Method

The main idea of the proposed method is to partition a complex task into multiple, smaller subtasks, which are less complex and of known type. These tasks can therefore be tackled by a swarm in a self-organized way, which is, because of their

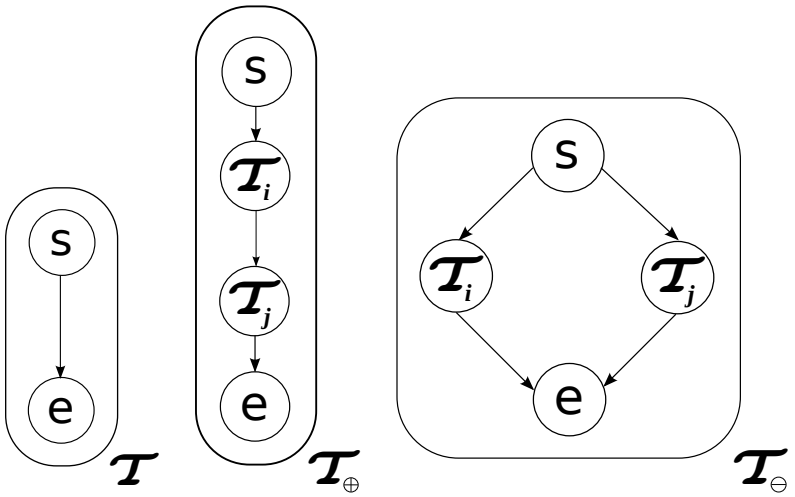


Figure 4.1: Possible types of subtasks. Arrows indicate the flow of execution, while nodes are subtasks. `s` and `e` nodes represent task execution start and end, respectively. (a) A atomic task. (b) A task that consists of two subtasks with a sequential interdependency. (c) A task that consists of two subtasks with a parallel interdependency.

simple nature, a lot easier than treating a complex task as a monolithic problem. Thus, the swarm should be able to decompose a complex task into smaller subtasks represented by a task dependency graph as introduced above. Determining whether this decomposition can be accomplished in a self-organized way is yet an unresolved issue. Having an indirect notion of the task dependencies, the swarm members should be able to allocate themselves to the tasks in the right order and quantity.

4.3.1 Research Directions

While the main idea of this work deems to be relatively simple, there are many unresolved issues. For example, it is not clear yet if it is reasonable to assume that task decomposition can be accomplished in a self-organized way, and to which extent the designer's knowledge has to be part of the system. Therefore, there exist a couple of main research directions rising from the analysis above.

How can we create a self-organized system, that is able to

1) partition complex tasks by itself;

2) make the transition between dependent subtasks;

3) allocate its individuals to these subtasks; and

4) tackle the two types of non-atomic subtasks mentioned above?

Finding answers and solutions to all this questions is a daunting task, which is well suited to provide topics for several researchers.

Scope of this Work

In this first work, we will focus on the last two questions of the ones listed above: How can we create a self-organized system that allocates automatically to all required subtasks, and how can a swarm tackle subtasks of the two mentioned types? We will use the decomposition method outlined before, with a partitioning set by the designer. In the rest of this work we will therefore study how to achieve self-organized task allocation to subtasks which are either sequentially or parallel dependent.

Limits of the Approach

Because of the simplicity of the main idea of this approach, there are many limits to it. Most important, it is yet not clear to which extent a self-organized task partitioning is possible, and to which extent the system designer has to integrate his knowledge into the system. Other limitations might be tasks that have different types of interdependencies, or incompatible task goals. Additionally, the proposed method does not allow for alternative ways of tackling a complex task and defines only a rather rigid way of execution.

4.4 Application Examples

In the following section we demonstrate the application of the task decomposition method outlined above by using two examples. First, we sketch a possible scenario for the *Swarmanoid*, in which the swarm solves a complex, distributed task using self-organized task allocation. The second example is the well known "rescue"-mission scenario, which demonstrates the application of the method to a real-world problem often used in the robotics literature.

4.4.1 A *Swarmanoid* Scenario

The following scenario is embedded in the *Swarmanoid*. The global task is a multi-foraging task. The swarm has to retrieve multiple objects of different types (i.e., building material) in a given ratio. The objects are clustered by type, but the clusters are distributed randomly in the environment. The environment can be large and

might include obstacles, which limit the vision of the foot-bots. For example, in a warehouse filled with shelves each different object type is stored on a different shelf. Consequently, the swarm has first to locate the shelves and identify the appropriate ones with the help of eye-bots. Next, the swarm has to collect a certain ratio of each object type and has to transport it to a central storage location.

Objects are collected from a shelf by one or more hand-bots, depending on the density of the objects on the shelf and the required quantity of objects. After an object is deposited on the floor by a collecting hand-bot, it has to be transported to the central storage location. Foot-bots are used as markers at the bottom of the shelf for two reasons: first, to coordinate distributed collection of objects on a single shelf; and second, to coordinate the transfer of collected objects to transporting robots. Object transport requires cooperation of several robots, as no single robot has the capability of moving on the ground while grasping an object. Thus, object transport is accomplished by a self-assembled robot which form a more powerful, symbiotic entity. In this case, the entity consists of several foot-bots and one hand-bot. The hand-bot is used to grasp the object, while the foot-bots move the hand-bot on the floor. Path-formation similar to the work published by Nouyan et al. (2008) will be employed by the eye-bots in order to lead foot-bots from the central storage location to each of the shelves. In the central storage location, objects are simply stored in a pile. Figure 4.2 shows a diagram of the overall task (top), and a screenshot taken from the ARGoS simulator (bottom).

Task allocation aspects of this scenario

The swarm has to form a so called "delivery line" from the central storage location to each of the shelves. Each delivery line represents the subtask of collecting one object type. These subtasks can be executed in parallel, but have the constraint of a certain required ratio between the object types. Each delivery line itself consists of two subtasks: object collection and object transportation. Because of the nature of the task, these subtasks have to be executed sequentially. From the task allocation point of view, the global task is partitioned into parallel subtasks, which themselves are partitioned into two sequential subtasks. Individuals have to be allocated to each delivery line, and inside the delivery line to either collect or transport objects. A possible reason for this might be that collection of objects on the shelf is faster than object transportation on the ground (e.g., because of the a long distance between the nest and the shelf). Thus, the swarm might use more robots for transportation than for gathering.

We consider a task with two object types, in the following referred to as "blue" and "red" objects. Therefore, the global task can be represented as a task consisting of two subtasks with parallel interdependency, $\mathcal{T} = \mathcal{T}_{\text{blue}} \ominus \mathcal{T}_{\text{red}}$. These subtasks in turn consist of two subtasks with a sequential interdependency, $\mathcal{T}_{\text{blue}} = \mathcal{T}_{\text{c}} \oplus \mathcal{T}_{\text{t}}$ (and $\mathcal{T}_{\text{red}}$ accordingly). The `collect` and `transport` subtasks ($\mathcal{T}_{\text{c}}$ and $\mathcal{T}_{\text{t}}$, respectively) are

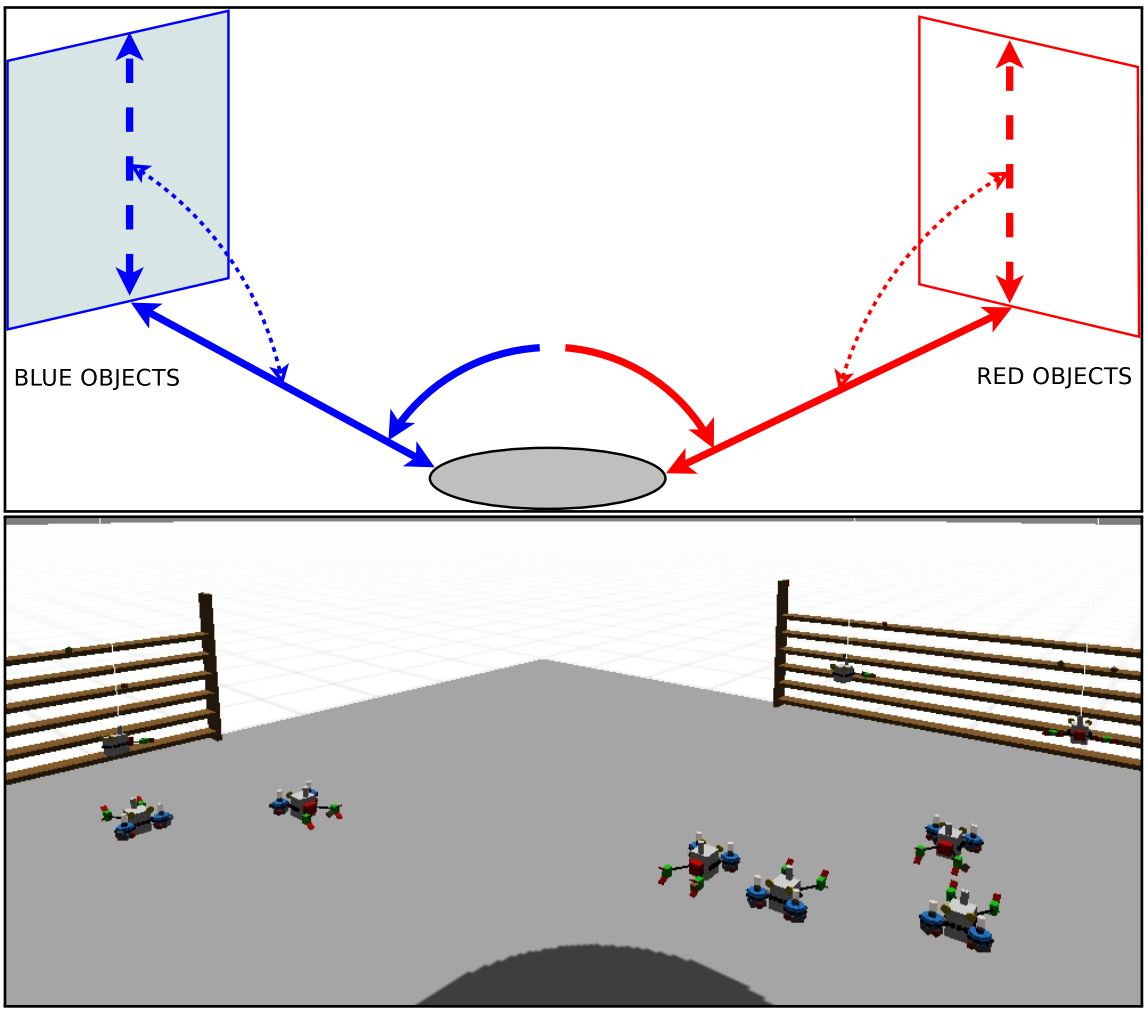


Figure 4.2: A task allocation experiment. The top image shows a schematic view of the experiment shown at the bottom (screenshot from the ARGoS simulator). The grey circle in the center represents the central storage location where all objects have to be transported to. There are two delivery lines, one for "blue" objects and one for "red" objects. Individuals can switch from one delivery line to the other. Inside each line, individuals can switch from one subtask (transporting on the ground) to the other subtask (collecting on the shelf).

atomic tasks and can be tackled directly by the swarm. Thus, the global task has a rank of 2. A graphical representation of this task as a task dependency graph can be seen in Figure 4.3

The proposed experiment relies on several submodules which are by themselves complex and interesting research topics. For example, shelf climbing using the hand-bot is a complex control problem involving close coordination of several joints. Other problems to research include collective motion of a self-assembled entity, path-

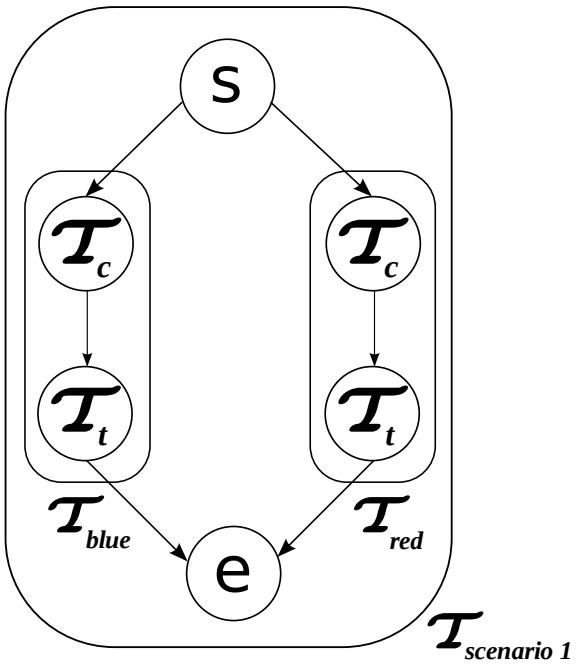


Figure 4.3: Task dependency graph for the *Swarmanoid* task allocation scenario depicted in Figure 4.2. There are two tasks in parallel, the `blue` foraging task ($\mathcal{T}_{\text{blue}}$) and the `red` foraging task ($\mathcal{T}_{\text{red}}$). Each task consists of a `collect` and `transport` task ($\mathcal{T}_{\text{c}}$ and $\mathcal{T}_{\text{t}}$, respectively).

formation using a swarm of eye-bots as well as navigation guided by eye-bots. These submodules are not part of this work as they are well outside the scope of task allocation. We rely on the *Swarmanoid* project partners to supply these submodules.

4.4.2 A Rescue-Mission Scenario

The following scenario is a "rescue"-mission scenario. We will use it here as a demonstrator for our method, as it is a common task of arbitrary complexity. Together with the "space"-mission scenario, the rescue-mission is one of the most prominent application examples in the robotics literature (see for example the Robocup Rescue Leagues[1]). Naturally, this scenario is more hypothetical, as large-scale robotic space missions are currently technologically unfeasible, and the environmental and task parameters were chosen arbitrarily. Nevertheless, this scenario serves well as a demonstrator for a possible application of the method.

In this scenario, we assume a swarm of homogeneous, wheeled robots which can communicate by local radio and have the ability to grip and pull objects. The envi-

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

ronment of the scenario consists of a site of some catastrophe or accident, for example a major train crash. The goals of the swarm in this scenario are:

- explore the environment;
- locate and identify human casualties;
- identify and overcome obstacles; and
- rescue humans (if possible).

The system operator need to set a goal or at least the environment boundaries upon mission start, so that the swarm knows in which space it needs to operate. The swarm should start by exploring the environment, finding human casualties along the way and identify possible obstacles that hinder their rescue. Possible obstacles might be for example rubble blocking the way or holes in the ground. The swarm members might have to work together in order to overcome these obstacles, and different obstacles might need different numbers of individuals working together. Finally, the swarm should rescue human individuals by dragging them along the cleared paths.

Task allocation aspects of this scenario

The swarm has to start exploring the environment. As soon as a human is found, the swarm determines the rescue path and obstacles on the way, while in parallel continuing to explore. Depending on the state of health of the human, different human casualties may have different priorities for rescue. Depending on this priority, the robots will clear the way of obstacles in sequential order or in parallel. By doing so, the robots are able to allocate the necessary amount of robots to each human in order to find and rescue every casualty in an efficient way. After the robots cleared the way, they will try to rescue the human. After completion, they will join forces with other robots working on other tasks, for example exploration.

The global task can be represented as a task consisting of two main subtasks with parallel interdependency, $\mathcal{T} = \mathcal{T}_{\text{explore}} \ominus \mathcal{T}_{\text{rescue}}$. The `explore` subtask is atomic, while the `rescue` subtasks in turn consist of three subtasks with a sequential interdependency, $\mathcal{T}_{\text{rescue}} = \mathcal{T}_{\text{locate}} \oplus \mathcal{T}_{\text{free}} \ominus \mathcal{T}_{\text{transport}}$. The `locate` and `transport` subtasks are atomic tasks and can be tackled directly by the swarm. The subtasks of the `free` task can be, depending on the state of health of the human, either executed in parallel or one after the other, with an atomic subtask $\mathcal{T}_{\text{clear}}$ for each obstacle. Thus, the global task has a rank of 3. A graphical representation of this task as a task dependency graph can be seen in Figure 4.4.

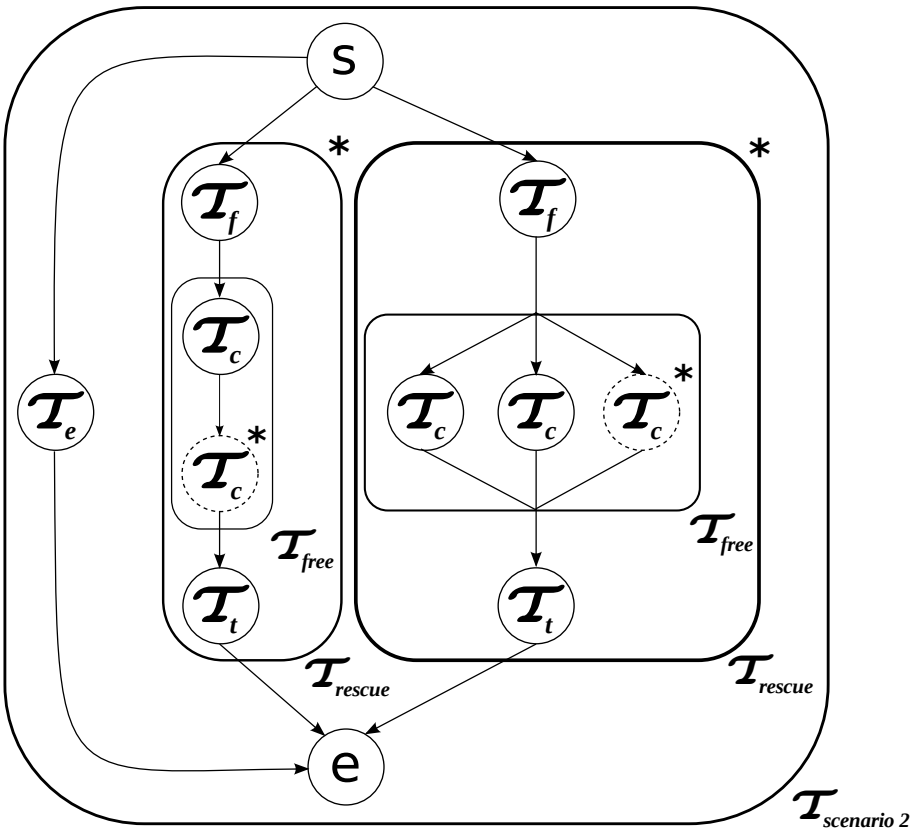


Figure 4.4: Task dependency graph for the rescue-mission task allocation scenario. There are three tasks in parallel, the `explore` task ($\mathcal{T}_\mathrm{e}$) and two `rescue` tasks ($\mathcal{T}_\mathrm{rescue}$). In the `rescue` tasks the robots first have to `find` the human ($\mathcal{T}_\mathrm{f}$), then `free` the way ($\mathcal{T}_\mathrm{free}$) and `transport` the human casualty back ($\mathcal{T}_\mathrm{t}$). Freeing the way includes clearing the path of all obstacles ($\mathcal{T}_\mathrm{c}$). `clear` tasks can be executed in sequential (left), or parallel (right) order, depending on the state of health of the human. Of course, working in parallel is faster but requires more robots. `*` symbolizes the cardinality of the `rescue` and `clear` subtasks, as there can be arbitrarily many of them.

Of course, the proposed scenario is rather hypothetical and more on a though-experiment. Nevertheless, we think it demonstrates well the flexibility of the framework and possible real-world counterparts to complex task allocation problems.

5 Sequential Task Allocation

In this chapter we present experimental work undertaken to study self-organized task allocation in the case of sequentially dependent tasks. Sequentially dependent tasks are one of the building blocks of the method for self-organized task allocation presented in the previous chapter. Therefore, we need to study extensively what the properties of such tasks are and how we might tackle these tasks in a self-organized way. More specifically, we first study an abstract version of a two-task problem. Afterwards, we try to transfer the knowledge gained in the abstract case to several, more specific applications with a practical background.

Please note that this chapter reports finished, ongoing, and future work. The studies are presented in their logical order and not their temporal order. Thus, ongoing work including preliminary results is presented in Section 5.2, whereas the following two Sections 5.3 and 5.4 describe future works. The final section of this chapter will report findings in a specialized case of task partitioning, and represents the only published study of this work (Pini et al., 2009).

5.1 Common Methods

We use the foraging problem, one of the canonical testbeds for collective robotics (see Section 3.1), as a base for our studies. In our experiments, a swarm of robots has to harvest prey objects from a source area and transport them to a home area. By spatially partitioning the environment, the global foraging task is partitioned into two subtasks: 1) `harvest` prey objects from a harvesting area (called `source`) and 2) `store` them to a home area (called `nest`). Robots working on the first subtask harvest prey objects from the source and pass them to the robots working on the second subtask, which store the objects in the nest. These subtasks have a sequential interdependency in the sense that they have to be performed one after the other in order to complete the global task once: delivering a prey object to the home area.

Thus, the task can be represented according to the the method introduced in Chapter 4 as follows. There are two atomic subtasks, $\mathcal{T}_{\text{harvest}}$ and $\mathcal{T}_{\text{store}}$, with the global subtask being $\mathcal{T}_{\text{sequential}} = \mathcal{T}_{\text{harvest}} \oplus \mathcal{T}_{\text{store}}$. See Figure 5.1 for a graphical representation of the task dependency graph.

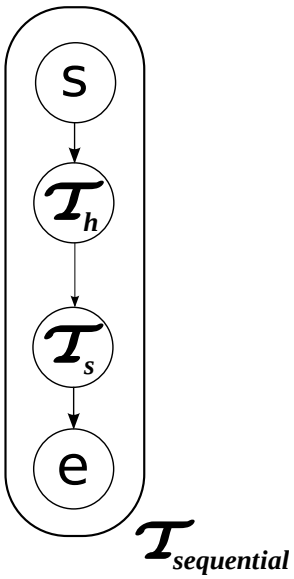


Figure 5.1: The general task dependency graph of the sequential task allocation problem studied here, according to the method presented in Chapter 4. First, objects have to be harvested (subtask $\mathcal{T}_\mathrm{h}$) and transported to the exchange zone. Afterwards, they have to be transported to the nest and be stored there (subtask $\mathcal{T}_\mathrm{s}$).

In this chapter, we limit ourselves to a harvesting task that is pre-partitioned by the designer into two subtasks with a sequential interdependency. Robots can decide to switch from one subtask to the other, thus creating a task allocation problem: individual robots have to be allocated to subtasks and different allocations yield different performance. As a prey object has to be passed directly from one robot to the other, a robot usually has to wait some time before passing a prey object to, or receiving a prey object from, a robot working on the other subtask. This waiting time can therefore give an indication of the allocation quality for the respective subtask: if the waiting time is very long, there might not be enough robots allocated to the other subtask. Thus, the robots can use this waiting time to decide whether to switch subtask or not. Ideally, the waiting time should be the same for the two subtasks in order for the system to reach a stable state and deliver optimal performance.

5.1.1 Environments

We study sequential task allocation in a common scenario: harvesting objects in environments which are spatially partitioned into two parts. In these environments, the nest is marked by a light source that can be perceived by all robots, thus providing directional information. The parts in which the environment is partitioned

Figure 5.2: General representation of the environments used for studying the sequential task allocation problem. Prey objects are transported from the source on the left to the nest on the right (gray stripes). The robots hand over the objects in the exchange zone (black stripe). The light source is marked with "L".

are the part containing the source, which is located on the left, and the part containing the nest, which is located on the right side of the arena. We refer to the two sides of the arena as *harvest area* and *store area*, respectively. The *exchange zone* is located between these two areas. Robots working on the left side, called *harvesters*, gather prey objects in the source and move them to the exchange zone, where they pass them to the robots working on the other side. These are referred to as *storers*: their role is to transport prey objects to the nest and store them there. The nest, the source, and the exchange zone can be detected through environmental cues (ground color). Figure 5.2 gives a graphical representation of the general layout of the environments used in the experiments.

5.1.2 Controllers

Although each of the individual experiments presented in the following sections requires extra methods or specialized behaviours, the overall structure of the controller is the same in all experiments. As we present the controller only on a fairly abstract level, we will present it once for all experiments. Individual specifics will be explained in the each experiment's subsection, titled "Controller Specifics".

All the robots share the same, hand coded, finite state machine controller depicted in Figure 5.3. The controller consists of two parts, each corresponding to a possible subtask a robot can perform. Gray states refer to the `harvest` subtask, white states to the `store` subtask. Since all the robots start in the harvest area, their controller is initially set to perform anti-phototaxis. In this way they will reach the source, where they can start retrieving prey objects. The behavior of each robot is a function of the task it is performing. Harvesters not carrying a prey object move towards the source, where they can find prey. Harvesters carrying a prey object, move to the exchange zone and wait for a free storer. Upon arrival of a storer, the harvester

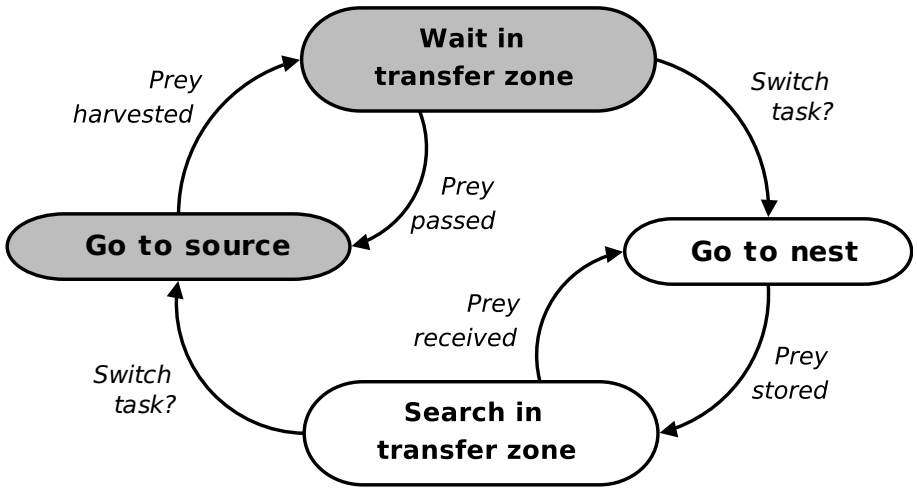


Figure 5.3: Simplified state diagram of the controller of the robots for all the sequential task allocation experiments. Gray states belong to the harvest task, white states to the store task. The obstacle avoidance state has been omitted for clarity, as it is applicable in all states.

passes the prey object to it. Storers carrying a prey object move towards the nest, where they can deposit the object. Storers not carrying a prey object head to the exchange zone and search for a harvester with a prey object. Robots can detect other robots carrying a prey on the basis of the color of their LED ring. While moving, each robot avoids obstacles (walls and other robots).

Task switches can occur: a harvester carrying a prey object can decide to become a storer, and a storer not carrying a prey object can decide to become a harvester. It depends on the different controller types how the decision of switching task is achieved, but usually some kind of threshold mechanism is employed. For example, the robots controller could employ a deterministic threshold model: if a robot remains in the transfer zone for a period of time that is bigger than its threshold without passing or receiving prey objects, it switches its task.

5.1.3 Research Questions

As the partition of the task has been fixed in the experiments presented in the following, we focus on the question on how to accomplish self-organized task allocation in a task with sequential interdependencies. In a self-organized system, there exists no central controller that decides which allocation might be the best. Instead, the global allocation emerges from the local decision of each individual of the group. Thus, the main question is: how can an individual make the decision in which task it should work? Or, more specifically to this task: when should an individual switch between the two subtasks?

The studies presented in this chapter focus on this question, and how it can be answered in a general and environment-independent way. Without using explicit communication, the robots have only the following estimates about the environment and tasks they are working on:

- The round-trip-time (t_{RTT}), describing the time they need to complete one subtask,
- the task switching delay (t_{TSD}), describing the cost of switching between tasks, and
- the waiting time (t_w), describing the time they wait in the exchange zone.

It is clear that an individual can only have an estimate of these environmental cues as experienced by itself. Thus, the individuals need some type of averaging or memory technique to keep track of estimates. Depending on these choices, the system needs a certain period of time before each individual has a stable estimate of these information, and needs therefore a certain period of time to stabilize in itself.

The main research question is therefore: How can an individual make the decision to switch tasks only based on the information it has, and how can this decision be independent from the environment it is working in? Moreover, we want to study the relationship between the information described above and their influence on the decision making process. Additionally, we will study applications of the proposed methods in different practical problems robotic designers are faced with.

5.2 Abstract Experiments

In the following experiments, we omitted a practical justification of the task as given in the other experiments presented in the following sections. The rationale behind this choice is that the environment and the choice of the problem parameters might be limited by such a practical application. We therefore study the problem in several abstract experiments and transfer the gained knowledge to the applications afterwards.

5.2.1 Methods

Simulation and Analysis

In case of the abstract experiments, we use a 3-tier approach to analyse and simulate the system. This approach, utilizing simulations on different levels of detail, allows

us to simulate and analyse larger parameter spaces than physics-based simulation only.

Mathematical Model We use a mathematical model to represent the task in an abstract way. This allows us to study the task analytically. The main advantage of this is the possible identification of theoretical optima and faster analysis of very large parameters spaces. Although the mathematical model is currently being designed, it has neither been finished nor validated against the simulation results. Thus, we will not include it in this work.

Abstract Simulation We used an abstract, agent-based simulator for simulating the system. Although the simulation is based on an individual representation of each robot, the agents are not embodied. This allows us to test algorithms and study general system dynamics in large populations without the high cost of a physics-based simulation. The abstract simulation has been implemented in Java using the Repast agent-based simulation framework (Collier, 2003).

Physics-based Simulation The most detailed experiments were carried out in a custom simulation environment that models geometries and functional properties of simple objects and robots, namely the ARGoS simulator of the *Swarmanoid* project (see Chapter 2 for more information). Our robots' model is purely kinematic. Prey objects are simulated as an attribute a robot can posses and not as physical entities. Although the experiments are conducted in simulation only, the simulated robots have a real counterpart: the foot-bots from the *Swarmanoid* project. For a detailed description of the hardware, see Section 2.2.2.

The simulated robots are of round shape, with a diameter of 0.116 m. Each of them is equipped with 24 infrared proximity sensors, used to perceive obstacles up to a distance of 0.15 m. Eight ambient light sensors can be used to perceive light gradients up to a distance of 5 m. The robots are equipped with 4 ground sensors used to perceive nest, source and exchange zone. A LED ring is used to signal when a prey object is carried. An omnidirectional camera allows the perception of LEDs in a circle of radius 0.6 m surrounding the robot. A uniform noise of 10% is added to all sensor readings at each simulation step. The robots can move at a maximum speed of 0.1 $^{m}/_{s}$ by means of a differential drive system.

Environments

The experiments are run in two different environments (see Figure 5.4). Both environments are 5 m long and 2 m wide. In the first environment (Figure 5.4a), the exchange zone is located exactly in the middle (rendering both areas 2.5 m long).

Figure 5.4: Depiction of the environments used in the abstract experiment. (a) *Symmetric* environment used in the first experiment. (b) *Asymmetric* environment used in the second experiment. The gray stripes are the source (left), and the nest (right), each 0.5 m deep. The black stripe is the exchange zone, that is 1 m deep. The light source is marked with "L".

Therefore, this environment is referred to as the *symmetric* environment. In the second environment (Figure 5.4b), the exchange zone is located in a way that the overall area is partitioned into 2/3 for the harvest area and 1/3 for the store area (rendering them 3.33 m and 1.66 m long, respectively). Therefore, this environment is referred to as the *asymmetric* environment.

In both environments, the exchange zone has a width of 1 m, whereas the nest and the source are each 0.5 m wide. The area of the environments is 10 m^2. As the overall area is the same in the two environments, the same group size results in the same robot density. Thus, results are comparable across the two environments.

Controller Specifics

In this experiment, we use a controller that switches task based on a probabilistic threshold θ. Thus, for each time step a robot is waiting in the exchange zone, it draws a uniform random number $p \in [0, 1)$. If $p < \theta$, the robot switches its subtask.

The robot's waiting time is a function of the average time the robots working in the other subtask need to complete their task. The task-completion time of a robot depends on two factors: 1) round-trip-time (i.e., distance to travel) and 2) time lost due to interference. Thus, the robot's threshold θ is a function of the round-trip-time and the interference of the robots in the other subtask. Therefore, the optimal task switching threshold depends on the task (i.e., time to harvest a prey object) and the environment (i.e., distance between the source and the nest). As the parameters of the environment are not pre-programmed into the robots, determining the optimal threshold can be a complex problem.

Experiments

The goal of the experiments is to uncover the relationship between the three environmental cues mentioned before, namely t_{RTT}, t_{TSD} and t_w, and their influence on the threshold. This information should be used to define a method for deciding when to switch task. As we base our controller on a probabilistic threshold, we therefore seek a general function how to determine this threshold θ based on the estimates each individual has. This threshold function should ideally be environment-independent and adaptive.

At time $t = 0$, the robots are randomly placed in the harvest area. The experiments run for $t_{max} = 18,000$ time steps (a simulated time of one hour, with a time step length of 200 ms). All experiments have been simulated with $N = 10$ robots.

Metrics

In order to quantify the influence of interference, we measure the *group performance* P by the number of prey objects collected by the swarm at the end of the experiment. Other metrics include time of convergence after experiment start and after changes in the population.

Another important metric is the ratio r between storers and harvesters the system converges to. We will compare this ratio to its theoretical optimal value r_t.

5.2.2 Preliminary Results and Discussion

In this section we present the ongoing experiments and their preliminary results. Most of them have been obtained by testing a hypothesis in the faster abstract simulation and were then transferred to the physics-based simulation.

Adaptive Threshold Function

We propose to set the threshold θ in dependence of the robot's waiting time, using a sigmoid function. Thus, task switching probability is initially nearly zero, while increasing over time and being certain after a long time span. We use a basic exponential function for creating a sigmoid curve,

$$p = \frac{1}{\left(1 - e^{\theta(t_w)}\right)} \,, \tag{5.1}$$

where $\theta(t_w)$ is a threshold function of the waiting time defined as follows:

$$\theta(t_w) = \frac{w_t - t_s}{a} - c \,, \tag{5.2}$$

where t_s is the waiting time at which the probability starts to rise, a influences the steepness of the curve and c is a calibration factor. The parameters a and c can be determined experimentally and should be fixed. The problem arises from t_s, which is dependent on the t_{RTT} of the other subtask. Figure 5.5 gives an example for such a function.

We performed a simple experiment in which the parameter t_s was set to the average waiting time experienced by the respective individual. That way, the probability of switching increased as soon as the robot waited longer than expected. The result of this preliminary experiment is shown in Figure 5.6. As it can be seen, the convergence speed of the swarm to a stable allocation is clearly influenced by the parameters. This means that the system is sensible to these parameters, which should therefore be set carefully. Additionally, we can observe that the system converges to a stable allocation, although the time of convergence is rather high. All experiments have been performed in the *symmetric* environment.

The performance P of each subtask is sequentially dependent on the performance of the other. The optimal allocation should be reached when both subtasks perform at an equal level, thus maximizing the flow of prey objects. Both subtasks perform equally when the waiting time t_w is equal among all robots across the subtasks.

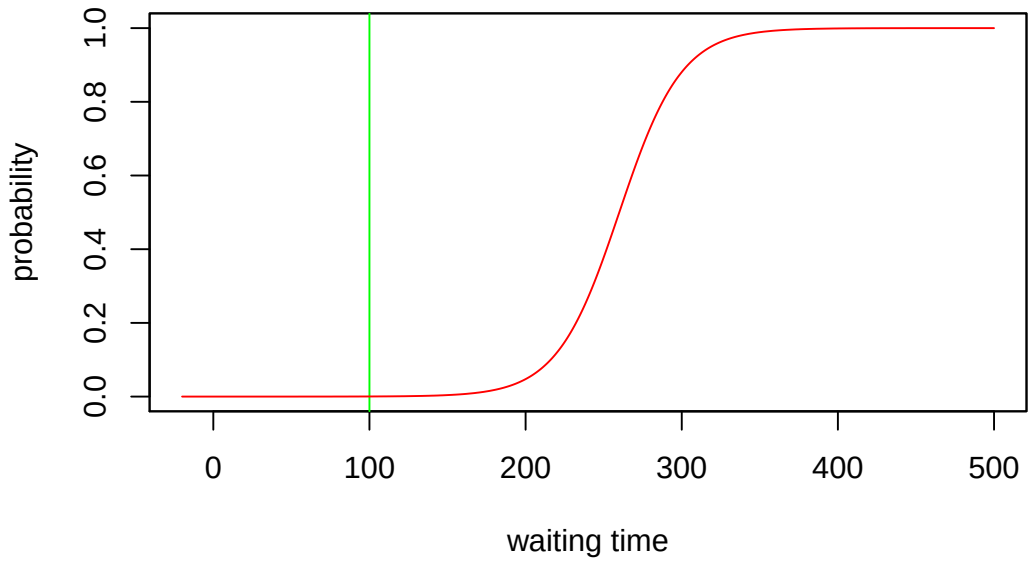


Figure 5.5: An example for the switching probability when using the threshold function $\theta(t_w)$, with $t_s = 100$, $a = 20$ and $c = 8$. The vertical line marks t_s, the point at which the probability to switch task starts to rise.

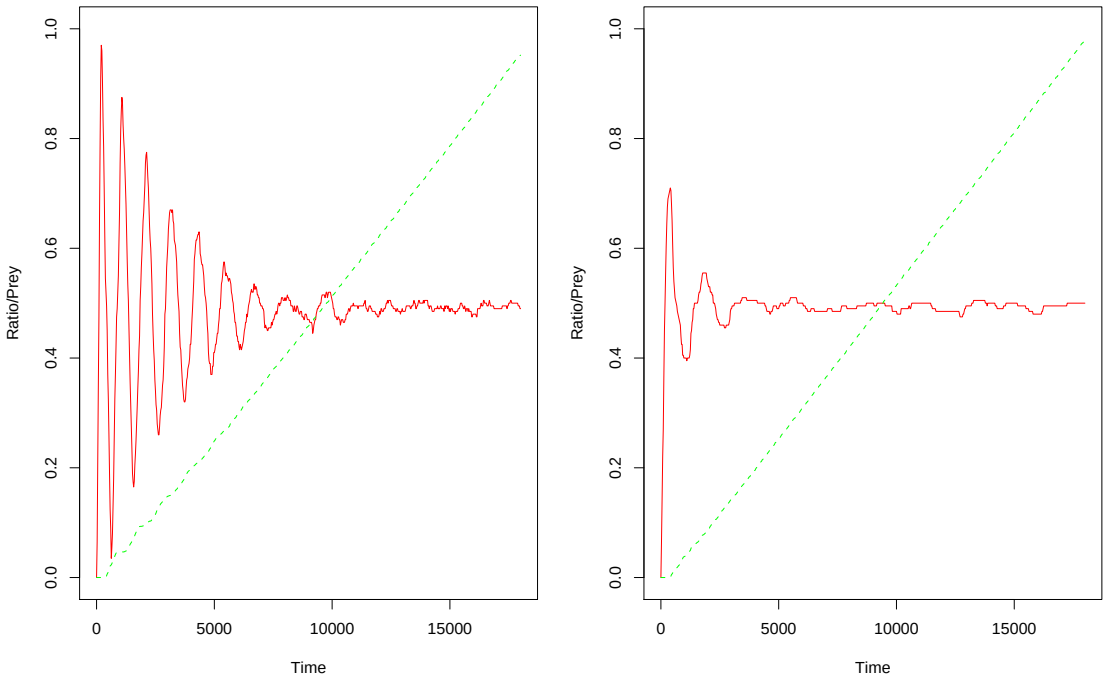
Thus, the parameter t_s should be set according to the ratio of the waiting times in the two subtasks:

$$t_s = \frac{\text{avg}(t'_w)}{\text{avg}(t_w)} \,, \tag{5.3}$$

with t'_w being the average waiting time experienced by the individual in the other task. When the current task is performing worse than the other (i.e., t_s is greater than 1), robots working in this task should continue working. On the other hand, robots working in the other task (t_s is lesser than 1), robots should switch to the lower performing task with higher probability. By combining this simple method with the curve shown in 5.5, we can generate a task switching function in which switching probability increases according to the difference of performance between the two tasks. Up to now, this hypothesis has has neither been implemented nor tested in physics-based simulation.

5.2.3 Future Work

One of the next steps will be the completion of the mathematical model mentioned above. We can use it to identify the theoretical optimum when concerning allocation and performance optima. This way, we can evaluate the existing system on a neutral basis.



(a) (b)

Figure 5.6: Convergence speed for different parameters of the threshold function. The solid line gives the ratio r between harvesters and storers, whereas the dotted line gives the percentage of gathered prey objects. Time is given in simulation time steps (1 step = 200 msec). (a) Convergence speed for $a = 10$ and $c = 5$. (b) Convergence speed for $a = 30$ and $c = 10$.

Additionally, we will continue the study concerning the threshold function and its parameters. We will implement and test the method proposed in Equation (5.3) in the abstract simulation, and, if successful, the physics-based simulation. We will try to eliminate the other parameters of the proposed threshold function and try to identify optimal values for them. Further, we need to investigate the influence of task switching delay on the performance. More specifically, we will study at what point it becomes advantageous to partition the task, depending on task switching delay.

5.3 Sequential Task Allocation in a Self-Assembled Swarm

The following experiment is the application of the sequential task allocation methods presented in the previous section to a "real-world" problem. Because of its

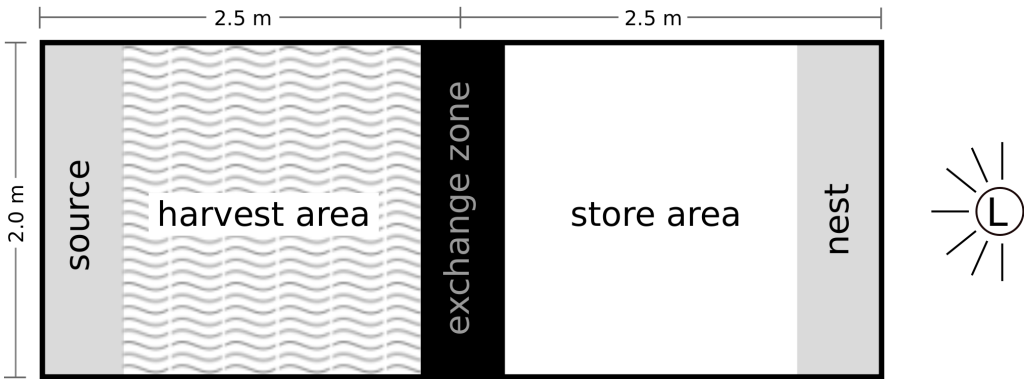


Figure 5.7: Depiction of the environment used in the self-assembly experiment. The shaded area is a rough terrain which renders a single robot incapable of navigating across it. Thus, robots have to team up in order to tackle this subtask.

dependence on the study of the threshold function, this work has not been extensively researched. Nevertheless, the basic implementation has been completed and the setup works. We will therefore outline the rationale and the goal of this research.

5.3.1 Outline of the Experiment

The rationale behind this experiment is to apply the method described in Chapter 4 and the threshold functions studied in the previous experiment to an environment which demonstrates the properties of the system in a "real-world" problem. We defined the task so that the two subtasks are not essentially the same as in the previous experiment and switching between subtasks is costly (i.e., there is a task switching delay).

The experiment will be conducted in an environment similar to the one specified in the previous experiment. The environment is again partitioned in two areas, one for harvesting and one for storing. The dimensions of the areas are as given in the previous Section 5.2.1, with an equal distribution of area between the two partitions. The difference compare to the previous experiment is that the surface of one of the subtasks is rough, thus rending a single robot incapable of navigating across the terrain. The subtask can therefore only be tackled by a self-assembled group of robots (i.e., two foot-bot attached to each other). Figure 5.7 gives a graphical representation of the environment.

5.3.2 Research Directions

The study will focus on the application of the threshold model mentioned in the previous Section 5.2 and the comparison to the self-assembly experiment. Interesting questions will be:

- Can conclusions be transferred between the two experiments, and if yes, at what level?
- Is the threshold model capable of converging to the optimal allocation?
- Are the parameter settings that proved to be optimal in the abstract experiment optimal in this experiment as well?
- What is the influence of a task switching cost to the system's performance?

Additionally, certain questions stem from the fact that the robots have to self-assemble. For example, an interesting question is which switching cost makes it advantageous not to disassemble after each run of the "rough-terrain" subtask and stay instead assembled during the course of the whole experiment? If this is the case, will this change in behaviour occur by itself and for all individuals? Will it occur at the same time or gradually? If it is not the case that the cost of task switching justifies always-assembled teams, will we observe the emergence of specialized individuals, only working on one task (be it assembled or individually)? We think that these questions are a substantial base for further studies and multiple interesting research directions.

5.4 Sequential Task Allocation in an Heterogeneous Swarm

In the following experiment we study another application of the sequential task allocation method. Because of the dependence on the study of the threshold function, this work has not been extensively researched. Nevertheless, the basic implementation has been completed and the setup works. We will therefore outline the rationale and the goal of this research.

5.4.1 Outline of the Experiment

The rationale behind this experiment is similar to the one presented in Section 5.3, but with a slightly different focus. In this experiment, we study the allocation to a task which has been partitioned into two subtasks which are essentially the same. The difference is that the swarm working on this task is heterogeneous, consisting of two types of robots: "strong" and "weak" ones. Only the strong robots can work

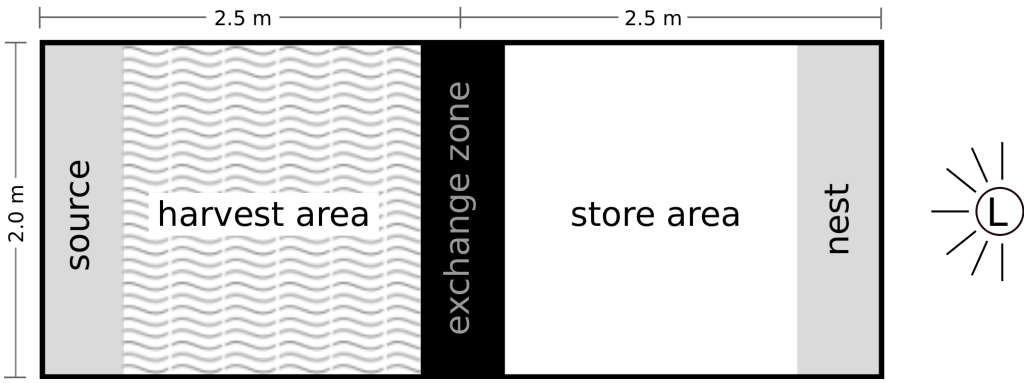


Figure 5.8: Depiction of the environment used in the heterogeneous experiment. The shaded area is a rough terrain on which some robots can only move with reduced speed. The other robots are able to move on both terrains with the same speed. Thus, robots have to specialize in order to reach optimal performance in this task.

in both subtasks efficiently, while the weak ones loose efficiency in one of them. Therefore, the swarm's individuals have to specialize and utilize division of labour in order to reach maximum performance. We will study if and how specialization in a task like this can be achieved in a self-organized way.

The experiment will be conducted in an environment similar to the one specified in the previous experiment. The environment is again partitioned in two areas, one for harvesting and one for storing. Similar to the environment of the previous experiment, described in Section 5.3.1, the surface of one of the subtasks is rough. Although all robots can navigate across this rough terrain, weak robots can do so only by driving very slow, while strong robots can proceed at normal speed. Thus, it would be advantageous that the strong robots specialize on the subtask with the rough terrain, while the weak robots work on the subtask where they can move at full speed. Figure 5.8 gives a graphical representation of the environment.

The study will focus on how the swarm's individuals can specialize in a self-organized way. Most probably, we will employ the well known *reinforced response threshold* model (Theraulaz et al., 1998), which has been proven to explain specialization and division of labour in colonies of social insects. In this model, individual i engages in task j with the following probability (from Bonabeau et al., 1996):

$$T_{\theta_{ij}}(s_j) = \frac{s_j^2}{s_j^2 + \theta_{ij}^2} \,, \tag{5.4}$$

where s_j is the stimulus of task j and θ_{ij} the threshold of individual i for task j.

For a stimulus $s_j \ll \theta_{ij}$, $T_{\theta_{ij}}(s_j)$ will be close to 0, rendering it nearly impossible that the individual i will start working on task j. On the other hand, for a stimulus $s_j \gg \theta_{ij}$, $T_{\theta_{ij}}(s_j)$ will be close to 1, making it nearly certain that the individual i will start working on task j.

Additionally, θ_{ij} is updated in a self-reinforced way as follows (from Theraulaz et al., 1998):

$$\theta_{ij} \rightarrow \theta_{ij} - \xi\Delta t \tag{5.5}$$

if individual j did not perform task j in the time period Δt, and

$$\theta_{ij} \rightarrow \theta_{ij} + \varphi\Delta t \tag{5.6}$$

if individual j did perform task j in the time period Δt. The factors ξ and φ are usually called forgetting and learning coefficients respectively, and control how fast individuals (de-)specialize. The overall threshold update function for a time period Δt is then:

$$\theta_{ij} \rightarrow \theta_{ij} - \xi\Delta + (1 - \theta_{ij}) + \varphi\Delta t \tag{5.7}$$

Replacing the stimulus s_j by a function similar to an inverted Equation (5.3) should allow the individuals to switch task according to necessity. As the value of t_w for the weak robots should always be larger for the subtask they can efficiently work on, they should specialize by lowering their threshold for this subtask. At the same time, their threshold for the other subtask should rise, making it less likely for them to work on that subtask. The strong robots on the other hand will be drawn to the subtask with the rough terrain, simply because of the lack of individuals working there.

5.4.2 Research Directions

As mentioned before, the study will focus on how the swarm's individuals can specialize in a self-organized way. Additionally, we will study how to embed this in the method described in Chapter 4. Interesting questions will be:

- Can the swarm reliably specialize?
- Is the swarm capable of converging to the optimal allocation?
- Is the specialization flexible enough, e.g., do weak robots start to work on both tasks when there are not enough strong robots present?
- How should the stimulus, threshold and update functions be designed?

- What is the influence of the parameters, and which values prove to be optimal? Are these environment-dependent?
- Can we extend the mathematical model from Section 5.2.1 to model the specialization of individuals?

We think that these questions are interesting and provide another direction for research: specialization in a self-organized robotic swarm. Although similar systems have already been researched, they were never fully analysed from the task allocation perspective. Moreover, combining these strategy with a method for self-organized allocation to complex task is certainly promising. Additionally, the parallels with social insect societies might provide further inside to the behavior of complex systems in both fields.

5.5 Interference Reduction through Sequential Task Partitioning

In collective robotics, interference is a critical problem limiting the growth of a group: the time each robot spends in non-task-relevant behaviors such as obstacle avoidance increases when the density of individuals rises—see e.g., Lerman and Galstyan (2002). The performance on tasks that suffer from physical interference can typically be improved by spatial partitioning; for example, by keeping each robot in its own "working area". A known approach that uses this rationale is the so called bucket-brigade (Fontán and Matarić, 1996; Shell and Matarić, 2006). In this approach, robots hand over objects to robots working in the following area, until the objects reach their destination. As tasks usually cannot be partitioned arbitrarily, this approach effectively limits the number of robots that can be employed in the task. A possible solution to this problem, treating working areas as non-exclusive, raises other problems: How should individuals be allocated to tasks? How can such an allocation help in limiting the amount of interference?

In this section, we study the application of self-organized task allocation to this problem. More specifically, we study a) how partitioning the global task in two subtasks with a sequential interdependency can help in reducing interference, and b) how we can apply the method introduced in Section 5.2 to this task. In contrary to the previously presented experiments, the robots exploit a simple, deterministic threshold-based model to decide when to switch task: when the waiting time t_w is higher than a threshold θ, a robot switches its subtask. In the following, we study the properties of this simple self-organized task allocation strategy, compare this strategy to a strategy without task partitioning, and analyze how it can help to reduce interference. We refer to the two strategies as *partitioned* and *non-partitioned*, respectively.

5.5.1 Methods

This section describes the environments in which the experiments are carried out, the simulated robots, and the robot's controller. Additionally, we describe how we run the experiments and we introduce some metrics that we use to evaluate the properties of the system.

Environments

At time $t = 0$, the robots are randomly placed in the harvest area. The experiments run for $t_{max} = 18,000$ time steps (a simulated time of one hour, with a time step length of 200 ms). The experiments are run in two different arenas (see Figure 5.9). The first arena (Figure 5.9a) is 4.125 m long with a width of 1.6 m at the source and exchange zone, whereas the nest is 0.4 m wide. The exchange zone is located 3.125 m away from the source. This arena is characterized by the presence of an area, critical for the task, in which high interference between robots can be expected (the nest). Thus, this arena is referred to as the *narrow-nest* environment.

The second arena (Figure 5.9b) has a rectangular shape: it is 3.75 m long and 1.6 m wide. Here as well the exchange zone is located 3.125 m from the source. The arena shape does not suggest the presence of any zone where interference can be higher than in other places. This arena is referred to as the *wide-nest* environment.

The area of both arenas is 6 m^2, 5 m^2 for the harvest area and 1 m^2 for the store area. The overall area is the same in the two arenas, so that the same group size results in the same robot density. Thus, results are comparable across the two environments.

Simulated Robots

As this experiment was the first one studied, the predecessor of the foot-bot, the s-bot, has been used. Although the experiments are conducted in simulation only, these simulated robots have a real counterpart: the swarm-bot robotic platform (Mondada et al., 2004). The platform consists of a number of mobile autonomous robots called s-bots, which have been used for several studies, mainly in swarm intelligence and collective robotics—see for instance Groß et al. (2006) and Nouyan et al. (2008). In the context of the experiments reported, the s-bots and the foot-bots are interchangeable, though. Therefore, we are certain that results are transferable across the two robot platforms.

Similar to the foot-bots, the simulated s-bots are of round shape, with a diameter of 0.116 m. Each of them is equipped with 16 infrared proximity sensors, used to perceive obstacles up to a distance of 0.15 m. Eight ambient light sensors can be

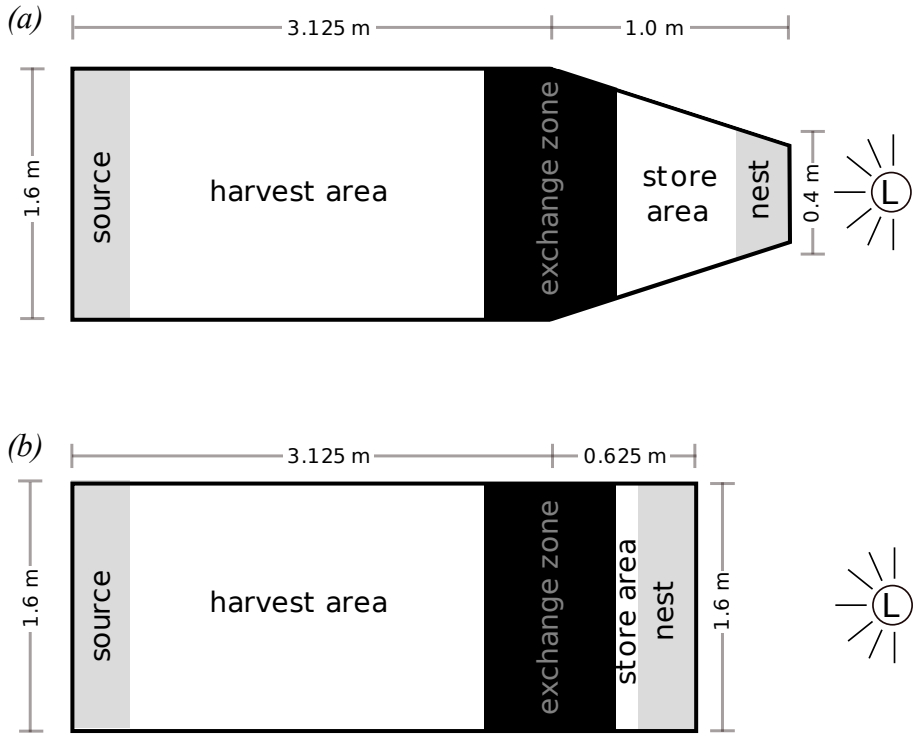


Figure 5.9: Depiction of the environments used in the interference-reduction experiment. (a) *Narrow-nest* environment used in the first experiment. (b) *Wide-nest* environment used in the second experiment. The gray stripes are the source (left), and the nest (right), each 0.25 m deep. The black stripe is the exchange zone, that is 0.5 m deep. The light source is marked with "L".

used to perceive light gradients up to a distance of 5.0 m. The robots are equipped with 4 ground sensors used to perceive nest, source and exchange zone. A 8 LEDs ring is used to signal when a prey object is carried. An omnidirectional camera allows the perception of LEDs in a circle of radius 0.6 m surrounding the robot. A uniform noise of 10% is added to all sensor readings at each simulation step. The robots can move at a maximum speed of 0.1 m/s by means of a differential drive system.

Controller Specifics

As mentioned before, this controller uses a deterministic threshold in order to determine when a robot should switch its task. The robots internal threshold θ represents the maximum amount of control cycles they can spend in the transfer zone waiting to pass (harvesters) or receive (storers) a prey object. If a robot remains in the

transfer zone longer than its threshold without passing or receiving prey objects ($t_w > \theta$), it switches its task. The optimal threshold value is not trivial to determine. In the work presented here, we use a simple method to set the threshold θ: at the beginning of the experiment, each robot draws a random threshold, sampled uniformly in the interval $[0, 1000]$.

We chose this method because it is sufficiently independent of the environment and does not rely on complex approximation techniques. The threshold value does not change during the experiment. In case of the non-partitioned strategy, the threshold is set to $\theta = 0$, causing the robots to switch subtask immediately as soon as they reach the exchange zone.

Experiments

The goal of the experiments is to investigate whether task partitioning can reduce interference in task-critical zones, and how to allocate a robotic swarm to partitions. As pointed out by Lerman and Galstyan (2002), interference is related to the number of individuals in the system. Additionally, the physical interference between robots is also a function of the environment the robots act in. The higher the group size, the higher the density, resulting in a higher amount of physical interference. Thus, in order to study interference in our experiments, we increase the size of the group in each of the two environments shown in Figure 5.9, while using both strategies (non-partitioned and partitioned). We study the performance of the system when the group size N ranges in the interval $[1, 40]$. We run 50 repetitions for each value of N and each experimental setting.

Metrics

In order to quantify the influence of interference, we measure the *group performance* P by the number of prey objects collected by the swarm at the end of the experiment ($t_{max} = 1$ hour). From the group performance measure we can derive the *individual efficiency* as follows:

$$I_{eff} = P/N, \tag{5.8}$$

where N is the size of the group. Individual efficiency can help to understand the effect of interference on the performance.

In order to measure the influence of environmental features on the interference, we define an interference measure taking inspiration from Rosenfeld et al. (2005). In their work, interference is measured as the time spent performing actions not strictly related to the task, but rather lost due to negative interactions with the environment (e.g., obstacle avoidance maneuvers). By registering the number of

collisions for each area of the arena, we can draw conclusions about where physical interferences happen most often. We measure interference through the state of the controller: in our case a robot is experiencing interference each time its controller perceives an obstacle.

In case of a partitioned task, there is another source of inefficiency that adds to interference: the time lost in the exchange zone. We define the *strategy cost* C as the sum of time lost because of physical interference and time lost in the exchange zone:

$$C = T_{int} + T_{part}, \tag{5.9}$$

where T_{int} is the amount of time steps during which the controller perceives an obstacle, and T_{part} is the total amount of time steps spent in prey passing maneuvers. By using this metric, the cost of the non-partitioned strategy is purely due to interference ($T_{part} = 0$), while in case of the partitioned strategy, prey passing costs add to interference costs. In a way, passing a prey object produces another kind of interference in the system. The strategy cost captures this effect, thus allowing for a comparison of strategies.

5.5.2 Results and Discussion

The graphs in Figures 5.10a and 5.11 show the performance P for different group sizes in the narrow-nest and wide-nest environment respectively. Figure 5.10b shows the individual efficiency I_{eff} of the robots in the narrow-nest environment. Black curves are the average computed over the 50 repetitions of each setting, gray curves indicate the 95% confidence interval on the expected value. The performance graph in Figure 5.10a shows that the partitioned strategy improves performance in the narrow-nest environment. The graph shows that the non-partitioned strategy performs better than the partitioned strategy for small group sizes (up to $N = 13$ robots). However, increasing the group size makes the non-partitioned strategy collapse: the number of gathered prey objects drops dramatically for groups larger than 13. The individual efficiency graph (Figure 5.10b) can explain the behavior of the system. The robots employing the partitioned strategy are less efficient, for small group sizes, than those performing the non-partitioned strategy. However, the addition of more individuals affects the efficiency of the non-partitioned group in a more dramatic way. At a certain point, the drop in efficiency becomes very steep for the non-partitioned strategy. On the other hand, the partitioned strategy scales better: individual efficiency drops smoothly. This explains why a group using the partitioned strategy performs better: it can benefit from the work of more individuals and therefore collects more prey objects. These considerations do not hold in the wide-nest environment. The performance graph in Figure 5.11 shows that the non-partitioned strategy performs better than the partitioned strategy for

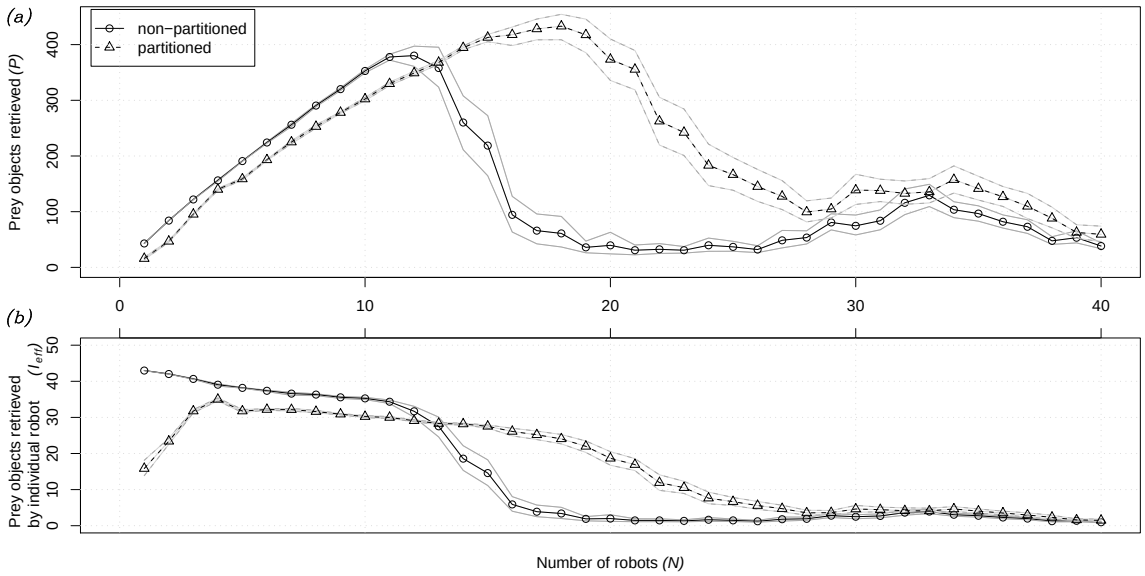


Figure 5.10: (a) Performance P and (b) individual efficiency I_{eff} for increasing number of robots in the narrow-nest environment. The black continuous line refers to the case of no task partitioning, the black dashed line to the case of partitioning. Gray lines indicate the 95% confidence interval on the expected value.

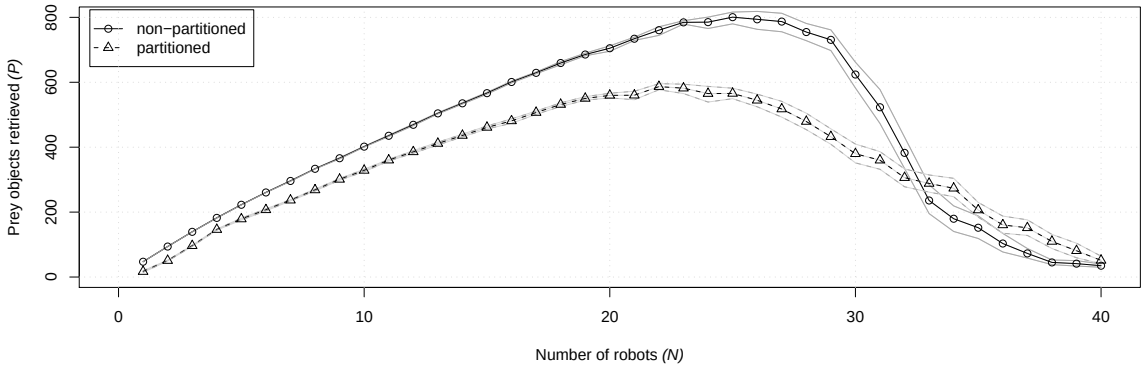


Figure 5.11: Performance P for increasing number of robots in the wide-nest environment. The black continuous line refers to the case of no task partitioning, the black dashed line to the case of partitioning. Gray lines indicate the 95% confidence interval on the expected value.

group sizes $N < 33$. In both the environments, independently of the strategy used to accomplish the task, the system collapses when the area is saturated by the swarm.

Figure 5.12 shows how an increasing number of robots effects the strategy cost C in the narrow-nest environment. The graph compares the cost C of each of the two strategies for different group sizes. In case of the partitioned strategy (Fig-

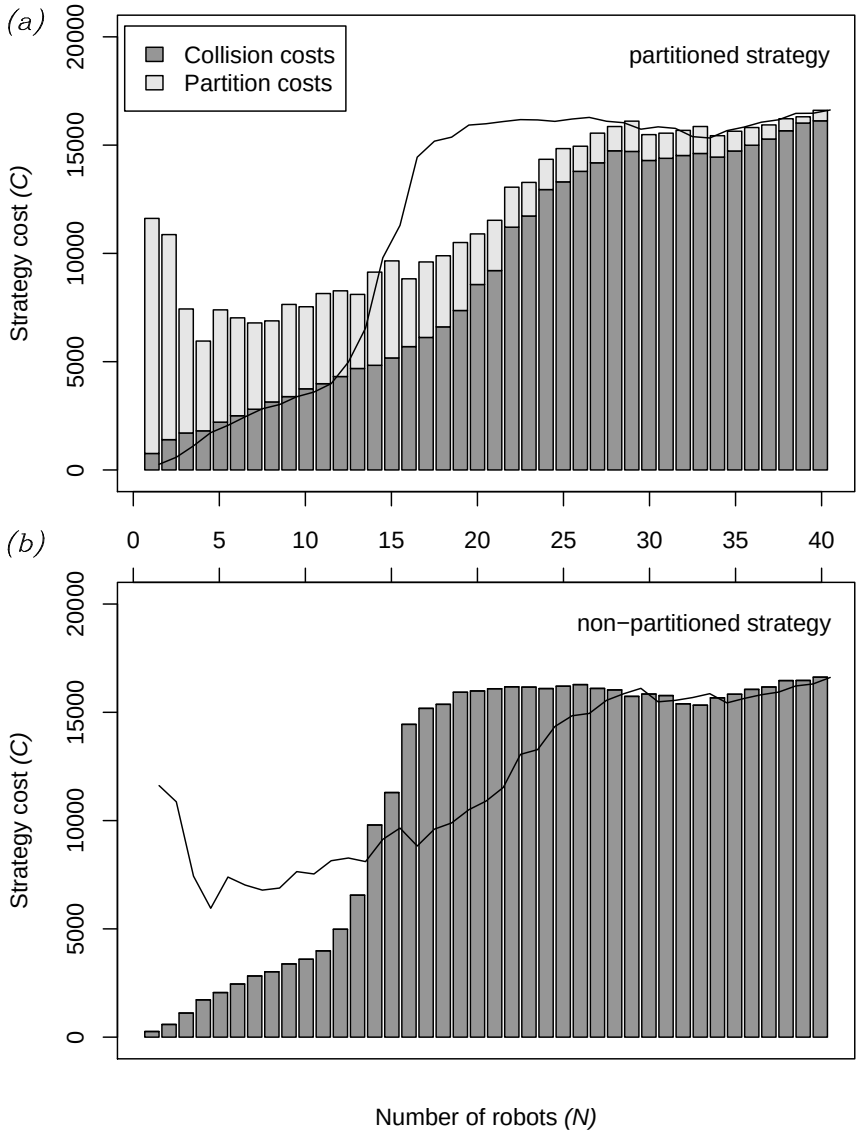


Figure 5.12: Cost of interference in the narrow-nest environment. Bars represent the cost C, sum of interference time T_{int} and partition time T_{part} (i.e., waiting times). For easy reference, the outline of the bars of the respective other graph has been added to each graph. (a) Costs for the partitioned strategy, where interference cost stem from waiting times and collisions. (b) Cost in case of the non-partitioned strategy, where only physical interference through collisions exists.

ure 5.12a), the graph shows each component of the cost (T_{int} and T_{part}). Clearly, task partitioning has the effect of reducing the cost due to interference but has the disadvantage of increasing the cost due to time lost in the exchange zone. The probability of two or more robots encountering each other increases with the robot density. Although this determines a higher interference cost (i.e., T_{int}), it decreases the cost due to lower waiting time (i.e., T_{part}) in the case of the partitioned strategy. Partitioning performs better when the gain from interference reduction is greater than the loss of performance due to partitioning inefficiencies. These considerations hold in the narrow-nest environment, where the likelihood of physical interference in a task-critical zone is very high. In the wide-nest environment, interference in the nest is as likely as interference in the exchange zone. Thus, it is not beneficial to pay the cost of waiting and the non-partitioned strategy performs better for any group size.

The mechanism by which partitioning reduces interference costs can be deduced by comparing the interference graphs in Figure 5.13. The graphs show the number of times that physical interference (as defined in Section 5.5.1) was registered in each region of the narrow-nest environment. The total area was discretized in squares of $1\,\mathrm{cm}^2$. Figure 5.13 shows the results obtained with 18 robots, in the case of the partitioned strategy (Figure 5.13a) and in the case of the non-partitioned strategy (Figure 5.13b). The graphs show that the use of the non-partitioned strategy leads to high interference in the nest, which becomes congested. Partitioning the task reduces the robot density in the nest, thus spreading the interference more uniformly across the arena. In addition, the overall interference diminishes because the majority of robot contact is moved into a area of the environment that is wider than the nest: the exchange zone. Thus, the robots have more freedom of movement and collide less often. Although the graphs show only data collected with 18 robots, experiments with different group sizes produced similar results.

5.5.3 Conclusions and Future Work

Interference can be an issue when working with swarms of robots. In this work, we used task partitioning and allocation to reduce interference between robots sharing the same physical space. We manually partitioned the environment and employed a simple self-organized strategy for allocating individuals to subtasks. Results show that a partitioning strategy improves performance in a constrained environment. Additionally, we identified cases in which partitioning is not advantageous and a non-partitioned strategy should be used. The proposed strategy is fairly simple and far from being an optimal solution, nevertheless we improved the performance of the swarm when interference was costly.

Future work will concern the identification of the optimal allocation in the studied environments as well as the development and study of a strategy that can find this optimal allocation in a self-organized and adaptive way. In addition, the interference

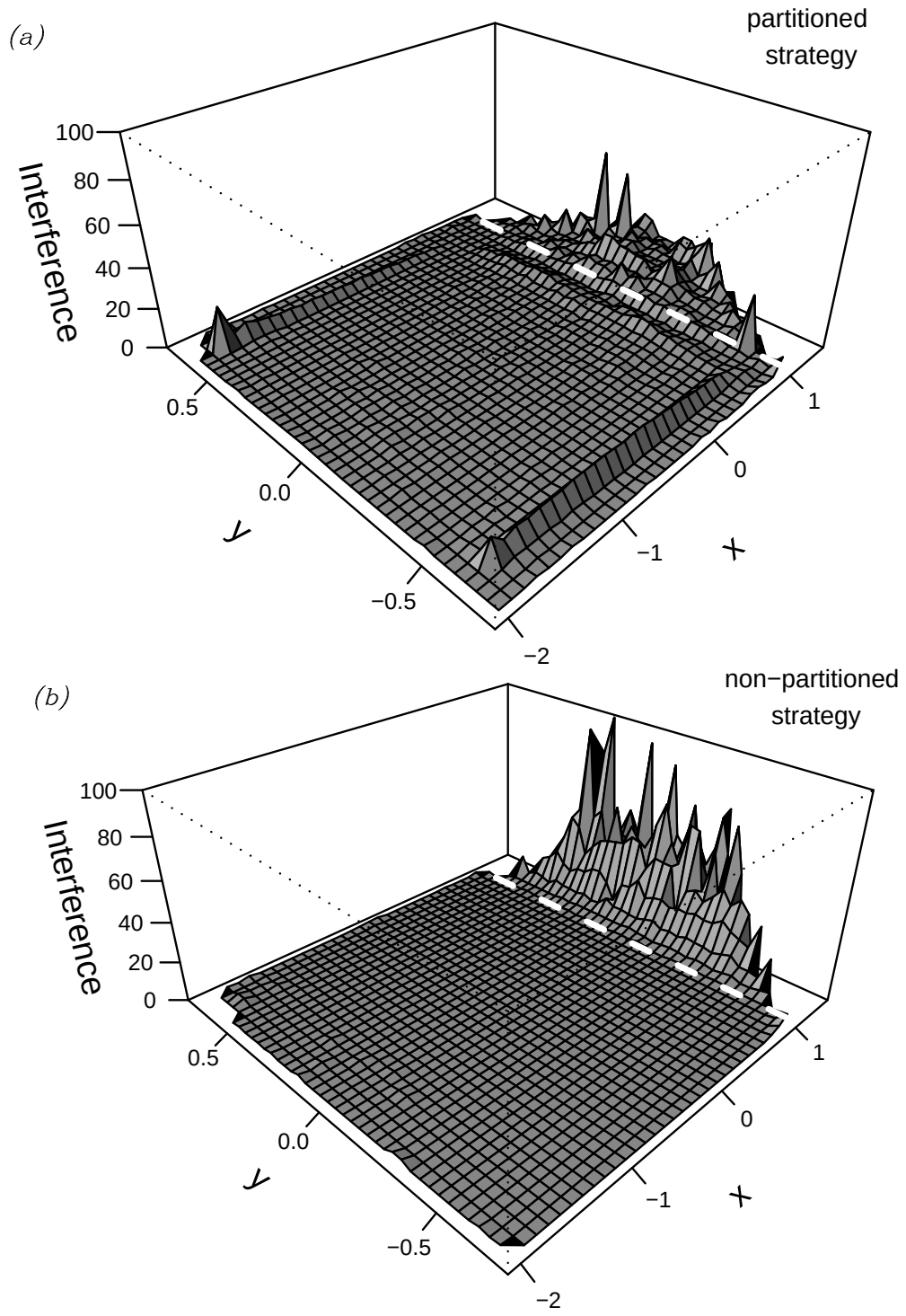


Figure 5.13: Mean interference values registered for (a) the partitioned strategy and (b) the non-partitioned strategy, both in the narrow-nest environment. Shown values are observation means of 50 repetitions with $N = 18$ robots. Coordinates on the x- and y-axis are given in meters. The arena is stretched along the y-axis for better visualization. The dashed white line marks the location of the exchange zone.

metric proposed in Section 5.5.1 could be used by the robots to decide whether to partition the task. In this way, we could achieve even better performance, since partitioning would be employed only when strictly needed. Finally, the goal is to validate the system using the real robots.

6 Parallel Task Allocation

In this chapter we will discuss self-organized allocation to subtasks with a parallel interdependency. Together with the sequential task allocation discussed in the previous chapter, parallel task allocation is one of the keystones for using the task allocation method presented in Chapter 4.

The work presented here is in a preliminary phase. Up to now, only tentative considerations and studies have been undertaken, but no real experiment has been developed out of them. We will therefore present only briefly the current state and give an outlook for possible experiments.

6.1 Outline of the Experiments

Similarly to the experiments presented in the previous chapter, we will use a foraging problem as a testbeds for our task allocation algorithms. A typical scenario is a harvesting task with two different types of prey objects (we will refer to them as "blue" and "red" objects from now on). The two distinct subtasks of harvesting one object type can be executed in parallel, thus creating a parallel task allocation problem as defined in Chapter 4. Usually, certain constraints are imposed on the subtasks, e.g., a given ratio of objects has to be harvested or a certain energy limit has to be kept. Several examples of this can be found in the literature, e.g., Campo and Dorigo (2007).

The task can be represented according to the the method introduced in Chapter 4 as follows. There exist two atomic subtasks, $\mathcal{T}_{\text{blue}}$ and $\mathcal{T}_{\text{red}}$, with the global subtask being $\mathcal{T}_{\text{parallel}} = \mathcal{T}_{\text{blue}} \ominus \mathcal{T}_{\text{red}}$. See Figure 6.1 for a graphical representation of the task dependency graph.

In this experiments, we will again limit ourselves to a harvesting task that is pre-partitioned by the designer into two subtasks with a parallel interdependency. Robots can decide to switch from one subtask to the other, thus creating a task allocation problem: individual robots have to be allocated to subtasks and different allocations yield different performance. Additionally, constraints as discussed above might influence the desired allocation.

The first group of planned experiments concerns exactly such a parallel task allocation problem with a constraint imposed on the ratio of objects. This means that the

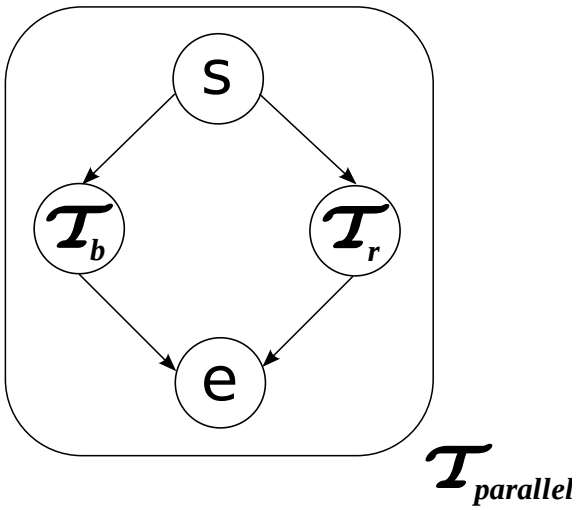


Figure 6.1: The general task dependency graph for the parallel task allocation problem studied here, according to the method presented in Chapter 4. The swarm has to harvest objects of two types (e.g., "blue" and "red" objects, their collection subtasks represented by $\mathcal{T}_\mathrm{b}$ and $\mathcal{T}_\mathrm{r}$, respectively). The two tasks can be executed in parallel, but might have some constraints imposed on them (e.g., harvest a specific ratio of objects).

swarm has to harvest the a specific, a-priori defined ratio of the two objects. The experiments will be concerned with researching the properties of the problem and possible methods to honor this constraint in a self-organized system.

6.2 Research Directions

The proposed studies open several research directions. Of course, the first group of experiments already rises several questions of mostly practical nature: how can we honor a ratio constraint as described above in a self-organized system? How can we achieve self-organized allocation to such tasks? Is it at all possible to achieve adaptive allocation without relying on explicit communication? If yes, which methods can be used? If no, how and which information should be passed among members of the swarm?

There are several higher-level questions stemming from these basic questions. The first is mostly related to the method presented in Chapter 4. The current method does not include the possibility to model constraints between the two subtasks running in parallel. In order to make this method general enough to allow for broad application, task allocation problems with parallel interdependencies have to be extensively researched. Therefore, the current method does not include any model for additional constraints. Additionally, although studies with parallel task allocation

exist in the literature (e.g., Campo and Dorigo, 2007), no study has yet analysed task interdependencies in an abstract way. Thus, questions concerned with this line of research are:

- Can we identify and model constraints imposed on tasks with parallel interdependencies?
- If yes, how do these models relate to our current method for complex task allocation, and, moreover, can they be integrated?
- Can we find general methods for tackling those constraints in a self-organized system?

Another research direction will focus on the application of the threshold models that were used for sequential task allocation. The most interesting question will be if we can use similar, threshold-based algorithms for tackling self-organized allocation to parallel dependent tasks. If yes, this raises related questions:

- How can we include the constraints imposed on the subtasks into the model?
- Can conclusions between the two types of task allocation problems be transferred, and if yes, at what level?
- Can we unify the approaches to sequential and parallel task allocation problems into a single method, general enough to tackle both?

Preliminary experiments and a review of the literature prove that threshold-based methods are certainly suitable for allocating a swarm to parallel dependent tasks. Nevertheless, much depends on the modeling of the constraints imposed on the tasks. Is is possible to find a versatile method to represent these? Can it be integrated in the threshold-based algorithms, and if not, what are the alternatives? All these questions are currently of fundamental nature, as the dynamics of the basic underlying system have not been researched yet. We will therefore focus the work on the first group of experiments described above, in order to gain insight into the more advanced topics outlined here.

7 Conclusions and Future Work

This final chapter summarizes the research work presented in the previous chapters and draws conclusions from it. Additionally, we summarize future research directions and possible improvements given in these chapters.

Conclusions

In this work, we presented our ongoing research in self-organized task partitioning and allocation. The current state of the art provides no unified understanding nor tools for modeling and designing swarm systems that are capable of self-organized task partitioning and allocation. The work presented here provides the first steps towards a method providing exactly this. More specifically, the goal of our research is to find a method that allows to partition tasks into smaller and simpler subtasks, which can be tackled by a set of algorithms readily available for the designer.

In order to achieve this, we first defined the problem and investigated related issues. We defined *task dependency graphs*, a first step towards a method to partition complex tasks into smaller subtasks. Although the partitioning of a complex task is currently defined a-priori by the designer, the method is intended to allow for self-organized task partitioning. Nevertheless, in the rest of the work we focused on researching self-organized task allocation in case of the two types of subtasks used by the mentioned method: subtasks with *sequential* and *parallel* interdependencies.

We proposed several experiments for the sequential task allocation, ranging from a more abstract experiment which allowed us to study the parameters of the system more freely to specific applications in current robotics research. We developed several simulation tools for each experiment, each providing us with a different level of detail. We identified and studied parameters of the problem and proposed a threshold-based algorithm for self-organized task allocation, which has been studied in case of the abstract experiment. We proposed several extensions of this algorithm and future experiments building on the gained knowledge, including an experiment which focuses on substantially different subtasks and an experiment focusing on specialization in a heterogeneous swarm. Additionally, we studied the use of task allocation and task partitioning for interference reduction. This

study is the only completed and published work of the present report (Pini et al., 2009).

Moreover, we outlined experiments for the study of parallel task allocation, the next keystone required for the task modeling method proposed in this work. We identified possible research directions and issues with this approach, but will hold further studies until the discussed experiments for sequential task allocation have been finished.

To summarize, we presented a method for tackling self-organized task allocation in a structured way, identified the keystones for achieving this in a real system and studied several instantiations of the problem. Overall, the proposed method and algorithms are well beyond the state of the art and extend the knowledge in the field of swarm robotics.

Future Work

As we gave detailed research directions specific to each experiment in each experiment's section, we will limit ourselves here to a summary and try to give a global outline of planned and possible future work.

First, we will continue investigating sequential task allocation problems as presented in Chapter 5. We will try to uncover the relationship of the environmental cues mentioned and to define a robust and adaptive method for allocating individuals in such a system. The methods researched on the abstract experiment (Section 5.2) will be transferred to the other two experiments (Section 5.3 and 5.4). We hope to find a general method to tackle self-organized task allocation in all of these problems, thus creating a method general enough for broad application.

The next milestone in creating a unified method for self-organized task allocation will be the research of problems with parallel interdependencies, as discussed in Chapter 6. As this is currently a very preliminary work, a detailed road map cannot be given. Nevertheless, future work will focus on modeling the problem as well as researching possible constraints between the parallel subtasks. We will try to integrate methods for handling constraints with threshold-based algorithms for allocation.

The overall goal is to identify and develop a set of methods that allows a swarm to self-organize allocation to complex tasks which have been pre-partitioned by the designer into the two mentioned types of subtasks. The following step will be concerned with how to self-organize partitioning and decomposition of complex tasks. It is currently unknown to which extend this is possible, although first studies have been already undertaken. Quantifying the amount of knowledge a designer has to

inject into the system on its creation is an essential step and might further push the boundaries of the state of the art.

The transfer of the knowledge gained in theoretical analysis and simulation to real robot systems that have to tackle a practical problem is envisioned for all of the proposed experiments.

List of Figures

Bibliography

Abraham, A., Grosan, C., and Ramos, V., editors (2006). *Swarm Intelligence in Data Mining*. Springer Verlag, Berlin/Heidelberg, Germany.

Agassounon, W. and Martinoli, A. (2002). 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 Multi-Agent Systems (AAMAS-02)*, pages 1090–1097, New York. ACM Press.

Arkin, R. C., Balch, T., and Nitz, E. (1993). Communication of behavorial state in multi-agent retrieval tasks. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, volume 3, pages 588–594.

Ben-Jacob, E., Cohen, I., and Levine, H. (2000). Cooperative self-organization of microorganisms. *Advance in physics*, 49(4):395–554.

Beni, G. (2005). From swarm intelligence to swarm robotics. In *Lecture Notes in Computer Science*, volume 3342, pages 1–9, Berlin/Heidelberg.

Beni, G. and Wang, J. (1989). Swarm intelligence in cellular robotic systems. In *Proceedings of the NATO Advanced Workshop on Robots and Biological Systems*, Tuscany, Italy. NATO Scientific Affairs Division.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (2000). Inspiration for optimization from social insect behaviour. *Nature*, 406(6791):39–42.

Bonabeau, E., Theraulaz, G., and Deneubourg, J.-L. (1996). Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. *Proceedings: Biological Sciences*, 263:1565–1569.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2003). *Self-Organization in Biological Systems*. Princeton Studies in Complexity. Princeton University Press, Princeton, NJ.

Campo, A. and Dorigo, M. (2007). Efficient multi-foraging in swarm robotics. In *Advances in Artificial Life: Proceedings of the VIIIth European Conference on Artificial Life, Lecture Notes in Artificial Intelligence LNAI 4648*, pages 696–705, Berlin/Heidelberg, Germany. Springer Verlag.

Cao, Y. U., Fukunaga, A. S., and Kahng, A. B. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:226–234.

Christensen, A., O'Grady, R., Birattari, M., and Dorigo, M. (2008). Fault detection in autonomous robots based on fault injection and learning. *Autonomous Robots*, 24(1):49–67.

Christensen, A., O'Grady, R., and Dorigo, M. (2007). Morphology control in a multirobot system. *IEEE Robotics & Automation Magazine*, 11(6):732–742.

Collier, N. (2003). Repast: An extensible framework for agent simulation. Technical report, Social Science Research Computing, University of Chicago.

Detrain, C. and Deneubourg, J.-L. (2006). Self-organized structures in a superorganism: do ants "behave" like molecules? *Physics of Life Reviews*, 3(3):162–187.

Di Caro, G. and Dorigo, M. (1998). Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365.

Dias, M. and Stentz, A. (2003). TraderBots: a Market-Based approach for resource, role, and task allocation in multirobot coordination. Technical Report CMU-RI -TR-03-19, Robotics Institute, Carnegie Mellon University.

Dorigo, M. and Şahin, E. (2004). Guest editorial. special issue: Swarm robotics. *Autonomous Robots*, 17:111–113.

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

Fontán, M. S. and Matarić, M. J. (1996). A study of territoriality: The role of critical mass in adaptive task division. In *From Animals to Animats 4: Proceedings of the Fourth International Conference of Simulation of Adaptive Behavior*, pages 553–561, Cambridge, MA. MIT Press.

Garnier, S., Gautrais, J., and Theraulaz, G. (2007). The biological principles of swarm intelligence. *Swarm Intelligence*, 1:3–31.

Gautrais, J., Michelena, P., Sibbald, A., Bon, R., and Deneubourg, J.-L. (2007). Allelomimetic synchronisation in merino sheep. *Animal Behaviour*, 74:1443–1454.

Gerkey, B. P. and Matarić, M. J. (2000). Murdoch: Publish/subscribe task allocation for heterogeneous agents. In Sierra, C., Gini, M., and Rosenschein, J. S., editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 203–204, Barcelona, Spain. ACM Press.

Gerkey, B. P. and Matarić, M. J. (2003). Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2003)*, pages 3862–3867, Pitscataway, NJ. IEEE.

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

Goldberg, D. (2001). *Evaluating the Dynamics of Agent-Environment Interaction*. PhD thesis, University of Southern California, Los Angeles, CA.

Goldberg, D. and Matarić, M. J. (2003). Maximizing reward in a non-stationary mobile robot environment. *Autonomous Agents and Multi-Agent Systems*, 6(3):287–316.

Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics*, 22(6):1115–1130.

Groß, R. and Dorigo, M. (2008). Evolution of solitary and group transport behaviors for autonomous robots capable of self-assembling. *Adaptive Behavior*, 16(5):285–305.

Grünbaum, D., Viscido, S., and Parrish, J. K. (2004). Extracting interactive control algorithms from group dynamics of schooling fish. *Lecture Notes in Control and Information Sciences*, 309:103–117.

Jeanne, R. L. (1986). The evolution of the organization of work in social insects. *Monitore Zoologico Italiano*, 20(2):119–133.

Jeanne, R. L. (1991). *Polyethism*, pages 389–425. Cornell University Press, Ithaca, New York.

Jones, C. and Matarić, M. J. (2003). Adaptive division of labor in large-scale minimalist multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1969–1974, Piscataway, NJ. IEEE.

Kalra, N., Dias, M., Zlot, R., and Stentz, A. (2005). Market-Based multirobot coordination: A comprehensive survey and analysis. Technical Report CMU-RI-TR-05-16, Robotics Institute, Carnegie Mellon University.

Kalra, N. and Martinoli, A. (2006). A comparative study of market-based and threshold-based task allocation. In *Distributed Autonomous Robotic Systems 7*, pages 91–102, New York. Springer.

Krieger, M. J. B. and Billeter, J.-B. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Journal of Robotics and Autonomous Systems*, 30:65–84.

Labella, T. H., Dorigo, M., and Deneubourg, J.-L. (2006). Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems*, 1(1):4–25.

Lein, A. and Vaughan, R. (2008). Adaptive multi-robot bucket brigade foraging. In *Proceedings of the Eleventh International Conference on Artificial Life (ALife XI)*, pages 337–342, Cambridge, MA. MIT Press.

Lerman, K. and Galstyan, A. (2002). Mathematical model of foraging in a group of robots: Effect of interference. *Auton. Robots*, 13(2):127–141.

Lerman, K., Jones, C., Galstyan, A., and Matarić, M. J. (2006). Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research*, 25(3):225–241.

Liu, W., Winfield, A., Sa, J., Chen, J., and Dou, L. (2007). Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive Behavior*, 15(3):289–305.

Magg, S. and Te Boekhorst, R. (2008). Task allocation by dynamic specialisation. In *Proceedings of the German Workshop on Artificial Life 2008*, pages 91–100, Leipzig, Germany.

Matarić, M. J. (1997). Learning social behavior. *Robotics and Autonomous Systems*, 20(2):191–204.

Mclurkin, J. and Yamins, D. (2005). Dynamic task assignment in robot swarms. In *Proceedings of Robotics: Science and Systems*.

Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I. V., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L. M., and Dorigo, M. (2004). SWARM-BOT: A new distributed robotic concept. *Autonomous Robots*, 17(2–3):193–221.

Nouyan, S., Campo, A., and Dorigo, M. (2008). Path formation in a robot swarm. Self-organized strategies to find your way home. *Swarm Intelligence*, 2(1):1–23.

Østergaard, E., Sukhatme, G., and Matarić, M. J. (2001). Emergent bucket brigading. *Autonomous Agents*, 37:2219–2223.

Parker, L. E. (1998). ALLIANCE: an architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on*, 14(2):220–240.

Pini, G., Brutschy, A., Birattari, M., and Dorigo, M. (2009). Interference reduction through task partitioning in a robotic swarm. In *Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics*. To appear.

Ratnieks, F. L. W. and Anderson, C. (1999). Task partitioning in insect societies. *Insectes Sociaux*, 46:95–108.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.

Rosenfeld, A., Kaminka, G. A., and Kraus, S. (2005). A study of scalability properties in robotic teams. In *Coordination of Large-Scale Multiagent Systems*, pages 27–51, New York. Springer.

Rybski, P. E., Larson, A., Veeraraghavan, H., LaPoint, M., and Gini, M. (2008). Performance evaluation of a multi-robot search and retrieval system: Experiences with mindart. *International Journal of Intelligent and Robotic Systems*, 52(3–4):363–387.

Shell, D. and Matarić, M. J. (2006). On foraging strategies for large-scale multi-robot systems. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2717–2723, Pitscataway, NJ. IEEE.

Sperati, V., Trianni, V., and Nolfi, S. (2008). Evolving coordinated group behaviours through maximization of mean mutual information. *Swarm Intelligence*. In press.

Theraulaz, G., Bonabeau, E., and Deneubourg, J.-L. (1998). Response threshold reinforcement and division of labour in insect societies. *Proceedings: Biological Sciences*, 265(1393):327–332.

Turgut, A., Çelikkanat, H., Gökçe, F., and Şahin, E. (2008). Self-organized flocking with a mobile robot swarm. In *AAMAS'08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 39–46, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

White, T. and Helferty, J. (2005). *Engineering Self-Organising Systems*, chapter Emergent Team Formation: Applying Division of Labour Principles to Robot Soccer, pages 180–194. Springer Verlag, Berlin/Heidelberg, Germany.