Task partitioning in a robot swarm: retrieving objects by transferring them directly between sequential sub-tasks

G. Pini, A. Brutschy, A. Scheidler, M. Dorigo, and M. Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2012-010

May 2012

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2012-010

Noname manuscript No.
(will be inserted by the editor)

Task partitioning in a robot swarm: retrieving objects by transferring them directly between sequential sub-tasks

Giovanni Pini · Arne Brutschy · Alexander Scheidler · Marco Dorigo · Mauro Birattari

Received: date / Accepted: date

Abstract In this work, we study task partitioning in the context of swarm robotics. Task partitioning refers to the decomposition of a task into sub-tasks that can be tackled by different workers. In this work, we focus on the case in which a task is partitioned into a sequence of sub-tasks that must be executed in a certain order. This implies that the sub-tasks must interface with each other, and that the output of a sub-task is used as input for the sub-task that follows. A distinction can be made between task partitioning with direct transfer and task partitioning with indirect transfer. We focus our study on the first case: the output of a sub-task is directly transferred from an individual working on that sub-task to an individual working on the sub-task that follows. As a testbed for our study, we use a swarm of robots performing foraging. The robots have to harvest objects from a source, situated in an unknown location, and transport them to a home location. When the source is found, the robots memorize its position and use dead-reckoning to return there. Dead-reckoning is appealing in robotics, since it is a cheap localization method and it does not require an additional external infrastructure. However, it leads to errors that grow unbounded in time if not corrected periodically. We compare a strategy that does not make use of task partitioning to one in which object retrieval is partitioned into sub-tasks and the objects are directly transferred between the robots. We show that cooperation through task partitioning can be used to limit the effect of dead-reckoning errors. This results in an improved capability of locating the object source and in an increased performance of the swarm. We use the implemented system as a testbed to study benefits and costs of task partitioning with direct transfer. We implement the system with real robots, demonstrating the feasibility of our approach in a foraging scenario.

Keywords task partitioning, foraging, swarm robotics, self-organization, localization, dead-reckoning

Giovanni Pini · Arne Brutschy · Alexander Scheidler · Marco Dorigo · Mauro Birattari
IRIDIA-CoDE, Université Libre de Bruxelles, Brussels, Belgium
E-mail: gpini@ulb.ac.be, arne.brutschy@ulb.ac.be, ascheidler@ulb.ac.be, mdorigo@ulb.ac.be, mbiro@ulb.ac.be

1 Introduction

Task partitioning is a technique for organizing the activities of groups of workers that consists in decomposing a task into a number of sub-tasks (Jeanne, 1986). This decomposition allows different groups of workers to tackle each of the sub-tasks in parallel, as well as to tackle the sub-tasks in different moments in time. In this work, we focus on the case in which a task is partitioned into a sequence of sub-tasks, which implies that the sub-tasks have to be performed in a defined order.

Task partitioning has been observed in different activities of social insects, such as ants, bees, and wasps (Ratnieks and Anderson, 1999a). Insects benefit from task partitioning in several ways. Task partitioning allows the physical separation of individuals working on different sub-tasks and therefore it can be beneficial in terms of reduction of physical interference (see examples in Hart and Ratnieks, 2000). Physical interference occurs when many individuals share a space and hamper the movements of each other. Interference can have a strong negative impact on performance, because the individuals have to spend resources to deal with it (e.g., spend time for avoiding other individuals instead of performing the task). Task partitioning also facilitates the exploitation of specialization and heterogeneity. For example, large workers of the leaf-cutting ant *Atta laevigata* climb plant stems and cut leaves at the petioles. The leaves are then dropped to the ground, where smaller individuals cut them in pieces. Petioles are harder to cut than leaves, therefore harvesting leaves is best performed by larger individuals (Vasconcelos and Cherrett, 1996). Partitioning a task into sub-tasks can also result in a gain of efficiency. An example is the foraging strategy of certain species of leaf-cutting ants, where some individuals climb trees and cut leaves and drop them to the ground. Other workers collect the leaves from the ground and transport them to the nest (Ratnieks and Anderson, 1999a). Partitioning the collection of leaves removes the need to repeatedly climb the tree, resulting in increased energy efficiency.

These examples highlight the benefits of task partitioning; however, it is important to notice that there are also costs associated with it. Although the nature of these costs is related to the specific tasks, when a task is partitioned into a sequence of sub-tasks, costs commonly occur where sub-tasks interface with each other. We can distinguish between two ways for interfacing sub-tasks, depending on how the output of a sub-task becomes an input for the sub-task that follows (Ratnieks and Anderson, 1999a). In *direct transfer* the output of a sub-task is transferred from an individual working on that sub-task to an individual working on the sub-task that follows (i.e., workers interact directly). In *indirect transfer* the output of a sub-task is stored in a temporary location (usually referred to as cache); in this case, there is no need for the two workers to be present at the same time to effectuate the transfer. When the transfer is direct, the costs are mainly due to the time needed to meet a transfer partner (Anderson and Ratnieks, 1999) and to perform the transfer itself (Anderson et al., 2002). When the transfer is indirect, the costs are due to inefficiencies of the cache. Examples of such inefficiencies are delays when accessing it (e.g., if the cache is empty and a worker must wait) or losses of materials stored at the cache (Ratnieks and Anderson, 1999a).

In this work, we study task partitioning in the context of swarm robotics. Swarm robotics is a branch of robotics that draws inspiration from social insects in the design and implementation of distributed robotic systems (Dorigo and Şahin, 2004; Beni, 2005). The main characteristics of a swarm robotics system are, similarly to what can be observed in social insects, simplicity of the individuals with respect to the task they must solve, restriction to local communication and information, and distributed control. Due to these characteristics, swarm robotics systems often exhibit desirable properties such as robustness, flexibility, fault tolerance, and scalability.

In the context of swarm robotics, task partitioning can provide similar benefits as for social insects. It is therefore interesting to explore the application of task partitioning to the organization of work in swarms of robots. In this paper we study the case in which a swarm of robots has to perform a foraging activity. In this case, the *task* of the robots is object retrieval: collect an object from a source and transport it to a home location, referred to as *nest*. The foraging activity consists in the robots of the swarm executing the object retrieval task multiple times. Foraging is widely studied in swarm robotics, since it is an abstraction of interesting real-world applications such as collection of samples, waste cleanup, and landmine clearance. For a review of robot foraging we refer the reader to the work of Winfield (2009).

In the foraging problem studied in this work, the objects are clustered in a single location, which we call *source*. Since the robots have no a priori knowledge about the environment, they need to explore it in order to find the source. When a robot finds the source, it maintains an estimate of its position, relatively to its own frame of reference. This estimate can be used by the robot to return to the source in the future. Determining the position of a robot relatively to a target is a problem known as *localization*. Several techniques have been proposed to tackle this problem. In this study, we use dead-reckoning. Dead-reckoning consists in the determination of the position of a robot by the integration over time of measures coming from motion sensors (Feng et al., 1994). Dead-reckoning is appealing because it is simple and it requires neither modifications of the environment nor a priori knowledge about it. The main problem of dead-reckoning is that the integration of noisy information over time leads to errors that can grow unbounded. In our specific case, dead-reckoning errors can result in a robot being unable to return to the source. As a consequence, to find the source the robot must explore the environment again.

Limiting the distance traveled by a robot reduces the magnitude of its dead-reckoning error, decreasing in this way the probability that it cannot find the source. We implement a strategy based on task partitioning that limits the distance traveled by each robot. The task of retrieving an object to the nest is decomposed into a sequence of sub-tasks. Each of these sub-tasks consists in transporting the object towards the nest, for a limited distance. Objects are delivered to the nest in progressive steps, with a robot working on one of the sub-tasks directly handing over objects to a robot working on the next sub-task. The process is fully decentralized and does not require the robots to communicate explicitly.

In simulated and real robot experiments, we show that cooperation through task partitioning allows the swarm to overcome the limitations of the single robot and increases the overall localization capabilities. The foraging strategy based on task partitioning can be used in cases where a swarm of robots has to retrieve objects situated in an unknown position in the environment. We use the implemented system as a case study to highlight costs, benefits and properties of task partitioning with direct transfer among individuals. This work is the first to explore the costs and benefits of task partitioning with direct transfer and to evaluate their impact on a real robotic system.

To the best of our knowledge, with the notable exception of the works of Fontan and Matarić (1996) and Goldberg and Matarić (2002), all the existing work on the topic of task partitioning has been carried out in simulation only. We deem real robot testing important, as often simulations overlook real-world phenomena and implementation issues that can strongly impact the behavior of the implemented systems. In the experiments studied in this paper, object handling and localization errors are critical issues. Therefore, we also run experiments with real robots foraging for physical objects. We do not make any assumption or simplification regarding the two issues, but instead we deal with them as they are in reality. The experiments performed with the robots demonstrate the applicability of an approach based on task partitioning to solve a problem in real-world settings.

The rest of the paper is organized as follows. In Sec. 2 we review the related work on the topics of task partitioning and localization. In Sec. 3, we describe the foraging problem that we use for our study and the strategies we implemented for tackling this problem. In Sec. 4, we describe the hardware and the simulation tools used to implement the studied system. In Sec. 5, we describe the real-robot and simulation experiments and we present the main results we obtained. In Sec. 6, we summarize the content of the paper, discuss the main contributions and propose directions for future research.

2 Related work

In this section, we review the work on the topics of task partitioning and robot localization. Task partitioning has been extensively studied in the field of biology, while only few works exist that study it in the context of swarm robotics. In Section 2.1 we review the work on task partitioning, focusing on swarm robotics. Contrary to task partitioning, several techniques have been proposed in the literature to tackle the problem of robot localization. In Section 2.2 we review work on localization outside the context of swarm robotics that is relevant to the study presented in this paper. In the same section we also review the state of the art of localization techniques proposed for swarm robotics systems.

2.1 Task partitioning

Task partitioning is a way of organizing work that consists in decomposing tasks into sequences of smaller sub-tasks (Jeanne, 1986; Ratnieks and Anderson, 1999a). Task partitioning is usually coupled with task allocation strategies: when a task is partitioned into multiple sub-tasks, workers must be assigned to each of these sub-tasks. The key difference between task partitioning and task allocation is that task allocation is a way of organizing the workforce, while task partitioning organizes the way work itself is performed (Ratnieks and Anderson, 1999a).

Task allocation has been extensively studied and several works on the topic exist both in the biology and the swarm robotics literature. In biology, task allocation can be observed in ants (Gordon, 1996; Bonabeau et al., 1996), bees (Ratnieks and Anderson, 1999b; Hart et al., 2002) and wasps (Akre et al., 1976). In swarm robotics, self-organized task allocation is often based on the response threshold model first introduced by Bonabeau et al. (1996). Threshold-based task allocation has been successfully employed in tasks such as foraging (Labella et al., 2006b) and object clustering (Agassounon and Martinoli, 2002).

The amount of research work on task partitioning is considerably lower and mainly studies natural systems. In nature, task partitioning is observed mainly in the organization of work of social insects (see Ratnieks and Anderson, 1999a, for a review). Instances of task partitioning have been reported in activities such as foraging (Seeley, 1995), garbage disposal (Hart and Ratnieks, 2001), and hunting (Schatz et al., 1996).

In the context of swarm robotics, task partitioning has received only marginal attention. Most of the work focuses on task partitioning as a means to reduce physical interference in groups of robots. Fontan and Matarić (1996) propose a system in which a foraging task is pre-partitioned into sub-tasks, each performed in a separate area. A robot is assigned a priori to each area (i.e., it works on the sub-task performed in that area). The authors show that efficiency is increased as an effect of the reduced competition for space (i.e., less physical interference). A similar result is presented by Pini et al. (2009) with the difference that the working areas are non-exclusive and are dynamically assigned. Goldberg and Matarić (2002) use a modified version of the system of Fontan and Matarić (1996) to study the design of behavior-based controllers that are robust with respect to failures and easy to modify.

Østergaard et al. (2001) compare bucket brigading and homogeneous foraging in a maze-like environment. The authors show that bucket brigading performs better when the corridors are narrow and it is hard (or impossible) for two robots traveling in opposite directions to pass at the same time. The idea of reducing interference using bucket brigading also appears in the work of Drogoul and Ferber (1992). However, in this work, the role of task partitioning is only marginal and the focus is on how global behaviors can be affected by variations of individual rules.

Shell and Matarić (2006) study the effect of interference on the performance of a swarm of robots in a foraging scenario. Each robot moves objects towards the nest, but never leaves an area of a given radius. As a result, objects are delivered to the nest by progressive movements from one area to the next. The authors show that, with increasing swarm sizes, a reduction of the working area size has a positive impact on the performance of the system. Lein and Vaughan (2008) extend the work of Shell and Matarić (2006) by adding a mechanism to dynamically adapt the size of the robots' working area. In another work, the same authors point out that the distribution of objects in the environment is critical for the performance of the system (Lein and Vaughan, 2009). To cope with clusters of objects, the authors further extend the proposed mechanism by allowing the robots to move the center of their working areas towards the position of clusters of objects.

In Pini et al. (2011), task partitioning is tackled explicitly. The authors propose a method for selecting between a partitioning and a non-partitioning strategy. In their work, the overall task is pre-partitioned into two sub-tasks, and one of the two strategies is selected on the basis of its costs. In a follow up work, Pini et al. (2012a) formulate the problem of choosing whether to employ task partitioning as a multi-armed bandit problem and show that algorithms employed in the literature to tackle multi-armed bandit problems can be successfully employed.

2.2 Localization

Localization is a common problem in robotics; several techniques to tackle this problem have been proposed in the literature (see Feng et al., 1994, for a review). Most of the existing techniques are not suitable for swarm robotics, either because they require complex sensing and computational capabilities (e.g., using maps) or because they depend on external infrastructures that are not always available (e.g., GPS). A low complexity alternative is using dead-reckoning: each robot estimates its own position by integrating information coming from sensors such as motor encoders. A known problem with dead-reckoning is that estimation errors accumulate with the traveled distance and that the quality of the estimate gradually degrades over time. Dedicated hardware (Hongo et al., 1987), calibration (Borenstein and Liqiang, 1996), or the use of Kalman filters (Kalman, 1960) can improve dead-reckoning.

In swarm robotics, different approaches have been proposed to perform localization. A common approach is to use robots of the swarm as landmarks. This is done by letting some of the robots move, while others stand still and serve as reference points for the moving robots. The robots need to be able to perceive each other and measure relative distances and/or orientations in order to use this solution. In the work of Kurazume and Hirose (2000), the swarm is divided into two groups. Groups move in turn, with the group of stationary robots acting as reference. Rekleitis et al. (2001) propose an analogous solution: robots are organized in pairs, while one of the two robots moves, the other is stationary and acts as a reference. Grabowski et al. (2000) use trilateration to calculate the relative position of the robots. The method requires a minimum of three robots to stand still while the others move. In the works of Nouyan et al. (2008), Nouyan et al. (2009), Dorigo et al. (2012)[1], and Werger and Matarić (1996), chains of robots that link points of interest in the environment are formed. The chains can subsequently be followed by other robots to help navigation in the environment. Sperati et al. (2011) use evolutionary robotics to synthesize a controller that allows a swarm of robots to create a chain that links two locations in the environment. The chain consists of a number of robots that navigate back and forth between the two locations. Connection between elements of the chain is maintained using vision. In the work of Ducatelle et al. (2011a), a swarm of flying robot attached to the ceiling provides directional information to robots moving on the ground. The two swarms cooperate to find obstacle free paths in the environment.

[1] A video (Dorigo et al., 2011) describing the system can be found online at `http://www.youtube.com/watch?v=M2nn1X9Xlps`

An alternative approach to chain formation is inspired by the pheromone trails used by ants. In this case, the robots mark the environment with information that can be used by others for navigation. In the work of Johansson and Saffiotti (2009), pheromone is implemented as information stored in RFID tags distributed across the environment. A robot can write information in the tags, which can be subsequently used for navigation. Vaughan et al. (2002) describe a system in which the robots create trails of waypoints between locations of interest in an initially unknown environment. Trails are virtual: waypoints, computed through dead-reckoning, are broadcast via radio and internally stored by receiving robots. Robots share common knowledge in the form of symbolic names given to the locations of interest, which are used to refer to the locations when communicating trails. A very similar approach is proposed by Ducatelle et al. (2011b). Also in their approach the robots exchange positional information that can be used to reach target locations. However, the robots themselves represent the waypoints that must be followed for reaching a target. In the work of Sugawara et al. (2004), a system composed of a CCD camera and an LC projector emulates the pheromone laying mechanism of the robots. The camera tracks the robots' movements and pheromone trails are then projected where needed. Drogoul and Ferber (1992) simulate a system in which robots can lay and follow "crumbs", used to mark paths in the environment. Mayet et al. (2010) implement a system in which the ground is painted with a synthetic resin that contains a phosphorescent material that reacts to ultraviolet light. UV-LEDs pointing at the ground are used by the robots to mark temporary trails on the floor. These trails can be perceived and followed using the on-board camera of the robots.

Gutiérrez et al. (2010) describe a localization strategy based on "social odometry": robots exchange the position of their target location with each other and use the information received by their peers to correct their dead-reckoning estimate. The method relies on the capability of the robots to measure the range and the bearing of the sender of each message.

3 Problem description and strategies

In this work, we study task partitioning in a swarm of robots performing foraging. Foraging consists in the repetition of the object retrieval task: collecting an object from a given location, called *source*, and transporting it to a central location, referred to as *nest*. The direction of the nest can be perceived by the robots from every position in the environment. The objects are clustered in the source that never depletes. The robots have no a priori knowledge about the location of the source. Therefore, in order to harvest objects, a robot needs first to explore the environment and to find the source. When the source is found, the robot memorizes its location, so that it can return to it again in the future.

Each robot faces a localization problem: when moving, the robot must update its information about the location of the source. The robots use dead-reckoning to tackle this problem: the positional information is derived from the integration of the measures of the motor encoders of the wheels. Due to noise, the position of the source

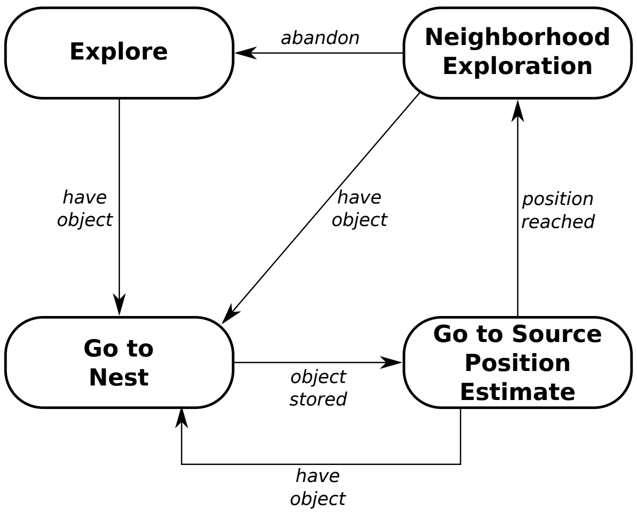


Fig. 1: Representation of the behavior of the robots when the non-partitioning strategy is employed

as maintained by a robot is only an estimate of the real one. In the rest of the paper, we will refer to a robot's estimate as the *source position estimate* of that robot.

The quality of the source position estimate depends on the distance traveled by the robot: the longer the distance traveled, the less accurate the source position estimate becomes. This is mainly due to three factors. First, errors on the bearing of the source position estimate have a stronger impact at longer distances from the source. Second, the longer a robot travels, the more it accumulates errors due to sensor and actuator noise. Third, traveling for a longer time increases the probability that the source position estimate is distorted by non-systematic components such as collisions, uneven terrain, and wheel slipping.

The robots can perform the object retrieval task using two different strategies: the *non-partitioning strategy* and the *partitioning strategy*. The non-partitioning strategy consists in performing the object retrieval task as a whole, unpartitioned task. The partitioning strategy consists in partitioning the object retrieval task into sub-tasks.

In Figure 1 we report a schematic representation of the behavior of the robots when foraging is performed employing the non-partitioning strategy. A video showing the behavior of the robots can be found in the online supplementary material (Pini et al., 2012b). Initially, a robot does not have any information about the location of the objects. Therefore, it needs to explore the environment to find the source. When the robot finds an object, it harvests it, sets its source position estimate to (0,0), and navigates towards the nest. Upon entering the nest, the robot stores the object and heads towards the source position estimate for harvesting another object. As mentioned, the source position estimate can diverge from the real source position. After reaching the position where it estimates that the source is located, the robot performs a *neighborhood exploration*. The neighborhood exploration consists in exploring an

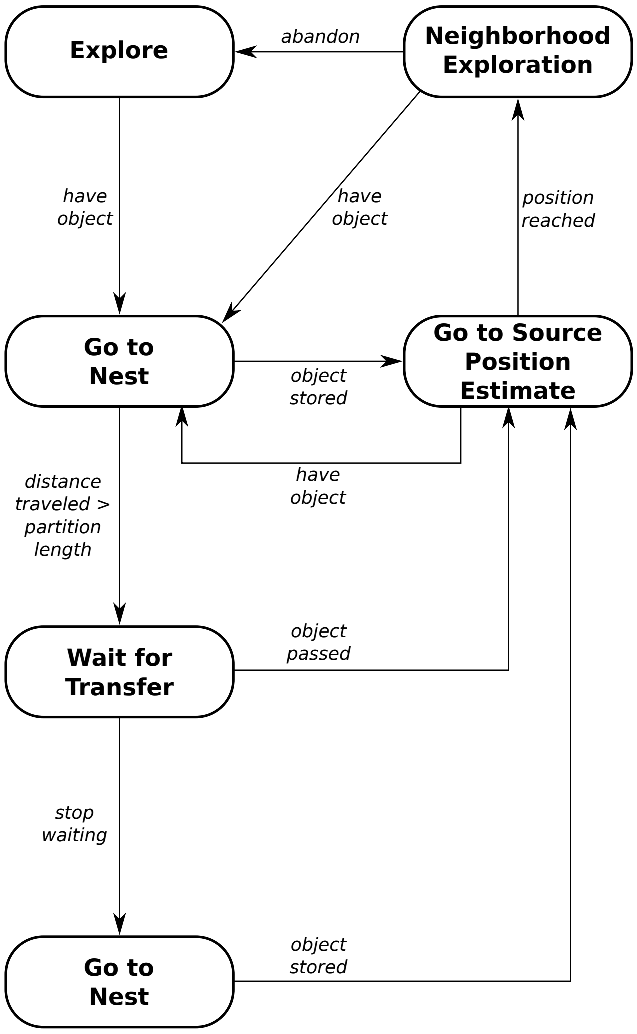


Fig. 2: Representation of the behavior of the robots when the partitioning strategy is employed

area of limited extension, centered around the source position estimate, with the goal of locating nearby objects. The exploration of the neighborhood can be abandoned by a robot, if it is unsuccessful. This may be caused by a large deviation between the source position estimate and the real source position. If the robot abandons the neighborhood exploration, it discards the source position estimate and explores the environment to search again for the source. In the rest of the paper we will say that the robot *got lost* when it is forced to abandon the neighborhood exploration and discard its source position estimate.

In Figure 2 we report a schematic representation of the behavior of the robots employing the partitioning strategy. The difference with the non-partitioning strategy is that, when heading to the nest, a robot stops if the distance from its source position estimate is bigger than a threshold, referred to as *partition length*. The robot waits in place for another robot, to which it can pass the object. The robot can decide to stop waiting; this prevents it from waiting indefinitely in place. In case a robot stops waiting, it navigates towards the nest to store the object.

As robots wait along the path to the nest, robots can find objects not only at the source, but also where other robots are waiting. Object transfer is direct: first the receiving robot must grip it, and only then the passing robot will release it. Therefore, an object is never left on the ground without at least one robot holding it. A robot that receives an object memorizes the location of the exchange; this location becomes the source position estimate for that robot. As for the non-partitioning strategy, the robots perform a neighborhood exploration upon reaching their source position estimate, be it the location of the source or the location where the robot received an object. A video showing the behavior of the robots employing the partitioning strategy can be found in the online supplementary material (Pini et al., 2012b).

The partitioning strategy results in the object retrieval task being partitioned into sub-tasks: an object can be transferred several times from robot to robot before it is delivered to the nest. Each robot only contributes to a portion of the overall work (i.e., it performs a sub-task of the object retrieval task). The number of sub-tasks in which the object retrieval task is partitioned depends on the distance between source and nest, as well as on the value of the partition length.

The two strategies described here entail different costs, which can render one preferable to the other. The non-partitioning strategy requires the robots to travel a longer distance from the nest to the source compared to the partitioning strategy. Traveling longer distances reduces the accuracy of the source position estimate. If the accuracy of the source position estimate is poor, the robot is likely to perform neighborhood exploration in a position that is far away from the actual position of the source. Consequently, neighborhood exploration fails, the robot gets lost, and the environment must be explored to locate the source again. The cost of such an event is linked to the size of the environment: the larger the environment, the higher the expected time needed to find the source.

Robots using the partitioning strategy, travel only a portion of the path that separates the source from the nest. This is advantageous because it reduces the effect of errors and non-systematic events mentioned above, since the distance traveled is shorter. The expected result is higher accuracy in the source position estimate, which in turn increases the chances to find an object during the neighborhood exploration. Consequently, the number of times the robots get lost is reduced. Disadvantages are the overheads due to object transfer: the time spent waiting for a partner that can receive an object, as well as to perform the transfer itself. An additional disadvantage is due to the potential overcrowding of certain areas of the environment (i.e., along the path from source to nest). Overcrowding impedes the movement of the robots and increases the chances of collisions. In turn, collisions have a strong negative impact on the source position estimate as they can cause the robots to be moved without being aware of it or to slip on the ground.

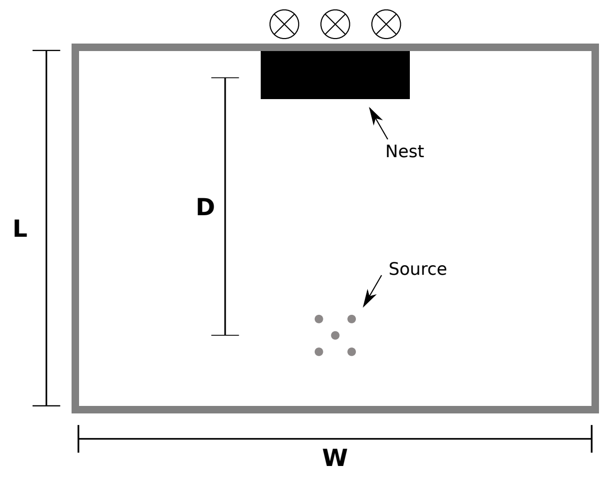


Fig. 3: Representation of the arena in which foraging takes place. The nest (black rectangle) is located at the center top and is marked by three lights (crossed circles), that can be perceived by the robots. The object source, located in front of the nest, is composed of 5 objects (grey circles), positioned as shown. The size of the arena (W, L) and the distance of the source from the nest (D) depend on the specific experiment.

4 Implementation of the system

In this section we describe the hardware and simulation software that we use to implement the system presented in Section 3. In Section 4.1, we describe the environment used for the foraging experiments. In Section 4.2, we present the *foot-bot* robotic platform and the objects that we use for carrying out experiments with the real robots. In Section 4.3, we describe ARGoS, the software we use for the simulation-based experiments. In Section 4.4, we provide details of the implementation of the controller used on the foot-bot, which implements the two foraging strategies presented in Section 3. In Section 4.5, we describe the dead-reckoning noise model we use in the simulation experiments.

4.1 Experimental environment

The robots perform foraging in a rectangular arena, surrounded by walls. The arena is represented in Figure 3. The length L and the width W of the arena depend on the specific experiment. The nest is located close to the border of the arena and it is marked by a black patch on the floor, which is 0.45 m long and 1.4 m wide. The robots can determine whether they are inside the nest by the color of this patch. Three lights (crossed circles in the figure) mark the location of the nest, and can be used by the robots to determine the direction of the nest.

The object source is located in front of the nest, at a distance D, measured from the center of the nest to the center of the source. The source is composed of 5 objects, positioned as shown in Fig. 3 at a distance of 8 cm (edge to edge) from the object in the center. Each time a robot harvests an object, a new object is added at the location where the previous object was harvested (i.e., the source never depletes).

The parameters of the environment (D, W, and L) influence the effectiveness of the strategy used for performing foraging. As mentioned in Sec. 3, the longer a robot travels, the lower the accuracy of its source position estimate becomes (i.e., the higher the number of times the robot gets lost). The relative improvement of using the partitioning strategy over using the non-partitioning strategy depends on the value of D: when D is small, the accuracy in locating the source can be good also using the non-partitioning strategy. Conversely, for high values of D, the partitioning strategy can lead to significant improvements over the non-partitioning strategy. In general, the higher the value of D (i.e., the longer the task), the more advantageous task partitioning becomes.

The parameters W and L define the difficulty of finding the source when exploring the environment: the larger the surface of the arena, the longer it takes (on average) to find the source. Therefore, the two parameters also determine which strategy is preferable. For example, in a large arena, the non-partitioning strategy could be disadvantageous even if D is small. In fact, even if the robots are accurate and get lost rarely (due to the short distance traveled), when they get lost they might need to search for a long time.

Conversely, for high values of D, the non-partitioning strategy can be preferable to the partitioning strategy if the arena is small. In fact, even if the robots are not accurate and get lost quite often, a small arena requires only a short period of time to be explored. Therefore, the cost of getting lost may be lower than the costs due to the partitioning strategy overhead, rendering the non-partitioning strategy preferable.

4.2 Robots and objects

The robots we employ in our experiments are the foot-bots. The foot-bot is a small (17 cm diameter) wheeled robot developed within the *Swarmanoid* project (Dorigo et al., 2012).[2] Figure 4 represents a foot-bot and highlights the sensors and actuators that have been employed in the experiments presented in this paper. In the following, we provide a brief description of those sensors and actuators. For a complete description of the foot-bot's hardware, refer to the work of Bonani et al. (2010).

The foot-bot can navigate autonomously using the *treels*, a combination of tracks and wheels that allows locomotion by means of a differential drive system. A gripper, combined with a rotating turret, is utilized for object gripping. The turret also hosts 12 RGB LEDs that can be controlled separately, which are employed by the robots to repel other robots (see Sec. 4.4). A traction sensor measures forces and torques applied to the turret. These measures are used when the robots transfer objects one to another. Obstacle avoidance is performed by utilizing data from the 24 infrared proximity sensors, which are also employed to measure the intensity of the ambient light and compute the direction of the nest. An RGB camera, pointing upwards to a

[2] `http://www.swarmanoid.org`

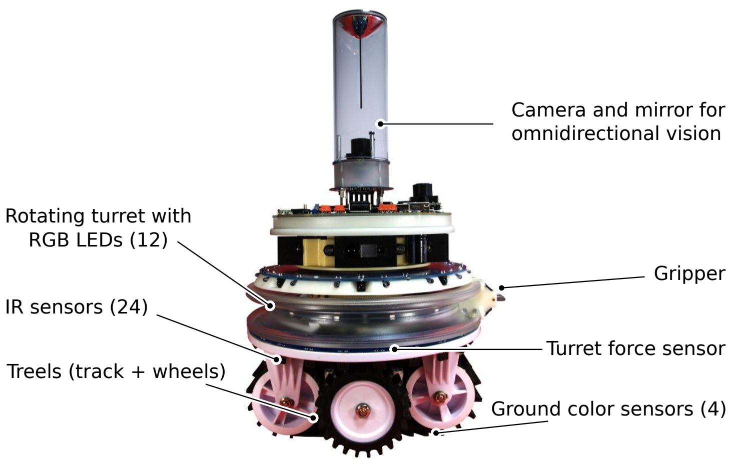


Fig. 4: Photography of a foot-bot. The sensors and actuators utilized in the experiments are highlighted.

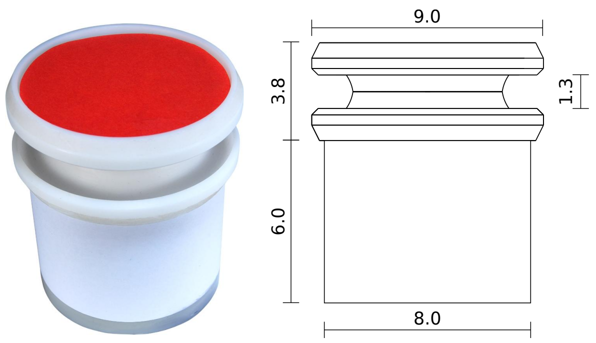


Fig. 5: Depiction of an object. Left: picture of the actual object. Right: schematic representation reporting dimensional information, values are in centimeters. An object is composed of a plastics ring mounted on a PVC pipe. A disk of red paper, perceivable by the robots with the omnidirectional camera, is glued on top of the object.

spherical mirror on top of a transparent tube, provides omnidirectional vision. The camera is employed to perceive objects and other robots' LEDs in the surroundings. Four infrared sensors, located underneath the robot, allow the perception of the color of the ground, and are employed by the robots to detect the nest. The readings of the encoders of the two motors are used to perform dead-reckoning.

To conduct the foraging experiments we employ objects that we specifically designed for the foot-bots (see Brutschy et al., 2012a, for more information). Figure 5 represents such an object. Each object features a 9 cm diameter plastic ring that can be gripped by the foot-bot. The ring is mounted on a PVC pipe that is 6 cm tall and has a diameter of 8 cm. A disk of red paper, perceivable by the robots' camera, is glued on top of the plastic ring. The distance from which an object can be perceived varies from robot to robot. Additionally, it depends on the relative position of the object with respect to the robot. These asymmetries are due to mirror imperfections and tubes that are not perfectly vertical. Overall, the perception range is between 45 cm and 55 cm.

4.3 Simulator

For the simulation-based experiments described in Section 5 we use ARGoS (Pinciroli et al., 2011), an open source software[3] also developed within *Swarmanoid*. ARGoS is a discrete time physics-based simulation environment designed for the real-time simulation of large heterogeneous swarms. ARGoS is highly customizable, and allows the user to tune the level of detail of the simulation to the experiment at hand. In particular, the user can select how physics is simulated: from simple kinematics in 2D environments to complex dynamics in 3D environments. The experiments described in this paper have been conducted using 2D dynamics physics. The simulation proceeds in discrete steps of 0.1 simulated seconds. A 5% uniform noise is added at each step to the measures of the light, proximity, and ground color sensors. Gaussian noise with a standard deviation of 2 cm is added to the measured distance of the objects perceived by the omnidirectional camera.

4.4 Robots controller implementation

In this section we provide details about the robots's controller, and highlight some differences that exist between simulation and real robots.

Exploration and neighborhood exploration: Both in simulation and with the real robots, the exploration of the environment is performed through random walk. Random walk is implemented as follows. The robot travels straight, with a maximum speed of $10\,\mathrm{cm/s}$. At each control step, the robot has a 5% probability of turning its direction of motion by a random angle, uniformly sampled in the interval $[-115°, 115°]$.

The neighborhood exploration is a special case of random walk: the robot performs random walk inside a circular area with a radius of 0.5 m, centered on the position estimate of the robot. Each time the robot reaches the boundary of the circular area, a new random direction pointing inward is generated. This prevents the robot from leaving the area.

As mentioned in Section 3, a robot can abandon the exploration of the neighborhood. Abandoning is governed by a timeout mechanism: when a robot has been exploring a neighborhood for 1 minute without locating objects, neighborhood exploration is abandoned (i.e., the robot assumes that it got lost). The timeout is increased to 3 minutes when the source position estimate is the location in which an object was received. The reason for a different duration of the neighborhood exploration is that the object source is always in the same location in the environment, while the

[3] ARGoS can be freely downloaded from `http://iridia.ulb.ac.be/argos/`

position of object transfer is not constant, due to robots' movements. Therefore, if the object source is not found within a minute it is because the robot ended up far away from its location and cannot perceive it during the neighborhood exploration. This is not necessarily true when a robot is exploring the neighborhood of a location where it received an object. In fact, a robot passing objects needs some time for harvesting a new object and come back to a location where the first robot can find it. This time span can be longer than a minute, which is taken into account in a longer neighborhood exploration time.

Object gripping - real robots: Gripping relies on the omnidirectional camera to perceive the distance and the bearing of the objects. An object is perceived by the robots as a red colored blob. Additionally, a repulsion mechanism is employed to avoid that the same object is gripped by several robots at the same time. The repulsion mechanism consists in the robots ignoring every red blob that is perceived near a blue blob. When gripping an object, a robot repels other robots from the same object by turning its LEDs to blue. The robot maintains the LEDs on while transporting the object towards the nest. The LEDs are turned off after dropping the object in the nest or when waiting for another robot that can receive the object (in this case, in fact, it is desirable that a second robot grips the same object).

Each time a robot grips an object, it tries to turn its turret first clockwise and then counter clockwise. Trying to turn the turret allows the robot to determine whether the object is being held by another robot (in which case the turret motion would be impeded). The robot needs to know whether the object it has gripped is held by another robot to decide what to do next. In case the non-partitioning strategy is used, the object is released. In case the partitioning strategy is employed, the gripping robot triggers the procedure by which the objects are transferred between robots (described next). A video showing a foot-bot gripping an object is available in the online supplementary material.

Object gripping - simulation: In simulation, the gripping procedure is a simplified version of the one implemented on the foot-bots. A first simplification resides in the fact that, upon gripping an object, robots do not check if it is held by another robot by moving the turret, but receive the information directly from the simulator. Additionally, when a robot grips an object, it is forced to wait in place for some time before it can undertake the next action. The amount of time that a robot waits after gripping an object is randomly sampled from a list of values that have been recorded with the real robots while performing the experiments described in Sec. 5.1. Figure 6 (left) plots the empirical distribution for these samples. The time samples are divided into two sets: one set contains samples of the grip times recorded when gripping objects from the source (80 samples), the other one contains samples of the grip times of objects held by other robots (81 samples). Each set of samples is used in the corresponding situation. Using this solution we can simulate the aspect of object gripping that is most relevant for our work: the time it takes to perform it.

An additional difference between simulation and real robots concerns the sensing capabilities of the omnidirectional camera. In simulation, the perception range is symmetrical and uniform across the robots, and it is set to 52.5 cm.

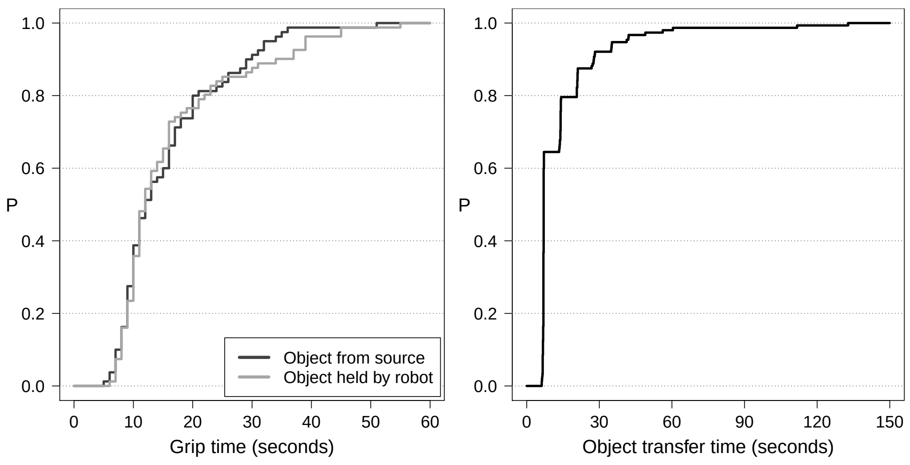


Fig. 6: Empirical distribution (P) of object grip times (left) and object transfer times (right) obtained with the real robots. The samples used to produce the two plots are used in simulation to implement object gripping and object transfer, respectively. The object grip times are divided into two sets. The dark gray line plots the empirical distribution of the grip times recorded when gripping objects from the source. The light gray line plots the empirical distribution of the grip times recorded when gripping objects held by other robots.

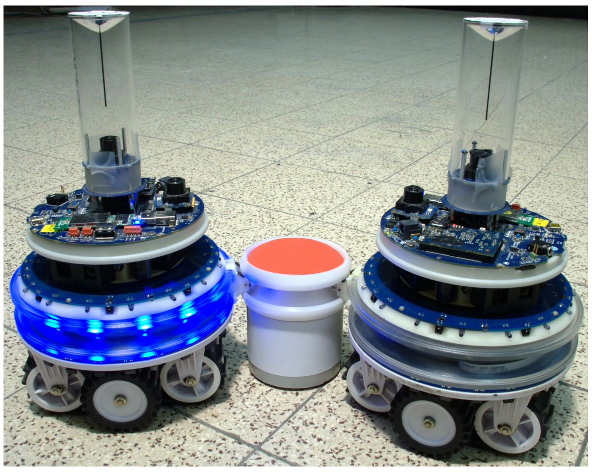


Fig. 7: Two robots transferring an object. The robot on the right is passing its object to the robot on the left

Object transfer - real robots: Object transfer takes place only when the robots are employing the partitioning strategy. Figure 7 shows two robots while transferring an object one to the other. When a robot grips an object being held by another robot, the procedure that implements object transfer is triggered. This procedure consists in the receiving robot repeatedly pushing and pulling the gripped object, with the result of producing force spikes on the turret of the passing robot. The spikes can be detected with the torque sensor. While trying to pass an object, a robot maintains a counter of the number of spikes perceived. If the robot does not perceive a spike within a second, it decrements the counter if its value is greater than zero. When the counter reaches the value of 8, the receiving robot releases the object and the passing robot can start moving towards the nest with the object.[4] During object transfer, the receiving robot lights up its LEDs in blue, to repel other robots from the object it is trying to receive.

As mentioned in Sec. 3, a robot that has been waiting too long without managing to pass its object, stops waiting and navigates to the nest to store its object. This mechanism is implemented with a timeout: the robots are allowed to wait for 3 minutes, after which they stop waiting and reach the nest to store the object. A video showing two foot-bots transferring an object to each other is available in the online supplementary material.

Object transfer - simulation: Analogously to object gripping, also object transfer is simulated as a simplified version of what happens in reality. Similarly to gripping, transfer times are sampled from the real robots and the resulting set of samples (152 samples) is used in simulation. When the receiving robot has gripped the object, a value is randomly selected from this set of samples. The two robots wait in place for the corresponding amount of time before the transfer is complete and the robots can leave. Figure 6 (right) plots the empirical distribution for the transfer time samples used in simulation. Again, this solution captures the aspect of object transfer that is important for our study, that is, the amount of time required to perform it. As for the real robot implementation, a robot holding an object is allowed to wait for a transfer partner a maximum time of 3 minutes. If that time passes and the robot did not manage to transfer its object, the robot stops waiting and navigates towards the nest.

Check for an unreachable destination: Due to dead-reckoning errors, it is possible that the source position estimate of a robot lies outside the arena boundaries. To prevent the robots from trying to reach a position that is outside the boundaries, the robots periodically check their progress towards their target location. If the distance to the target location is not decreasing in time, a robot discards its source position estimate. A non-decreasing distance to the target position means that an obstacle (wall or other robot) is preventing the robot from moving towards its destination. When the distance does not decrease for a long time, it is likely that the obstacle is a wall rather than a robot, which means that the target position lies outside the arena boundaries. The mechanism described here is used both in simulation and with the real foot-bots.

Notice that it is possible that, when this mechanism is triggered, a robot's movements are impeded by another robot. This represents a false positive: the first robot is not trying to reach a position that lies outsides the boundaries, but instead the obstacle avoidance is not managing to overcome the second robot.

[4] The parameters of the spike detection system have been determined empirically, with the goal of finding a satisfactory balance between the duration of the transfer procedure and its reliability.

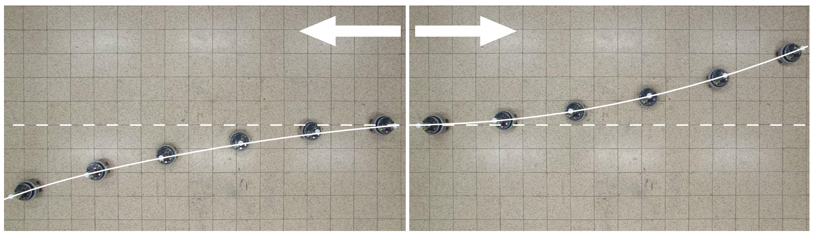


Fig. 8: Trajectory followed by a foot-bot when the speed of the two wheels is set to $-10\,\mathrm{cm/s}$ (left part of the figure) and $10\,\mathrm{cm/s}$ (right part of the figure). The white arrow on top of each figure marks the direction of motion of the robot. The white continuous line marks the trajectory of the robot. The dashed line reports a straight trajectory, for comparison.

4.5 Dead-reckoning noise model

In order to simulate the system studied in this paper, we need a model of the dead-reckoning noise that accounts for the error on the source position estimate of each robot. While performing experiments with the real foot-bots, we noticed a bias in the motion of each robot. Figure 8 shows a typical trajectory of a robot, when the wheel speeds are set to $-10\,\mathrm{cm/s}$ (left part of the figure) and $10\,\mathrm{cm/s}$ (right part of the figure). The figure is composed of 12 snapshots taken from a video, which is available in the online supplementary material. The figure reports the direction of motion of the robot (white arrow on top). The trajectory followed by the robot is marked with the continuous line; for reference a straight trajectory is also marked (dashed line). The trajectory clearly shows a drift towards the left-hand side with respect to the direction of motion. The amount by which the robot drifts is not constant: the same robot can drift more or less towards the left-hand side in subsequent trips. Notice that the robot does not measure such a drift: the encoder measures would tell the robot that it is moving straight. All the robots drift towards the left-hand side, therefore we speculate this behavior is linked to some asymmetry in the two motors of the wheels.

Solutions such as the calibration procedures described by Borenstein and Liqiang (1996) or Kalman filtering (Kalman, 1960) could improve dead-reckoning. Nevertheless, they cannot completely remove errors. Eventually, the same problems would arise after traveling longer distances than the one considered in this study. The key idea that task partitioning can improve localization would therefore still be valid at a larger scale. For simplicity, in this study, we do not try to correct or reduce the dead-reckoning errors in any way. In real-world applications, techniques for dealing with dead-reckoning errors can be coupled with task partitioning to further improve the system.

To implement the noise model, we collected samples of the robots' dead-reckoning errors when trying to reach the source position estimate. The experimental area is a rectangular arena (W = 6.7 m, L = 4.5 m), the source is located at a distance D = 4.0 m from the nest, and the swarm is composed of 6 robots. We sample data from four of the runs in which the object retrieval task is not partitioned into sub-tasks. Each sam-

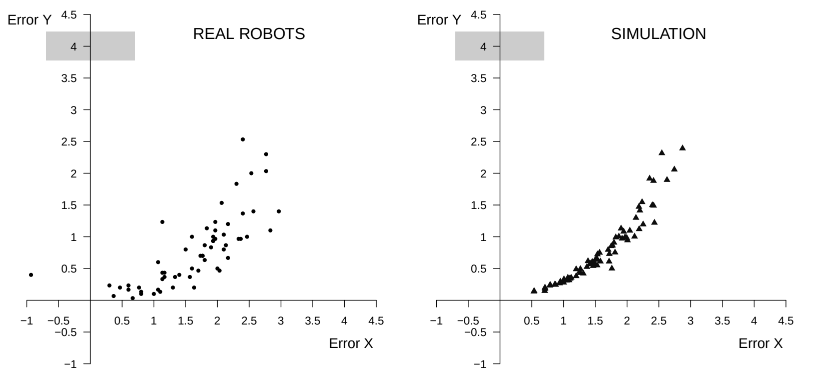


Fig. 9: Dead-reckoning error of real (left) and simulated robots (right) for several trips from source to nest. The source to nest distance D is 4.0 m. The robots employ the non-partitioning strategy. Each point reports the X,Y difference between the position at which a robot harvested an object (that corresponds to 0,0 in the picture) and the position at which it returned, after storing the object in the nest. The closer a point to the origin, the smaller the dead-reckoning error. The nest is represented as a light grey rectangle at the top of the figure.

ple records error on the two dimensions x and y on a trip from the source to the nest and back. The floor of the arena is covered by tiles that are used as a reference for calculating the error. The samples are collected manually from videos of the robots performing the experiment described in Sec. 5.1. Gathering samples from a video in a manual fashion is not as precise as taking direct measures with tests designed ad-hoc for the purpose. On the other hand, the samples include the results of stochastic events (e.g., collisions, shadowing, avoidance) that cannot be easily derived in ad-hoc tests.

A total of 61 samples were collected from the videos of the experiments. The points are reported in Figure 9 (left); the closer a point to the origin, the smaller the dead-reckoning error for the corresponding trip. In the figure, the origin marks the position at which the object was gripped (which becomes the source position estimate of the robot). The bias towards the left-hand side can be clearly seen in the plot (the nest is located at the top of the figure).

The dead-reckoning noise model used in simulation is implemented as follows (refer to Alg.1). The bias towards the left is produced by acting on the actuated values for the left and the right wheel speeds (lines 2 to 7 of Alg.1). The right wheel speed is increased when its actuated value is positive, the left wheel speed is decreased when its actuated value is negative (therefore its absolute value increases).

Notice that the robot computes dead-reckoning on the basis of *actuatedLeft-WheelSpeed* and *actuatedRightWheelSpeed*. It is therefore not aware of the drift. The trajectory that the robot follows is defined by the parameter μ_{noise}, which is the percentage by which the wheel speed is increased, with respect to the value actuated by

Algorithm 1 Pseudo-code for the dead-reckoning noise

```
1: $actuatedRightWheelSpeed, actuatedLeftWheelSpeed \leftarrow ControllerStep()$
2: if $actuatedRightWheelSpeed > 0$ then
3:     $rightWheelSpeed = actuatedRightWheelSpeed + \mu_{noise} * actuatedRightWheelSpeed$
4: end if
5: if $actuatedLeftWheelSpeed < 0$ then
6:     $leftWheelSpeed = actuatedLeftWheelSpeed + \mu_{noise} * actuatedLeftWheelSpeed$
7: end if
8: if Object gripped then
9:     $\mu_{noise} = RAYLEIGH(\sigma)$
10: end if
```

the robot. The higher the value, the larger the error in the robot movements and thus in its source position estimate.

Each time a robot grips an object, be it from the source or from another robot, a new value for μ_{noise} is sampled from a Rayleigh distribution with parameter σ (lines 8-10 of Alg.1). The sampled value of μ_{noise} characterizes the dead-reckoning error of the robot until the next object is gripped. The initial value of μ_{noise} is sampled from the same distribution. The value of σ has been determined by trial and error and is set to 0.0134. The goal was producing an error pattern similar to the one of Fig. 9 (left) and percentages of success in finding the source close to the ones recorded with the real robots. Figure 9 (right) reports 77 error points taken from simulation with experimental conditions analogous of those in which the real-robot noise samples where taken.

Notice that our goal is not to exactly model what produces the dead-reckoning error on the robots. Dead-reckoning errors are the result of systematic and non-systematic components (Borenstein, 1998). Modeling these components is beyond the scope of this work. Instead, we use a minimalistic model that allows us to mimic in simulation the behavior observable with the real robots in our setup.

5 Experiments and results

In this section we describe the experiments we ran to evaluate the system and we report the results obtained. In Sec. 5.1 we describe the experiments performed with real foot-bots. The goal of the experiments is to compare the partitioning strategy and the non-partitioning strategy in real world settings. The experiments described in Sec. 5.1 are the only ones carried out with real robots; the rest of the experiments have been performed in simulation. In Sec. 5.2 we describe simulation experiments in which we test a setup identical to the one of the real robot experiments and we compare the results across simulation and reality. These experiments are used to validate the simulator, which is then used for other experiments to test experimental conditions that are not feasible in reality. In the experiments presented in Sec. 5.1 and Sec. 5.2, half of the swarm starts close to the source and half in the center of the arena. This setup allows us to study the behavior of the system at the steady state, skipping the initial phase in which the robots explore the environment and search for the source.

Table 1: Default values of the experimental parameters

Parameter	**Default value**
Arena width W	4.5 m
Arena length L	6.7 m
Source to nest distance D	4.0 m
Number of robots	6
Duration of an experiment	90 minutes
Number of repetitions per experiment	200

This choice is motivated by the need of reducing the duration of the experiments with the real robots.

In Sec. 5.3, we study how the system reaches the steady state in case the swarm starts inside the nest, and the exploration phase is not skipped. The goal is to evaluate whether the system stabilizes using either strategy and to determine the time it takes to reach a stable state. In Sec. 5.4, we study the effects of different environments and swarm sizes on the performance of the system. The results of the experiments highlight costs and benefits of task partitioning with direct transfer. Finally, in Sec. 5.5, we describe experiments ran with different values for the source to nest distance D and for the number of sub-tasks into which the overall task is partitioned. We study, depending on the distance D and the environment, which is the best way to partition the task in terms of number of sub-tasks.

Table 1 reports the default values for the parameters used in the experiments. When not explicitly mentioned, a parameter has the value reported in the table. The default environment consists in a 4.5 m by 6.7 m rectangular arena, with the source positioned at a distance D = 4.0 m from the nest. The swarm is composed of 6 robots. Each experiment lasts 90 minutes. We perform 200 experimental runs for each experimental condition.

5.1 Real robot experiments

The experiments using real robots start with three foot-bots located in proximity of the source and three foot-bots located in the center of the arena, at a distance D/2 = 2.0 m from the nest (see Fig. 10). In case the partitioning strategy is employed, the 3 robots starting at the center of the arena initially perform neighborhood exploration, centered around their starting position. The partition length is set to 2.0 m. A robot that receives an object from another robot delivers it to the nest (i.e., an object can be transferred only once). As a result, the task is partitioned into two sub-tasks of (approximately) equal length. If the non-partitioning strategy is employed, the 3 robots in the center of the arena start by exploring the environment. Refer to the real robot experiment videos in the online supplementary material for more information.

For each strategy, we performed 6 experimental runs, each with different sets of robots starting at the source and in the center of the arena at the beginning of the run. A summary of the results of each experimental run can be found in the online

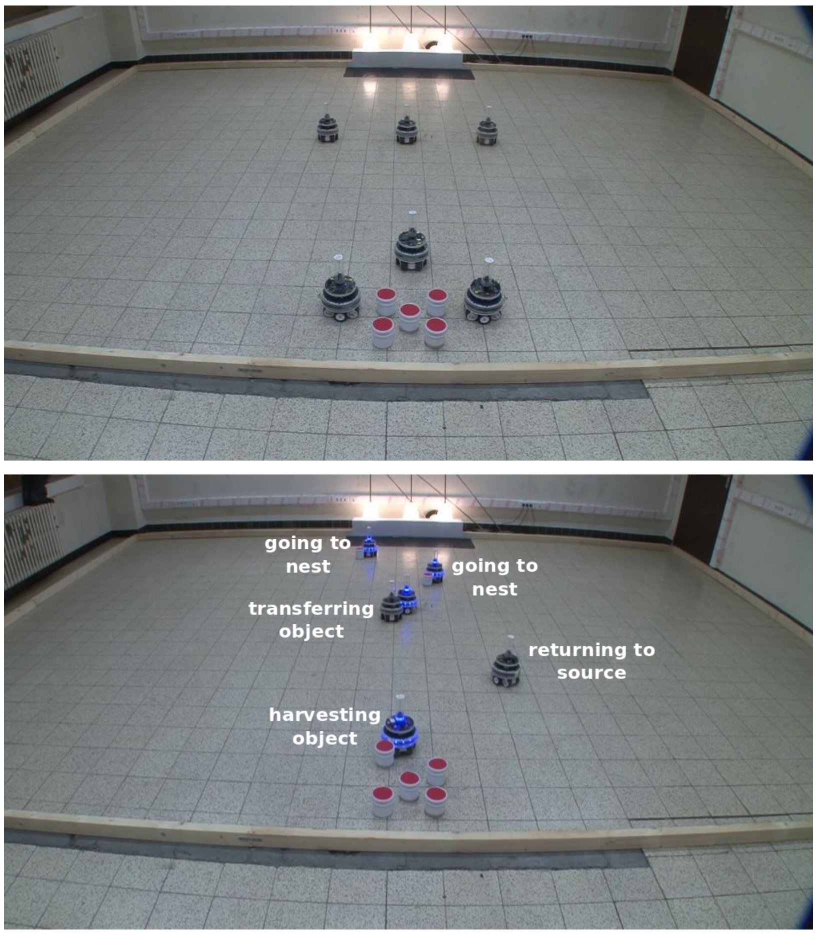


Fig. 10: Snapshots of a real robot experimental run. The robots employ the partitioning strategy. (Top) initial configuration: 3 robots are positioned in proximity of the object source, the rest at half distance between source and nest. (Bottom) situation after 1.5 minutes. Two robots passed their object: one is returning to the source, the other is harvesting another object. The two robots that received an object are heading to the nest. The remaining two robots are in the process of transferring an object.

supplementary material. Each time an object is harvested from the source, a new one is placed at the same position, to maintain a constant amount of 5 objects at the source. If a robot loses an object during transportation, the experimenter removes the object from the arena. Two videos with complete runs can be found in the online supplementary material. One of the two videos shows the robots performing foraging using the partitioning strategy, the other video shows the robots performing foraging without partitioning. Figure 10 reports two snapshots taken from the first video at the beginning of the run (top) and after 1.5 minutes of experiment (bottom).

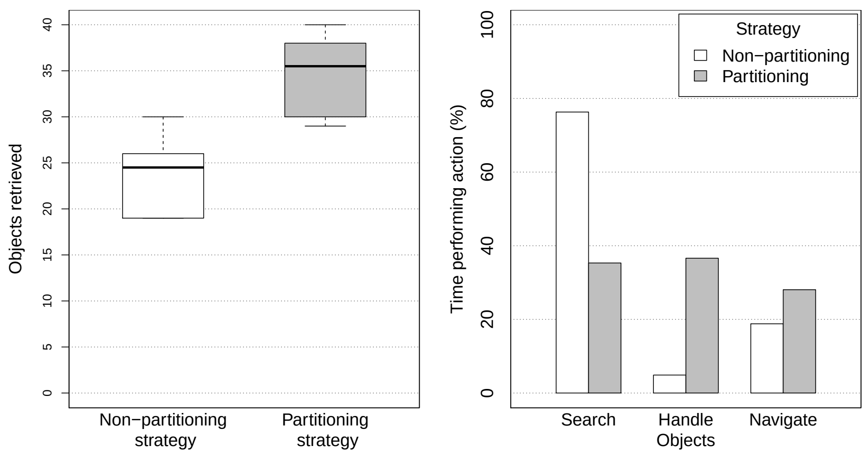


Fig. 11: Results of the real robot experiments, given as the total amount of object collected using the two strategies (left) and actions taken by the robots (right). In the plot on the right, each bar reports the percentage of time, computed across all the runs, the robots spent performing the corresponding action. The actions are grouped into three sets. *Search* accounts for the time spent exploring the environment and performing neighborhood exploration. *Handle Objects* accounts for the time spent gripping objects and includes the time spent waiting and transferring objects in case the partitioning strategy is employed. *Navigate* accounts for the time spent going towards the nest with an object or reaching the source position estimate.

Figure 11 (left) reports the total amount of objects delivered to the nest by the robots using the two strategies. We performed an unpaired Wilcoxon test (two sided), to assess whether there is a statistically significant difference between the performance of the swarm using the partitioning strategy and using the non-partitioning strategy. The test indicates that there is a significant difference between the two, with confidence level of 1%.

Figure 11 (right) provides a summary of the actions performed by the robots. Each pair of bars reports the percentage of time, computed across all the experimental runs, that the robots spent performing each action for the non-partitioning strategy and the partitioning strategy. The actions of the robots are sampled every second during the course of the experiment. The actions are grouped into three sets. *Search* includes the time the robots spent exploring the environment and performing neighborhood exploration. *Handle Objects* includes the time the robots spent approaching and gripping objects. In case the partitioning strategy is employed, object handling also includes the time the robots spent waiting for a partner that received the object and the time required to perform the transfer. *Navigate* accounts for the time the robots spent going to the nest while carrying an object, or going towards their source position estimate.

The graph highlights the costs and benefits of each of the two strategies. The non-partitioning strategy has relatively low object handling costs, that are only due to object gripping, but very high search costs. As mentioned, this is due to the fact that

Table 2: Probability of getting lost at the source for the two strategies. Data computed over the 6 runs for each strategy. The original robot identifiers have been reported for future reference.

Robot	**Non-partitioning**	**Partitioning**
footbot4	90.91	26.09
footbot10	85.19	15.15
footbot18	73.33	7.14
footbot22	86.96	14.63
footbot23	69.23	6.78
footbot37	47.62	2.22
OVERALL	76.12	10.7

robots using the non-partitioning strategy travel longer distances. Consequently, the accuracy of the source position estimate degrades as effect of dead-reckoning errors and the robots get lost more often (and further away from the source). On the other hand, the partitioning strategy entails high handling costs, but the robots spend less time searching. The result is that the percentage of time the robots spend navigating, which determines the foraging performance, is higher.

Table 2 reports the percentage of times each robot got lost after searching for the object source, for the non-partitioning strategy and the partitioning strategy. The percentage is computed across all the experimental runs. The table highlights that the tight cooperation resulting from task partitioning allows the swarm to overcome the localization limits of single robots. In fact, the overall localization capabilities of the swarm are increased and this results in the higher foraging performance reported in Fig. 11.

5.2 Validation

In this set of experiments we replicate the setup described in the previous Section 5.1 in simulation. The goal is to compare the results between simulation and reality in order to validate our simulation, particularly regarding the noise model presented in Sec. 4.5. Figure 12 (left) reports the total amount of objects transported to the nest by the robots using the two strategies. The results confirm those of Fig. 11 (left): when the robots use the partitioning strategy, they transport more objects to the nest than when the non-partitioning strategy is employed.

Figure 12 (right) summarizes the actions performed by the robots. The results confirm what observed in the experiments with the real robots: the partitioning strategy is costly in terms of objects handling, but it reduces the search time and increases the navigation time. Table 3 reports, for both strategies, the probabilities of getting lost measured in simulation and in the real robot experiments. The table reports, for each entry, the mean of the observations (in bold) and the 95% confidence interval on the value of the mean. The table also confirms that simulation replicates the results observed in reality in terms of probability of getting lost, for both the non-partitioning

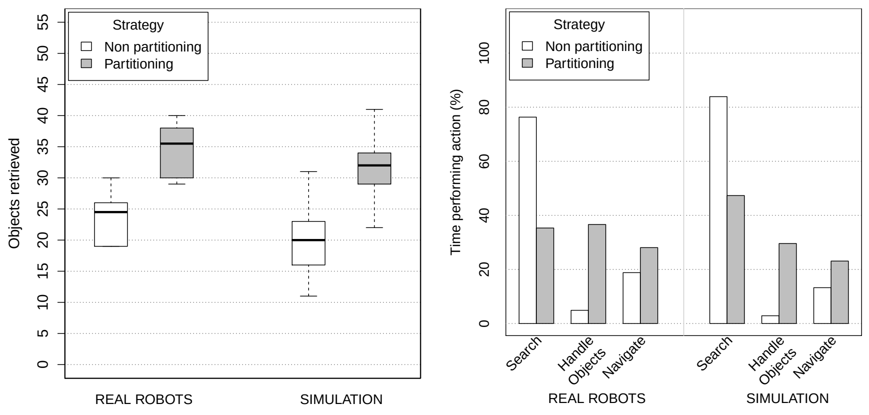


Fig. 12: Results of the validation experiments: total amount of object collected using the two strategies (left) and actions taken by the robots (right). In the plot on the right, each bar reports the percentage of time, computed across all the runs, the robots spent performing the corresponding action. The actions are grouped into three sets. *Search* accounts for the time spent exploring the environment and performing neighborhood exploration. *Handle Objects* accounts for the time spent gripping objects and includes the time spent waiting and transferring objects in case the partitioning strategy is employed. *Navigate* accounts for the time spent going towards the nest with an object or reaching the source position estimate. In each plot, also the corresponding data of the real robot experiment is reported (see Fig. 11)

Table 3: Probability of getting lost in real robot and simulation experiments. The data for both strategies is reported. In case the partitioning strategy is employed, the probability of getting lost is reported also for the cases in which the source position estimate of the robots refers to the location of an object transfer. For each measure, the mean and the 95% confidence interval on the value of the mean is reported.

Measure	**Real robots**	**Simulation**
NO PARTITION		
Get lost - searching the source (%)	70.93 / **77.15** / 85.22	76.66 / **77.91** / 79.1
PARTITION		
Get lost - searching the source (%)	6.31 / **11.37** / 17.16	13.29 / **14.05** / 14.81
Get lost - searching other robots (%)	11.48 / **14.14** / 17.65	11.35 / **11.94** / 12.56

strategy and the partitioning strategy. A similar table, reporting additional measures not presented here, can be found in the online supplementary material.

Figure 13 reports the empirical distribution of the time a robot takes to find the source after it got lost, for the non-partitioning strategy (left) and the partitioning strategy (right). Simulation and real robot experiment data is reported for comparison. The graph shows that, on average, in simulation the robots take more time to find the object source when getting lost. This can explain the discrepancy between simulation

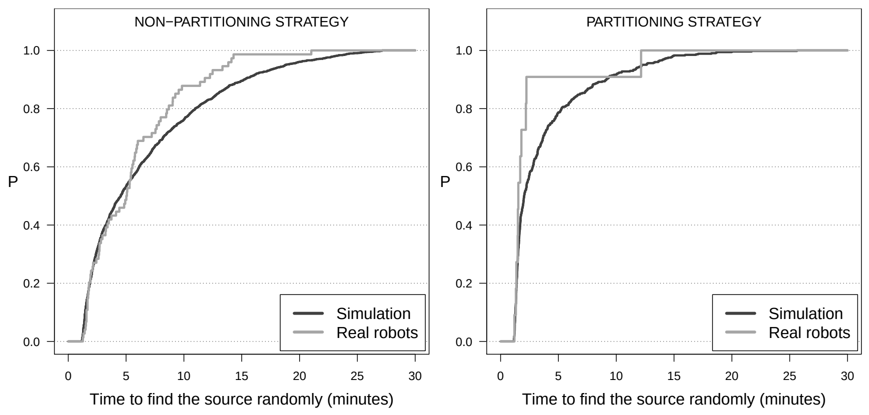


Fig. 13: Empirical distribution (P) of the time needed to find the object source when searching randomly. The plotted data only includes the time needed to find the source after a neighborhood exploration failed (i.e., it does not include the time required to find the source for the first time). The plot on the left reports the data collected when the non-partitioning strategy is employed by the robots, the plot on the right when the partitioning strategy is employed. The dark gray line plots the data collected in simulation, the light gray line the data collected in the real robot experiments.

and reality in terms of number of objects retrieved and actions performed observed in Fig. 12. Notice that the robots using the partitioning strategy employ less time, on average, to find the source after they got lost than the robots employing the non-partitioning strategy. This indicates that when the partitioning strategy is used, the robots get lost in locations that are closer to the object source compared to the non-partitioning strategy.

The experiments reported in this section served to validate the simulation of our system. We pointed out some discrepancies in the numerical results. For the purposes of this study, we consider the differences acceptable. The trends we observed with real robot experiments are also observable in their simulation counterpart. We are therefore confident that our simulation indeed captures the properties of the studied system.

5.3 Convergence

The experiments described in Sec. 5.1 and 5.2 start with half of the swarm in proximity of the source, and the other half in the center of the arena. We chose this setup to study the behavior of the system at the steady state, that is, once the robots have converged to a stable rate of objects delivered to the nest. In the experiments described in this section, we study the behavior of the system when all the robots are deployed in the nest. In fact, in real world applications, it is likely that the robots are deployed in a single location and need to explore the environment first. The goal of the experiments

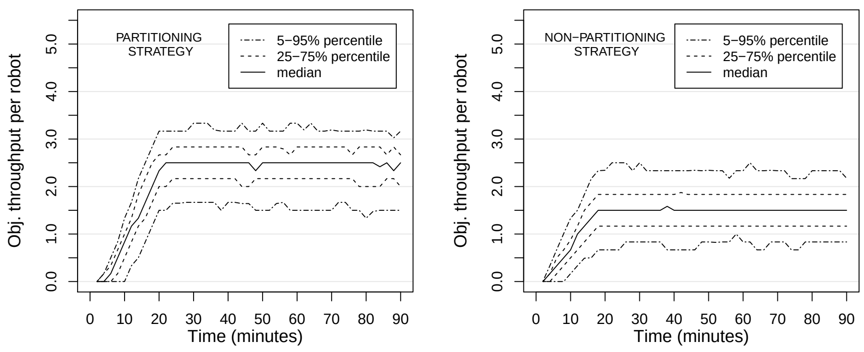


Fig. 14: Throughput of the swarm using the partitioning strategy (left) and the non-partitioning strategy (right). The throughput is computed in a time window of 15 minutes and expressed as objects per robot. The 5, 25, 50, 75, 95% percentiles are reported in each plot.

described in this section is to assess whether the system is able to reach a steady state and to determine how such a state is reached.

In Figure 14, we report the throughput of the system during the course of the experiments. The throughput of the system is sampled every 2 simulated minutes and it is expressed as the number of objects delivered to the nest per robot, in the preceding 15 minutes. The graph on the left reports the results of the swarm using the partitioning strategy, the graph at the right the results for the non-partitioning strategy. Each curve reports the median (continuous line) and the 5-95% and 25-75% percentiles (dashed lines).

For both strategies, the system reaches the steady state. The non-partitioning strategy reaches the steady state in approximately 18 minutes, while the partitioning strategy takes approximately 22 minutes. The value of the throughput at the steady state matches the one recorded in the simulation experiments described in Sec. 5.2 (see Plot V1 of the online supplementary material). It is therefore reasonable to conclude that the behavior we observe in the experiments described in Sec. 5.1 and 5.2 is a faithful representation of the behavior of the system at the steady state.

The results show that the system is able to reach a steady state also if the robots start with no notion about the location of the object source. This indicates that the partitioning strategy can be applied in realistic scenarios in which the robots initially need to explore the environment looking for objects to forage.

5.4 Environment and swarm size

The goal of the set of experiments described in this section is to evaluate the influence of the size of the environment and of the number of robots. As mentioned, the size of the environment influences the time it takes for the robots to find the object source, or another robot waiting to pass an object, when searching randomly: the larger the environment, the higher the expected time. We test four different environments of the following dimensions (L x W): a 4.5 m by 4.5 m environment, referred to as the *small environment*, the 4.5 m by 6.7 m *standard environment*, a 6.7 m by 6.7 m *large environment*, and a 6.7 m by 9.97 m *huge environment*. In all the environments, the source to nest distance D is 4.0 m. The swarm size is taken from the set $\{2, 4, 6, 8, 10, 15, 20, 30\}$. All the robots are placed inside the nest at the beginning of the experiment.[5]

Each graph in Fig. 15 reports, for a given environment and different swarm sizes, the individual performance using the non-partitioning strategy and the partitioning strategy. The individual performance is computed as the total amount of objects collected by the swarm, divided by the number of robots.

In general it is expected that, for a given number of robots, a bigger environment leads to a lower individual performance. Surprisingly, this is not true when comparing the standard and the large environments: for a given number of robots, the individual performance is slightly higher in the large environment. This is linked to the fact that in the normal environment the source is very close to a wall. This position makes it harder to find it: due to obstacle avoidance, the robots tend to navigate away from the walls. Consequently, the robots take more time, on average, to find the source in the normal environment than in the large environment (see plots E1 of the online supplementary material). Therefore, the geometry of the environment affects the cost of getting lost expressed by the average time needed to find the source when searching randomly. The probability of getting lost is instead dependent on the value of D. Since D is the same in all the environments, the probability of getting lost is constant across the different environments (see table 2 of the online supplementary material).

The size of the swarm has an impact on the performance as well, but the effects depend on the strategy employed. The results obtained by the swarm employing the non-partitioning strategy show a monotonic decrease of performance for an increasing swarm size. The performance decreases more steeply in smaller environments. This is an effect of physical interference: the smaller the environment, the higher the robot density, and therefore the frequency at which the robots encounter and interfere with each other. In a swarm employing the non-partitioning strategy, each robot performs the task on its own and consequently the interactions between the robots are only due to physical interference. The partitioning strategy instead requires the robots to collaborate with each other. Therefore, there are both negative interactions due to interference and positive interactions in the form of collaboration. This explains why, when the partitioning strategy is employed, the individual performance does not drop monotonically with an increasing swarm size.

Notice that the partitioning strategy requires a sequence of events to occur before the robots can start delivering objects to the nest at a stable rate. First, a robot needs to find the source and harvest an object. Second, once that robot is waiting for a transfer partner, another robot must encounter it and receive the object. This has to happen before the first robot stops waiting. The probability of both events to happen increases with the density of robots. In environments where the area to explore is small (small and normal environments), the two events are likely to happen also when using small

[5] If the swarm size is too large, only some robots are placed inside the nest, while the rest of the swarm is placed in close proximity.

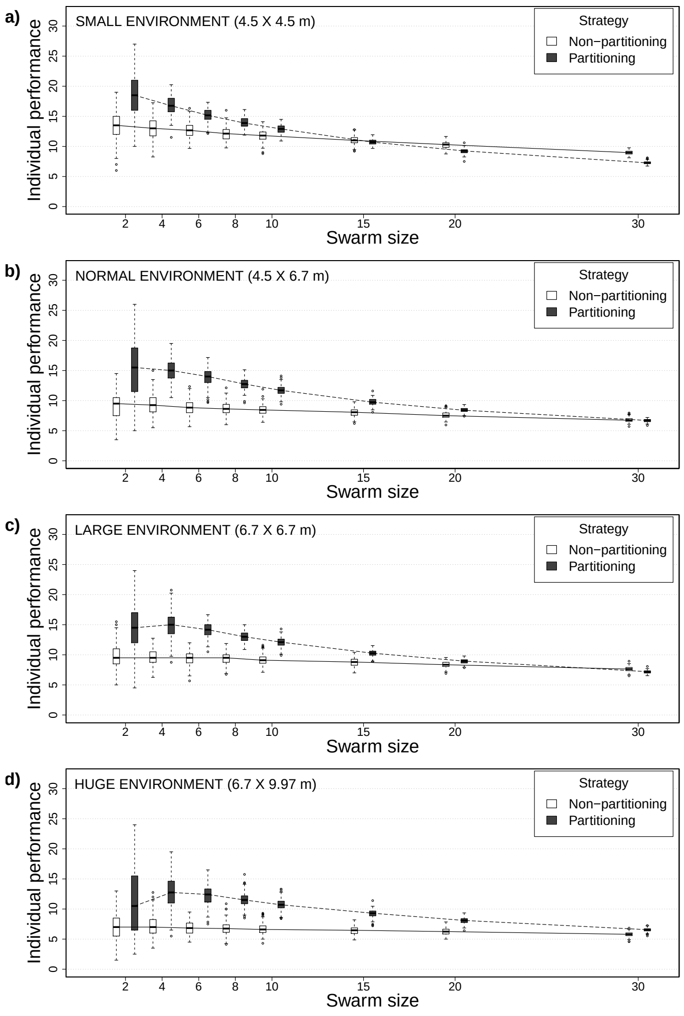


Fig. 15: Individual performance for different swarm sizes using the partitioning strategy and the non-partitioning strategy in the a) small, b) normal, c) large, and d) huge environments. The individual performance is computed as amount of objects retrieved divided by the number of robots.

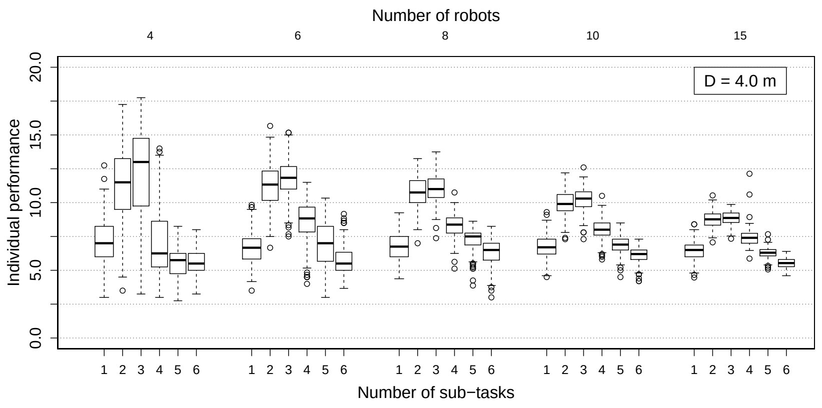
swarms. Therefore, the resulting individual performance is high. Instead, in the large and huge environments there is an initial increase of performance for an increasing swarm size that indicates that two robots are too few for the partitioning strategy to work.

The size of the swarm also influences the stability of the partitioning strategy: the smaller the swarm, the less stable the partitioning strategy. Also with the partitioning strategy there is a non-null probability for the robots to get lost after reaching the source position estimate. If one of the robots gets lost, the sequence of sub-tasks by which the objects are delivered to the nest may break. In the experiments described in this section, this sequence is composed of two sub-tasks, but in general the same issue may arise when the task is partitioned into more sub-tasks. The sequence of sub-tasks is more likely to be interrupted in small swarms, since it involves few robots. If the sequence of sub-tasks is interrupted the whole system is affected: some robots may not manage to pass an object, while others to receive one. A robot that cannot manage to pass an object will eventually abandon waiting and navigate towards the nest, therefore traveling a longer distance. As a consequence, its source position estimate is likely to be less precise, and it is more likely that it will get lost afterwards. Analogously, when a robot cannot receive an object for a long time, it will return exploring the environment. These events further reduce the number of robots transferring objects, which can result in the whole swarm returning to the initial state where none of the robots has a source position estimate. The plots of Fig. 15 show that the variability of the results decreases for increasing swarm sizes. This indicates that, with more robots, it is less likely that the sequence of sub-task is interrupted and therefore the results become more stable.

The plots also highlight that, in large swarms, the performance decreases more steeply with the partitioning strategy than with the non-partitioning strategy. This is a consequence of the fact that, when the partitioning strategy is employed, physical interference concentrates in areas of the environment that are critical for the task. Since the robots using the partitioning strategy get lost less frequently, they spend more time navigating along the path from source to nest (i.e., performing the task). Therefore, the density of robots is high in specific areas of the environment and consequently also the physical interference in those areas is high. In particular, the robots cluster at the locations where objects are transferred. Here, many stationary robots interfere with the movements of the other robots. Stationary robots also increase the frequency of false positive detected by the mechanism described in Sec. 4.4, with which a robot checks if it is heading outside the arena (see Table 3 of the online supplementary material).

The impact of interference on the partitioning strategy is more visible in the small environment. When the robots are few, physical interference is low and the partitioning strategy is advantageous over the non-partitioning strategy. This is not true when the swarm size is increased: interference progressively increases to a point at which it overcomes the benefits. In other words, the cost of the partitioning strategy due to interference is higher than the cost of getting lost of the non-partitioning strategy. Thus, the non-partitioning strategy becomes preferable.

To summarize, the results presented in this section highlight benefits and costs of employing task partitioning with direct transfer. We pointed out that a minimum amount of robots is needed in order to exploit task partitioning efficiently. However, increasing the number of robots can also increase physical interference, specially in the areas where the objects are transferred. Depending on the characteristics of the environment, interference can cancel out the benefits of employing task partitioning. Therefore, it is important to properly select the strategy on the basis of the costs and benefits, for example using mechanisms as the one described by Pini et al. (2011). Alternatively, physical interference could be reduced by regulating the swarm size.



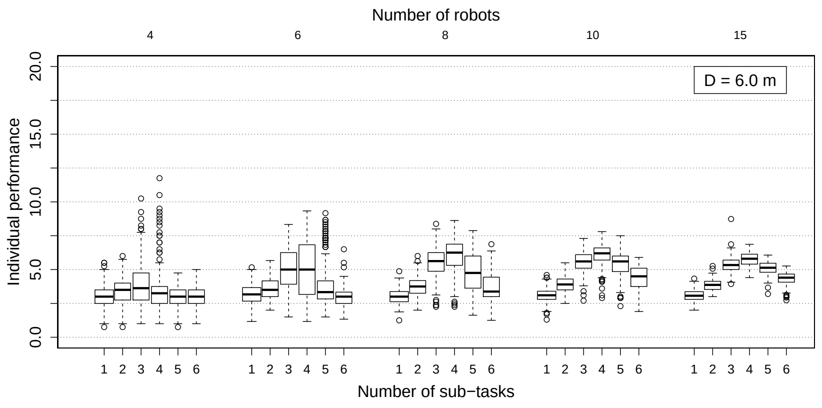


Fig. 16: Individual performance for $D = 4.0\,\mathrm{m}$ (top) and $D = 6.0\,\mathrm{m}$ (bottom) for different swarm sizes. The individual performance is computed as number of object retrieved divided by the number of robots. Each bar reports the individual performance when the task is partitioned into a different number of sub-tasks.

Mechanisms of division of labor, such as the one proposed by Labella et al. (2006a), could be used to limit the number of foraging robots and cope with interference.

5.5 Best partitioning strategy

The goal of the experiments described in this section is to study the behavior of the system in relation to the distance between source and nest and the number of sub-tasks. We run the experiments in the huge environment (L= 6.7 m, W= 9.97 m). In a first set of experiments the value of D is set to 4.0 m, in another to 6.0 m. In both cases, we compare strategies that partition the object retrieval task in a different number of sub-tasks: from a minimum of 1 sub-task (i.e., without partitioning) to a maximum of 6 sub-tasks. The swarm size is taken from the set $\{4, 6, 8, 10, 15, 20, 30\}$. All the robots are placed inside the nest at the beginning of the experiment.

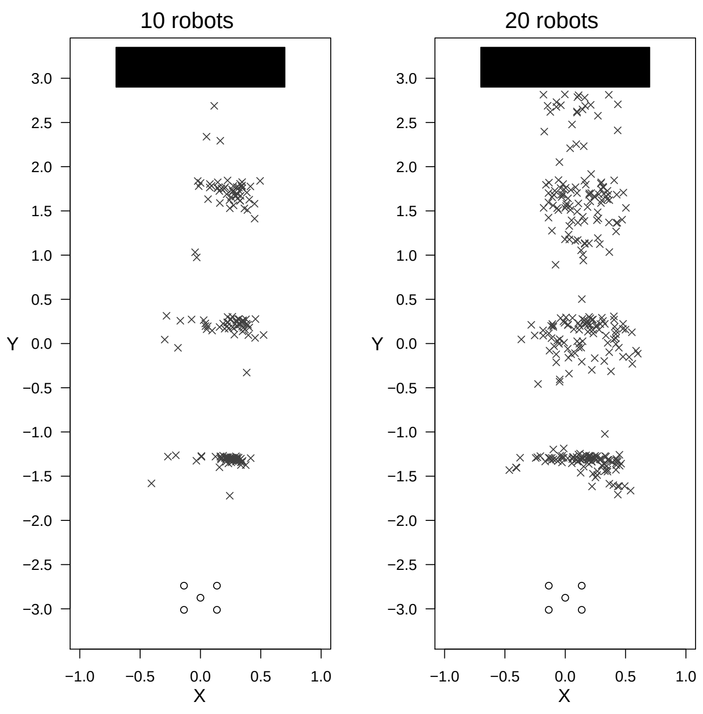


Fig. 17: Coordinates at which objects are transferred when the object retrieval task is partitioned into four sub-tasks. The source to nest distance $D = 6.0\,\mathrm{m}$. The plot on the left reports data collected in a single experimental run with a swarm composed of 10 robots, the one on the right with a swarm composed of 20 robots.

Figure 16 plots the individual performance (objects retrieved divided by the number of robots) for different swarm sizes. For clarity, only the results for a sub-set of the tested swarm sizes are reported in the figure (see the online supplementary material for the complete results). The graph on the top reports the results for $D = 4.0\,\mathrm{m}$, the one at the bottom for $D = 6.0\,\mathrm{m}$. The results show that the distance D and the size of the swarm define which is the best strategy in terms of number of sub-tasks. For $D = 4.0\,\mathrm{m}$, the best strategy is to partition the object retrieval task into two or three sub-tasks, depending on the size of the swarm. For $D = 6.0\,\mathrm{m}$, the best is to partition the object retrieval task into three or four sub-tasks, depending on the size of the swarm.

The graph at the bottom confirms the importance of the number of robots for the success of a strategy. A minimum number of robots is required in order for a strategy to work well. In general, the higher the number of sub-tasks, the higher the number of robots required.

Figure 17 shows the coordinates in the environment at which objects are transferred. The plot reports data taken from a single experimental run, with $D = 6.0\,\mathrm{m}$

and with the object retrieval task partitioned into four sub-tasks. The plot on the left reports the results for a swarm of 10 robots, the one on the right for a swarm of 20 robots. The plots focus on the area of the environment between source and nest, where the object transfers take place (i.e., the horizontal dimension does not include the whole available space).

In both plots, three clusters of points can be clearly identified: as mentioned, object transfers concentrate at specific locations in the environment. A comparison between the shape of the clusters in the two plots explains why the performance drops when the number of robots exceeds certain limits. With many robots, the locations at which objects are transferred spread out both horizontally and vertically (see plot on the right). This is an effect of obstacle avoidance: the robots traveling towards the nest avoid clusters of stationary robots that are transferring objects. Therefore, the line linking the position at which a robot collects an object and the position at which the robot transfers the object is not perfectly straight in the direction of the nest, but slanted. Since the robots travel a fixed distance (i.e., the partition length) before stopping, the effect is to spread vertically and horizontally the positions at which the robots transfer objects. The higher the number of the robots in the environment, the bigger the clusters of stationary robots and the higher the effect of obstacle avoidance. This behavior has a negative impact on the performance, as it increases the frequency at which object transfers are executed in close proximity of the nest (see Fig. 17 left), which is inefficient. A possible improvement would be for the robots to maintain also an estimate of the position of the nest that could be used to avoid transferring objects in close proximity of the nest.

6 Conclusions

Task partitioning can be an advantageous way of organizing work in swarms of autonomous robots. Benefits of task partitioning include: reduction of physical interferences, higher parallelism, exploitation of specialization, and higher efficiency. However, there are also costs associated with task partitioning, which are mainly due to coordination efforts needed to link the different sub-tasks one to another. Depending on how the output of a sub-task becomes the input of the following, the costs have a different nature and affect the system in different ways. Understanding the origin and the nature of these costs is an important step towards a better comprehension of task partitioning, that is a requirement for exploiting the benefits of task partitioning efficiently.

In this paper, we focused on task partitioning with direct transfer between sub-tasks, using a swarm of robots in a foraging scenario. Since the transfer is direct, the objects are directly handed over from robot to robot and progressively delivered to the nest. We identified physical interference to be the main factor in determining the cost of task partitioning when the transfer between sub-tasks is direct. We showed that the benefits of task partitioning depend on the relation between the characteristics of the environment and of the task (i.e., its length in our specific scenario). Additionally, we pointed out the critical role of the size of the swarm: task partitioning requires cooperation and its benefits cannot be fully exploited when the swarm is composed of few robots. On the other hand, the cost of physical interference increases with the number of robots. Therefore, one should carefully select when to employ task partitioning and when not to. Mechanisms such as the one described by Pini et al. (2011) can be employed to adaptively select whether to employ task partitioning. Alternatively, algorithms of division of labor, such as the one proposed by Labella et al. (2006a) could be used to regulate the number of individuals performing the task and reduce the impact of physical interference.

In the foraging scenario studied in this paper, the objects can be found at a source located in an unknown position in the environment. Upon locating the source, the robots maintain an estimate of its position using dead-reckoning. Dead-reckoning is suitable for swarm robotics because of its simplicity, cheapness and the fact that it does not require complex capabilities nor prior modifications to be made to the environment. A known problem with dead-reckoning is that it leads to an unbounded accumulation of errors. In the studied scenario, this renders the position estimates imprecise and makes it hard for the robots to return to the object source. We show that cooperation through task partitioning allows the swarm to overcome the individual limitations in terms of capability of returning to the object source. This results in a higher success rate in finding the object source and increases the performance in foraging.

We performed experiments using a real robotic platform. As in most of the experiments in robotics, the environmental conditions are controlled and the environment is artificial. Nevertheless, the key aspects of the problem have not been neglected: the robots spend time for gripping and transferring objects and are subject to real dead-reckoning errors. The results obtained with the robots prove the applicability of an approach based on task partitioning to solve a common problem in robotics: retrieve objects from an unknown location to a nest.

We wish to point out that the works of Vaughan et al. (2002), Gutiérrez et al. (2010), and Ducatelle et al. (2011b) tackle the very same problem and share many similarities with our study. However, in these works, the capability of the robots to reach targets located in the environment is enhanced by the use of explicit communication. This choice increases the requirements on the robot capabilities and introduces issues related to communication, for example related to bandwidth limitations (Ducatelle et al., 2011b). In the works of Gutiérrez et al. (2010) and Ducatelle et al. (2011b), the robots need a special communication device that allows the robots to measure the range and the bearing of the sender of a message. Gutiérrez et al. (2010) point out that the device is sensitive to reflections and can interfere with other sensors. An additional requirement of all the mentioned works is that the robots need to share knowledge that is used to refer in a common way to their target locations. Building such knowledge autonomously is not trivial. Gutiérrez et al. (2010) and Vaughan et al. (2002) give this knowledge a priori, while in the work of Ducatelle et al. (2011b) robots discovering a target remain stationary to mark its location. The approach based on task partitioning presented in this paper requires the robots neither to communicate nor to share common knowledge. This removes many of the requirements imposed by the mentioned works and makes the implementation of the system feasible also with minimalistic robotics platforms.

Our research aims at developing methods that allow swarms of robots to autonomously partition tasks into sub-tasks. If robots were able to do so, they could adapt the way they organize their task to specific environmental conditions. This would result in an increased autonomy and flexibility of swarm robotics systems. The system presented in this work will be used as a testbed for such methods. Autonomous task partitioning requires the robots to be able to regulate autonomously the length of the sub-tasks (i.e., the partition length, in the studied setup). Direct transfer is appealing in this sense since it provides feedback information that can be used for regulating the length of the sub-tasks. For example, the waiting time when transferring objects can be used to infer how the task should be partitioned. Communication could also be employed: two robots transferring an object could negotiate where to perform the next object transfer in order to minimize for both the probability of getting lost.

An additional direction for future research will be to focus on the task allocation component of the studied problem: in this study, the robots do not decide explicitly which sub-task to perform. A poor allocation of robots to sub-tasks can introduce inefficiencies and can reduce the benefits of task partitioning. In a separate study, we propose a self-organized method to allocate robots to tasks that have been partitioned (Brutschy et al., 2012b). In the future, we will combine the task allocation method to task partitioning and let each robot decide which sub-task to perform. The addition of the task allocation method is likely to improve the effectiveness of task partitioning and to further increase the overall system performance.

Acknowledgements The research leading to the results presented in this paper has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 246939. Marco Dorigo, Mauro Birattari, and Arne Brutschy acknowledge support from the Belgian F.R.S.–FNRS. Giovanni Pini acknowledges support from Université Libre de Bruxelles through the "Fonds David & Alice Van Buuren".

References

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 Multiagent Systems*, pages 1090–1097. ACM Press, New York.

Akre, R. D., Garnett, W. B., MacDonald, J. F., Greene, A., and Landolt, P. (1976). Behaviour and colony development of *Vespula pensylvanica* and *V. atropilosa* (hymenoptera: Vespidae). *Journal of the Kansas Entomological Society*, 49:63–84.

Anderson, C., Boomsma, J. J., and Bartholdi, J. J. (2002). Task partitioning in insect societies: bucket brigades. *Insectes Sociaux*, 49:171–180.

Anderson, C. and Ratnieks, F. L. W. (1999). Task partitioning in insect societies (I): Effect of colony size on queueing delay and colony ergonomic effciency. *American Naturalist*, 154:521–535.

Beni, G. (2005). From swarm intelligence to swarm robotics. In Şahin, E. and Spears, W. M., editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 1–9. Springer, Berlin, Germany.

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. *Proc. R. Soc. Lond. B.*, 263(1376):1565–1569.

Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., and Mondada, F. (2010). The MarXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, pages 4187–4193. IEEE Press, Piscataway, NJ.

Borenstein, J. (1998). Experimental results from internal odometry error correction with the omnimate mobile robot. *IEEE Transactions on Robotics and Automation*, 14(6):963–969.

Borenstein, J. and Liqiang, F. (1996). Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(6):869–880.

Brutschy, A., Pini, G., and Decugnière, A. (2012a). Grippable objects for the foot-bot. Technical Report TR/IRIDIA/2012-001, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., and Dorigo, M. (2012b). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. Technical Report TR/IRIDIA/2012-008, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Dorigo, M., Birattari, M., O'Grady, R., Gambardella, L. M., Mondada, F., Floreano, D., Nolfi, S., Baaboura, T., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A. L., Decugnière, A., Di Caro, G., Ducatelle, F., Ferrante, E., Martinez Gonzales, J., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., Montes de Oca, M., Pinciroli, C., Pini, G., Rétornaz, P., Rey, F., Roberts, J., Rochat, F., Sperati, V., Stirling, T., Stranieri, A., Stützle, T., Trianni, V., Tuci, E., Turgut, A. E., and Vaussard, F. (2011). Swarmanoid, the movie. In *AAAI-11 Video Proceedings*. AAAI Press. Winner of the "AAAI-2011 Best AI Video Award".

Dorigo, M. and Şahin, E. (2004). Guest editorial. Special issue: Swarm robotics. *Autonomous Robots*, 17(2–3):111–113.

Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A. L., Decugnière, A., Caro, G. D., Ducatelle, F., Ferrante, E., Förster, A., Gonzales, J. M., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., de Oca, M. M., O'Grady, R., Pinciroli, C., Pini, G., Rétornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stützle, T., Trianni, V., Tuci, E., Turgut, A. E., and Vaussard, F. (2012). Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*. (2012), in press.

Drogoul, A. and Ferber, J. (1992). From Tom Thumb to the dockers: Some experiments with foraging robots. In Meyer, J.-A., Herbert, L. R., and Stewart, W. W., editors, *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 451–459. MIT Press, Cambridge, MA.

Ducatelle, F., Di Caro, G., Pinciroli, C., and Gambardella, L. M. (2011a). Self-organized cooperation between robotic swarms. *Swarm Intelligence*, 5(2):73–96.

Ducatelle, F., Di Caro, G., Pinciroli, C., Mondada, F., and Gambardella, L. M. (2011b). Communication assisted navigation in robotic swarms: self-organization and cooperation. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, pages 4981–4988. IEEE Computer Society Press, Los Alamitos, CA.

Feng, L., Borenstein, J., and Everett, H. R. (1994). *"Where am I" Sensors and Methods for Autonomous Mobile Robot Positioning*. University of Michigan Press, Ann Arbor, MI.

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

Goldberg, D. and Matarić, M. J. (2002). Design and evaluation of robust behavior-based controllers for distributed multi-robot collection tasks. In Balch, T. and Parker, L. E., editors, *Robot Teams: From Diversity to Polymorphism*, pages 315—-344. A K Peters/CRC Press.

Gordon, D. M. (1996). The organization of work in social insect colonies. *Nature*, 380:121–124.

Grabowski, R., Navarro-Serment, L. E., Paredis, C. J. J., and Khosla, P. K. (2000). Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, (8):293–308.

Gutiérrez, A., Campo, A., Monasterio-Huelin, F., Magdalena, L., and Dorigo, M. (2010). Collective decision-making based on social odometry. *Neural Computing & Applications*, 19(6):807–823.

Hart, A., Anderson, C., and Ratnieks, F. L. W. (2002). Task partitioning in leafcutting ants. *Acta ethologica*, 5:1–11.

Hart, A. G. and Ratnieks, F. L. W. (2000). Leaf caching in *Atta* leafcutting ants: Discrete cache formation through positive feedback. *Animal behaviour*, 59(3):587–591.

Hart, A. G. and Ratnieks, F. L. W. (2001). Task partitioning, division of labour and nest compartmentalisation collectively isolate hazardous waste in the leafcutting *Atta cephalotes*. *Behavioral Ecology and Sociobiology*, 49:387–392.

Hongo, T., Arakawa, H., Sugimoto, G., Tange, K., and Yamamoto, Y. (1987). An automatic guidance system of a self-controlled vehicle. *IEEE Transactions on Industrial Electronics*, IE-34(1):5–10.

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

Johansson, R. and Saffiotti, A. (2009). Navigating by stigmergy: A realization on an RFID floor for minimalistic robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 245–252, Kobe, JP.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME, ser. D, Journal Of Basic Engineering*, 82(1):35–45.

Kurazume, R. and Hirose, S. (2000). An experimental study of a cooperative positioning system. *Autonomous Robots*, 1(8):43–52.

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

Labella, T. H., Dorigo, M., and Deneubourg, J.-L. (2006b). 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. T. (2008). Adaptive multi-robot bucket brigade foraging. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 337–342. MIT Press, Cambridge, MA.

Lein, A. and Vaughan, R. T. (2009). Adapting to non-uniform resource distributions in robotic swarm foraging through work-site relocation. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, pages 601–606, Piscataway, NJ. IEEE Press.

Mayet, R., Roberz, J., Schmickl, T., and Crailsheim, K. (2010). Antbots: a feasible visual emulation of pheromone trails for swarm robots. In Dorigo, M., Birattari, M., Di Caro, G., Doursat, R., Engelbrecht, A. P., Floreano, D., Gambardella, L., Groß, R. Şahin, E., Stützle, T., and Sayama, H., editors, *Proceedings of the 7th international conference on swarm intelligence (ANTS 2010)*, volume 6234 of *Lecture notes in computer science*, pages 84–94. Springer, Berlin, Germany.

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.

Nouyan, S., Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2009). Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711.

Østergaard, E. H., Sukhatme, G. S., and Matarić, M. J. (2001). Emergent bucket brigading: A simple mechanisms for improving performance in multi-robot constrained-space foraging tasks. In *AGENTS '01: Proceedings of the Fifth International Conference on Autonomous Agents*, pages 29–30. ACM Press, New York.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G. A., Ducatelle, F., Stirling, T., Gutiérrez, A., Gambardella, L. M., and Dorigo, M. (2011). ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, pages 5027–5034. IEEE Computer Society Press, Los Alamitos, CA.

Pini, G., Brutschy, A., Birattari, M., and Dorigo, M. (2009). Task partitioning in swarms of robots: reducing performance losses due to interference at shared resources. In Cetto, J. A., Filipe, J., and Ferrier, J.-L., editors, *Informatics in Control, Automation and Robotics*, volume 85 of *LNEE*, pages 217–228. Springer, Berlin, Germany.

Pini, G., Brutschy, A., Francesca, G., Dorigo, M., and Birattari, M. (2012a). Multi-armed bandit formulation of the task partitioning problem in swarm robotics. Technical Report TR/IRIDIA/2012-009, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Pini, G., Brutschy, A., Frison, M., Roli, A., Birattari, M., and Dorigo, M. (2011). Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, 5(2):283–304.

Pini, G., Brutschy, A., Scheidler, A., Dorigo, M., and Birattari, M. (2012b). Task partitioning in a robot swarm: retrieving objects by transferring them directly between sequential sub-tasks – Online supplementary material. `http://iridia.ulb.ac.be/supp/IridiaSupp2012-001/`.

Ratnieks, F. L. W. and Anderson, C. (1999a). Task partitioning in insect societies. *Insectes Sociaux*, 46(2):95–108.

Ratnieks, F. L. W. and Anderson, C. (1999b). Task partitioning in insect societies. II. use of queueing delay information in recruitment. *The American naturalist*, 154(5):536–548.

Rekleitis, I., Dudek, G., and Milios, E. (2001). Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, (31):7–40.

Schatz, B., Lachaud, J.-P., and Beugnon, G. (1996). Polyethism within hunters of the ponerine ant, *Ectatomma ruidum* roger (formicidae, ponerinae). *Insectes Sociaux*, 43:111–118.

Seeley, T. D. (1995). *The wisdom of the hive*. Harvard University Press, Cambridge, MA.

Shell, D. J. and Matarić, M. J. (2006). On foraging strategies for large-scale multi-robot systems. In *Proceedings of the 19th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2717–2723, Pitscataway, NJ. IEEE Press.

Sperati, V., Trianni, V., and Nolfi, S. (2011). Self-organised path formation in a swarm of robots. *Swarm Intelligence*, 5(3–4):97–119.

Sugawara, K., Kazama, T., and Watanabe, T. (2004). Foraging behavior of interacting robots with virtual pheromone. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04)*, volume 3, pages 3074–3079, Pitscataway, NJ. IEEE Press.

Vasconcelos, H. L. and Cherrett, J. M. (1996). The effect of wilting on the selection of leaves by the leaf-cutting ant *Atta laevigata*. *Entomologia Experimentalis et Applicata*, 78(2):215–220.

Vaughan, R. T., Stoy, K., Sukhatme, G. S., and Matarić, M. J. (2002). LOST: localization-space trails for robot teams. *IEEE Transactions on Robotics and Automation*, 18(5):796–812.

Werger, B. B. and Matarić, M. J. (1996). Robotic "food" chains: externalization of state and program for minimal-agent foraging. In Maes, P., Matarić, M. J., Meyer, J. A., Pollack, J., and Wilson, S. W., editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 625–634. MIT Press˜/ Bradford Books, Cambridge, MA.

Winfield, A. F. T. (2009). Foraging robots. In Meyers, R. A., editor, *Encyclopedia of Complexity and System Science*, pages 3682–3700. Springer, Berlin, Germany.