Task Partitioning in a Robot Swarm: Object Retrieval as a Sequence of Subtasks with Direct Object Transfer

Giovanni Pini*,**
Université Libre de Bruxelles

Arne Brutschy**
Université Libre de Bruxelles

Alexander Scheidler†
Fraunhofer IWES

Marco Dorigo**
Université Libre de Bruxelles

Mauro Birattari**
Université Libre de Bruxelles

Abstract We study task partitioning in the context of swarm robotics. Task partitioning is the decomposition of a task into subtasks that can be tackled by different workers. We focus on the case in which a task is partitioned into a sequence of subtasks that must be executed in a certain order. This implies that the subtasks must interface with each other, and that the output of a subtask is used as input for the subtask that follows. A distinction can be made between task partitioning with direct transfer and with indirect transfer. We focus our study on the first case: The output of a subtask is directly transferred from an individual working on that subtask to an individual working on the subtask that follows. As a test bed 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 a robot finds the source, it memorizes its position and uses dead reckoning to return there. Dead reckoning is appealing in robotics, since it is a cheap localization method and it does not require any additional external infrastructure. However, dead reckoning leads to errors that grow in time if not corrected periodically. We compare a foraging strategy that does not make use of task partitioning with one that does. We show that cooperation through task partitioning can be used to limit the effect of dead reckoning errors. This results in improved capability of locating the object source and in increased performance of the swarm. We use the implemented system as a test bed 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, localization, dead reckoning

A version of this paper with color figures is available online at http://dx.doi.org/10.1162/artl_a_00132. Subscription required.

1 Introduction

Task partitioning is a technique to organize work that consists of decomposing a task into a number of subtasks [30]. This decomposition allows different groups of workers to tackle each

* Contact author.
** IRIDIA-CoDE, Université Libre de Bruxelles, Brussels, Belgium. E-mail: gpini@ulb.ac.be (G.P.); arne.brutschy@ulb.ac.be (A.B.); mdorigo@ulb.ac.be (M.D.); mbiro@ulb.ac.be (M.B.)
† Fraunhofer IWES, Koenigstor 59, D-34119, Kassel, Germany. E-mail: ascheidler@ulb.ac.be

of the subtasks in parallel, as well as to tackle the subtasks at different moments in time. In this work, we focus on the case in which a task is partitioned into a sequence of subtasks, which implies that the subtasks 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 [50]. Insects benefit from task partitioning in several ways. Task partitioning allows the physical separation of individuals working on different subtasks, and therefore it can be beneficial in the reduction of physical interference (see examples in [27]). Physical interference occurs when many individuals share a space and hamper each other's movements. Interference can strongly degrade performance, because the individuals have to spend resources to deal with it (e.g., spend time 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 workers cut them in pieces. Petioles are harder to cut than leaves; therefore harvesting leaves is best performed by larger individuals [59]. Partitioning a task into subtasks can also result in a gain of efficiency. An example is the foraging strategy of certain species of leaf-cutting ants, where some workers climb trees, cut leaves, and drop them to the ground, and other workers collect the leaves from the ground and transport them to the nest [50]. 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 note 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 subtasks, costs commonly occur where subtasks interface with each other. We can distinguish between two cases, depending on how the output of a subtask becomes an input for the subtask that follows [50]. In *direct transfer*, the output of a subtask is transferred from an individual working on that subtask to an individual working on the subtask that follows (i.e., workers interact directly). In *indirect transfer*, the output of a subtask is stored in a temporary location (usually referred to as a *cache*) and 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 [4] and to perform the transfer itself [3]. 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 [50].

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 [9, 53]. The main characteristics of a swarm robotics system are, similar to what can be observed in social insects, simplicity of the individuals with respect to the task they must perform, 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 to those for social insects. It is therefore interesting to explore the application of task partitioning to the organization of work in swarms of robots. We study the case in which a swarm of robots has to perform a foraging activity. The *task* of the robots is object retrieval: Collect an object from a source and transport it to a home location, referred to as the *nest*. The foraging activity consists of multiple executions of the object retrieval task. 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 land-mine clearance. In the context of foraging, researchers have focused on different aspects. In most cases, foraging is used as a test bed to study aspects such as, for example, task allocation and division of labor [12, 34], communication [55], or interference among robots [37]. For a review of robot foraging we refer the reader to the work of Winfield [62]. Foraging has also received attention in the field of multi-agent systems, where it is also used as a test bed to study theoretical aspects of machine learning [42], or to propose and study distributed algorithms [43].

In this work, we use foraging as a test bed to study task partitioning. In the foraging problem studied, the objects are clustered in a single location (*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, relative 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 relative 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*: the determination of the position of a robot by the integration over time of measures coming from motion sensors [19]. 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 with dead reckoning is that the integration of noisy information over time leads to errors that can grow without bound. In our specific case, dead-reckoning errors can result in a robot being unable to return to the source.

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 object retrieval task is decomposed into a sequence of subtasks, each consisting of transporting the object toward the nest for a limited distance. Objects are delivered to the nest in progressive steps, with a robot working on one of the subtasks, directly handing over objects to a robot working on the next subtask. The process is fully decentralized and does not require the robots to communicate explicitly.

Simulation and real-robot experiments show that cooperation through task partitioning allows the swarm to overcome the limitations of the single robot and increases the overall localization capabilities. We use the implemented system as a case study to highlight costs, benefits, and properties of task partitioning with direct transfer among individuals. To the best of our knowledge, this work is the first to explore the costs and benefits of task partitioning with direct transfer and to evaluate their influence on a real robotic system. With the notable exception of the works of Fontan and Matarić [20] and Goldberg and Matarić [22], 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 affect the behavior of the implemented systems. Therefore, we also run experiments with real robots foraging for physical objects. In the experiments studied in this article, object handling and localization errors are critical issues. We do not make any assumptions or simplifications 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 our approach based on task partitioning to real-world settings.

2 Related Work

Task partitioning has been extensively studied in the field of biology, but only a 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. In contrast 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 article. 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 of decomposing tasks into sequences of subtasks [30, 50]. Task partitioning is usually coupled with task allocation strategies: When a task is partitioned into multiple subtasks, workers must be assigned to each of these subtasks. The key difference between task partitioning and task allocation is that task allocation organizes the workforce, while task partitioning organizes the way the work itself is performed [50].

Task allocation has been extensively studied, and several works on the topic exist both in the biology and in the swarm robotics literature. In biology, task allocation can be observed in ants [5, 23], bees [26, 51], and wasps [2]. In swarm robotics, self-organized task allocation is often based on the response threshold model first introduced by Bonabeau et al. [5]. Threshold-based task allocation has been successfully employed in tasks such as foraging [34] and object clustering [1].

The amount of research work on task partitioning is considerably smaller and mainly studies natural systems. In nature, task partitioning is observed mainly in the organization of the work of social insects (see [50] for a review). Instances of task partitioning have been reported in activities such as foraging [21], garbage disposal [28], and hunting [54].

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 [20] propose a system in which a foraging task is pre-partitioned into subtasks, each performed in a separate area. A robot is assigned a priori to each area (i.e., it works on the subtask performed in that area). The authors showed 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. [45] with the difference that the working areas are non-exclusive and are dynamically assigned. Goldberg and Matarić [22] used a modified version of the system of Fontan and Matarić [20] to study the design of behavior-based controllers that are robust with respect to failures and easy to modify.

Østergaard et al. [41] compared bucket brigading and homogeneous foraging in a mazelike environment. The authors showed 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 [14]. In that work, however, 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 [56] studied the effect of interference on the performance of a swarm of robots in a foraging scenario. Each robot moves objects toward 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 showed that, with increasing swarm sizes, a reduction of the working area size improves the performance of the system. Lein and Vaughan [35] extended the work of Shell and Matarić [56] by adding a mechanism to dynamically adapt the size of the working areas. In another work, the same authors point out that the distribution of objects in the environment is critical for the performance of the system [36]. To cope with clusters of objects, the authors further extended the proposed mechanism by allowing the robots to relocate the center of their working areas toward the position of clusters of objects.

In the work of Pini et al. [46], task partitioning is tackled explicitly. The authors proposed a method for selecting between a partitioning and a nonpartitioning strategy. The overall task is pre-partitioned into two subtasks, and one of the two strategies is selected on the basis of its costs. In follow-up research [47, 49], the authors formulated the problem of choosing whether to employ task partitioning as a multi-armed-bandit problem and showed that algorithms employed in the literature to tackle multi-armed-bandit problems can be successfully employed.

Existing studies of task partitioning in swarm robotics, with the exception of the work presented by Pini et al. [45], consider the case with indirect transfer. The studies of Anderson and Ratnieks [4, 51] show that task partitioning with direct transfer also occurs in nature, and its costs and benefits are different from those that appear in the case of indirect transfer. Studying the effects of these costs is an important step toward a better understanding of task partitioning, that is a requirement to exploit its benefits efficiently. The work presented in this article is the first to study costs and benefits of task partitioning with direct object transfer in robotics and one of the few works in robotics in which task partitioning is demonstrated with real robots. Additionally, the work proposes a novel strategy that can be used in groups of robots performing foraging. This strategy is based on the use of dead reckoning and task partitioning, and it can be applied to robotic platforms with limited sensing capabilities.

2.2 Localization

Localization is a common problem in robotics; several techniques to tackle this problem have been proposed in the literature (see [19] for a review). However, most 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 infrastructure that is 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 distance traveled and that the quality of the estimate gradually degrades over time. Dedicated hardware [29], calibration [8], or the use of Kalman filters [32] 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 [33], the swarm is divided into two groups. The groups move in turn, with the group of stationary robots acting as a reference. Rekleitis et al. [52] proposed an analogous solution in which robots are organized in pairs: While one of the two robots moves, the other is stationary and acts as a reference. Grabowski et al. [24] used 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. [39, 40], of Dorigo et al. [15],[1] and of Werger and Matarić [61], 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. [57] used 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. [17], a swarm of flying robots 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.

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 [31], 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. [60] described 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 these locations when communicating. A very similar approach has been proposed by Ducatelle et al. [16, 18]: The robots exchange positional information that can be used to reach target locations. However, the robots themselves represent the waypoints that must be followed to reach a target. In the work of Sugawara et al. [58], a system composed of a CCD camera and an LC projector emulates the pheromone-laying mechanism. The camera tracks the robots' movements, and pheromone trails are then projected where needed. Drogoul and Ferber [14] simulated a system in which robots can lay and follow "crumbs," used to mark paths in the environment. Mayet et al. [38] implemented a system in which the ground is painted with a synthetic resin containing a phosphorescent material that reacts to ultraviolet light. On-board 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.

1 A video [13] describing the system can be found online at http://www.youtube.com/watch?v=M2nn1X9Xlps

Gutiérrez et al. [25] described a localization strategy based on "social odometry": Robots exchange the positions 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 is the repetition of the object retrieval task: collecting an object from a given location (*source*), and transporting it to a predefined location (*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, which 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 find the source. When the source is found, the robot memorizes its location, so that it can return to it in the future.

Each robot faces a localization problem: When moving, the robot must update its information about the source location. As mentioned, the robots tackle this problem using dead reckoning. Due to errors, the position of the source as maintained by a robot is only an estimate of the real one. In the rest of the article we refer to such an 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, the less accurate the estimate. This is mainly due to three factors. First, errors on the bearing of the source position estimate have a stronger effect 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 nonsystematic components such as collisions, uneven terrain, and wheel slipping.

The robots can perform the object retrieval task using two strategies: the *nonpartitioning strategy* and the *partitioning strategy*. The nonpartitioning strategy consists of performing the object retrieval task as a whole, unpartitioned task. The partitioning strategy consists of partitioning the object retrieval task into subtasks.

Figure 1 reports a schematic representation of the behavior of the robots when foraging is performed employing the nonpartitioning strategy. A video showing the behavior of the robots is available with the online supplementary material [48]. Initially, a robot does not have any information about the location of the objects. Therefore, it needs to explore the environment to find the

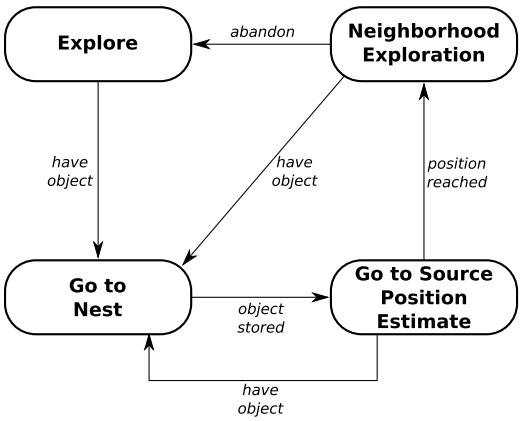

Figure 1. Representation of the behavior of the robots employing the nonpartitioning strategy.

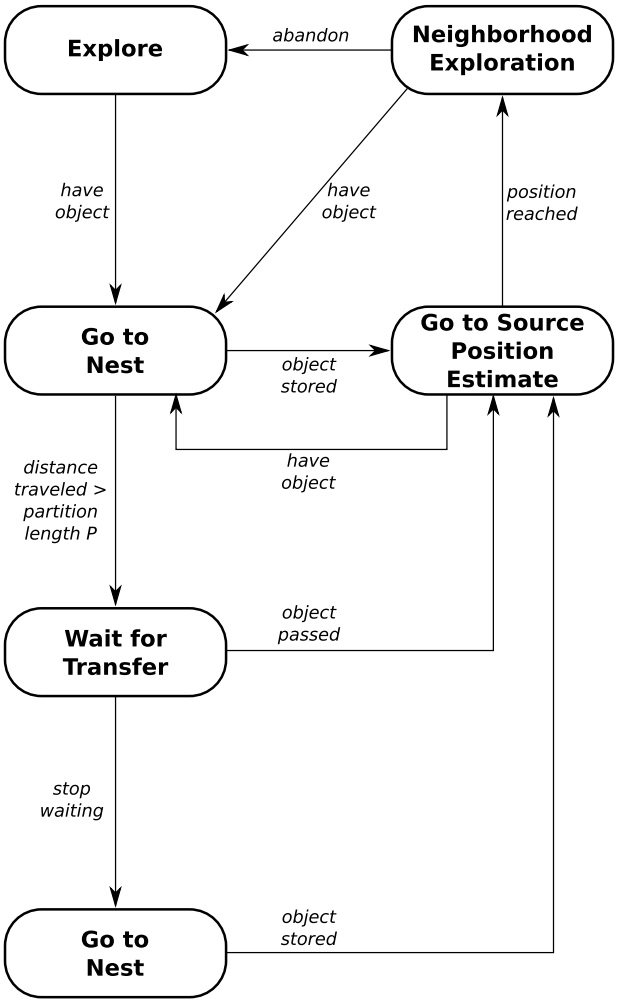

Figure 2. Representation of the behavior of the robots employing the partitioning strategy.

source. When the robot finds an object, it harvests it, sets its source position estimate to (0,0), and navigates toward the nest. Upon entering the nest, the robot stores the object and heads toward the source position estimate to harvest 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 collects a new object. If no objects are perceived, the robot performs a *neighborhood exploration*. The neighborhood exploration consists of exploring an 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 article we say that the robot *got lost* when it abandoned the neighborhood exploration and discarded its source position estimate.

Figure 2 reports a schematic representation of the behavior of the robots employing the partitioning strategy. The difference from the nonpartitioning strategy is that, when heading to the nest, a robot stops if the distance from its source position estimate is larger than a threshold, referred to as *partition length P*. 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 toward the nest to store the object.

As robots wait along the path to the nest, objects are found 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 releases it. Therefore, an object is never left on the ground without at least one robot holding it. A robot receiving an object memorizes the location of the exchange; this location becomes the source position estimate for that robot. As for the nonpartitioning 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 is available with the online supplementary material [48].

The partitioning strategy results in the object retrieval task being partitioned into subtasks: An object may be transferred several times from robot to robot before it is delivered to the nest. Each robot contributes to a portion of the overall work (i.e., it performs a subtask of the object retrieval task). The number of subtasks into 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 nonpartitioning strategy requires the robots to travel a longer distance than the partitioning strategy. Traveling longer distances reduces the accuracy of the source position estimate. If the accuracy of the source position estimate is low, 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 again. The cost of such an event depends on the size of the environment: The larger the environment, the longer 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 nonsystematic events mentioned above, since the distance traveled is shorter. The expected result is a higher accuracy in the source position estimate, which in turn increases the probability of finding an object during the neighborhood exploration. Consequently, the number of times the robots get lost is reduced. Disadvantages include 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. An additional disadvantage is 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 degrade the source position estimate, as they can cause the robots to be moved without being aware of it or to slip on the ground.

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. Additionally, we provide some implementation details required to replicate the experiments presented in this article.

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 five objects, positioned as shown

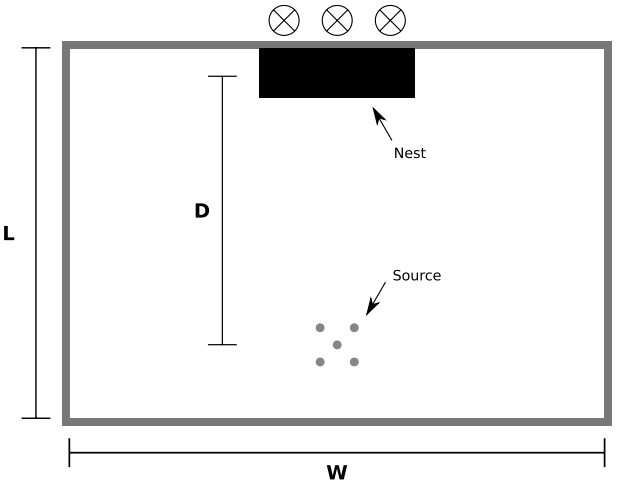

Figure 3. Representation of the arena in which foraging takes place. The nest (black rectangle) is located at the center top and is marked by three lights (crossed circles). The object source, located in front of the nest, is composed of five objects (gray circles), positioned as shown. The size of the arena (*W*, *L*) and the distance of the source from the nest (*D*) depend on the specific experiment.

in Figure 3. 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 to perform foraging. As mentioned in Section 3, the longer a robot travels, the lower the accuracy of its source position estimate becomes. The relative improvement of using the partitioning strategy over using the nonpartitioning strategy depends on the value of *D*: When *D* is small, the accuracy in locating the source can also be good using the nonpartitioning strategy. For large *D*, the partitioning strategy can lead to significant improvements over the nonpartitioning 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 have an influence on which strategy is preferable. For example, in a large arena, the nonpartitioning strategy could be disadvantageous even if *D* is small. In fact, even if the robots are accurate and rarely get lost (due to the short distance traveled), when they get lost they might need to search for a long time.

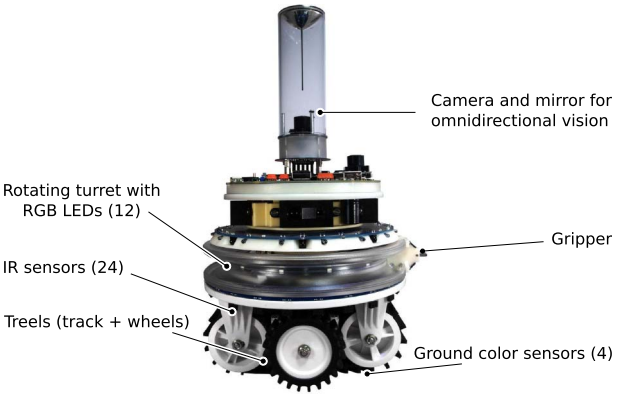

Figure 4. Photograph of a foot-bot. The sensors and actuators used in the experiments are highlighted.

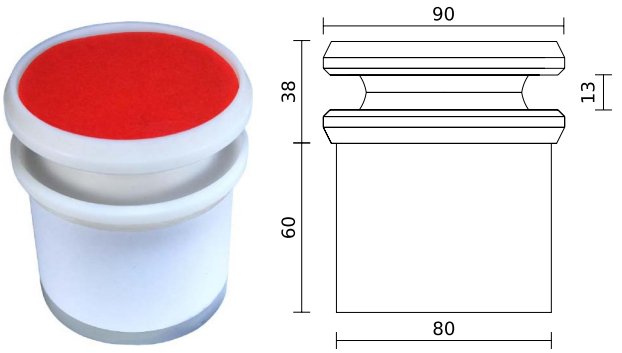

Figure 5. Depiction of an object. Left: picture of the actual object. Right: schematic representation reporting dimensional information; dimensions are in millimeters.

In contrast, for high values of D, the nonpartitioning 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.

4.2 Robots and Objects

The robots we employ in our experiments are *foot-bots*. The foot-bot is a small robot developed within the *Swarmanoid* project [15].[2] Figure 4 represents a foot-bot and highlights the sensors and actuators employed in the experiments. 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. [6].

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 used for object gripping. The turret also hosts 12 RGB LEDs that can be controlled separately. The LEDs are employed by the robots to repel other robots (see Section 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 using 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 upward to a 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 [10] for more information). Figure 5 represents such an object. Each object features a plastic ring that can be gripped by the foot-bot. The ring is mounted on a PVC pipe. The red top of the object is perceivable by the robots' camera. The distance from which an object can be perceived varies from robot to robot; overall, the perception range is between 0.45 and 0.55 m.

4.3 Simulator

For the simulation-based experiments described in Section 5 we use ARGoS [44], an open source simulator[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

2 http://www.swarmanoid.org

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

kinematics in 2D environments to complex dynamics in 3D environments. We conducted the experiments 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 outputs of the light, proximity, and ground-color sensors. Gaussian noise with a standard deviation of 20 mm is added to the measured distance of the objects perceived by the omnidirectional camera.

4.4 Robot Controller Implementation

4.4.1 Exploration and Neighborhood Exploration

Both in simulation and with the real robots, the exploration of the environment is performed through random walk, implemented as follows. The robot travels straight, with a maximum speed of 0.1 m/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 a 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.

As mentioned in Section 3, a robot can abandon the exploration of the neighborhood. Abandoning is governed by a time-out mechanism: When a robot has been exploring a neighborhood for 1 min without locating objects, neighborhood exploration is abandoned. The time-out is increased to 3 min 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 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. 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 to collect 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.

4.4.2 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 blob. Additionally, a repulsion mechanism is employed to prevent the same object from being gripped by several robots at the same time. The repulsion mechanism consists of the robots' ignoring every object that is perceived near a blue LED. Robots gripping an object can repel other robots from the same object by turning on their LEDs in blue.

Each time a robot grips an object, it tries to turn its turret 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 nonpartitioning strategy is used, the object is released. In case the partitioning strategy is used, 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 [48].

4.4.3 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 (80 samples) that have been recorded with the

real robots while performing the experiments described in Section 5.1. The empirical distribution of these samples is available in the online supplementary material. 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.

4.4.4 Object Transfer—Real Robots

Object transfer takes place only when the robots employ the partitioning strategy. When a robot grips an object being held by another robot, the procedure that implements object transfer is triggered. This procedure consists of 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 by the receiving robot, which releases the object. 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 Section 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 time-out: The robots are allowed to wait for 3 min, 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 [48].

4.4.5 Object Transfer—Simulation

Analogous to object gripping, also object transfer is simulated as a simplified version of what happens in reality. Similar 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 empirical distribution of these samples is available online). The two robots wait in place for the corresponding amount of time before the transfer is complete and the robots can leave. 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 for a maximum time of 3 min.

4.4.6 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 toward 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 toward 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.

4.5 Dead-Reckoning Noise Model

In order to simulate the system studied in this article, 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: The robots drift toward 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 toward the left-hand side in subsequent trips. Note that the robot cannot measure such a drift. All the robots drift toward the left-hand side; therefore we speculate that this behavior is due to some asymmetry in the two motors of the wheels.

Solutions such as the calibration procedures described by Borenstein and Liqiang [8] or Kalman filtering [32] 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, 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. Details about the procedure used to collect the samples are available with the online supplementary material [48].

Algorithm 1 (Pseudocode 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: $\mu_{noise} = RAY\ LEIGH(\sigma)$

10: **end if**

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

Note that the robot computes dead reckoning on the basis of *actuatedLeftWheelSpeed* 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. 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 Algorithm 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 and percentages of success in finding the source close to the ones observed with the real robots.

Note 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 nonsystematic components [7]. 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. Table 1 reports the default values for the parameters used in the experiments. When not explicitly mentioned in the text, a parameter assumes the value reported in the table.

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 min
Number of repetitions per experiment	200

5.1 Real-Robot Experiments

The experiments using real robots start with three foot-bots located in proximity to the source and three foot-bots located at the center of the arena, at a distance $D/2 = 2.0$ m from the nest (see Figure 6). 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. This choice is motivated by the need of reducing the duration of the experiments with the real robots. In case the partitioning strategy is employed, the three robots starting at the center of the arena initially perform neighborhood exploration, centered around their starting position. The partition length P is set to 2.0 m. A robot that receives an object from another robot delivers it to the nest; that is, an object can be transferred only once. As a result, the task is partitioned into two subtasks of (approximately) equal length. If the nonpartitioning strategy is employed, the three robots at the center of the arena

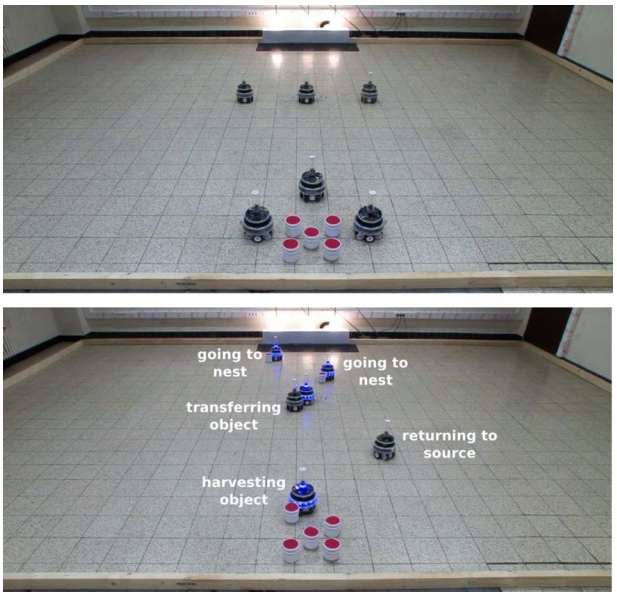

Figure 6. Snapshots of a real-robot experimental run. The robots employ the partitioning strategy. Top: initial configuration: Three robots are positioned in proximity to the object source, the rest midway between source and nest. Bottom: Situation after 1.5 min. Two robots passed their object: One is 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.

Table 2. Total number of objects collected using the two strategies and actions taken by the robots, for both real-robot and simulation experiments. For each strategy, we perform 6 real-robot and 200 simulation runs. The actions performed by the robots (bottom part of the table) are grouped into three sets: *Search*, *Handle Objects*, and *Navigate* (refer to the text for details). For each measure, the mean and the 95% confidence interval on the value of the mean are reported.

Measure	Nonpartitioning	Partitioning
Performance (total number of objects)		
Real robots	20.8/**23.8**/26.8	31.3/**34.7**/37.8
Simulation	19.0/**19.5**/20.1	31.4/**32.0**/32.4
Actions (% of time)		
Real robots—*Search*	73.1/**76.4**/80.4	31.4/**35.3**/38.8
Real robots—*Handle Objects*	3.4/**4.8**/6.4	32.9/**36.6**/39.4
Real robots—*Navigate*	16.0/**18.7**/21.5	26.1/**28.0**/30.5
Simulation—*Search*	83.0/**83.9**/84.7	45.8/**47.3**/48.8
Simulation—*Handle Objects*	2.4/**2.9**/3.4	28.1/**29.6**/31.0
Simulation—*Navigate*	12.4/**13.2**/14.1	21.9/**23.1**/24.3

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 six experimental runs, each with different sets of robots starting at the source and at 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 supplementary material [48]. Each time an object is harvested from the source, a new one is placed at the same position, so that there are always five 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 (one for each strategy) can be found in the online supplementary material. Figure 6 shows two snapshots taken from one of the videos at the beginning of the run (top) and after 1.5 min of experiment (bottom).

Table 2 reports, for the two strategies, the total number of objects delivered to the nest and the actions performed by the robots.[4] The data reported in the top part of the table indicates that the swarm collects more objects when the robots employ the partitioning strategy.

The bottom part of Table 2 summarizes the actions performed by the robots. The actions are sampled every second during the course of the experiment and 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 toward their source position estimate.

The data in Table 2 highlights costs and benefits of each of the two strategies. The nonpartitioning strategy has relatively low object handling costs, which are only due to object gripping, but very

4 The table also reports the data collected in simulation, which is discussed in Section 5.2.

Table 3. Probability of getting lost at the source for the two strategies. For each strategy, the data is computed over the corresponding six real-robot experimental runs. The mean and the 95% confidence interval on the value of the mean are reported. The original robot identifiers have been reported for future reference.

Robot	Nonpartitioning	Partitioning
foot-bot4	83.3/**92.5**/100.0	18.7/**41.7**/81.2
foot-bot10	75.8/**86.9**/97.2	5.1/**13.7**/22.3
foot-bot18	33.2/**67.6**/94.3	0.0/**5.0**/15.0
foot-bot22	75.0/**86.1**/95.8	6.7/**30.5**/61.7
foot-bot23	51.0/**74.5**/95.0	0.0/**6.4**/17.3
foot-bot37	34.0/**49.3**/62.0	0.0/**3.3**/10.0
Overall	68.2/**77.3**/85.3	8.9/**17.8**/23.4

high search costs. As mentioned, this is due to the fact that robots using the nonpartitioning strategy travel longer distances. Consequently, the accuracy of the source position estimate degrades due to dead-reckoning errors, and the robots get lost more often (and farther 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 3 reports the percentage of times each robot got lost after searching for the object source, for the nonpartitioning strategy and the partitioning strategy. 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 Table 2.

5.2 Validation

We replicate in simulation the setup described in the preceding Section 5.1 with the the goal of validating our simulator. To do so, we compare the results between simulation and reality, particularly regarding the noise model presented in Section 4.5.

The results of the experiments are summarized in Table 2.[5] The top part of Table 2 reports the total number of objects transported to the nest by the robots using the two strategies. The simulation results confirm the ones observed with real robots: When the robots use the partitioning strategy, they transport more objects to the nest. The bottom part of Table 2 summarizes the actions performed by the robots. Also in this case, the simulation results confirm what was observed in the real-robot experiments: The partitioning strategy is costly in object handling, but it reduces the search time and increases the navigation time.

Table 4 reports, for both strategies, the probabilities of getting lost measured in simulation and in the real-robot experiments. The data confirms that the simulation replicates the results observed in reality in terms of probability of getting lost, for both the nonpartitioning strategy and the partitioning strategy. A similar table, reporting additional measures not presented here, can be found in the online supplementary material [48].

5 The table also reports the data collected in real-robot experiments, which is discussed in Section 5.1.

Table 4. Probability of getting lost in real-robot (six runs per strategy) and simulation experiments (200 runs per strategy). In case the partitioning strategy is employed, the probability of getting lost is also reported 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 are reported.

Measure	Real robots	Simulation
No partition		
Get lost—searching for the source (%)	70.9/**77.1**/85.2	76.7/**77.9**/79.1
Partition		
Get lost—searching for the source (%)	6.3/**11.4**/17.2	13.3/**14.0**/14.8
Get lost—searching for other robots (%)	11.5/**14.1**/17.6	11.3/**11.9**/12.6

Figure 7 reports the empirical distribution of the time a robot takes to find the source after it got lost, for the nonpartitioning strategy (left) and the partitioning strategy (right). Simulation and real-robot experiment data is reported for comparison. The graphs show that, on average, in simulation the robots take more time to find the object source on getting lost. This can explain the discrepancy observed in Table 2 between simulation and reality in the numbers of objects collected and actions performed. Note 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 nonpartitioning strategy. This indicates that when the partitioning strategy is used, the robots get lost in locations that are closer to the object source than with the nonpartitioning strategy.

5.3 Convergence

The experiments described in Sections 5.1 and 5.2 start with half of the swarm in proximity to the source, and the other half at 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

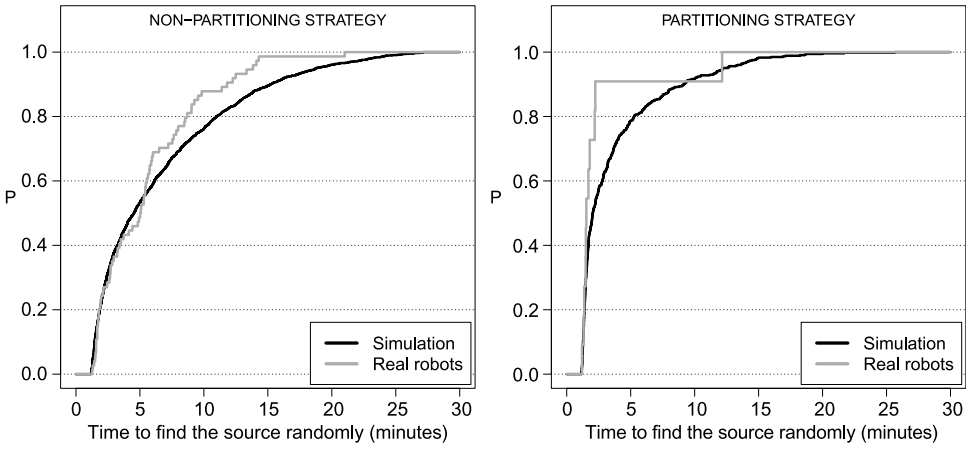

Figure 7. Empirical distribution (P) of the time needed to find the object source when searching randomly. The plotted data only includes the time needed to find the source after a neighborhood exploration failed (i.e., it does not include the time required to find the source for the first time). The plot on the left reports the data collected when the nonpartitioning strategy is employed by the robots; the plot on the right, when the partitioning strategy is employed.

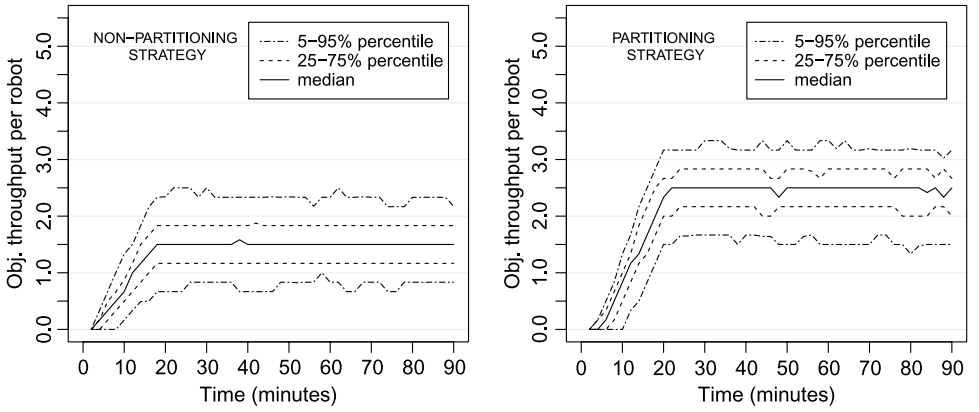

Figure 8. Throughput of the swarm using the nonpartitioning strategy (left) and the partitioning strategy (right). The throughput is computed in a time window of 15 min and expressed as objects per robot. The 5th, 25th, 50th, 75th, 95th percentiles are reported in each plot. The figure reports data collected in simulation-based experiments.

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 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 8, we report the throughput of the system during the course of the experiments. The throughput 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 min. The graph on the left reports the results of the swarm using the partitioning strategy; the graph on the right, the results for the nonpartitioning strategy.

For both strategies, the system reaches the steady state. The nonpartitioning strategy reaches the steady state in approximately 18 min; the partitioning strategy takes approximately 22 min. The value of the throughput at the steady state matches the one recorded in the simulation experiments described in Section 5.2 (see Plot V1 of the online supplementary material [48]). It is therefore reasonable to conclude that the behavior we observe in the experiments described in Sections 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 also able to reach a steady state 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 longer the expected time. We test four different environments of the following dimensions ($L \times W$): 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.

Each graph in Figure 9 shows, for a given environment and different swarm sizes, the individual performance using the nonpartitioning strategy and the partitioning strategy. The individual

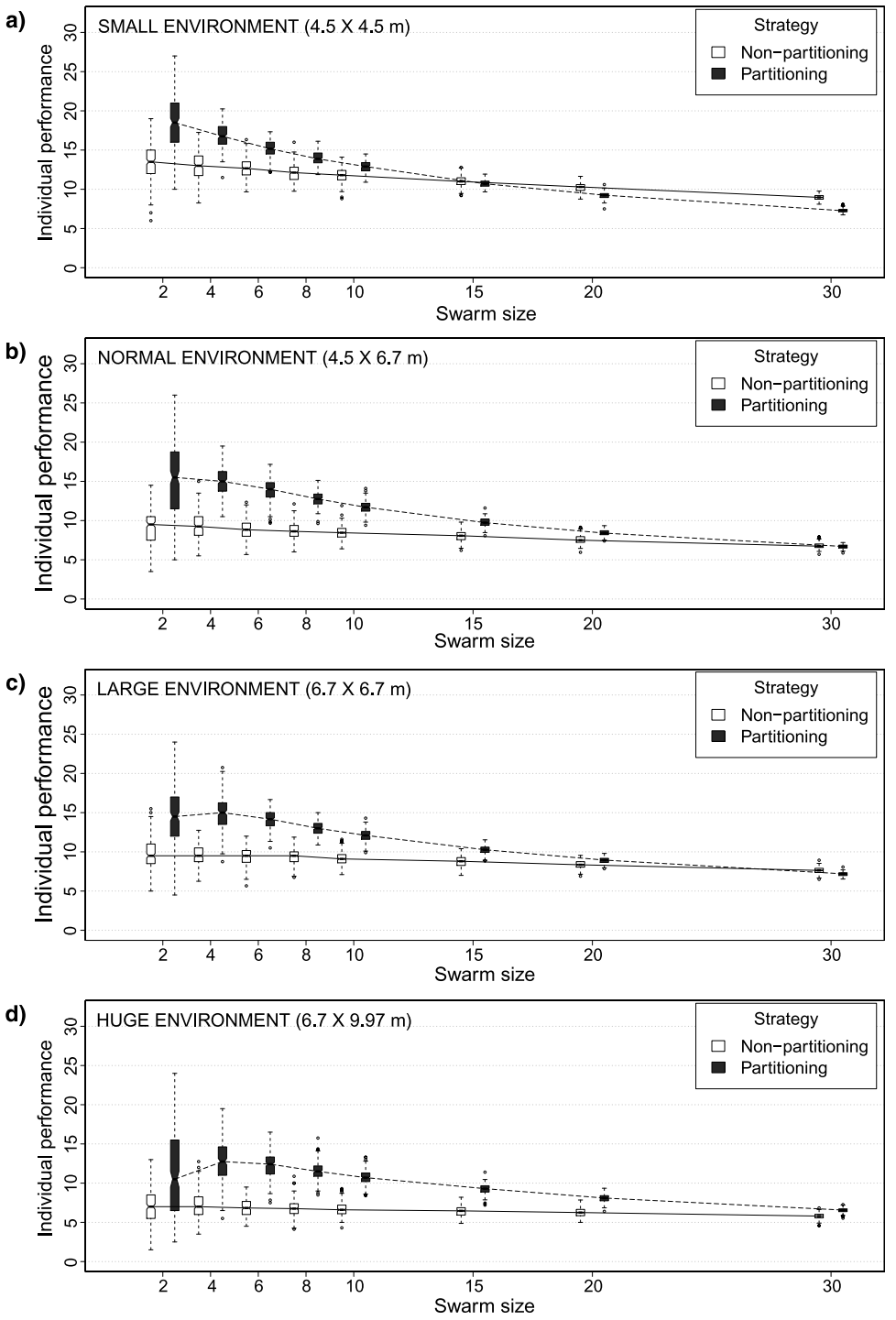

Figure 9. Individual performance for different swarm sizes using the partitioning strategy and the nonpartitioning strategy in the (a) small, (b) normal, (c) large, and (d) huge environments. The individual performance is computed as the total number of objects retrieved divided by the number of robots. The figure reports data collected in simulation-based experiments.

performance is computed as the total number of objects collected by the swarm, divided by the number of robots. The results confirm that the dimension of the environment has an effect on the performance of the swarm, as it 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 [48]).

The size of the swarm has an effect on the performance as well, but that depends on the strategy employed. The results obtained by the swarm employing the nonpartitioning 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 nonpartitioning strategy, each robot performs the task on its own and the robots' interactions only result in physical interference and therefore are negative. The partitioning strategy instead requires the robots to collaborate with each other. Therefore, there are both negative interactions due to interference and positive interactions that result from collaboration. 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.

This explains the shape of the curves in Figure 9 for the case in which the partitioning strategy is employed. In environments where the area to explore is small (small and normal environments, Figure 9a and b), the two events are likely to occur also when there are only two robots. In the large and huge environments (Figure 9c and d), the individual performance initially increases with the swarm size. This indicates that the positive interactions due to collaboration can be exploited only if there are more robots in the swarm.

Note that, for large swarms, the individual performance decreases more steeply with increasing swarm size for the partitioning strategy than for the nonpartitioning strategy. Thus, the negative interactions due to physical interference have a stronger effect on the partitioning strategy. This is a consequence of the fact that, when the partitioning strategy is employed, physical interference concentrates in task-critical areas of the environment. 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 the physical interference in those areas is also 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.

The tradeoff between positive interactions in the form of collaboration and negative interactions in the form of interference is particularly marked in the small environment. When the robots are few, physical interference is low and the benefits of collaboration render the partitioning strategy advantageous over the nonpartitioning strategy. This is not true when the swarm size is increased: Interference progressively increases until it reaches a point (for a swarm of 15 robots) at which its negative effects cancel the positive effects of collaboration. Here, the cost of the partitioning strategy due to interference is higher than the benefits due to a reduced probability of getting lost, and the nonpartitioning strategy becomes preferable.

The results presented in this section highlight benefits and costs of employing task partitioning with direct transfer. We pointed out that a minimum number of robots is needed in order to exploit task partitioning efficiently. However, increasing the number of robots can also increase physical interference, especially 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.

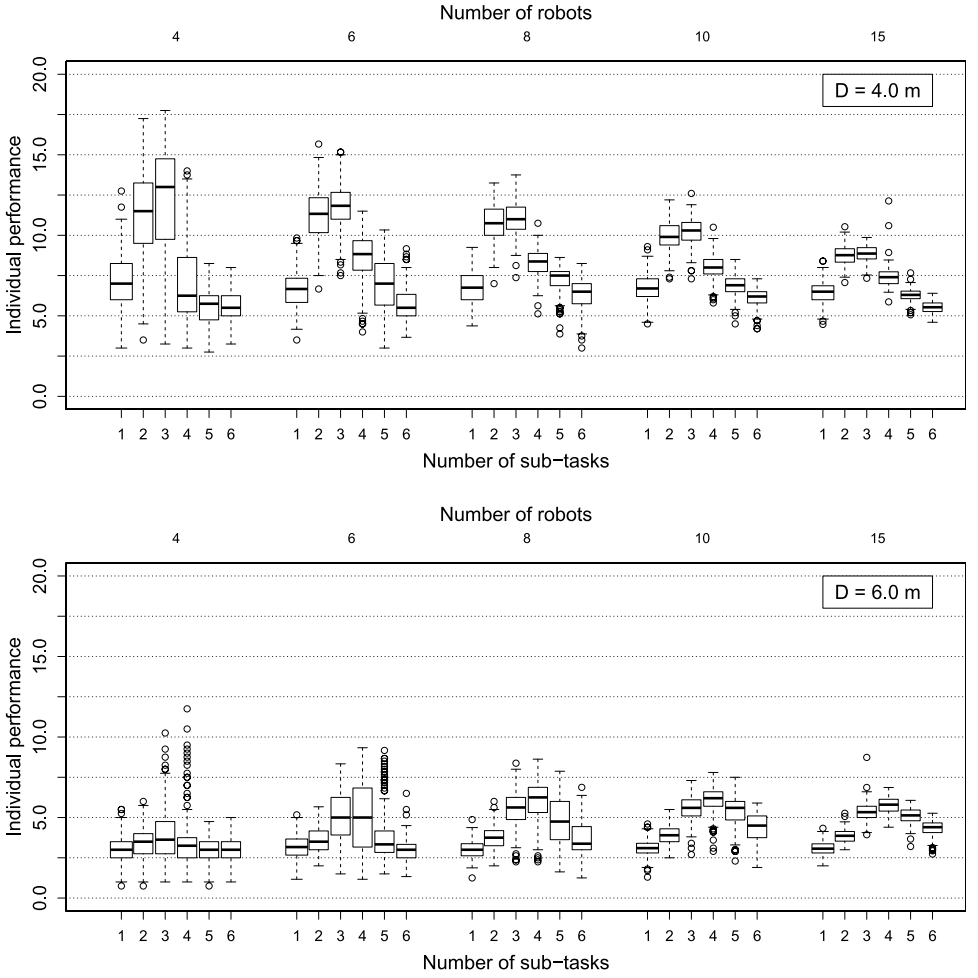

Figure 10. Individual performance for $D = 4.0$ m (top) and $D = 6.0$ m (bottom) for different swarm sizes. The individual performance is computed as the number of objects collected divided by the number of robots. The figure reports data collected in simulations carried out in the huge environment ($L = 6.7$ m, $W = 9.97$ m).

5.5 Number of Subtasks

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 subtasks. 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 the second, set to 6.0 m. In both cases, we compare strategies that partition the object retrieval task in different numbers of subtasks: from a minimum of one (i.e., without partitioning) to a maximum of six.[6] 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.

Figure 10 plots the individual performance (objects retrieved divided by the number of robots) for different swarm sizes.[7] For clarity, only the results for a subset of the tested swarm sizes are reported in the figure (see the online supplementary material [48] for the complete results). The

6 To set the number of subtasks into which transportation is partitioned, we select the value of the partition length P depending on the value of D (number of subtasks $= D/P$).

7 The online supplementary material [48] reports the same data in notched boxplots (notches are not included in Figure 10, as they are not visible).

graph on the top reports the results for $D = 4.0$ m, the one at the bottom for $D = 6.0$ m. The results show that the distance D and the size of the swarm determine which strategy is preferable in terms of number of subtasks. For $D = 4.0$ m, it is preferable to partition the object retrieval task into two or three subtasks, depending on the size of the swarm. For $D = 6.0$ m, it is preferable to partition the task into three or four subtasks, again 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 subtasks, the higher the number of robots required.

Figure 11 shows the coordinates in the environment at which objects are transferred. The plot reports data taken from a single experimental run, with $D = 6.0$ m and with the object retrieval task partitioned into four subtasks. 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 shapes of the

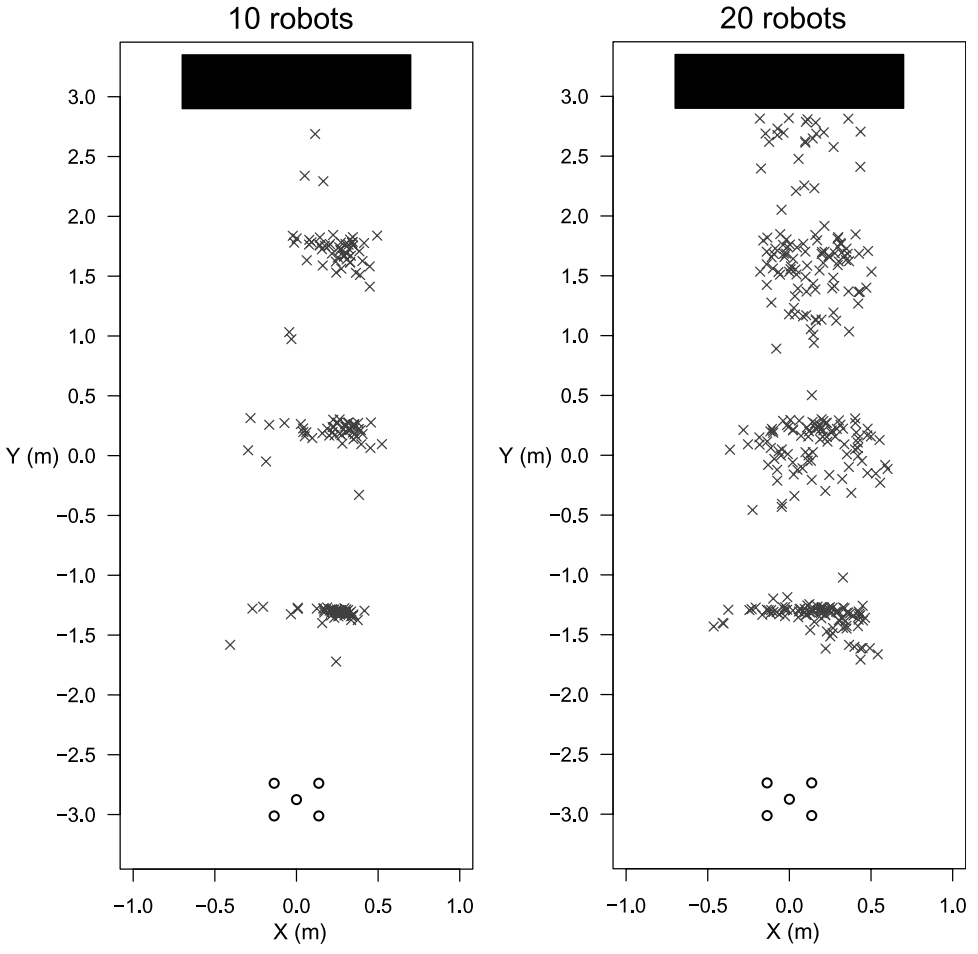

Figure 11. Coordinates at which objects are transferred when the object retrieval task is partitioned into four subtasks. The figure reports data collected in simulations carried out in the huge environment ($L = 6.7$ m, $W = 9.97$ m), for a source-to-nest distance $D = 6.0$ m. The plot on the left reports data collected in a single experimental run with a swarm composed of 10 robots; the plot on the right, with a swarm composed of 20 robots.

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 toward 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 precisely in the direction of the nest, but slanted. Since the robots travel a fixed distance P (i.e., the partition length) before stopping, the effect is to vertically and horizontally spread 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 larger the effect of obstacle avoidance. This behavior has a negative effect on the performance, as it increases the frequency at which object transfers are executed in close proximity to the nest (see Figure 11, left), which is inefficient. A possible improvement would be for the robots to also maintain an estimate of the position of the nest that could be used to avoid transferring objects in its close proximity.

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 subtasks one to another. Depending on how the output of a subtask becomes the input of the following, the costs have a different nature and affect the system in different ways. Studying the origin and the nature of these costs is an important step toward a better understanding of task partitioning, which is a requirement for exploiting its benefits efficiently.

We focused on task partitioning with direct transfer between subtasks, 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 as the main factor in determining the cost of task partitioning when the transfer between subtasks 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 only composed of a 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. [46] 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. [34], could be used to regulate the number of individuals performing the task and reduce the effect of physical interference.

In the foraging scenario studied in this article, 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 or prior modifications of 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 on capability of returning to the object source. This results in a higher success rate in finding the object source and increases the foraging performance.

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 on 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 point out that the works of Vaughan et al. [60], Gutiérrez et al. [25], and Ducatelle et al. [16, 18] 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 [16, 18]. In the works of Gutiérrez et al. [25] and Ducatelle et al. [16, 18], the robots need a special communication device that allows them to measure the range and the bearing of the sender of a message. Gutiérrez et al. [25] 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. The approach based on task partitioning presented in this article requires the robots neither to communicate nor to share common knowledge. This removes many of the requirements imposed by the mentioned works and also makes the implementation of the system feasible with minimalistic robotic platforms.

Our research aims at developing methods that allow swarms of robots to autonomously partition tasks into subtasks. If robots were able to do so, they could adapt the way they organize their task to specific environmental conditions. This would result in increased autonomy and flexibility of swarm robotics systems. The system presented in this work will be used as a test bed for such methods. Autonomous task partitioning requires the robots to be able to regulate autonomously the length of the subtasks (i.e., the partition length P, in the studied setup). Direct transfer is appealing in this context, since it provides feedback information that can be used to regulate the length of the subtasks. 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 is the analysis of the task allocation component of the studied problem: In this study, the robots do not decide explicitly which subtask to perform. A poor allocation of robots to subtasks 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 [11]. In the future, we will combine the task allocation method with task partitioning and let each robot decide which subtask 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.

Acknowledgments

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 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

1. Agassounon, W., & 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* (pp. 1090–1097). New York: ACM Press.

2. Akre, R. D., Garnett, W. B., MacDonald, J. F., Greene, A., & Landolt, P. (1976). Behaviour and colony development of *Vespula pensylvanica* and *V. atropilosa* (Hymenoptera: Vespidae). *Journal of the Kansas Entomological Society, 49*(1), 63–84.

3. Anderson, C., Boomsma, J. J., & Bartholdi, J. J. (2002). Task partitioning in insect societies: Bucket brigades. *Insectes Sociaux*, *49*(2), 171–180.

4. Anderson, C., & Ratnieks, F. L. W. (1999). Task partitioning in insect societies (I): Effect of colony size on queueing delay and colony ergonomic efficiency. *American Naturalist*, *154*(5), 521–535.

5. Bonabeau, E., Theraulaz, G., & Deneubourg, J.-L. (1996). Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. *Proceedings of the Royal Society of London B*, *263*(1376), 1565–1569.

6. Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., & 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)* (pp. 4187–4193). Taipei, Taiwan: IEEE Press.

7. 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.

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

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

10. Brutschy, A., Pini, G., & Decugnière, A. (2012). *Grippable objects for the footbot* (Technical report TR/IRIDIA/2012-001). Brussels: IRIDIA, Université Libre de Bruxelles.

11. Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., & Dorigo, M. (2014). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, *28*(1), 101–125.

12. Campo, A., & Dorigo, M. (2007). Efficient multi-foraging in swarm robotics. In M. Capcarrere, A. A. Freitas, P. J. Bentley, C. G. Johnson, & J. Timmis (Eds.), *Advances in Artificial Life: Proceedings of the 8th European Conference on Artificial Life (ECAL 2005)* (pp. 696–705). Berlin: Springer.

13. 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. A., 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., & Vaussard, F. (2011). Swarmanoid, the movie. In *AAAI-11 Video Proceedings*. AAAI Press.

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

15. 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., Di Caro, G. A., Ducatelle, F., Ferrante, E., Förster, A., Gonzales, J. M., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., Oca, M. M. de, 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., & Vaussard, F. (2013). Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, *20*(4), 60–71.

16. Ducatelle, F., Di Caro, G. A., Förster, A., Bonani, M., Dorigo, M., Magnenat, S., Mondada, F., O'Grady, R., Pinciroli, C., Rétornaz, P., Trianni, V., & Gambardella, L. M. (2014). Cooperative navigation in robotic swarms. *Swarm Intelligence*, *8*(1), 1–33.

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

18. Ducatelle, F., Di Caro, G. A., Pinciroli, C., Mondada, F., & Gambardella, L. M. (2011). 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)* (pp. 4981–4988). Piscataway, NJ: IEEE Computer Society Press.

19. Feng, L., Borenstein, J., & Everett, H. R. (1994). *"Where am I" sensors and methods for autonomous mobile robot positioning*. Ann Arbor, MI: University of Michigan Press.

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

21. Fowler, H. G., & Robinson, S. W. (1979). Foraging by *Atta sexdens* (Formicidae: Attini): Seasonal patterns, caste and efficiency. *Ecological Entomology*, *4*(3), 239–247.

22. Goldberg, D., & Matarić, M. J. (2002). Design and evaluation of robust behavior-based controllers for distributed multi-robot collection tasks. In T. Balch & L. E. Parker (Eds.), *Robot teams: From diversity to polymorphism* (pp. 315–344). Natick, MA: A K Peters/CRC Press.

23. Gordon, D. M. (1996). The organization of work in social insect colonies. *Nature*, *380*(6570), 121–124.

24. Grabowski, R., Navarro-Serment, L. E., Paredis, C. J. J., & Khosla, P. K. (2000). Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, *8*(3), 293–308.

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

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

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

28. Hart, A. G., & 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*(5), 387–392.

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

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

31. Johansson, R., & Saffiotti, A. (2009). Navigating by stigmergy: A realization on an RFID floor for minimalistic robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 245–252).

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

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

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

35. Lein, A., & Vaughan, R. T. (2008). Adaptive multi-robot bucket brigade foraging. In S. Bullock, J. Noble, R. Watson, & M. A. Bedau (Eds.), *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems* (pp. 337–342). Cambridge, MA: MIT Press.

36. Lein, A., & 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)* (pp. 601–606). Piscataway, NJ: IEEE Press.

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

38. Mayet, R., Roberz, J., Schmickl, T., & Crailsheim, K. (2010). Antbots: A feasible visual emulation of pheromone trails for swarm robots. In M. Dorigo, M. Birattari, G. Di Caro, R. Doursat, A. P. Engelbrecht, D. Floreano, L. Gambardella, E. Groß, R. Şahin, T. Stützle, & H. Sayama (Eds.), *Proceedings of the 7th International Conference on Swarm Intelligence (ANTS 2010)* (pp. 84–94). Berlin: Springer.

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

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

41. Østergaard, E. H., Sukhatme, G. S., & 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* (pp. 29–30). Montreal: ACM Press.

42. Panait, L., & Sean, L. (2003). Learning ant foraging behaviors. In J. Pollack, M. Bedau, P. Husbands, T. Ikegami, & R. A. Watson (Eds.), *Proceedings of the 9th International Conference on the Simulation and Synthesis of Living Systems* (pp. 569–574). Cambridge, MA: MIT Press.

43. Panait, L., & Sean, L. (2004). A pheromone-based utility model for collaborative foraging. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, Vol. 1 (pp. 36–43). Piscataway, NJ: IEEE Computer Society Press.

44. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G. A., Ducatelle, F., Birattari, M., Gambardella, L. M., & Dorigo, M. (2012). ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, *6*(4), 271–295.

45. Pini, G., Brutschy, A., Birattari, M., & Dorigo, M. (2009). Task partitioning in swarms of robots: Reducing performance losses due to interference at shared resources. In J. A. Cetto, J. Filipe, & J.-L. Ferrier (Eds.), *Informatics in Control, Automation and Robotics* (pp. 217–228). Berlin: Springer.

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

47. Pini, G., Brutschy, A., Francesca, G., Dorigo, M., & Birattari, M. (2012). Multi-armed bandit formulation of the task partitioning problem in swarm robotics. In M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. P. Engelbrecht, R. Groß, & T. Stützle (Eds.), *Proceedings of the 8th International Conference on Swarm Intelligence, ANTS 2012* (pp. 109–120). Berlin: Springer.

48. Pini, G., Brutschy, A., Scheidler, A., Dorigo, M., & Birattari, M. (2012). *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/.

49. Pini, G., Gagliolo, M., Brutschy, A., Dorigo, M., & Birattari, M. (2013). Task partitioning in a robot swarm: A study on the effect of communication. *Swarm Intelligence*, *7*(2), 173–199.

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

51. Ratnieks, F. L. W., & Anderson, C. (1999). Task partitioning in insect societies. II. Use of queueing delay information in recruitment. *The American Naturalist*, *154*(5), 536–548.

52. Rekleitis, I., Dudek, G., & Milios, E. (2001). Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, *31*(1–4), 7–40.

53. Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In *Proceedings of the SAB 2004 Workshop on Swarm Robotics* (pp. 10–20). Berlin: Springer.

54. Schatz, B., Lachaud, J.-P., & Beugnon, G. (1996). Polyethism within hunters of the ponerine ant, *Ectatomma ruidum* Roger (Formicidae, Ponerinae). *Insectes Sociaux*, *43*(2), 111–118.

55. Schmickl, T., & Crailsheim, K. (2008). Trophallaxis within a robotic swarm: Bioinspired communication among robots in a swarm. *Autonomous Robots*, *25*(1–2), 171–188.

56. Shell, D. J., & 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)* (pp. 2717–2723). Piscataway, NJ: IEEE Press.

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

58. Sugawara, K., Kazama, T., & 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)*, Vol. 3 (pp. 3074–3079). Piscataway, NJ: IEEE Press.

59. Vasconcelos, H. L., & 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.

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

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

62. Winfield, A. F. T. (2009). Foraging robots. In R. A. Meyers (Ed.), *Encyclopedia of Complexity and System Science* (pp. 3682–3700). Berlin: Springer.