Original Paper

Adaptive Behavior

Adaptive Behavior
21(2) 118–136
© The Author(s) 2013
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1059712313484771
adb.sagepub.com
SAGE

Autonomous task partitioning in robot foraging: an approach based on cost estimation

Giovanni Pini, Arne Brutschy, Carlo Pinciroli, Marco Dorigo and Mauro Birattari

Abstract
We propose an approach for autonomous task partitioning in swarms of foraging robots. Task partitioning is the process of decomposing tasks into sub-tasks. Task partitioning impacts tasks execution and associated costs. Our approach is characterized by the use of a cost function, mapping the size of sub-tasks to the overall task cost. The robots model the cost function and use the model to select sub-tasks to perform, aiming to minimize costs. Our approach separates the task partitioning process from task-specific actions and it does not require *a priori* assumptions to be made about the best partitioning strategy to employ. We study a foraging scenario in which object transportation is performed by different robots, each moving objects for a limited distance. The robots autonomously decide the distance traveled on the basis of our approach. The robots use odometry for navigational purposes; we show that task partitioning reduces the impact of odometry errors and improves performance. We validate our approach using simulation-based experiments. We study how the swarm partitions transportation under a number of experimental conditions characterized by different levels of odometry accuracy, size of the environment and the swarm, and total transportation distance. Our approach leads to partitioning solutions that are appropriate for each condition.

Keywords
Task partitioning, swarm robotics, swarm intelligence, self-organization, foraging

1 Introduction

Task partitioning is the process by which tasks are decomposed into sub-tasks (Jeanne, 1986). In artificial systems, task partitioning is used in the fields of computer science and robotics. In nature, examples of task partitioning have been observed in social insects (see Ratnieks and Anderson (1999)), which utilize task partitioning in activities involving material transportation. Examples are foraging (Seeley, 1995), hunting (Schatz, Lachaud, & Beugnon, 1996), nest excavation (Anderson & Ratnieks, 2000), and garbage disposal (Hart, Francis, & Ratnieks, 2001).

The benefits of task partitioning are manifold. Task partitioning allows for physical separation of the workers, diminishing the negative effects of interference (Hart & Ratnieks, 2000) and competition for shared resources (Pini, Brutschy, Birattari, & Dorigo, 2009). Partitioning a task also allows for a better exploitation of specialization: sub-tasks can be assigned to the workers that are better suited to perform them (Ratnieks & Anderson, 1999). Finally, task partitioning can increase the efficiency in performing a task (Fowler & Robinson, 1979).

Our research focuses on the application of task partitioning to swarm robotics, a branch of robotics that studies the implementation and control of large groups of autonomous robots (see Brambilla, Ferrante, Birattari, and Dorigo (2013) for a review). Swarm robotics is based upon the principles of swarm intelligence (Dorigo & Birattari, 2007): decentralization, distributed control, limited perception, and local communication. The goal is to implement systems that are robust, tolerant to faults, scalable, and flexible (Dorigo & Şahin, 2004). Swarms of robots are frequently implemented taking inspiration from insect colonies that exhibit such characteristics. Swarms in which the robots are capable of autonomously partitioning tasks would be extremely flexible since they

IRIDIA, Université Libre de Bruxelles, Belgium

Corresponding author:
Giovanni Pini, IRIDIA, Université Libre de Bruxelles, 50 Av. Franklin Roosevelt CP 194/6, 1050 Brussels, Belgium.
Email: gpini@ulb.ac.be

could adapt the way they organize task execution to specific environments and goals.

In this work, we focus on foraging. Foraging is an important benchmark in swarm robotics, because it abstracts practical problems such as search and rescue, mine clearance, and cleaning (Winfield, 2009). We propose an approach for robot swarms to autonomously partition the task of transporting an object.

So far, all of the existing work on task partitioning in foraging proposes methods built with the sole purpose of reducing physical interference. These methods are blended in with the task-specific actions the robots perform and rely upon characteristics of the environment. Typically, either the tasks are partitioned *a priori*, or the sub-tasks dimension is regulated on the basis of measures of interference (e.g. perception of obstacles). These approaches have two limitations. First, each method is only suitable for the specific setup for which it is built. Second, the assumption that interference is the only factor determining the best way to partition a task is too restrictive and limiting. As mentioned, the benefits of task partitioning go beyond reduction of interference. In addition, task partitioning entails overhead costs and therefore using it to reduce interference in contexts where interference is not a key factor is likely to introduce costs without entailing benefits.

The key idea that differentiates our from others is to associate *costs* with the execution of tasks and to drive task partitioning towards a minimization of these costs. The one of cost is a generic concept: costs can be used to measure performance independently of the tasks performed by the robots and the context in which the robots operate. The advantages of reasoning upon costs are: (i) the task partitioning process can be decoupled from the specific robot actions and setup; and (ii) the factors determining the best way of partitioning a task are implicitly taken into account (i.e. no *a priori* assumptions about these factors must be made). These are necessary steps towards a general method for autonomous task partitioning.

In our foraging scenario, we consider the case in which the objects to be foraged are clustered in a unique location in the environment, the *source*. Each robot must initially locate the source. Upon finding it, a robot utilizes odometry to keep an estimate of its own position relative to the source. This estimate is used by the robot to return to the source on subsequent trips. Odometry is suitable to robotics because of its simplicity and because it can be implemented with sensors that are available on many mobile robots. The drawback of odometry is that it suffers from estimation errors that grow with the distance traveled.

In our previous study (Pini, Brutschy, Scheidler, Dorigo, & Birattari, 2012), we study the same setup to show that task partitioning can be employed to reduce the negative impact of odometry errors therefore increasing the foraging performance. An object is delivered to the nest through sequential steps, performed by different robots; each robot transports the object for a limited distance and hands it over to another robot, which continues transportation.

In this work, we extend our previous study with a mechanism that allows each robot to decide autonomously the distance to cover. We use simulation-based experiments to show that the optimal way of partitioning object transportation depends on the accuracy of the odometry and on properties of the environment. We present a swarm that autonomously partitions the transportation of objects, performing well across all of the tested experimental conditions.

The contents of the paper are organized as follows. In Section 2, we present related work on the topic of task partitioning, focusing on swarm robotics. In Section 3, we describe the foraging scenario, the behavior of the robots, and the application of our approach. In Section 4, we present the simulation tools we employed to perform the experiments. In Section 5, we describe the experiments and comment on the results. In Section 6, we summarize the contents of the paper and discuss directions for future research.

2 Related work

Most of the research on the topic of task partitioning has been conducted in the field of entomology (for a review refer to Ratnieks and Anderson (1999) and Hart, Anderson, and Ratnieks (2002)). Here, we focus on works devoted to task partitioning in artificial systems.

Many examples of task partitioning applied to artificial systems are found in computer science. Algorithms based on the *divide and conquer* paradigm employ task partitioning (Cormen, Leiserson, Rivest, & Stein, 2001). In operating systems, the execution of processes on a CPU is managed via task partitioning (Bovet & Cesati, 2005). Similarly, in architectures with multiple processors, programs are divided into tasks to be performed in parallel (Ennals, Sharp, & Mycroft, 2005).

In swarm robotics, task partitioning has been employed in foraging to reduce physical interference. Physical interference is an issue in multi-robot systems: the robots share the same physical space and interfere with each other's movements. As the robot density increases, the robots spend more resources dealing with physical interference rather than performing tasks and performance is affected negatively (Lerman & Galstyan, 2002).

In foraging, task partitioning is implemented by limiting the size of the area within which each robot operates. Objects are delivered to the target by several robots. Different robots contribute to transportation; the objects are either handed over from robot to robot or deposited on the ground. The result is that the robots interfere less with each other and performance improves over a non-partitioning solution.

Drogoul and Ferber (1992) were the first to propose a solution based on task partitioning in foraging. In their work, the robots hand over objects one another, forming chains along which objects are transferred. An analogous solution is proposed in Østergaard, Sukhatme, and Matarić (2001). In Fontan and Matarić (1996), Goldberg and Matarić (2002), and Pini et al. (2009), the environment is divided into predefined areas, within which one or more robots perform foraging. In Shell and Matarić (2006), Lein and Vaughan (2008), and Lein and Vaughan (2009), the working areas are not associated with a given location in the environment, but defined with respect to the position of the robot.

Pini et al. (2011) proposed an algorithm for robot swarms to decide whether to employ task partitioning to perform a given task. In follow-up works, the authors showed that the same decision can be taken using algorithms proposed in the literature to solve multi-armed bandit problems (Pini, Brutschy, Francesca, Dorigo, & Birattari, 2012; Pini, Gagliolo, Brutschy, Dorigo, & Birattari, 2013).

Pini et al. (2012) studied the same setup proposed in this article. The robots forage for objects from a unique location and use odometry to estimate their position relative to this location. The authors propose a solution based on task partitioning to tackle the problem: each robot transports objects for a limited distance. The objects are handed over directly from robot to robot. The authors show that task partitioning improves performance. In addition, they use the system as a testbed for studying the costs of direct object transfer.

Note that most of the work presented here has been performed with simulation tools only. The only exceptions are the works of Fontan and Matarić (1996), Goldberg and Matarić (2002), and Pini et al. (2012).

3 Description of the approach

In this section, we present our approach to autonomous task partitioning in a swarm of foraging robots. In Section 3.1, we describe the foraging scenario and in Section 3.2 the application of the approach to such a scenario. In Section 3.3, we present the task partitioning algorithms tested in the experiments, including an algorithm based on the proposed approach.

3.1 The foraging scenario

The goal of the robots is to collect objects and transport them to a unique location referred to as *nest*. We focus on the case in which the objects are located in a unique position in the environment, the *source*. In this context, a *task* consists in transporting an object from the source to the nest. Foraging consists in the repetition of the transportation task.

Figure 1 illustrates a finite state machine that describes the behavior of a robot. Initially, robots have no information and they perform a random walk to search for objects. Upon finding and collecting an object, a robot navigates towards the nest. The robot can reach the nest by heading towards a landmark (three lights) that marks its location.

While navigating towards the nest, the robot maintains an estimate of its position relative to the location where the object was collected. Upon reaching the nest, the robot releases the object and uses the position estimate to return to the location where the object was collected. While traveling towards this location the robot ignores other objects it may encounter on the way.

Estimating the position of a robot with respect to a target is a problem known as localization. Several techniques have been proposed in the literature to tackle this problem (see Feng, Borenstein, and Everett (1994) for a review). Most techniques are not suited for swarm robotics, because either they rely upon external devices and infrastructure or they require complex sensing and computational capabilities. In our experiments, the robots use odometry to estimate their position with respect to the source. Odometry consists in the integration of motion sensor information to estimate the position of the robot. In contrast to other techniques, odometry is suited for swarm robotics as it can be easily

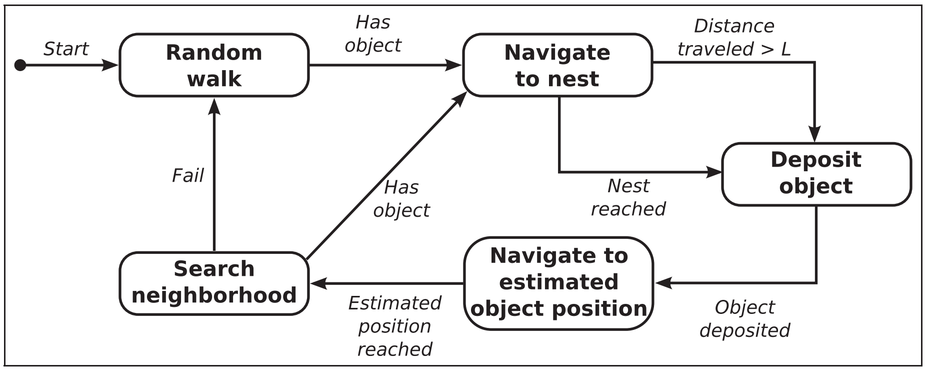

Figure 1. Finite state machine describing the high-level behavior of each robot.

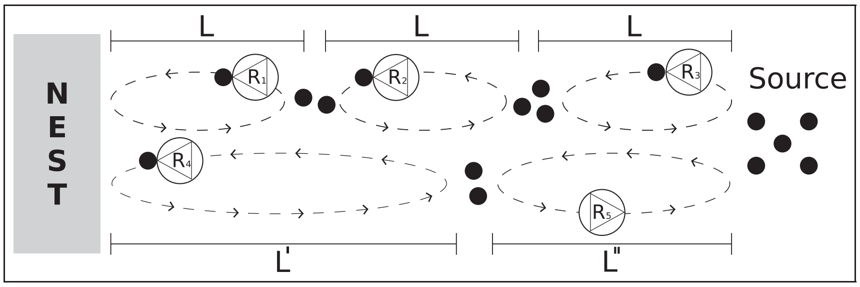

Figure 2. Representation of the task and contribution of the robots. Each robot R_i transports objects (black circles) towards the nest for a limited distance L, which can be different from robot to robot. The way the task is partitioned depends on the values L used by the robots.

implemented using sensors available on many mobile robots (typically motor encoders) and it requires low computational capabilities. However, it suffers from the accumulation of errors in time that are caused by several phenomena (e.g. mechanical imperfections, finite sensors resolution, wheel slippage). Owing to these errors, the position estimate of the robot can deviate from the real position where the robot collected the object. Upon reaching the estimated position at which the object was collected, the robot performs a *neighborhood search*, to locate objects nearby. Neighborhood search can compensate for relatively small estimation errors. In case the error is large, it is likely to fail. If this happens, the robot resumes performing random walk, to locate again objects. We say that the robot *got lost* when the neighborhood search fails and the robot must random walk again. Random walk is a time-consuming operation, therefore it is desirable to minimize the frequency at which it is necessary.

In a previous work, we showed that task partitioning reduces the negative effect of odometry noise and improves the localization capabilities of the robots (Pini et al., 2012). Instead of traveling directly from source to nest, each robot travels a limited distance L from the location where an object was collected (i.e. performing only a sub-task of the transportation task). Upon traveling such a distance, referred to as *partition length*, the robot deposits the object on the ground and returns to the location where the object was collected using its position estimate. Since the odometry error grows with the distance traveled, the expected estimation error is smaller compared with the case in which the robot travels directly to the nest. The result is that the probability for a robot to get lost diminishes, therefore reducing the total random walk time of the swarm.

As objects can be deposited everywhere in the environment, the robots that are performing random walk can find them not only at the source, but also in other locations. The object transportation task is therefore executed as a sequence of sub-tasks performed by different robots, as represented in Figure 2. The number of sub-tasks depends on the distance between source and nest and on the partition length L used by the robots. Note that the behavior of the robots is the same regardless of the fact that their odometry estimate refers to the actual position of the source (as for the robots R_3 and R_5 in Figure 2) or another location of the environment (as for the robots R_1, R_2, and R_4). The robots have no means to determine whether they are collecting objects from the source or from other locations.

The value of the partition length L affects the system in different ways. On the one hand, a large value is desirable, since the robots travel further while carrying objects and reduce the number of times each object is deposited on the ground. Depositing an object introduces overheads because the same object must be found and picked up by a different robot. On the other hand, traveling further increases the magnitude of the odometry error and the likelihood that the robot gets lost.

3.2 Autonomous task partitioning in foraging: the proposed approach

We tackle the problem of autonomous task partitioning with an approach that has been employed in other contexts in robotics (for example, task allocation (Torbjørn, Matarić, & Sukhatme, 2009) and foraging (Tangamchit, Dolan, & Khosla, 2002)): the decisions are taken on the basis of cost or utility measures. The idea at the basis of our approach is illustrated in Figure 3 (left). A *cost function* maps the *amount of work* contributed by a robot to the *cost* of executing the overall task. With its contribution in terms of amount of work, a robot performs a sub-task of the overall task. The way the overall task is partitioned and the cost for its execution therefore depend on the amount of work contributed by each robot.

Each robot employs the cost function to decide its contribution to the overall task in terms of amount of work (i.e. the sub-task it performs), with the goal of minimizing the resulting overall task cost. The cost function cannot be defined *a priori* since it depends on properties of the environment and the tasks that may be unknown. In addition, the cost function may change

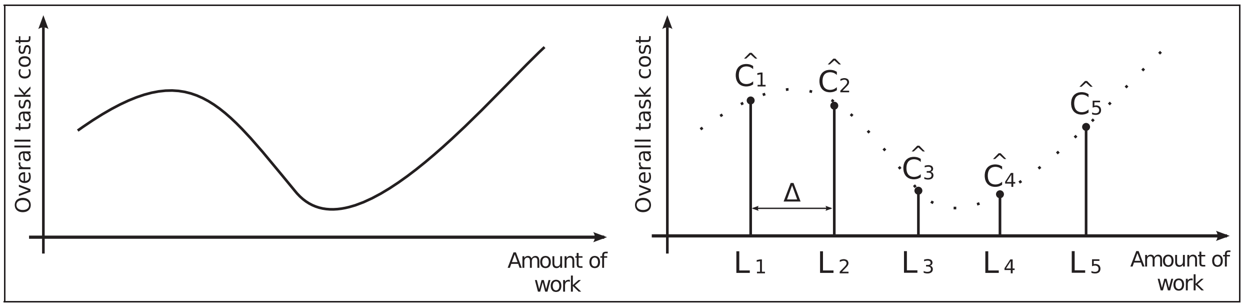

Figure 3. Cost function (left) and its model as built by a robot (right). The function maps the amount of work contributed by a robot that performs a sub-task to the resulting cost for performing the overall task. This information is used by each robot to decide how to partition the overall task.

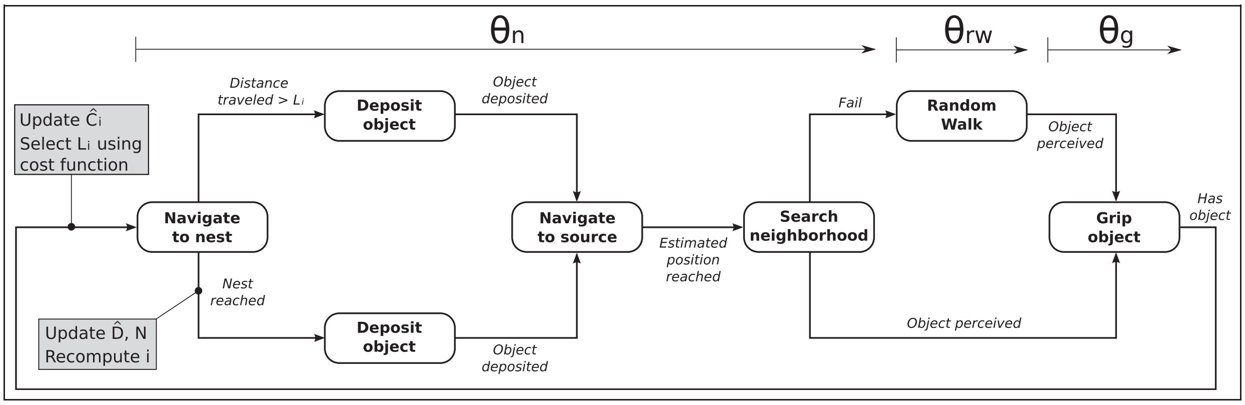

Figure 4. High-level representation of the behavior of the robots.

in time, since the cost associated to the execution of tasks is likely to vary as a result of the actions of the robots. For these reasons, each robot is merely able to build a *model* of the real cost function. This model must be updated in time by the robot.

The way costs and amount of work are measured depends on the specific tasks to be performed and the goals to be attained. In the foraging scenario, the amount of work contributed by a robot can be measured in terms of the distance L traveled by that robot with an object. In the following, we refer to the *length* of a sub-task performed by a robot to indicate the amount of work contributed by that robot. Since the goal of the swarm is to maximize the total number of objects transported to the nest, which corresponds to the number of task instances performed, we express *costs* as time. Therefore, the *cost function* maps the distance traveled by a robot to the time required to perform the transportation task.

The robots build a discrete model of the cost function, as represented in Figure 3 (right). The model consists of a finite set of cost estimates $\hat{C}_i$, each associated to a given partition length value L_i. In this context, selecting the length of a sub-task corresponds to selecting the index i (i.e. a value of L_i) on the basis of the estimates $\hat{C}_i$. In the rest of this section we describe how each robot builds and uses the cost function model to achieve autonomous task partitioning.

Figure 4 illustrates the high-level behavior of each robot and visualizes the concepts explained in the rest of this section. In the figure, the white rectangles indicate actions performed by the robot, the gray ones indicate internal information update.

The rest of this section is organized as follows. In Section 3.2.1, we illustrate how discretization is performed and its implications on the task partitioning process. In Section 3.2.2, we explain how the robots estimate costs.

3.2.1 Discretization.

Each robot individually builds a set $\mathbb{L}$ of potential partition lengths. Here $\mathbb{L}$ corresponds to the domain of the cost function modeled by a robot (see Figure 3, right). Each robot selects a value from $\mathbb{L}$ as partition length. In this section, we explain how the robots build the set $\mathbb{L}$. In Section 3.3, we illustrate possible algorithms to select a partition length value.

Number of elements in $\mathbb{L}$. The set $\mathbb{L}$ is composed of N values, each corresponding to a different partition length. Each robot builds the set by discretizing an estimate of the overall task length D (i.e. of the distance

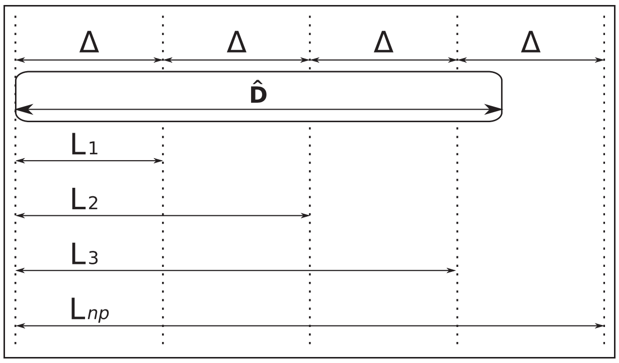

Figure 5. Example representing the set $\mathbb{L}$ from which a robot selects the partition length value. The robot builds the set by discretizing the estimated task length $\hat{D}$ with a step Δ, obtaining $N-1$ values L_i ($N=4$ in this figure). Here L_{np} corresponds to performing the transportation task as an unpartitioned task.

between source and nest), with a discretization step Δ (see Figure 3, right). Here N is computed by a robot as

$$N = \left\lceil \frac{\hat{D}}{\Delta} \right\rceil \tag{1}$$

In our setup, we fixed the discretization step Δ to 0.5 m which corresponds (roughly) to the visual range of the robots. Discretization is performed by a robot on the basis of $\hat{D}$, which is its estimate of the distance D between source and nest. Note that the robot can find objects deposited by other robots along the way between source and nest (i.e. closer to the nest than the source is) and therefore the value $\hat{D}$ of a robot can be an underestimate of D. For this reason, the robot must keep $\hat{D}$ and $\mathbb{L}$ up to date.

Elements of $\mathbb{L}$. The values L_i that compose the set $\mathbb{L}$ are calculated as follows:

$$L_i = \Delta \cdot i \quad \text{with } i \in \{1, 2, \ldots, N-1\} \tag{2}$$

In addition to the $N-1$ values L_i, $\mathbb{L}$ contains the special value L_{np}, which corresponds to performing transportation as an unpartitioned task. A robot selecting L_{np} directly transports the object to the nest.

Figure 5 represents the set $\mathbb{L}$ and illustrates the relationship between N, the estimated length of the task $\hat{D}$, the discretization step Δ, and the values L_i.

Every time a robot grips an object, it selects a partition length L_i that is used for the subsequent trip towards the nest (see Figure 4, left-hand side). Consequently, different task instances (i.e. the transportation of different objects) can be partitioned in different ways, depending on the partition length L_i selected by the robots involved in transportation (see, for example, Figure 2). Note that each robot is not aware of the value L_i selected by other robots. The global process by which the overall task is partitioned into sub-tasks results from independent choices in a self-organized manner.

Estimation of D. Each time a robot reaches the nest (label *nest reached* in Figure 4), it checks whether $\hat{D}$ underestimates the distance to the source. In this case, the robot sets $\hat{D}$ to the estimated distance to the source and it updates the set $\mathbb{L}$, using Equations (1) and (2). Note that odometry errors can cause a robot to overestimate D, however the robots can only determine whether $\hat{D}$ underestimates D (by measuring a value greater than $\hat{D}$) but not whether it overestimates it. Initially, robots have no information about the source-to-nest distance. The value of $\hat{D}$ is initialized to zero and $\mathbb{L}$ only contains L_{np}. Consequently, the robots travel directly to the nest in their first trip, thus calculating a first estimate $\hat{D}$.

3.2.2 Estimation of the costs. Each robot approximates the overall task cost $\hat{C}_i$ associated with the partition length L_i on the basis of the cost experienced when performing its own sub-tasks.

The moment an object is gripped marks the beginning of a new sub-task for a robot. Upon gripping an object, a robot updates the cost estimate $\hat{C}_i$ and selects a partition length value L_i to employ next (see Figure 4, left). The cost estimates $\hat{C}_i$ are updated as follows:

$$\hat{C}_i = (1-\alpha)\, \hat{C}_i' + \alpha\, \bar{C} \tag{3}$$

where $\alpha \in (0, 1]$ is a memory factor, $\hat{C}_i'$ is the value of the cost estimate before the update, and $\bar{C}$ is the cost associated with the last trip towards the nest (i.e. the last sub-task performed). Here $\bar{C}$ is computed as

$$\bar{C} = \frac{\hat{D}}{L_i}\left(\Theta_n + \Theta_g\right) + \Theta_{rw} \tag{4}$$

where $\Theta_n + \Theta_g$ is the measured cost of the last sub-task performed (see Figure 4, top). This cost is composed of two parts. Here Θ_n accounts for the navigational costs (i.e. transporting and depositing an object, returning to the source, and performing neighborhood search), Θ_g is the cost of gripping another object, an action that must be performed before the following sub-task can be started, and Θ_{rw} measures the time spent performing random walk in case the robot got lost. Note that $\Theta_n + \Theta_g$ measures the cost of the actions needed to perform sub-tasks, while Θ_{rw} is a penalty due to a robot getting lost.

The cost estimate $\hat{C}_i$ updated by a robot depends on whether or not the robot reached the nest during the last execution of the task. If the robot reached the nest (transition labeled *nest reached* in Figure 4), it recomputes the index i on the basis of the actual distance traveled. In fact, if the value L_i selected by the robot upon gripping an object exceeds the distance between the robot and the nest, the robot actually travels a shorter distance than L_i. In this case, the cost to be updated is that associated with a shorter sub-task (i.e. a smaller L_i). If, on the other hand, the robot does not reach the nest, the cost $\hat{C}_i$ subsequently updated is that associated with the selected L_i (i.e. the robot actually travels a distance L_i).

The cost estimates are initialized randomly: when a robot reaches the nest and, according to Equation 2, adds new values L_i to $\mathbb{L}$, it initializes the associated costs $\hat{C}_i$ with a random value.

3.3 Task partitioning algorithms

In the experiments, we compare different task partitioning algorithms to select the value of the partition length from $\mathbb{L}$. We compare an algorithm based on the approach proposed in this paper with a set of reference algorithms. The reference algorithms utilize the set $\mathbb{L}$ to select the value of the partition length L_i, but in none of them the selection is based upon the cost function model.

The cost-based partitioning algorithm. In this work, the robots use the *ε-greedy* algorithm (Sutton & Barto, 1998) to select the value of the partition length. ε-greedy selects with probability $1 - \varepsilon$ the value L_i associated to the minimum cost $\hat{C}_i$ and with probability ε a random value. We call *cost-based partitioning* algorithm the task partitioning algorithm that utilizes ε-greedy and the cost estimates to select the partition length value.

Note that the proposed approach does not require any specific algorithm to select the value L_i. Other algorithms, such as reinforcement learning techniques (see Sutton and Barto (1998)), can be used in place of ε-greedy. We decided to use ε-greedy because it is simple and because its only parameter ε directly expresses the degree of exploration.

The fixed algorithms. In the *fixed* algorithms, the partition length L_i is fixed to a given value, which is the same for all of the robots and remains constant over time. We refer to a fixed algorithm with the label *fixed X*, where X identifies the partition length value L_i. A special case of fixed algorithm is the *never partition algorithm*: in this case, the robots do not partition the transportation task (i.e. $L_i = L_{np}$).

The random initialization algorithm. The *random initialization algorithm* consists in stochastically selecting the partition length value in $\mathbb{L}$ at the beginning of the experiment. Each robot selects a value L_i and never changes it during the course of the experiment. The algorithm requires the initialization of the set $\mathbb{L}$. To this aim, the first time a robot grips an object, it transports it directly to the nest so that D can be estimated. We add 10% of the value to the estimate, to partially compensate for underestimation errors. The resulting value is used to initialize the set $\mathbb{L}$ as described by Equations (1) and (2). The robot stochastically selects a value L_i from the set $\mathbb{L}$ and uses it throughout the rest of the experiment.

We also tested a variation of the algorithm, whereby the robots stochastically select the partition length value in $\mathbb{L}$ each time an object is gripped. The algorithm performed badly across all of the experiments and therefore we do not include it in this work.

4 Experimental tools

In this section, we describe the tools that we utilized to perform the experiments. All of the experiments are performed in simulation. In the experiments, we simulate the foot-bot robot using the ARGoS simulator (Pinciroli et al., 2012). ARGoS simulates the physics and the devices of the foot-bot and the properties of the environment which are relevant for our foraging scenario. The contents of this section are organized as follows. In Section 4.1, we describe the environment in which the robots perform foraging. In Section 4.2, we introduce features of the foot-bot that are relevant for this work. In Section 4.3, we describe the simulation software ARGoS and illustrate how the studied system is implemented in simulation.

4.1 Environment

The robots forage in the environment represented in Figure 6 (left). The environment is a rectangular arena surrounded by walls. Its dimensions W and L depend on the specific experiment. The nest, located near one of the walls, is marked by a 1.4 m by 0.45 m black rectangular patch on the ground. Three lights, located in proximity of the nest, are used by the robots to determine the direction of the nest.

The distance D between source and nest, measured from the center of the nest to the object in the center of the source, depends on the specific experiment. Five objects, arranged as represented in Figure 6, are positioned at the source. The objects are positioned at a distance of 0.17 m from the object in the center. Each time a robot removes an object from the source, a new one is added in the same location. When a robot releases an object within the nest, an instance of the transportation task is complete and the object is removed.

4.2 Foot-bots and objects

The foot-bot is a mobile ground robot developed within the *Swarmanoid* project (Dorigo et al., 2013). The foot-bot is roughly cylindrical-shaped, it has a diameter of

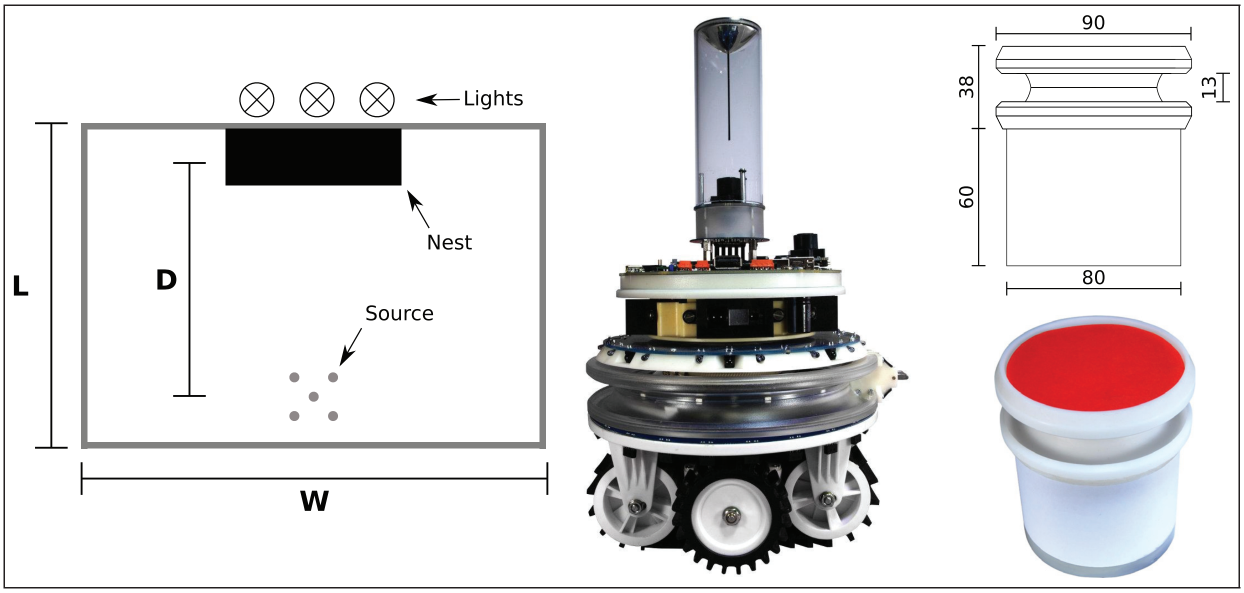

Figure 6. Foraging environment (left), a foot-bot (center), and one of the objects to be collected (right). The length *L* and the width *W* of the environment depend on the specific experiment. The source, composed of five objects, is at a distance *D* from the nest. The nest (black rectangle) can be reached by approaching the three lights located behind.

170 mm, a height of 290 mm, and it weighs 1.8kg. Figure 6 depicts a foot-bot and one of the objects we designed to perform foraging experiments with the foot-bots. For more information about these objects, refer to Brutschy, Pini, and Decugnière (2012).

The foot-bot navigates the environment by means of a differential drive system that combines tracks and wheels. A gripper is used to grip objects for transportation. The gripper is mounted on a rotating plastic ring, that allows the robot to point the gripper towards a desired direction (e.g. to position an object before releasing it). The ring also hosts 12 RGB LEDs that we use to convey information about the internal status of the robot.

Among the sensors available on the foot-bot, we use the omnidirectional camera, the infrared ground sensors, and the infrared proximity sensors. The omnidirectional camera is a system composed of a color camera pointing to a convex mirror mounted on top of a transparent tube. Objects and LEDs can be perceived up to a distance of roughly 0.5m. The infrared ground sensors are used to detect the black patch representing the nest. Twenty-four infrared sensors are utilized to perceive the lights marking the nest. They also serve as virtual bumpers, used to detect obstacles. For more information about the foot-bot, see Bonani et al. (2010).

In the rest of this section, we provide implementation details about the behavior of the robots.

Random walk. By default, the robots move straight, at a maximum velocity of 0.1m/s, while avoiding obstacles. With a 5% probability per control-step,[1] a new direction of motion is uniformly sampled in $[-115°, 115°]$. The neighborhood search is also performed as a random walk, but the robot remains in a circular area of radius 0.5 m, centered around the expected position of the source. If a robot is about to leave the search area, it generates a new random direction biased towards the center. A robot abandons the neighborhood search in case it has been performing it for 60 s without detecting any object.[2] In this case, in fact, it is likely that the robot reached a position far away from the actual source position.

Object gripping. The foot-bot uses the omnidirectional camera to perceive and approach objects in the surroundings. While approaching an object with the intent of gripping it, a robot lights up its LEDs in blue, to repel other robots from the same object. For the same reason, the LEDs are lit up in blue also during the transportation of an object.

Check for unreachable destination. Owing to odometry errors, the robot may try to reach a position outside the perimeter of the arena while navigating to the source. To detect such an event, each robot periodically checks its estimated distance to the source. If the distance does not decrease in time, the robot assumes that it is navigating to an unreachable destination and it resumes random walk (i.e. it got lost). In fact, if the estimated distance to the source is not decreasing, an obstacle blocks the movements of the robot. If this happens for a long period of time (30 consecutive seconds), it is likely that the obstacle is a wall, rather than another

Algorithm 1 Pseudo-code of the odometry noise model.

```
 1: $actuatedWheelSpeedR$; $actuatedWheelSpeedL \leftarrow ControllerStep()$
 2: if $actuatedWheelSpeedR > 0$ then
 3:     $wheelSpeedR = actuatedWheelSpeedR + \mu_{\text{noise}} * actuatedWheelSpeedR$
 4: end if
 5: if $actuatedWheelSpeedL < 0$ then
 6:     $wheelSpeedL = actuatedWheelSpeedL - \mu_{noise} * actuatedWheelSpeedL$
 7: end if
 8: if Object gripped then
 9:     $\mu_{noise} = RAYLEIGH(\sigma)$
10: end if
```

robot. The mechanism is prone to errors: if the density of robots is high, the movements more constrained and the mechanism can be triggered by the presence of other robots.

4.3 Simulation of the system using ARGoS

ARGoS[3] (Pinciroli et al., 2012) is the robot simulator that we employed for the experiments presented in this paper. ARGoS can simulate thousands of robots in real time and it is highly customizable.

In the experiments presented in this work, we employ the two-dimensional dynamics physics engine offered by ARGoS. The simulation proceeds at discrete time steps, 0.1 simulated seconds long. The simulated sensors and actuators are subject to noise. Gaussian noise with a standard deviation of 0.02m is added to the distance readings of the camera. At each time step, a uniform random value between -5% and 5% of the reading is added to the measures of the ground, light, and proximity sensors.

In simulation, we model object gripping using 80 time samples collected with six real foot-bots performing foraging in a $W = 6.7$m by $L = 4.5$m arena. Each sample records the time spent by a robot to grip an object (i.e. from the moment the object was perceived to the moment it was gripped). A figure reporting the empirical distribution of the time samples is available in Pini, Brutschy, Pinciroli, Dorigo, and Birattari (2012). The samples are utilized in simulation to model the object gripping time. Each time a robot grips an object, a random value is selected from the set of samples. The robot waits in place for a time corresponding to the selected value, before it can undertake the following action. This solution allows us to model in a simple but realistic way the variability observed with the real foot-bots in the object gripping time.

The odometry noise plays an important role in our experimental setup as it defines how successful the robots are in finding objects when returning to the source. In this work, we use the same model that we proposed in Pini et al. (2012). The model is built on the basis of odometry error samples, collected with real foot-bots performing foraging. A plot reporting these samples is available online in Pini et al. (2012).

Algorithm 1 reports the pseudo-code describing the implementation of odometry noise in simulation. The foot-bots suffer from a systematic drift towards the left-hand side with respect to the direction of motion, obtained in simulation by modifying the actuated values of the two wheel speeds (lines 2–7). The robot is not aware of the drift: odometry is computed using *actuatedWheelSpeedR* and *actuatedWheelSpeedL*. The parameter μ_{noise} represents the percentage by which the wheel speed is increased or decreased. Its value changes each time a robot grips an object. The new value is sampled from a Rayleigh distribution with parameter σ (lines 8–10). The same distribution is used to randomly initialize μ_{noise}.

The value of σ characterizes the noise of the robots: the higher its value, the higher the expected value of μ_{noise}, and the larger the odometry error of a robot. The default value of σ, indicated as $\bar{\sigma}$, is set to 0.0134, the same value utilized by Pini et al. (2012). Other values of σ are used in the experiments to evaluate the impact of noise on the system.

5 Experiments and results

In this section, we describe the simulation-based experiments that we run to test our approach. When not specified otherwise, the experimental parameters are set as follows. The environment measures $W = 6.7$ m by $L = 4.5$ m (see Figure 6); the source is at a distance $D = 4$ m from the nest. Each experimental run lasts 20 simulated hours. At the beginning of each run, the robots are positioned inside the nest, with a random orientation. For each experimental condition we run 20 randomly seeded simulations.

The rest of this section is organized as follows. In Section 5.1, we present experiments that have the goal of evaluating the basic properties of the system, and of selecting the values of the parameters α and ε, used in the cost-based partitioning algorithm. In Section 5.2, we describe experiments in which we test the impact of the environment size. In Section 5.3, we present experiments that aim to evaluate the behavior of the system for different values of the source-to-nest distance D. Finally, in Section 5.4, we analyze a case in which the environmental conditions vary in time and discuss the exploitation versus exploration trade-off.

5.1 Basic properties of the system

The first goal of the experiments presented here is to determine the best way of partitioning the transportation task as a function of the swarm size and the odometry noise. We perform experiments testing the reference algorithms in swarms of different sizes (4, 6, 8, 10, 15, and 20 robots) and for different values of σ (0.0, $0.2\bar{\sigma}$, $0.5\bar{\sigma}$, $\bar{\sigma}$, $1.2\bar{\sigma}$). The second goal of the experiments is to select a value for the parameters of the cost-based partitioning algorithm, and to compare the algorithm to the reference algorithms. We select values for α in the set $\{0.001, 0.1, 0.25, 0.9, 1.0\}$, and ε in $\{0.0, 0.05, 0.15, 0.25, 0.5\}$.

The performance of the fixed algorithms in relation to the size of the swarm and the amount of noise provides insights about the basic properties of the system. We refer to the *performance of an algorithm* as the total number of objects transported to the nest when that algorithm is employed.

For reasons of space, we cannot include in this paper all of the results we obtained with the fixed algorithms. In the following we summarize the main findings; the whole set of results can be found in Pini et al. (2012). For a given swarm size, which fixed algorithm performs best depends on the odometry noise: the higher the noise, the smaller the partition length of the best performing fixed algorithm. Therefore, if the noise is high, the number of sub-tasks should be increased and their length decreased. Conversely, for low values of noise it is preferable to use few, long sub-tasks. These results confirm that task partitioning is beneficial to reduce the negative impact of the odometry noise.

The results also show that, given certain noise conditions, in large swarms it is preferable to partition the task into many, short sub-tasks. Physical interference is higher in large swarms than in small ones. Task partitioning distributes the robots along the path between source and nest, thus diminishing the interference among robots.

As mentioned, another goal of the experiments was to select the values of the parameters ε and α. We do not report the results of the experiments here, for reasons of space. The results are available in Pini et al. (2012). The best results of the cost-based partitioning algorithm are obtained for $\alpha = 0.25$ and $\varepsilon = 0$. We select these values and use them in all of the experiments presented in the paper, with the exception of those presented in Section 5.4. The case $\varepsilon = 0$ corresponds to a maximally greedy version of the ε-greedy: the algorithm always selects the value L_i associated with the minimal cost estimate $\hat{C}_i$. This is not surprising given the nature of the experiments presented here. The only variations occurring in the system are those introduced by the robots depositing objects. The remaining conditions, such as number of robots, odometry noise, size of the environment, etc. do not vary in time. Consequently, no exploration is needed and an exploiting version of ε-greedy performs well. Further considerations on the trade-off between exploration and exploitation are presented in Section 5.4.

Figure 7 reports the performance of the cost-based partitioning algorithm and of three reference algorithms. The top plot reports the results for $\sigma = 0.2\bar{\sigma}$ (low noise), the bottom plot for $\sigma = \bar{\sigma}$. For a given setting (i.e. odometry noise and swarm size), the fixed algorithm reported in the figure is the one performing best. The performance of the fixed algorithm is an upper bound for the performance of a swarm that uses task partitioning. In general, to reach such a performance the swarm needs prior knowledge. If the robots do not have such knowledge, they have to use other algorithms which, in general, perform worse.

The results reported in Figure 7 confirm prior hypotheses that we made about the properties of the studied system. The performance of all of the algorithms depends on the noise: as the noise increases, the performance of each algorithm decreases. In fact, the higher the noise, the larger the odometry error and the lower the success in finding the source. The never partition algorithm performs well when the noise is low and the swarm is small. In larger swarms, algorithms that use task partitioning are more advantageous. This is due to interference: the robots employing the never partition algorithm travel directly from source to nest and interfere with each other. The negative effect of interference increases for a decreasing noise: the robots get lost rarely and the traffic along the path between source and nest is high. The cost-based partitioning algorithm performs well in the majority of the tested conditions, indicating that the partitioning strategy utilized by the swarm suits the conditions in which the robots operate.

Figure 8 reports the values of the partition length selected in time by robots employing the cost-based partitioning algorithm, for $\sigma = \bar{\sigma}$. The plots report the percentage of times each value L_i was selected, computed across the 20 experimental runs. The sequence of plots in each row reports the data collected with a certain swarm size. Each column reports the data relative to a time period of 5 hours, corresponding to a quarter of the experimental run.

Each sequence of plots shows that there is an initial phase (first column of plots) during which the robots sample the values L_i. In time (plots from left to right), the robots converge towards defined values. The values selected the most in the final part depend on the swarm size: the larger the swarm, the lower the selected

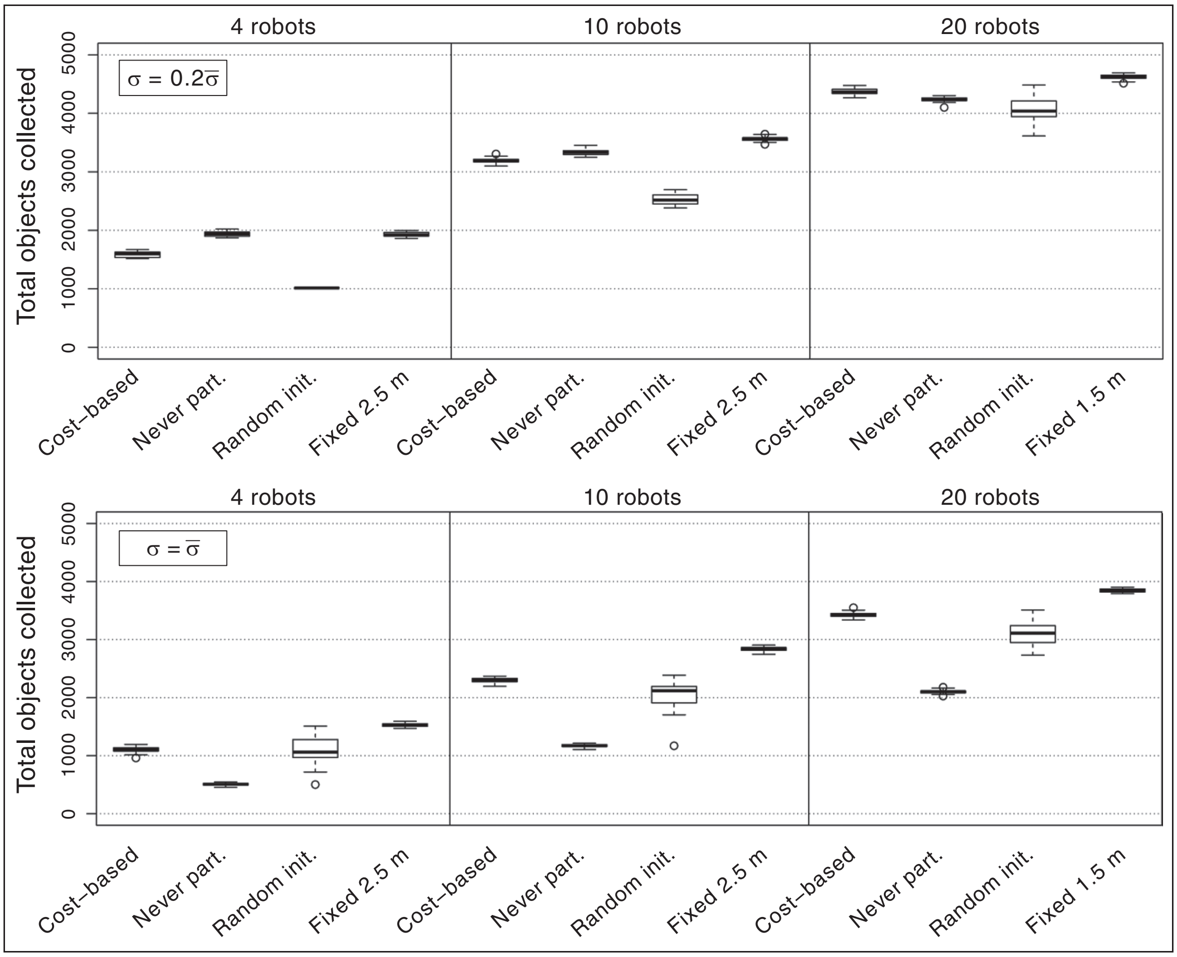

Figure 7. Objects collected by swarms of different size employing different partitioning algorithms. The top plot reports the data for $\sigma = 0.2\bar{\sigma}$, the bottom plot for $\sigma = \bar{\sigma}$. The fixed algorithms reported are the ones performing best in the corresponding setting.

partition length values (compare plots in the last column). Therefore, the cost-based partitioning algorithm responds to a growing swarm size by reducing the length of the sub-tasks. As mentioned, the analysis of the behavior of the fixed algorithms suggests that this is advantageous.

Figure 9 reports the values of the partition length selected in time by the robots for $\sigma = 0.5\bar{\sigma}$. The plots confirm the observed trends: in large swarms the robots select lower values of the partition length. Comparing with Figure 8, however, the overall choice is oriented towards higher values of the partition length. This is due to the lower odometry noise compared with the previous case.

The results presented in this section demonstrate that our approach based on cost can be used to obtain autonomous task partitioning. The swarm using the cost-based partitioning algorithm partitions the transportation task autonomously, on the basis of the environmental conditions (odometry noise and physical interference). Recall that the robots do not take direct measures of the frequency at which they get lost, the odometry error, or the amount of interference (i.e. they do not explicitly regulate task partitioning on the basis of these factors). Instead, they only estimate costs of performing sub-tasks. The partitioning strategy employed by the robots varies with the environmental conditions in a self-organized manner, due to the impact of such conditions on the costs.

5.2 Effect of the size of the environment

The goal of the experiments presented in this section is to test the system in environments of different size. In larger environments, random walk is costly: the robots take longer, on average, to locate objects. We test two environments: a *small environment* ($W = 4.5$ m by $L = 4.5$ m) and a *huge environment* ($W = 6.7$ m by $L = 10.0$ m).[4] In both cases, the source-to-nest distance D is 4.0 m and $\sigma = \bar{\sigma}$.

Figure 10 reports the performance of the cost-based partitioning algorithm and three reference algorithms for the different swarm sizes tested in the experiments (4, 10, and 20 robots). The top plot reports the results

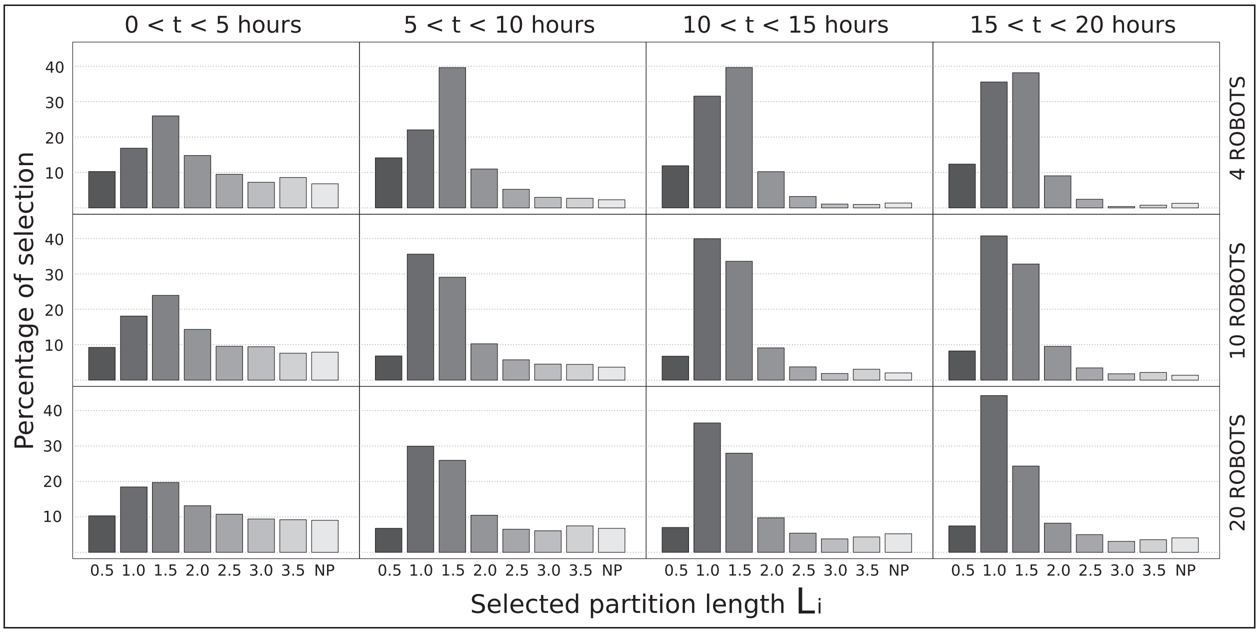

Figure 8. Partition length values L_i selected in time the robots using the cost-based partitioning algorithm, for $\sigma = \bar{\sigma}$. Each plot reports the percentage of selection of each value L_i in a time window of 5 hours.

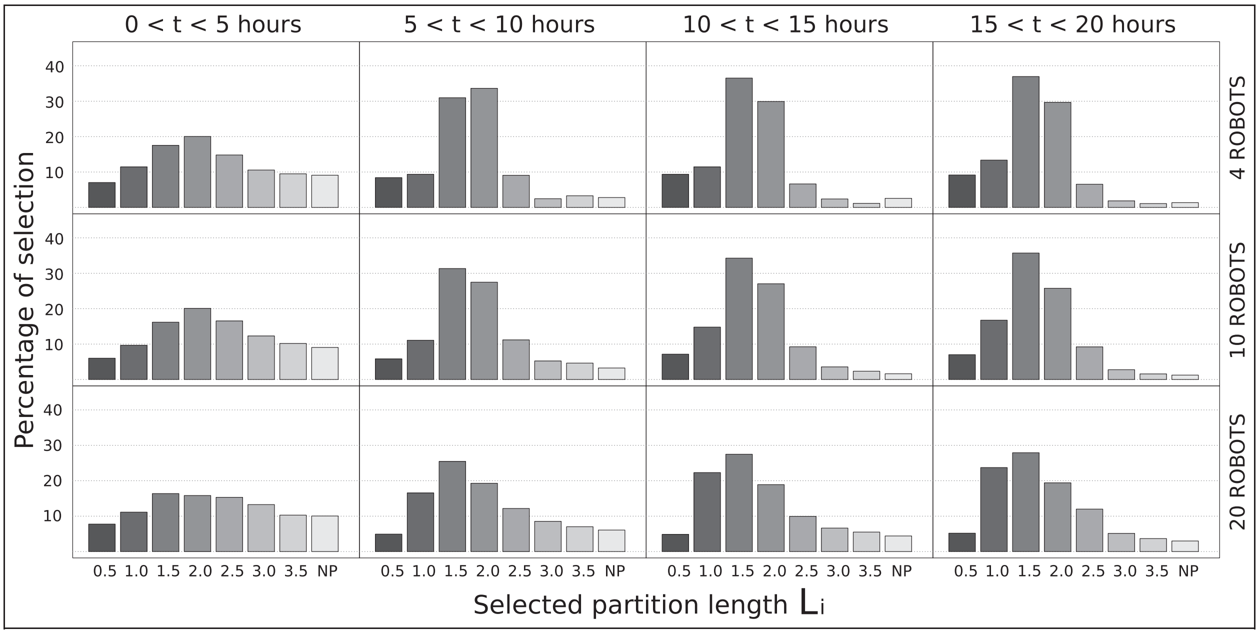

Figure 9. Partition length values L_i selected in time by the robots employing the cost-based partitioning algorithm, for $\sigma = 0.5\bar{\sigma}$. Each plot reports the percentage of selection of each value L_i in a time window of 5 hours.

obtained in the small environment, the bottom plot the results obtained in the huge environment. The results of the experiments indicate that the cost-based partitioning algorithm performs well across environments.

Figure 11 reports the partition length values selected by the robots in the final quarter of the experiment (last 5 hours). Plots in the same row refer to the same environment: small environment on top and huge environment at the bottom. Plots in the same column report data for the same swarm size. The plots show a trend in the partition length values selected. For a given swarm size, the selected values are influenced by the environment: the larger the environment, the lower the partition length values selected. This indicates that the cost-based partitioning algorithm identifies the high cost that derives from getting lost in larger environments and adapts the way the task is partitioned accordingly, by reducing the distance traveled by the robots and consequently the frequency at which they get lost.

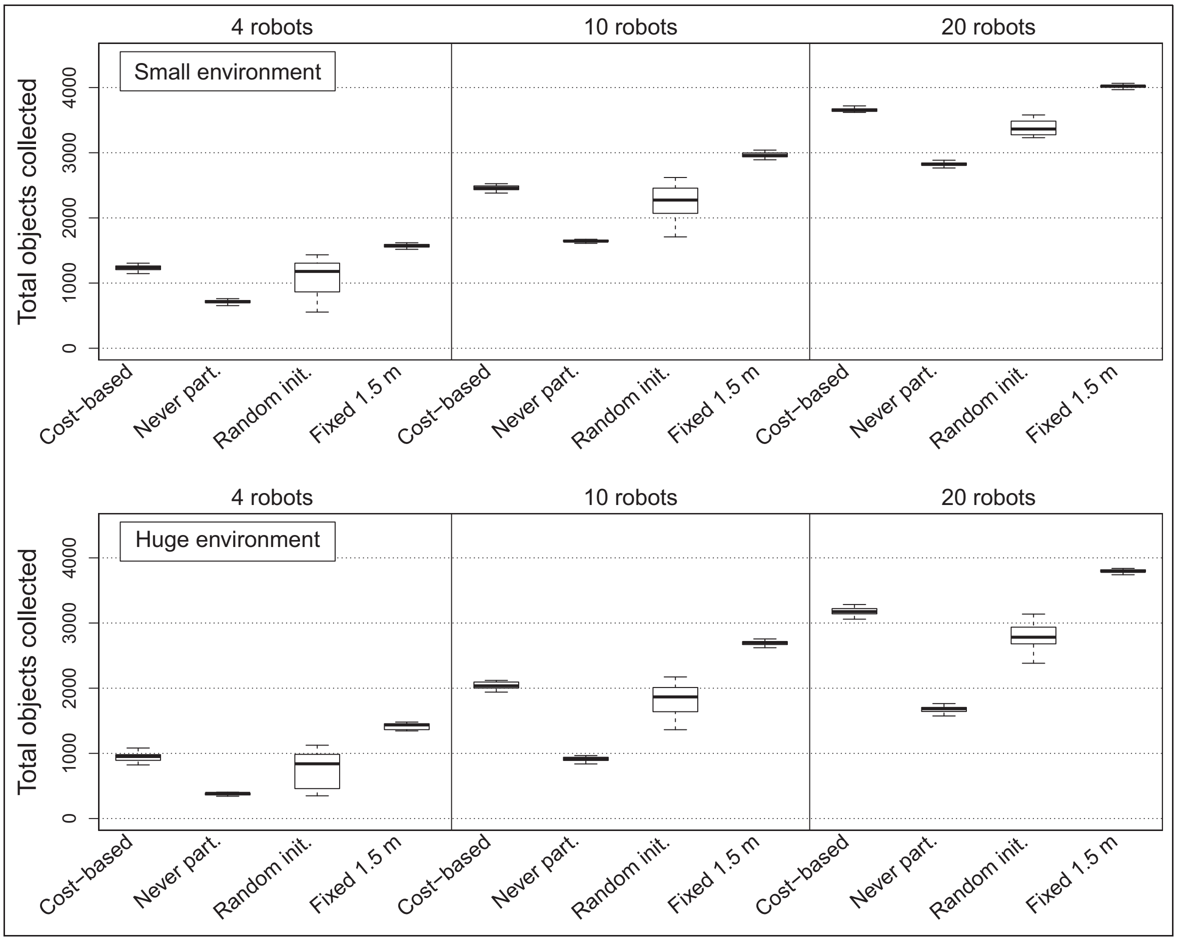

Figure 10. Objects collected by swarms of different size when different partitioning algorithms are employed. The top plot reports the data collected in the small environment, the bottom plot in the huge environment.

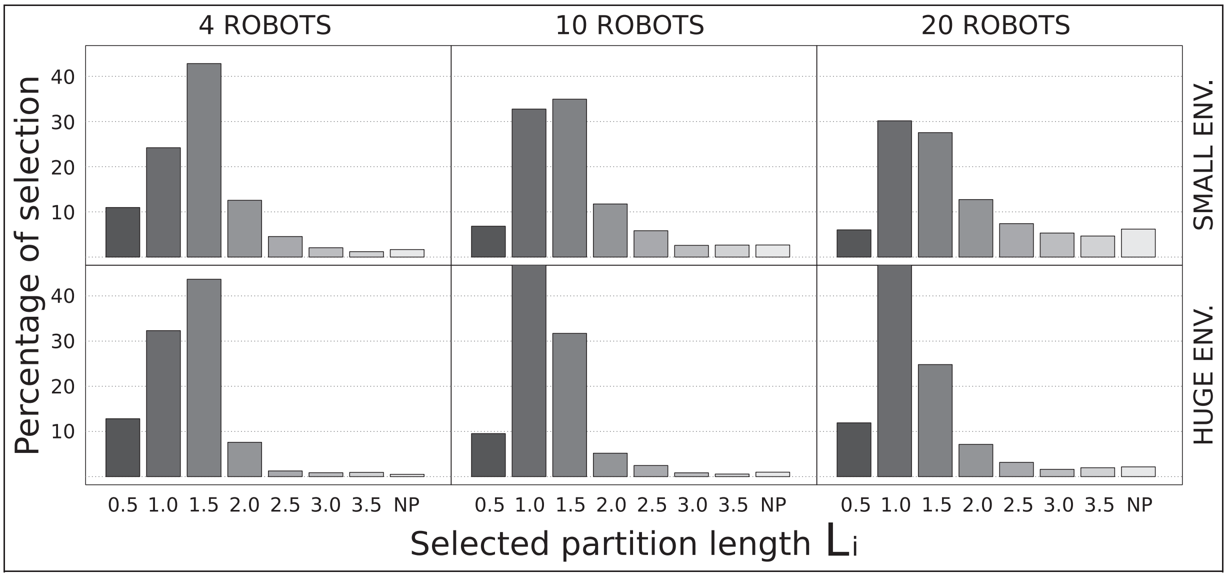

Figure 11. Partition length values L_i selected in time by robots employing the cost-based partitioning algorithm, for $\sigma = \bar{\sigma}$. Each plot reports the percentage of selection of each value L_i in the final quarter of the experiments.

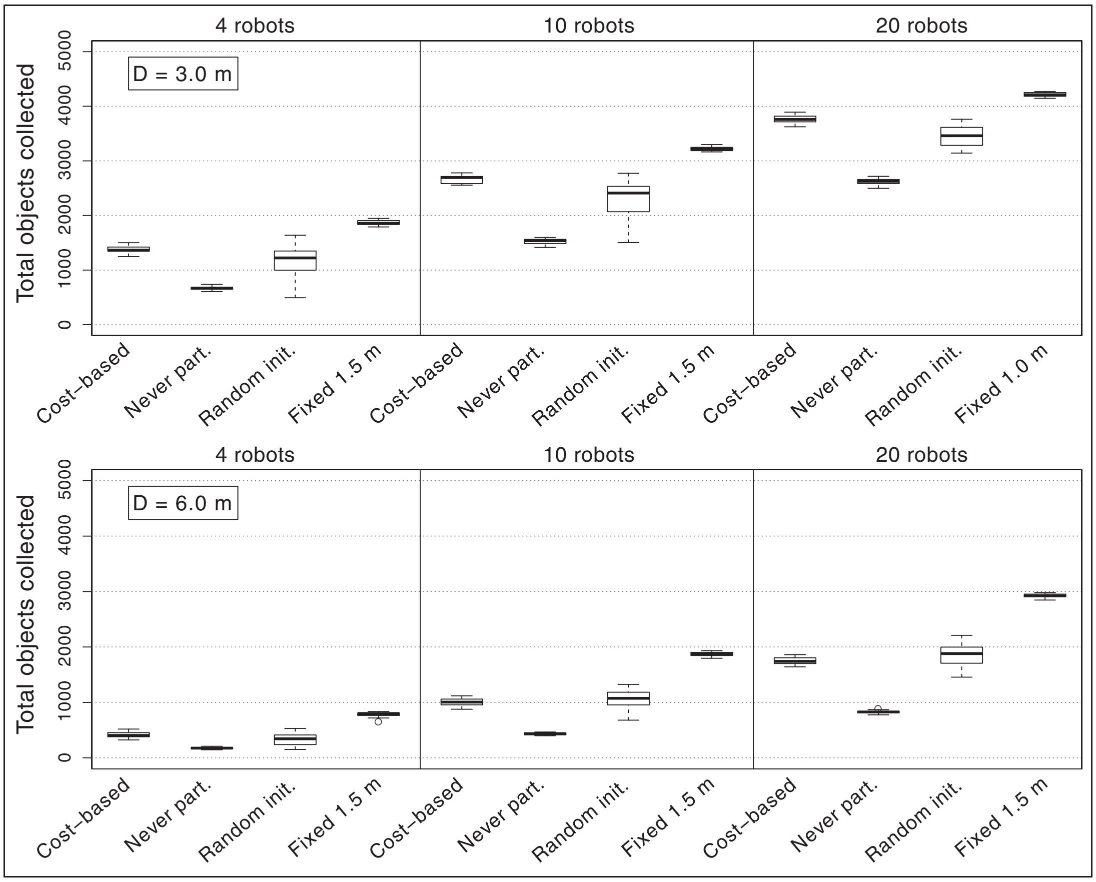

Figure 12. Objects collected by the swarm when different partitioning algorithms are employed. The top plot reports the data for $D = 3.0$ m, the bottom plot for $D = 6.0$ m.

5.3 *Effect of the source-to-nest distance*

The goal of the experiments presented in this section is to study the effect of the source-to-nest distance D (i.e. of the overall task length). We perform the experiments in the huge environment ($W = 6.7$ m by $L = 10.0$ m), with $\sigma = \bar{\sigma}$. The value of D is set to 3.0 m and 6.0 m.

Figure 12 reports the performance of the cost-based partitioning algorithm and three reference algorithms for the different swarm sizes tested in the experiments (4, 10, and 20 robots). The top plot displays the results for $D = 3.0$ m, the bottom plot for $D = 6.0$ m.

As expected, the performance of a given algorithm for $D = 3.0$ m is higher than the corresponding performance for $D = 6.0$ m: in the latter case, the objects must be transported for a longer distance and the throughput of the swarm is therefore inferior. In addition, in the case of the never partition algorithm, the longer distance traveled increases the probability that a robot gets lost. For the remaining algorithms, all of which utilize task partitioning, a longer distance corresponds to a higher number of sub-tasks and therefore higher overheads due to object transfer.

Figure 12 shows that, for $D = 3.0$ m, the robots using the cost-based partitioning algorithm perform well across the tested values of noise and swarm size. This is not the case for $D = 6.0$ m: the performance of the cost-based partitioning algorithm is close, or even inferior, to that of the random initialization algorithm. A separate analysis of the throughput (available in Pini et al. (2012)) shows that in the final part of the experiment, the throughput of the cost-based partitioning algorithm is higher than that of the random initialization algorithm. However, the throughput grows slowly in time and this results in a low performance. We speculate that the slow growth of the throughput in time is due to the fact that the actions performed by each robot influence the cost estimation process of the others. This influence between robots is more marked for larger values of D, which in general correspond to a higher number of object transfers (i.e. more robots are involved in the transportation of a given object).

5.4 *The tradeoff between exploration and exploitation*

In all of the experiments presented so far, the cost-based partitioning algorithm achieves its best performance with a maximally greedy version of the ε-greedy

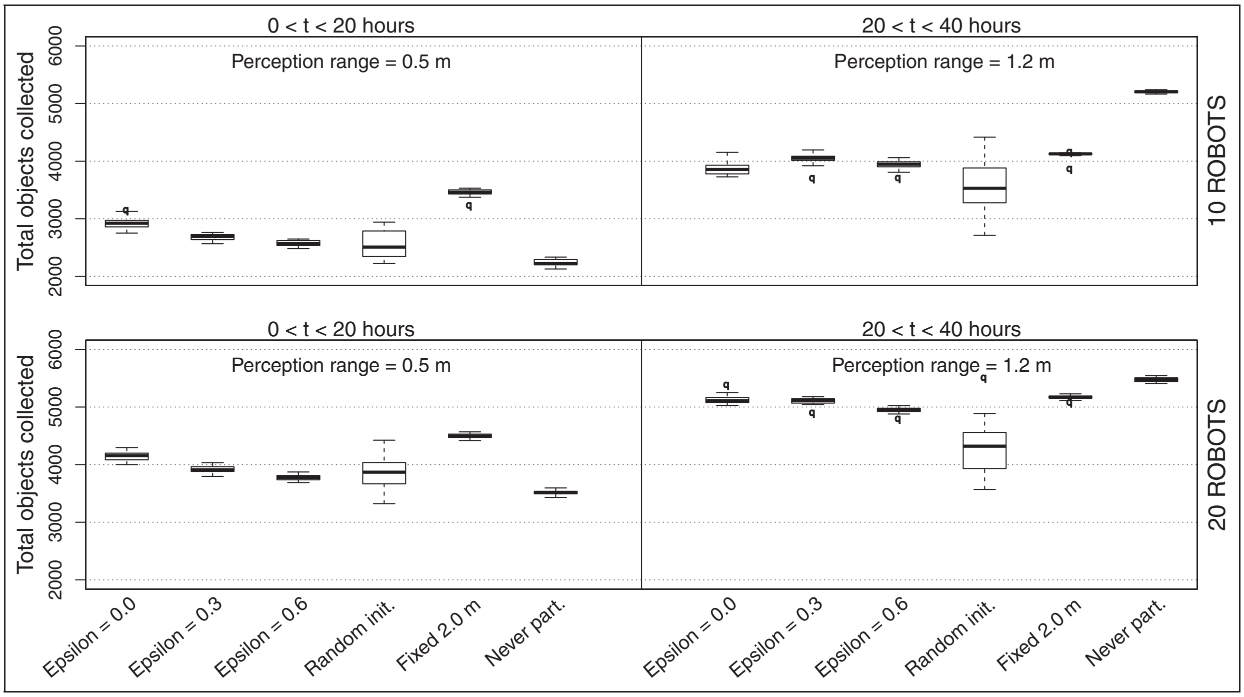

Figure 13. Objects collected by the swarm when different partitioning algorithms are employed. In the experiments, the perception range of the robots is increased from 0.5 m to 1.2 m at half the experimental time. In each plot, the first group of bars refers to the first half of experiment, the second group to the second half. The plot on top reports the results for 10 robots, the plot at the bottom for 20 robots.

algorithm (i.e. for $\varepsilon = 0$). This is due to the fact that we tested only static conditions. In certain cases an exploring version of the algorithm may be better: variations occurring in the environment may render advantageous a partition length value that is initially costly. If the robots sample the different partition length values in time (i.e. they use exploration), they are more likely to detect variations. Exploration may be beneficial only if one of the costs $\hat{C}_i$ decreases and renders the associated value L_i advantageous over the partition length value that previously had the lowest cost associated.

To test such a situation, we perform an experiment in which we introduce a variation in the environmental conditions. The variation consists in incrementing the distance at which the robots perceive objects. A broader perception diminishes the probability that a robot gets lost: objects can be perceived from farther away and therefore odometry errors have a smaller impact.

We perform the experiments with swarms of 10 and 20 robots. When the experimental run reaches half-time, we vary the perception range of the robots from the value of 0.5 m to 1.2 m. The duration of a run is extended to 40 simulated hours, to allow the cost-based partitioning algorithm to identify good partition length values before the perception change is introduced. We test different versions of the algorithm, each with a different value of the parameter ε. The tested values for ε are $\{0, 0.3, 0.6\}$, corresponding to increasing degrees of exploration. We report here the results for $\sigma = 0.5\bar{\sigma}$ since this value highlights the most the effects of exploration, compared with other values we tested in our experiments.

Figure 13 reports the performance of the cost-based partitioning algorithm and of three reference algorithms for swarms of 10 (top) and 20 robots (bottom). In each plot, the group of boxes on the left-hand side reports the data collected in the first half of the experiment, before the perception range is modified. The group on the right-hand side reports the data collected after the perception range has been modified.

The results show that, in the first half of experiment, the best performing algorithm is the fixed algorithm with a partition length of 2.0 m. Among the different versions of the cost-based partitioning algorithm, the best performing one is non-exploring ($\varepsilon = 0$).

In the second half of experiment, the situation is different: the broader perception renders the never partition algorithm preferable. In swarms of 20 robots, the relative performance gain of the never partition algorithm over the fixed algorithm is limited. This is again an effect of interference, which is the dominant factor in large swarms.

The results obtained by the different versions of the cost-based partitioning algorithm indicate that exploration is beneficial. In a swarm of 10 robots, the cost-based partitioning algorithm with $\varepsilon = 0.3$ performs better that the non-exploring version. In a swarm of 20 robots, the performance reached by the two is

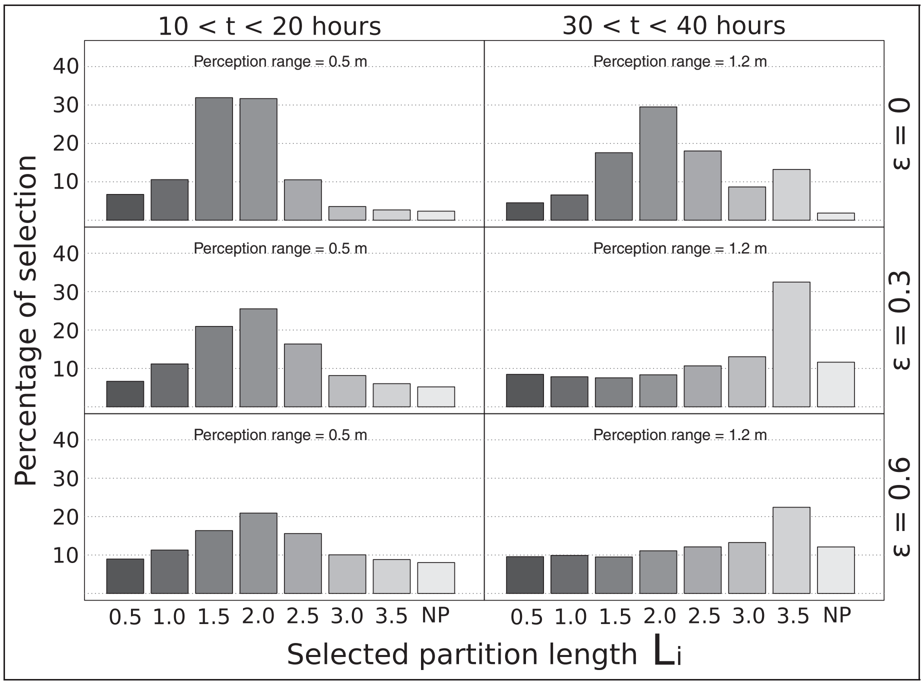

Figure 14. Partition length values L_i selected in a swarm of 10 robots for different values of ε (increasing exploration from top to bottom). In the experiments, the perception range of the robots is increased from 0.5 m to 1.2 m at half the experimental time. The first column of plots reports the percentage of selection of each value L_i in the second quarter of experiment. The second column analogous data collected in the final quarter.

comparable. Exploration is beneficial up to a certain point: for $\varepsilon = 0.6$, the performance is always inferior than for $\varepsilon = 0.3$. Therefore, exploration and exploitation should be balanced carefully.

Figure 14 reports the partition length values selected in time by a swarm of 10 robots. Each row of plots corresponds to a value of ε. The plots in the left-hand side column report the frequency at which each partition length value was selected in the second quarter of the experiment. The plots in the right-hand side column report analogous data, collected in the last quarter. The plots highlight different behaviors of the cost-based partitioning algorithm, depending on the value of ε.

The plots in the first column show that all of the versions the cost-based partitioning algorithm mostly select values around 2.0 m. The effect of increasing exploration can be observed by comparing the plots from top to bottom: the peaks flatten and the selected values distribute more evenly, affecting performance negatively, as shown in Figure 13.

The effect of exploration is also visible in the right-hand side column of plots in Figure 14. For $\varepsilon = 0.3$ and $\varepsilon = 0.6$, the peak moves from 2.0 m to 3.5 m. This indicates that the cost-based partitioning algorithm detects the change occurring in the environment and reacts by selecting a partitioning strategy more suited for the new conditions. The non-exploring version of the algorithm, on the other hand, continues to select values around 2.0 m. The value 3.5 m is selected more often compared to the second quarter of experiment (Figure 14, top left), indicating that in the studied system the robots sample different partition length values independently of the value of ε, which regulates exploration. We believe that this is due to stochasticity in the system (e.g. robots getting lost due to odometry errors), which introduces variations in the cost estimates. Analogous variations may not be present in other systems and therefore explicit exploration is an option to consider, in general.

6 Conclusion and future work

In this paper, we have proposed an approach to achieve autonomous task partitioning in robot foraging. What characterizes our approach is the use of a cost function, mapping the amount of work contributed by a robot to the resulting cost of performing the overall task. The concept of cost is generic, and it can be related to the performance of the robots independently of the context in which they operate and the tasks they perform. Each robot models the cost function and uses this model to decide its contribution in terms of amount of work, therefore defining the sub-task it performs in an autonomous way.

There are two aspects that differentiate our from the other approaches proposed in the literature. First, our approach allows for a separation between the process implementing task partitioning and the actions performed by the robots to execute tasks. Second, it does not require *a priori* assumptions to be made about the factors that determine the best way of partitioning a tasks. These two aspects render our approach more general with respect to what proposed so far. We built our approach focusing on foraging, however we are confident that the same approach can be applied directly to other contexts, by adapting the definition of cost function to the specific case. The underlying decision process that implements task partitioning does not need to be reimplemented.

In the scenario presented, the goal of the robots is to collect objects from a location in the environment, the source, and transport them to a nest. The robots use odometry to estimate their location with respect to the source. We show empirically that the benefits of task partitioning depend on the odometry noise and the swarm size. We compare an algorithm based on our approach to a set of reference algorithms in different experimental setups. We show that, by using our approach, the swarm performs well across the tested conditions. The partitioning strategy employed by the swarm depends on different factors: the accuracy of the odometry, the number of robots, and the size of the environment. The robots do not take measures to directly estimate any of these factors, yet the strategy employed to perform foraging is appropriate in relation to such factors.

We study the exploration/exploitation trade-off in a setup in which the environmental conditions change over time. The results show that, in the studied system, stochastic events lead the robots to sample over time different partitioning options. However, we also remark that built-in exploration may be beneficial and therefore it is an option to consider, in general.

Short-term research aims to verify the applicability of the approach to tasks different from transportation, for example patrolling or construction. Another research direction is integrating explicit communication in the studied system. We remarked that, in certain situations, the swarm is slow in converging to a certain partitioning strategy. Communication could improve this aspect of the system. The robots could directly exchange information and integrate their cost estimates with the data received from other robots. This is likely to reduce the negative impact of cost estimation errors made by the robots. In addition, communication would allow the robots to take a collective decision on the way the overall task is partitioned, with the result of an increased coordination between robots and consequently a more responsive collective behavior.

Our long-term research goal is the development of a general task partitioning framework, that can be applied to different tasks and contexts. The capability of autonomously partition tasks into sub-tasks would increase the flexibility of swarm robotics systems: the robots could adapt the way tasks are performed to specific environmental conditions and goals. The approach proposed in this paper makes the first significant steps in this direction.

Notes

1. Each control step lasts 0.1 seconds.
2. The neighborhood search time and radius are the same as in Pini et al. (2012).
3. See http://iridia.ulb.ac.be/argos
4. The name and the size of the environments are the same as in Pini et al. (2012).

Funding

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 number 246939). Giovanni Pini acknowledges support from Université Libre de Bruxelles through the "Fonds David & Alice Van Buuren". Arne Brutschy, Marco Dorigo, and Mauro Birattari acknowledge support from the Belgian F.R.S.–FNRS.

References

Anderson, C., & Ratnieks, F. L. W. (2000). Task partitioning in insect societies: novel situations. *Insectes sociaux*, *47*, 198–199.

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). Piscataway, NJ: IEEE Press.

Bovet, D. P., & Cesati, M. (2005). *Understanding the linux kernel*. O'Reilly Media.

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

Brutschy, A., Pini, G., & Decugnière, A. (2012). *Grippable objects for the foot-bot* (Tech. Rep. No. TR/IRIDIA/2012-001). Brussels, Belgium: IRIDIA, Université Libre de Bruxelles.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms*, second edition. Cambridge, MA: MIT Press.

Dorigo, M., & Birattari, M. (2007). Swarm intelligence. *Scholarpedia*, 2 (9), 1462.

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

Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A. L., Decugnière, A., Di Caro, G. Ducatelle, F., Ferrante, E., Förster, A., Gonzales, J. M., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., Montes de Oca, M., O'Grady, R., Pinciroli, C., Pini, G., Rétornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stützle, T., Trianni, V., Tuci, E., Turgut, A. E., & Vaussard, F. (2013). Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*. (In press)

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.

Ennals, R., Sharp, R., & Mycroft, A. (2005). Task partitioning for multi-core network processors. In *Compiler construction* (Vol. 3443, p. 76–90). Springer, Berlin/Heidelberg, Germany.

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.

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.

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

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.

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

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

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.

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

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.

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.

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

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 E. Andre, S. Sen, C. Frasson, & P. M. Jörg (Eds.), *Agents '01: Proceedings of the fifth international conference on autonomous agents* (pp. 29–30). New York: ACM Press.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., 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.

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* (Vol. 85, pp. 17–228). Berlin/Heidelberg, Germany: Springer.

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, G. A. Di Caro, R. Doursat, A. P. Engelbrecht, D. Floreano, L. M. Gambardella, R. Groß, E. Şahin, H. Sayama, & T. Stützle (Eds.), *Swarm intelligence, 8th international conference, ants 2012* (Vol. 6234, pp. 287–298). Berlin/Heidelberg, Germany: Springer.

Pini, G., Brutschy, A., Frison, M., Roli, A., Dorigo, M., & Birattari, M. (2011). Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, *5*(3–4), 283–304.

Pini, G., Brutschy, A., Pinciroli, C., Dorigo, M., & Birattari, M. (2012). Autonomous task partitioning in robot foraging: An approach based on cost estimation – *Online supplementary material*. (http://iridia.ulb.ac.be/supp/IridiaSupp2012-0014/)

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 subtasks* (Tech. Rep. No. TR/IRIDIA/2012-010). Brussels, Belgium: IRIDIA, Université Libre de Bruxelles.

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. doi: 10.1007/s11721-013-0078-7

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

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

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

Shell, D. J., & Matarić, M. J. (2006). On foraging strategies for large-scale multi-robot systems. In 2006 *IEEE/RSJ* international conference on intelligent robots and systems (*IROS'06*) (pp. 2717–2723). Pitscataway, NJ: IEEE Press.

Sutton, R., & Barto, A. (1998). *Reinforcement learning, an introduction*. Cambridge, MA: MIT Press.

Tangamchit, P., Dolan, J. M., & Khosla, P. (2002). The necessity of average rewards in cooperative multirobot learning. In *IEEE international conference on robotics and automation (ICRA'02)* (Vol. 2, pp. 1296–1301). Piscataway, NJ: IEEE Press.

Torbjørn, D., S., Matarić, M. J., & Sukhatme, G. S. (2009). Multi-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, *6–7*(57), 674–687.

Winfield, A. F. T. (2009). Foraging robots. In R. A. Meyers (Ed.), *Encyclopedia of complexity and system science* (pp. 3682–3700). Berlin/Heidelberg, Germany: Springer.

Author biographies

Giovanni Pini is a PhD candidate in applied sciences at IRIDIA, Université Libre de Bruxelles, Belgium. He holds a Masters in computer science engineering, received from Politecnico di Milano, Italy, in 2007. His research work focuses on swarm intelligence and swarm robotics, and self-organized task partitioning and task allocation

Arne Brutschy received a Diploma in computer science from the University of Leipzig, Germany. He is currently a PhD candidate in applied sciences at IRIDIA, Université Libre de Bruxelles, Belgium. He received a scholarship from the fund for scientific research F.R.S.-FNRS of Belgium's French community for the duration of his PhD studies. His research interests are in artificial intelligence and swarm robotics, with a focus on self-organized task allocation and task partitioning.

Carlo Pinciroli is a PhD student at IRIDIA, CoDE, Université Libre de Bruxelles in Belgium. Before joining IRIDIA, in 2005 he obtained a Masters in computer engineering at Politecnico di Milano, Milan, Italy and a second Masters in computer science at University of Illinois at Chicago, IL, USA. In 2007 he obtained a Diplôme d'études approfondies from the faculty of engineering of the Université Libre de Bruxelles, Belgium. The focus of his research is computer simulation and swarm robotics.

Marco Dorigo received his PhD in electronic engineering in 1992 from Politecnico di Milano, Italy, and the title of Agrégé de l'Enseignement Supérieur, from ULB, in 1995. Since 1996, he has been a tenured researcher of the fund for scientific research F.R.S.-FNRS of Belgium's French community, and a research director of IRIDIA, ULB. He is the inventor of the ant colony optimization metaheuristic. His current research interests include swarm intelligence, swarm robotics, and metaheuristics for discrete optimization. He is the editor-in-chief of *Swarm Intelligence*. He is a fellow of the IEEE and of ECCAI. He was awarded the Italian prize for artificial intelligence in 1996, the Marie Curie excellence award in 2003, the Dr. A. De Leeuw-Damry-Bourlart award in applied sciences in 2005, the Cajastur international prize for soft computing in 2007, and an ERC advanced grant in 2010.

Mauro Birattari received his Masters in electronic engineering from Politecnico di Milano, Italy, in 1997; and his PhD in information technologies from Université Libre de Bruxelles, Belgium, in 2004. He is currently with IRIDIA, Université Libre de Bruxelles, as a research associate of the fund for scientific research F.R.S.-FNRS of Belgium's French community. He has co-authored about 100 peer-reviewed scientific publications in the field of computational intelligence. He is an associate editor for the journal *Swarm Intelligence*, and an area editor for the journal *Computers & Industrial Engineering*.