AUTONOMOUS TASK PARTITIONING IN SWARMS OF ROBOTS: AN APPROACH BASED ON COST ESTIMATION

G. Pini, A. Brutschy, C. Pinciroli, M. Dorigo, and M. Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2012-018

November 2012

IRIDIA – Technical Report Series
ISSN 1781-3794

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

Technical report number TR/IRIDIA/2012-018

International Journal of Intelligent Computing and Cybernetics
© World Scientific Publishing Company

AUTONOMOUS TASK PARTITIONING IN SWARMS OF ROBOTS: AN APPROACH BASED ON COST ESTIMATION

GIOVANNI PINI*
IRIDIA, Université Libre de Bruxelles
50, Av. F. Roosevelt, CP 194/6 Brussels, 1050,Belgium
gpini@ulb.ac.be
http://iridia.ulb.ac.be/∼gpini

ARNE BRUTSCHY
IRIDIA, Université Libre de Bruxelles
50, Av. F. Roosevelt, CP 194/6 Brussels, 1050,Belgium
arne.brutschy@ulb.ac.be

CARLO PINCIROLI
IRIDIA, Université Libre de Bruxelles
50, Av. F. Roosevelt, CP 194/6 Brussels, 1050,Belgium
carlo.pinciroli@ulb.ac.be

MARCO DORIGO
IRIDIA, Université Libre de Bruxelles
50, Av. F. Roosevelt, CP 194/6 Brussels, 1050,Belgium
mdorigo@ulb.ac.be

MAURO BIRATTARI
IRIDIA, Université Libre de Bruxelles
50, Av. F. Roosevelt, CP 194/6 Brussels, 1050,Belgium
mbiro@ulb.ac.be

Received (Day Month Year)
Revised (Day Month Year)

In this work, we study task partitioning in swarms of robots. Task partitioning is the process by which a task is decomposed into sub-tasks. We consider here the case in which a given task requires that a total amount of work is performed, and each sub-task incrementally contributes a fraction of that amount of work.

We propose an approach that enables a swarm of robots to autonomously partition a task. Each robot performs a sub-task; the entity of the amount of work that the sub-task contributes to the execution of the overall task is decided autonomously by the robot. The entity of the contribution has an impact on the cost of performing the overall task:

*Université Libre de Bruxelles, 50, Av. F. Roosevelt, CP 194/6 Brussels, 1050,Belgium

in some cases, partitioning the task in many small sub-tasks may be more advantageous than partitioning it in few larger sub-tasks, or vice versa.

Each robot decides the entity of the amount of work that it contributes with its sub-task on the basis of a model that it builds. This model provides an estimate of the cost of performing the overall task as a function of the amount of work contributed by a sub-task, which is an indicator of the number of sub-tasks into which the original task is partitioned.

We test our approach on a foraging scenario: the robots collect objects from a source location, and transport them to a target location. The robots use odometry to be able to move back to the source location. Odometry is appealing due to its simplicity and the fact that it does not require external infrastructures. However, it suffers from the accumulation of errors in time.

Task partitioning can be used to reduce the effects of the odometry error. A robot transporting an object travels a limited distance before depositing the object on the ground. There, another robot picks it up and continues transportation. In such context, autonomous task partitioning consists in deciding the distance traveled by a robot when carrying an object, that is, the amount of work that the robot contributes to the transportation task.

We validate our approach using simulation-based experiments. We study how the task is partitioned by the swarm under a number of experimental conditions characterized by different levels of odometry accuracy, size of the environment and the swarm, and distance between source and nest. Our approach leads to partitioning solutions that are appropriate for each experimental condition studied and that result in good performance.

Keywords: Task partitioning; swarm robotics; swarm intelligence; self-organizing systems; foraging.

Research paper

1. Introduction

Task partitioning is the process by which tasks are decomposed into sub-tasks. The term was first introduced by [Jeanne, 1986] to indicate situations in which insects divide tasks into sequences of sub-tasks. Task partitioning is applied to organize the execution of tasks in many contexts. In artificial systems, task partitioning is applied, among others, in the fields of computer science and robotics (see Section 2 for details). In nature, many examples of task partitioning have been observed in the organization of work of social insects (see [Ratnieks and Anderson, 1999]). Social insects utilize task partitioning in activities that involve the transportation of material; examples have been reported in foraging [Seeley, 1995], hunting [Schatz et al., 1996], nest excavation [Anderson and Ratnieks, 2000], and garbage disposal [Hart et al., 2001].

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

Our research focuses on the study of task partitioning in the context of swarm robotics, a branch of robotics that studies the implementation and control of large groups of autonomous robots. Swarm robotics focuses on properties such as decentralization, distributed control, limited perception, and local communication. The goal is to implement systems that are robust, tolerant to faults, scalable, and flexible. Swarms of robots are frequently built taking inspiration from insect colonies, in which such characteristics can often be observed. Since task partitioning is advantageous for social insects, it is interesting to study its application to robot swarms. Swarms in which the robots are capable of autonomously defining sub-tasks of a given task would be extremely flexible. In fact, the way tasks are partitioned and therefore performed could be adapted to specific environments and to the goals to be reached.

In this work, we study autonomous task partitioning in foraging. Foraging is an important benchmark in swarm robotics, since it is an abstraction of practical problems such as search and rescue, mine clearance, and cleaning [Winfield, 2009]. We propose an approach that can be utilized by a swarm of robots to autonomously partition the task of transporting an object. Our approach is based on the idea of associating *costs* to the execution of tasks that must be performed. In general, the nature of the costs depends on the specific tasks and goals to be attained. In the foraging scenario we tackle in this work, the goal is to maximize the number of foraged objects and costs are expressed as time.

Task partitioning affects the costs associated to the execution of a task. On the one hand, as mentioned, task partitioning can reduce certain costs (e.g., physical interference in foraging). On the other hand, task partitioning also introduces overhead costs, mainly due to coordination efforts between individuals working on different sub-tasks. Therefore, the best way of partitioning a task into sub-tasks depends on the task at hand and on the goals to be attained. Moreover, since the properties of the environment and the goals to be attained can vary, the best way of partitioning a task can also change in time.

We propose an approach whereby the robots decide how to partition the object transportation task on the basis of cost estimates. The use of cost estimates characterizes our approaches and differentiates it from other task partitioning methods proposed in the context of foraging. In all the existing work on the topic, the mechanisms that implement task partitioning are designed for the tasks at hand and are highly dependent on the specific context. Our approach, on the other hand, decouples the task partitioning process from the tasks being performed by the robots, through the concept of cost. Decoupling the task partitioning process from the task to be executed is a necessary step towards a general method for autonomous task partitioning.

In the foraging scenario studied in this work, we consider the specific case in which the objects to be collected are clustered in a unique location in the environment, referred to as *source*. The robots must initially explore the environment and locate the source. Upon finding the source, a robot utilizes odometry to maintain an estimate of its position relatively to the source. This estimate is used by the robot to return to the source. Odometry is suited to robotics because of its simplicity and low cost, but it suffers from estimation errors that grow with the distance traveled.

In [Pini et al., 2012c], we use the same scenario to show that task partitioning can be utilized to enhance the capability of the robots to return to the source. We show that task partitioning can be used to limit the distance traveled by each robot, therefore reducing the negative impact of odometry errors. An object is delivered to the nest through a sequence of steps, performed by different robots: each robot transports the object for a limited distance and the object is directly handed over from robot to robot till it eventually reaches the nest.

In this work, we test an analogous solution based on task partitioning, with the difference that the objects are not handed over directly, but deposited on the ground. We use simulation-based experiments to show that the optimal way of partitioning object transportation depends on the accuracy of the odometry and on the characteristics of the environment. We present a swarm that autonomously partitions the foraging task described. The swarm performs well across all the experimental conditions we tested.

The contents of the paper are organized as follows. In Section 2, we present related work on the topic of task partitioning, focusing on artificial systems, swarm of robots in particular. In Section 3, we describe in detail the foraging scenario, the behavior of the robots, and the application of our approach for autonomous task partitioning to the studied scenario. In Section 4, we present the simulation tools we utilized to carry out the experiments. In the same section we describe the simulated robots and the environment in which foraging is performed. In Section 5, we describe the experiments and comment on the results. In Section 6, we summarize what presented in the paper and discuss directions for future research.

2. Related Work

Most of the research on the topic of task partitioning has been carried out in the field of enthomology. A review of the state of the art of the field is beyond the scope of this article; comprehensive reviews can be found in the works of [Ratnieks and Anderson, 1999] and [Hart et al., 2002]. In this section, we review works devoted to task partitioning in artificial systems and, in particular, in swarm robotics.

Several examples of task partitioning applied to artificial domains can be found in computer science. A first example are recursive algorithms, often based on the *divide and conquer* approach: the problem to be solved is decomposed into smaller sub-problems that are solved recursively and their solutions are combined [Cormen et al., 2001]. In modern operating systems, each process is executed on a CPU only for a limited amount of time [Bovet and Cesati, 2005]. The result is that the execution of each process consists of a number of discrete steps, progressively performed at different times. To the end user, the technique gives the illusion of parallelism.

In architectures with multiple processors, task partitioning is the problem of dividing the execution of a program into tasks that can be performed in parallel by the processors [Ennals et al., 2005].

In swarm robotics, task partitioning has been mainly utilized for reducing physical interference between robots. Physical interference is a common problem in multi-robot systems: the robots share the same physical space and interfere with the movements of each other. As the density of robots in the environment increases, the robots spend resources dealing with physical interference rather than performing useful tasks and the performance of the swarm is affected negatively [Lerman and Galstyan, 2002].

The use of task partitioning to deal with interference was originally proposed by [Drogoul and Ferber, 1992]. In their work, the authors propose a solution based on task partitioning to deal with “traffic jams” forming in certain locations of the environment. In their experiments, the robots hand over objects one to another. The result is that chains of robots form in the environment and objects are transferred along these chains till they reach the nest. Notice, however, that the role of task partitioning in the work of [Drogoul and Ferber, 1992] is only minor and the focus is on how individual behavior affects group behavior.

In the work of [Fontan and Matarić, 1996], the environment is divided into territories, each exclusively assigned to one robot. The task of the robots is to forage for pucks and transport them to a target location. Each robot transports the pucks found in its territory towards the target location. Each robot delivers objects to the neighboring territory, so that the robot assigned to that territory can continue the transportation. The authors show that their solution increases the performance of the system, due to a reduction of physical interference. The same robotic system and experimental setup are utilized in the work of [Goldberg and Matarić, 2002], that aim at studying the design of robust and easily modifiable behavior-based controllers.

[Pini et al., 2009] present a work similar to the one of [Fontan and Matarić, 1996]. The work focuses on the usage of task partitioning to reduce competition for accessing a shared resource. In this case, the territories of the robots are not exclusive and are dynamically assigned in a self-organized manner.

[Shell and Matarić, 2006] also restrict the area in which each robot operates to reduce physical interference. Differently from the work of [Fontan and Matarić, 1996], the working area of a robot is not associated to a given location in the environment, but it is defined relatively to the current position of the robot. Each robot performs foraging in a circular area of a certain radius. The robots estimate the position of the working areas using odometry. The robots collect objects found in their area and transport them towards a target location. If a robot leaves its area while transporting an object, it drops the object and returns towards the center of its area. Each object is progressively delivered to the nest crossing several areas. The authors show the relation between the number of robots in the environment and the radius of the working area of each robot. They demonstrate that reducing the radius impacts positively the performance for an increasing swarm size.

[Lein and Vaughan, 2008] extend the work of [Shell and Matarić, 2006] with an algorithm that dynamically regulates the radius of the working area on the basis of the interference perceived by a robot. In a follow-up work, the authors point out that the performance of the system depends on the distribution of objects in the environment. Therefore, the authors further extend their algorithm to allow the relocation of the working areas towards zones of the environment where the density of objects is high [Lein and Vaughan, 2009].

In the work of [Østergaard et al., 2001], a group of robots forages in a maze-like environment. The authors compare an algorithm based on task partitioning to a non-partitioning algorithm, in environments that differ in the width of the corridors. The authors conclude that the algorithm based on task partitioning performs better in spatially constrained environments.

[Pini et al., 2011] propose a distributed algorithm for robot swarms to decide whether to employ task partitioning to perform a given task. Each robot takes its decision independently from the others, on the basis of cost estimates. The authors study the algorithm in situations in which task partitioning is advantageous and others in which it is not. They show that the algorithm performs well in the tested conditions and that it renders the system flexible with respect to variations occurring in the environment. In a follow-up work, [Pini et al., 2012a] show that the problem of deciding whether to employ task partitioning can be approached with algorithms proposed in the literature to tackle multi-armed bandit problems.

[Pini et al., 2012c] study the same setup proposed in this article. The robots forage for objects clustered in a certain location of the environment. The robots use odometry to estimate their position relatively to this location. The authors propose a solution based on task partitioning to tackle the problem: each object is transported to the nest through a sequence of steps, performed by different robots. Each robot contributes to transportation by carrying objects for a limited distance. The objects are handed over directly from robot to robot. The authors show that the approach based on task partitioning improves the foraging performance. Additionally, the authors use the system as a test-bed for studying the costs of task partitioning in case of direct object transfer.

[Parker and Zhang, 2010] study the case in which a group of robots performs sequences of mutually exclusive tasks: a task can begin only when the preceding one in the sequence is completed and no robot is working on it anymore. Analogous situations may occur when a task is partitioned into a sequence of sub-tasks. The focus of the study is on the decision making process that allows the robots to collectively estimate whether a sub-task is complete and the group can start working on the following.

Notice that most of the work presented in this section has been carried out using simulation tools only. The only exceptions are the works of [Fontan and Matarić, 1996], [Goldberg and Matarić, 2002], and [Pini et al., 2012c].

3. Description of the Approach

In this section, we present the approach we propose for obtaining autonomous task partitioning in a swarm of robots performing foraging. The foraging scenario is described in Section 3.1; our approach and its application to such a scenario are discussed in Section 3.2. In Section 3.3, we present task partitioning algorithms that are tested in the experiments, including an algorithm based on the proposed approach.

3.1. *The Foraging Scenario*

In the foraging scenario studied in this work, the goal of the robots is to collect objects from an environment and transport them to a unique location referred to as *nest*. We focus on the specific 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. An instance of such a task is completed when an object is successfully transported to the nest. Foraging consists in the parallel repetition of the transportation task performed by the robots of the swarm.

Figure 1 illustrates a finite state machine that describes how foraging is performed by a robot. Initially, the robot does not have information about the location of the source and therefore it must explore the environment to find it. Upon finding the source and collecting an object, the robot navigates towards the nest. The robot can reach the nest by heading towards a landmark (three lights) that marks the location of the nest. While navigating towards the nest, the robot keeps an estimate of its position, relatively to the location in which the source was found. Upon reaching the nest, the robot releases the object and uses the position estimate to return to the source.

Estimating the position of a robot relatively to a target (the source in our setup) is a problem known as localization. Several techniques have been proposed in the robotics literature to tackle this problem (see [Feng et al., 1994] for a review). Most of these techniques are not suited to swarm robotics, because either they rely upon external devices and infrastructures or they require complex sensing and computational capabilities. In our experiments, the robots use odometry to estimate their position relatively to the source. Odometry consists in the integration of motion sensor information to estimate the position of the robot. Contrary to other techniques, odometry is suited for swarm robotics as it can be easily implemented using onboard sensors (i.e., motor encoders) and it requires low computational capabilities. However, odometry suffers from the accumulation of errors in time that are caused by several phenomena (e.g., mechanical imperfections, finite sensors resolution, wheel slippage). Due to these errors, the estimate of the relative position of the robot with respect to the source can deviate from the real one. Upon reaching the

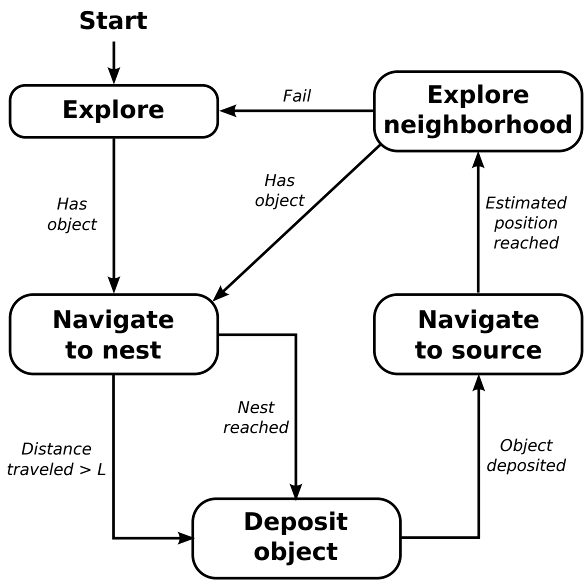


Fig. 1. Finite state machine describing the high level behavior of each robot.

position at which the source is estimated to be, the robot performs a *neighborhood exploration*, which consists in searching for the source within a limited area centered around the estimate. Neighborhood exploration allows to compensate for relatively small estimation errors. In case the estimation error is big, the neighborhood exploration is likely to fail. If this happens, the robot explores again the environment to locate the objects source. We say that the robot *got lost* when the neighborhood exploration fails and the robot must explore the environment again. Exploring the environment is a time consuming operation due to its stochastic nature, therefore it is desirable to minimize the frequency at which a robot gets lost.

In a previous work, we showed that task partitioning can be used to reduce the negative effect of odometry errors and to improve the localization capabilities of the robots [Pini et al., 2012c]. Instead of traveling all the way from source to nest, each robot only travels a limited distance L from the source (i.e., it performs only a part of the transportation task). Upon traveling such distance, referred to as *partition length*, the robot deposits the object on the ground and returns to the source using its position estimate. Since the odometry error grows with the distance traveled, the expected estimation error is smaller compared to the case in which the robot travels all the way from the source to the nest. The result is that the probability that a robot gets lost diminishes.

As objects can be deposited everywhere in the environment, the robots can find them not only at the source, but also in other locations along the way from source to nest. The result is that the object transportation task is carried out 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. We refer to the way a transportation task instance is performed in terms of number and length of the sub-tasks as the *partitioning strategy* employed by the robots to tackle that task instance. For example in Figure 2, the robots R_1, R_2, and R_3 use a partitioning strategy such that the task is partitioned into three sub-tasks, each contributing with the same amount of work to transportation. The robots R_4 and R_5 employ a partitioning strategy such that the task is partitioned into two sub-tasks, each contributing a different amount of work to the transportation task.

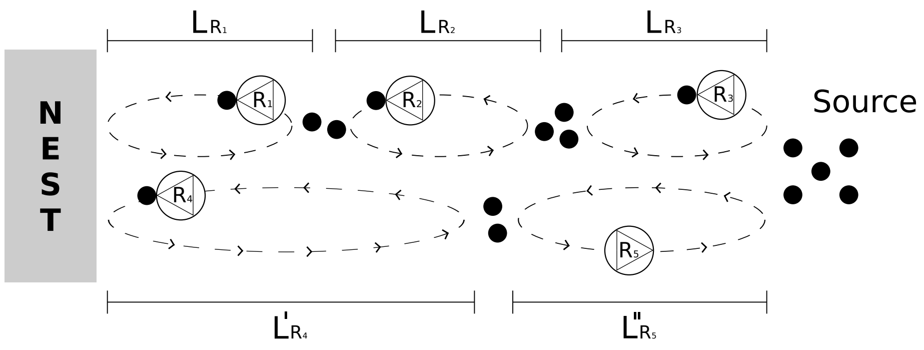


Fig. 2. Representation of the task and contribution of the robots to object transportation. Each robot transports objects towards the nest for a limited distance L, which can be different from robot to robot. The way the object transportation task is partitioned into sub-tasks depends on the values L selected by the robots.

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. This reduces the number of times each object is deposited on the ground before reaching the nest. Depositing an object on the ground 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. *The Approach Applied to Foraging*

The general idea upon which our approach is based is represented in Figure 3 (left). A *cost function* maps the *amount of work* contributed by a robot performing a sub-task to the *cost* of executing of the overall task. Each robot utilizes the cost function to decide the amount of work it contributes to the overall task, with the goal of minimizing the resulting cost for the execution of the overall task. The cost function cannot be defined a priori since it depends on properties of the environment and the tasks that may be unknown. Additionally, the cost function may change in time, since the cost associated to the execution of tasks is likely to vary as a result of the actions performed by the robots. As a consequence, the cost function must be determined with an online process performed by the robots. Each robot is merely able to build a *model* of the real cost function, based on sensory input. To apply our approach to a given scenario, the robots must be equipped with the means of modeling the cost function.

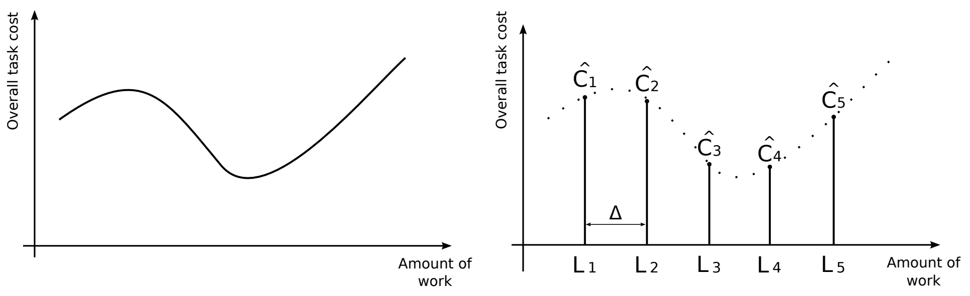


Fig. 3. Cost function (left) and its modeling as done by each robot (right). The cost function maps the amount of work contributed by robot performing a sub-task to the resulting cost for performing the overall task. This information is used by each robot to decide how to partition the overall task into sub-tasks.

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 context of the foraging scenario studied in this work, the amount of work contributed by a robot is proportional to the distance traveled by that robot with an object. Therefore we directly express the amount of work as the *distance* L traveled by a robot when carrying an object. Given this relation between amount of work and distance traveled, we refer to the *length* of a sub-task performed by a robot to indicate the amount of work contributed by the robot performing that sub-task. 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 object 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 value L_i of the partition length. In this context, selecting the length of a sub-task consists in 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 the model of the cost function is built by the robots and used to implement autonomous task partitioning.

Figure 4 illustrates the high level behavior of each robot and can be used as a reference to better understand the concepts explained throughout the rest of this section. In the figure, the white rectangles indicate actions performed by the

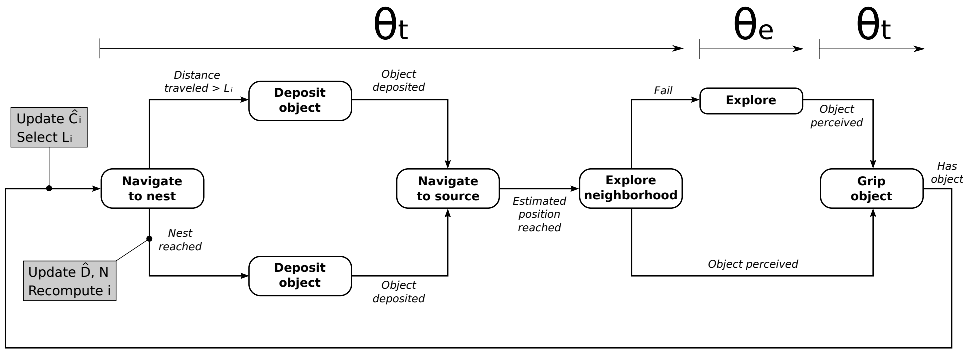


Fig. 4. High level representation of the behavior of the robots. The state machine also indicates when information is updated by the robot (gray rectangles).

robot. The gray rectangles indicate moments in which the robot updates internal information that is related to the approach described in this section.

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. The set $\mathbb{L}$ corresponds to the domain of the cost function modeled by a robot (see Fig. 3, right). Each robot selects a particular value from this set and utilizes it as its partition length. In this section, we explain how robots build the set $\mathbb{L}$. In Section 3.3, we illustrate possible ways by which the selection of the partition length can be implemented.

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 $\mathbb{L}$ by discretizing the overall task length D, which corresponds to the distance between source and nest in our scenario, with a discretization step Δ (see Figure 3, right). The value of 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 perception range of the robots (see Section 4). The discretization is performed by a robot on the basis of $\hat{D}$, which is its estimate of the real distance D between source and nest (i.e., an estimate of the overall task length). Notice that the value $\hat{D}$ of a robot can underestimate D: 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). For this reason, the robot must keep $\hat{D}$ and the set $\mathbb{L}$ up to date. Details about the way the robots update $\hat{D}$ and $\mathbb{L}$ are given at the end of this section.
Elements of $\mathbb{L}$: The values L_i that compose the set $\mathbb{L}$ are calculated as follows:

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

In addition to the $N-1$ values L_i, the set $\mathbb{L}$ also contains the special value L_{np}, which corresponds to performing the transportation task without employing task partitioning. A robot selecting L_{np} transports the carried object all the way to the nest. Figure 5 represents the set $\mathbb{L}$ and illustrates the relationship between N, the

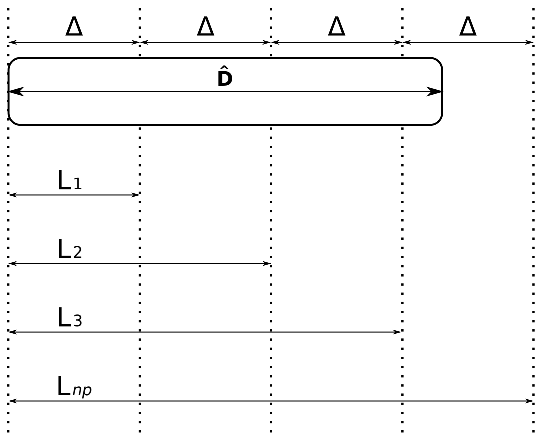


Fig. 5. Example representing the set $\mathbb{L}$ from which a robot selects the value of the partition length. The robot builds the set by discretizing the estimated task length $\hat{D}$ with a step Δ, obtaining $N-1$ values L_i. The figure reports the case in which $N = 4$. The special value L_{np} is also an element of $\mathbb{L}$ and corresponds to performing the task without employing task partitioning.

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 following trip towards the nest (see Fig. 4, left-hand side). Consequently, different task instances (i.e., objects to be transported) can be partitioned in different ways, depending on the partition length L_i selected by the robots involved in the transportation (see, for example, Fig. 2). Notice that each robot is not aware of the choice of L_i made by the 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 Fig. 4), it checks if its current value $\hat{D}$ underestimates the distance to the nest.[a] In this case, the robot sets $\hat{D}$ to the estimated distance to the source. In case the value of $\hat{D}$ is updated, the robot also updates the set $\mathbb{L}$, using Equations 1 and 2. Initially, robots do not have information about the source to nest distance. Initially the value of $\hat{D}$ is set to zero and the set $\mathbb{L}$ only contains L_{np}. Consequently, the robots travel all the way to the nest in their first trip, thus calculating a first estimate of the value of D.

3.2.2. *Estimation of the costs*

The robots construct a model of the cost function. Such model consists of a set of values $\hat{C}_i$, each associated with a value in $\mathbb{L}$. Each robot approximates the overall task cost 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 that robot. Upon gripping an object, a robot updates its cost estimates $\hat{C}_i$ and selects a new partition length value L_i to employ next (see Figure 4, left). The cost estimates $\hat{C}_i$ are updated as follows:

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

where α is a memory factor and $C^{'}$ is the cost associated to the last trip towards the nest (i.e., the last sub-task performed) using a given partition length L_i. $C^{'}$ is computed as:

$$C^{'} = \frac{\hat{D}}{L_i}\vartheta_t + \vartheta_e \tag{4}$$

where ϑ_t is the measured cost of the last sub-task performed by the robot (see Figure 4, top). ϑ_t is measured from the moment an object was gripped (excluding the time needed to grip that object) to the moment the following object was gripped (including the object gripping time). ϑ_t does not include the time spent exploring the environment in case the robot got lost (if any), which is measured by ϑ_e (also indicated in Figure 4, top).

ϑ_t takes into account not only the cost of the actions performed to complete a sub-task (i.e., transporting and dropping an object), but also the cost of the actions that must be performed before the following sub-task can be started (i.e., returning to the source, performing neighborhood exploration, and gripping the following object).

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

[a]Notice that the robots can only determine whether $\hat{D}$ underestimates D and not whether it overestimates D.

(transition labeled *nest reached* in Fig. 4), it recomputes the index i on the basis of the distance that was actually traveled. In fact, if the value L_i selected by a robot upon gripping an object exceeds the distance between the robot and the nest, the robot actually travels a smaller distance than L_i. In this case, the cost to be updated is the one associated to a shorter sub-task (i.e., a smaller L_i). If, on the other hand, the robot does not reach the nest, the cost $\hat{C}_i$ that will subsequently be updated is the one associated to the selected value 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 the list of values that can be selected, it initializes the associated costs $\hat{C}_i$ with a random value.

3.3. *Task partitioning algorithms*

In the experiments presented in this paper, we compare different task partitioning algorithms that can be utilized to select the value of the partition length from the set $\mathbb{L}$. We compare an algorithm based on the approach proposed in the previous sections to a set of reference algorithms. All the reference algorithms utilize the set $\mathbb{L}$ to select the value of the partition length L_i. However, in none of them the selection is done on the basis of the cost estimates $\hat{C}_i$ that model the cost function. In the rest of this section we describe in detail each algorithm.

3.3.1. *The cost-based partitioning algorithm*

The application of the approach presented in the Section 3.2 requires a mechanism to select the value of the partition length L_i among the possible values in $\mathbb{L}$, on the basis of the cost estimates $\hat{C}_i$. In this work, we use the *ε-greedy* algorithm for selecting the value of the partition length. ε-greedy selects with a probability $1-\varepsilon$ the value L_i with the minimal associated cost 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 value of the partition length.

Notice 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]), could be used in place of ε-greedy. We decided to use ε-greedy because of its simplicity and because its only parameter ε directly expresses the degree of exploration of the algorithm.

3.3.2. *The fixed algorithms*

A first family of reference algorithms are the *fixed* algorithms: the partition length L_i is fixed a priori to a given value. This value is the same for all 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 that is used by the algorithm. A special case of fixed algorithm is the *never partitioning algorithm*: in this case, the robots do not employ task partitioning and transport objects from the source to the nest.

3.3.3. *The random initialization algorithm*

The *random initialization algorithm* consists in stochastically selecting the value of the partition length L_i in $\mathbb{L}$ at the beginning of the experiment. Each robot selects its own value L_i and never changes it during the course of the experiment.

The algorithm requires to initialize the set $\mathbb{L}$. To this aim, the first time a robot grips an object, it transports it directly to the nest, so that $\hat{D}$ can be estimated. 10% of the value is then added to the estimate, to partially compensate for under-estimation errors. The resulting value is used to initialize the set $\mathbb{L}$ as described by Equations 1 and 2. The robot then stochastically selects a value L_i from the set $\mathbb{L}$ and the selected value is used by that robot throughout the rest of the experiment. We also tested a variation of the algorithm, whereby the robots stochastically select the value of the partition length among the possible values L_i each time an object is gripped. The algorithm performed badly across all the experiments and therefore we do not include its results in this work.

4. Experimental tools

In this section, we describe the tools that we utilized to carry out the experiments presented in this paper. All the experiments have been carried out in simulation. In the experiments, we simulate the foot-bot robotics platform using the ARGoS simulator. ARGoS allows a faithful simulation 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 the features of the foot-bot robotic platform that are relevant for the experiments presented in this work. In the same section, we also present objects that are used to perform foraging experiments with the foot-bot. In Section 4.3, we describe the simulation software ARGoS and illustrate how the system is implemented in simulation, focusing in particular on the model of the odometry noise.

4.1. *Environment*

The robots perform foraging in the environment represented in Figure 6. The environment is a rectangular arena surrounded by walls. Its dimensions W and L depend on the specific experiment (see Section 5). The nest, located in proximity of one of the walls, is marked by a black rectangular patch on the ground, 1.4 m wide and 0.45 m long. Three lights (represented by crossed circles) are located outside the perimeter of the arena, in proximity of the nest. The lights are used by the robots to determine the direction of the nest.

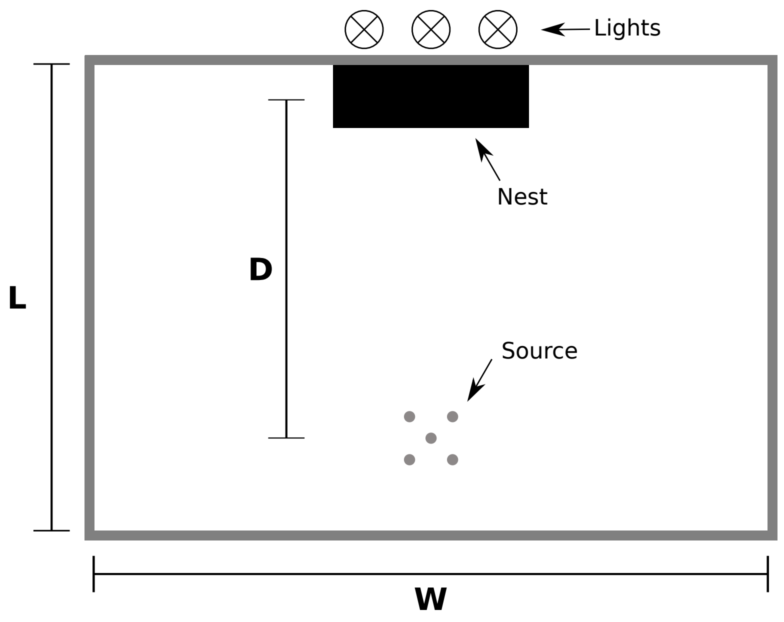


Fig. 6. Environment in which the robots perform foraging. The length L and the width W depend on the specific experiment. The source is located at a distance D from the nest and it is composed of 5 objects. The nest (black rectangle represented on top), can be reached by approaching three lights located in its vicinity.

The object source is located on the opposite side with respect to 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 of the group. Each time a robot removes an object from the source, a new one is added in the same location. Thus, the source never depletes. When a robot releases an object within the boundaries of the nest, transportation is completed and the object is removed from the environment.

4.2. *Foot-bots and Objects*

The foot-bot is a mobile ground robot developed within the *Swarmanoid* project [Dorigo et al., 2012]. The foot-bot is roughly cylindrical-shaped, it has a diameter 170 mm and an height of 290 mm, and it weights 1.8 kg. Figure 7 depicts a foot-bot and highlights sensors and actuators that have been utilized in the experiments presented in this work.

The foot-bot navigates the environment by means of a differential drive system that combines tracks and wheels. A metal gripper is used to connect to objects

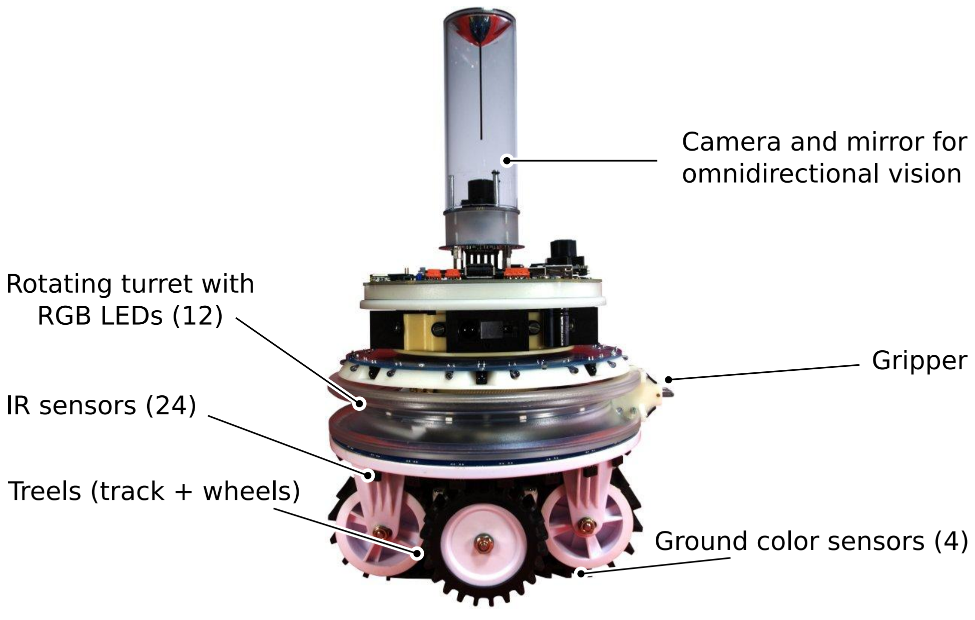

Camera and mirror for omnidirectional vision
Rotating turret with RGB LEDs (12)
Gripper
IR sensors (24)
Treels (track + wheels)
Ground color sensors (4)

Fig. 7. The foot-bot robotic platform. The sensors and actuators utilized in the experiments are highlighted in figure.

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 as colored blobs up to a distance of roughly 0.5 m. The infrared ground sensors are positioned underneath the robot and allow the robots to recognize the color of the ground. In the experiments, the ground sensors are used by the robots to detect the black patch representing the location of the nest. Twenty-four infrared sensors are evenly distributed below the plastic ring that hosts the gripper. These sensors can be utilized to measure the intensity of the ambient light and they also serve as bumpers, to detect obstacles in the vicinity. In the experiments, the infrared sensors are employed to determine the direction of the lights marking the nest as well as to provide information about the presence of obstacles to be avoided (walls and other robots). For more information about the foot-bot, we refer the reader to the work of [Bonani et al., 2010].

To perform transportation experiments with the foot-bots, we designed objects that can be grasped by the robots using their metal gripper. In Fig. 8, we show one of these objects and report its dimensions. The object is composed of a 90 mm plastic ring that can host the gripper of a foot-bot. The ring is fitted on top of a 60 mm tall PVC pipe with a diameter of 80 mm. A red cardboard disk, on top of the plastic ring, can be perceived by the robots using the omnidirectional camera. For more details about the design and the features of the objects refer to the technical information reported in [Brutschy et al., 2012].

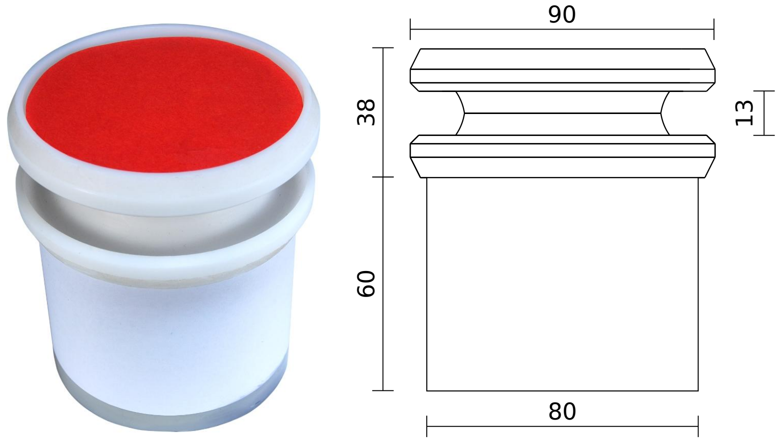


Fig. 8. Objects utilized to perform foraging. On the left an actual picture of an object. On the right a schematic representation reporting its dimensions in mm.

In the rest of this section, we provide implementation details about the behavior of the robots and describe the odometry noise that characterizes the foot-bots.

Exploration of the environment: The robots explore the environment using a random walk implemented as follows. By default, the robots move straight, at a maximum velocity of 0.1 m/s, avoiding obstacles. The stochastic component consists in randomly generating a new direction of motion, uniformly sampled in $[-115°, 115°]$. The random direction is generated with a 5% probability per control-step. The neighborhood exploration is also performed using random walk, but the robot remains in a circular area of radius 0.5 m, centered around the position at which the robot expected to find the source. If a robot is about to leave the exploration area, it generates a new random direction biased towards the center of the area. As mentioned, a robot may abandon the exploration of a neighborhood when unsuccessful. A timeout mechanism governs abandoning. The robot abandons in case it has been performing the neighborhood exploration for 60 s without detecting any object.[b] In this case, in fact, it is likely that the robot reached a position far away from the object source.

Object gripping: Object gripping is based on vision; the foot-bot uses information from the omnidirectional camera to perceive objects in the surrounding and to approach them. Once the robot is close to an object to be gripped, it uses the front proximity sensors to refine its alignment. A repulsion mechanism is also part of object gripping: the robots ignore any red blob (i.e., an object) which is perceived in proximity of a blue blob. This prevents that more robots grip the same object at the same time. 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 towards the nest.

Check for unreachable destination: Due to odometry errors, the robot may try to reach a position that lies outside the perimeter of the arena while navigating to the source. To determine whether it is trying to reach a position that is not within the arena boundaries, each robot periodically checks its estimated distance to the source. If the distance does not decrease in time, the robot assumes that it is trying to reach a position outside the arena perimeter and it returns exploring the environment (i.e., it gets lost). In fact, if the estimated distance to the source is not decreasing, it means that an obstacle blocks the movements of the robot. If this happens for a long period of time (30 consecutive seconds in our experiments), it is likely that the obstacle is a wall marking the perimeter of the arena, rather than another robot. Notice that the mechanism is prone to errors: if the density of robots is high, the movements are harder and the mechanism described here can be triggered by the presence of other robots.

Odometry noise: The foot-bots suffer from a systematic drift towards the left-hand side with respect to the direction of motion. Figure 9 shows the shape of the trajectory followed by a foot-bot when the speed of both wheels is set to $-0.1\,\mathrm{m/s}$ (left-hand side) and $0.1\,\mathrm{m/s}$ (right-hand side). The figure is built using snapshots taken from a video that is available with the online supplementary material [Pini et al., 2012b]. The figure reports the direction of motion of the robot (white arrow on top), the trajectory followed by the robot (white continuous line) and a reference straight trajectory (white dashed line). The drift towards the left-hand side is not constant: the amount by which the same foot-bot drifts varies from trip to trip. The foot-bot cannot measure such drift: using odometry the robot would estimate its trajectory to be (roughly) straight.

The odometry of the robots could be improved by the usage of calibration pro-

[b]The values of the neighborhood exploration time and radius are the same utilized in our previous work [Pini et al., 2012c].

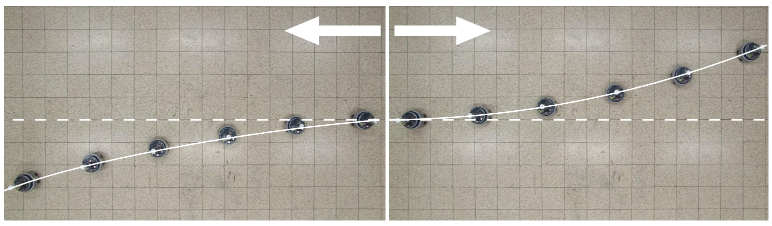


Fig. 9. Trajectory followed by a real foot-bot when the speed of both wheels is set to $-0.1\,\mathrm{m/s}$ (left-hand side) and $0.1\,\mathrm{m/s}$ (right-hand side)

cedures or Kalman filtering [Kalman, 1960]. However, in our experiments the goal is not to maximize the foraging efficiency of the robots, but to test the proposed approach for autonomous task partitioning in a realistic setup. Therefore we do not implement any mechanism that aims at correcting or limiting the odometry error.

4.3. *Simulation of the system using ARGoS*

ARGoS[c] is the physics-based robot simulator that we employed for the experiments presented in this paper. ARGoS allows the real-time simulation of thousands of robots and it is highly customizable. The level of detail at which the robots, their sensors and actuators, and the physical space are simulated is defined by the user. For additional details about ARGoS, we refer the reader to the original contribution of Pinciroli et al., 2011 [Pinciroli et al., 2011].

In the foraging experiments presented in this work, we employ the 2D-dynamics physics engine offered by ARGoS. The simulation proceeds at discrete time-steps, 0.1 simulated seconds long. All the simulated sensors and actuators are subject to noise. Gaussian noise with $0.02\,\mathrm{m}$ standard deviation is added to the distance readings of the omnidirectional camera. At each simulation 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 $W = 6.7\,\mathrm{m}$ by $L = 4.5\,\mathrm{m}$ arena. Each sample records the time spent by a robot for gripping an object (i.e., the time from the moment the object was perceived to the moment it was gripped). Figure 10 reports the empirical distribution of the grip time samples. The samples are utilized in simulation to model the time spent by a robot to grip an object. 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 that can be observed with the real foot-bots in the

[c]`http://iridia.ulb.ac.be/argos`

time needed for gripping objects.

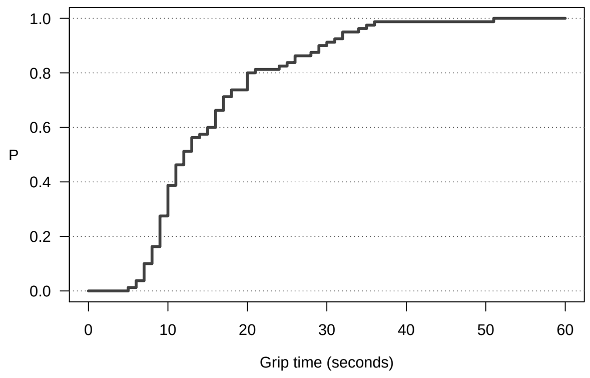


Fig. 10. Empirical distribution function (P) of the 80 samples utilized to simulate the object gripping time. The samples have been collected with a group of 6 foot-bots performing foraging.

The odometry noise plays an important role in our experimental setup since it defines how successful the robots are in finding objects when returning to the source. The model of the odometry noise that we use in simulation is the same that we originally proposed in [Pini et al., 2012c]* The model is built on the basis of odometry error samples collected with real foot-bots performing foraging. The setup of the experiments is as follows: a group of six robots performs foraging in the 6.7 m by 4.5 m rectangular arena represented in Figure 6. The objects source is located at a distance of $D = 4$ m from the nest. Upon gripping an object from the source, a robot travels all the way to the nest (i.e., task partitioning is not utilized). When the robot reaches the nest, it releases the object and it returns to the source using odometry.

TODO: Change from tech report to journal, if it ever will be published - FIX BIB ENTRY

The data upon which the odometry noise model is built was sampled from video recordings of the experiment described. A total of 61 error samples were collected from the videos, using the tiles on the floor as a reference to calculate the odometry error. These measures include effects such as collisions, wheel slippage, and avoidance as they are in the experiment at hand.

Figure 11 reports the 61 error samples collected from the videos. The origin of the axes in the plot represents the position in which an object is gripped by a robot. A point in the plot reports the X, Y error between the position at which the object was gripped, and the final position reached by the robot when returning to the source using odometry.

Algorithm 1 reports the pseudo-code that describes how the odometry noise is implemented in simulation. The drift towards the left-hand side is obtained by modifying the actuated values of the two wheel speeds (lines 2 to 7): the speed of the right wheel is incremented when the actuated value is positive, the speed of

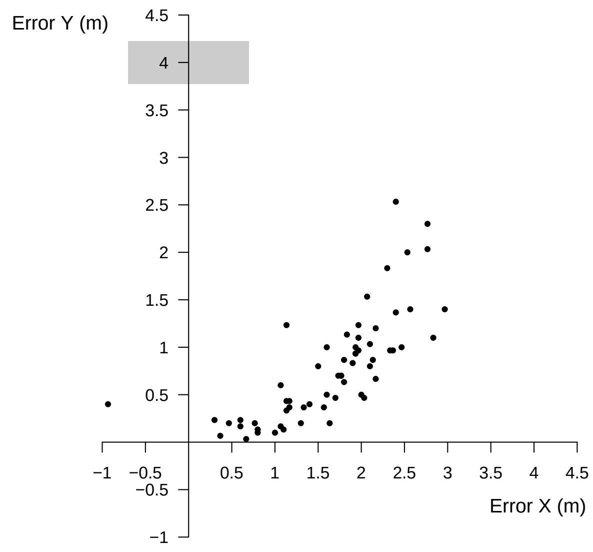


Fig. 11. Odometry error samples collected in experiments performed with the foot-bots (61 samples). Each sample reports the X, Y estimation error at the end of a trip from source to nest and back. The light gray rectangle marks the position of the nest.

Algorithm 1 Pseudo-code for the odometry noise model

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

the left wheel is decremented when the actuated value is negative. The robot is not aware of the drift: the odometry is computed using *actuatedRightWheelSpeed* and *actuatedLeftWheelSpeed*. The parameter μ_{noise} represents the percentage by which the wheel speed is increased or decreased. The value of μ_{noise} changes each time a robot grips an object. The new value is sampled from a Rayleigh distribution with parameter σ (lines 8 to 10). The same distribution is used to randomly initialize μ_{noise}.

The value of the parameter 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 value of the parameter σ was fitted through a set of targeted experiments. The default value of σ, indicated as $\bar{\sigma}$, was set to 0.0134. The same value has been utilized in all the experiments presented in our previous work (see [Pini et al., 2012c]). $\bar{\sigma}$ allows to replicate in simulation a pattern similar to the one reported in Fig. 11. To test the impact of noise on our system, in the experiments presented in this work we vary the value of the parameter σ. We express the value of σ used in an experiment as a fraction of $\bar{\sigma}$. Figure 12 reports error points (70 in each plot) collected in simulation for the different noise conditions tested in the experiments. In the figure, plots from left to right correspond to increasing values of the odometry noise.

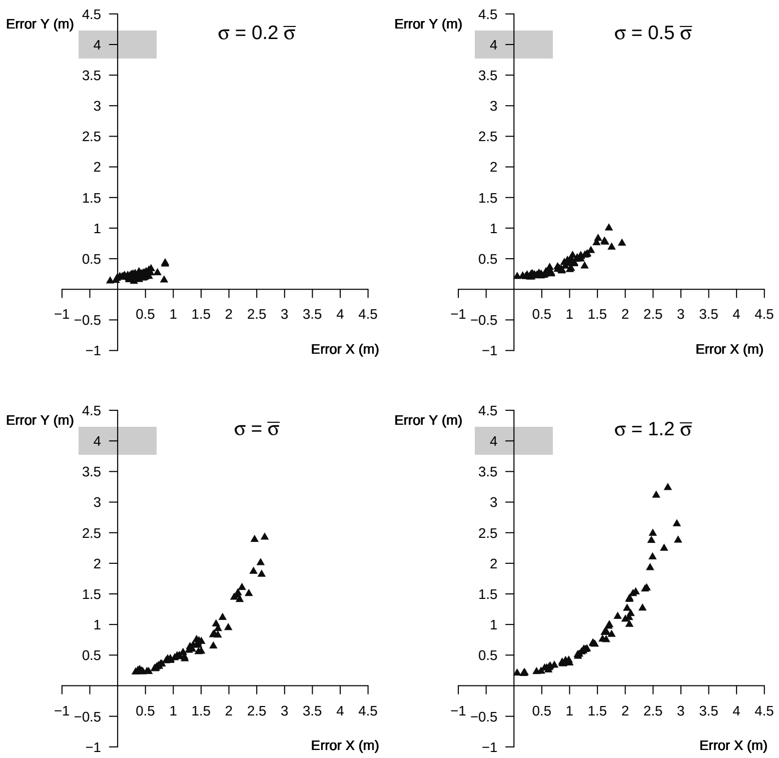


Fig. 12. Odometry error samples collected in simulation for different noise conditions.

The presented noise model does not take into account internal aspects of the real foot-bot. For our simulation this is acceptable, because we only aim at reproducing the observable behavior of the foot-bot with a sufficient degree of accuracy.

Table 1. Default experimental settings

Parameter	**Value**
Experiment duration (simulated time)	20 hours
Swarm size	4, 10, 20
Size of the environment	$W = 6.7\,\mathrm{m}$ by $L = 4.5\,\mathrm{m}$
Source-to-nest distance	$D = 4\,\mathrm{m}$
Number of repetitions	20
Default noise parameter σ	$\bar{\sigma} = 0.0134$

5. Experiments and Results

In this section, we describe the simulation-based experiments that we carried out to test our approach for autonomous task partitioning. The experiments are divided into five sets, each aiming to test different aspects. When not differently specified, the experimental parameters are set as follows (see Table 1). The experimental environment measures 6.7 m in width and 4.5 m in length (see Fig. 6). Source and nest are placed at a distance $D = 4\,\mathrm{m}$ from each other. Each experimental run lasts a total of 20 simulated hours. At the beginning of each run, the robots are positioned inside the nest, with a random orientation. We test three different swarm sizes: 4, 10, and 20 robots. 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 a set of experiments that have the goal of evaluating the basic properties of the system, and selecting the values of the parameters α and ε, used in the cost-based partitioning algorithm. In Section 5.2 we present experiments in which we test whether our approach is able to cope with environments of different size. In Section 5.3 we describe experiments that aim to evaluate the behavior of the system in relation to different values of the source-to-nest distance D. In Section 5.4, we describe experiments in which we study the effect of heterogeneity within the swarm. Finally, in Section 5.5, we evaluate a case in which the environmental conditions vary in time, and discuss the exploitation vs. exploration trade-off.

5.1. *Experimental set 1: Basic Properties*

The first set of experiments has two main goals. The first goal is to assess the best way of partitioning the transportation task in relation to the size of the swarm and the amount of noise in the odometry system. We perform experiments in which we test the reference algorithms with 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 test

Table 2. Best performing fixed algorithm for different values of the odometry noise parameter σ and different swarm sizes.

	$\sigma = 0.0$	$\sigma = 0.2$	$\sigma = 0.5$	$\sigma = 1.0$	$\sigma = 1.2$
4 robots	2.5 m	2.5 m	2.0 m	1.5 m	1.5 m
6 robots	2.5 m	2.5 m	2.0 m	1.5 m	1.5 m
8 robots	2.5 m	2.5 m	2.0 m	1.5 m	1.5 m
10 robots	2.5 m	2.5 m	2.0 m	1.5 m	1.5 m
15 robots	2.0 m	2.0 m	1.5 m	1.5 m	1.5 m
20 robots	1.5 m	1.5 m	1.5 m	1.5 m	1.0 m

different versions of the cost-based partitioning algorithm, that vary for what concerns the value of the parameter α, used for computing cost estimates according to Equation 3, and ε of the ε-greedy algorithm. The parameter α is selected from the set $\{0.001, 0.1, 0.25, 0.9, 1.0\}$, while ε from the set $\{0.0, 0.05, 0.15, 0.25, 0.5\}$. The remaining experimental conditions are as reported in Table 1.

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 by the robots when that algorithm is employed.

Table 2 reports, for the different values of σ and the different swarm sizes, the partition length value of the best performing fixed algorithm in the corresponding setting. The complete results of these experiments can be found with the online supplementary material [Pini et al., 2012b]. The results reported in the table highlight two aspects. First, they confirm that the higher the noise, the more task partitioning becomes advantageous. In fact, for a given swarm size (i.e., a given table row), which fixed algorithm performs best varies in relation to the value of the odometry noise: the higher the noise, the smaller the partition length of the best performing fixed algorithm. In other words, if the value of 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 second aspect highlighted by the experiments is that, given certain noise conditions (i.e., a given column in Table 2), in large swarms it is preferable to partition the given task into many, short sub-tasks. This comes from the fact that 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, in addition to evaluating the effect of the odometry noise and of the swarm size, the experiments of the first set were also used to select the values of the parameters ε and α. Figure 13 summarizes the effect of the parameters ε and α on the performance of the cost-based partitioning algorithm, for the case in

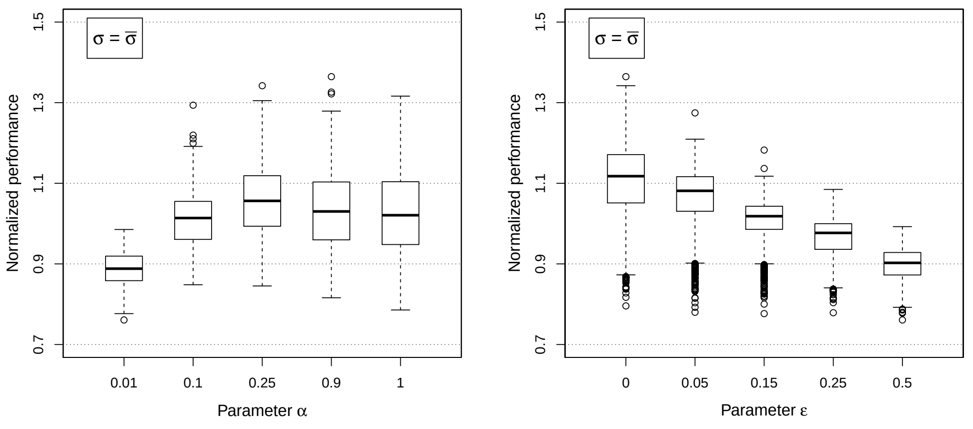


Fig. 13. Effect of the parameter α (left) and ε (right), for $\sigma = \bar{\sigma}$. The data reported in the plots is computed as follows. We calculate the average performance P_N, computed over all the values of ε and α, for each swarm size N. The value P_N is used to normalize the performance values recorded for swarms of size N. Each box in the left-hand side plot aggregates the normalized performance for different swarm sizes and values of ε, for a given value of α. Analogously, in the right-hand side plot the boxes aggregate the normalized performance for different swarm sizes and values of α, for a given value of ε.

which $\sigma = \bar{\sigma}$. The complete results of the experiments are reported with the online supplementary material [Pini et al., 2012b].

The data reported in the Fig. 13 is computed as follows. First, we calculate the average performance P_N, computed over all the values of ε and α, for each swarm size N. Each box in the plots of Fig. 13 aggregates the performance of swarms of different size, divided by the corresponding value P_N. The left-hand side plot of Fig. 13 reports the data for different values of α (i.e., each bar aggregates all the values of ε and swarm size N). Analogously, the plot on the right-hand side of Fig. 13 reports the data for different values of ε. The plot on the left-hand side shows the overall effect of α, the one on the right-hand side the overall effect of ε. The plots show that the highest levels of performance are obtained for $\alpha = 0.25$ and $\varepsilon = 0$. We select these values and utilize them in all the experiments presented in the paper, with the exception of the ones presented in Section 5.5.

The value 0 for the parameter ε corresponds to a purely exploiting version of the ε-greedy algorithm: the algorithm always selects the partition length L_i associated to the minimal cost estimate $\hat{C}_i$. This is not surprising given the nature of the experiments of the first set. In fact, the only variations occurring in the system are those introduced by the robots dropping objects along the path from source to nest. The remaining conditions, such as the number of robots, the noise on the odometry system, the size of the environment, etc. do not vary in time. Consequently, no exploration is needed and a pure exploiting version of the ε-greedy algorithm performs well. Further considerations about the trade-off between exploration and exploitation are presented in Section 5.5.

Figure 14 reports the performance of the cost-based partitioning algorithm and of three reference algorithms for swarms of 4, 10, and 20 robots. The top plot reports the results for a low value of noise ($\sigma = 0.2\ \bar{\sigma}$), the bottom plot for $\sigma = \bar{\sigma}$. For clarity, we did not report the performance of all the reference algorithms. For a given experimental setting (i.e., odometry noise and swarm size), the fixed algorithm reported in the figure is the one performing the best in the corresponding setting. 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 about the environment. If the robots do not have such knowledge, they have to utilize other partitioning algorithms which, in general, perform worse than the fixed algorithm.

The results reported in Figure 14 highlight several aspects. The performance of all the algorithms decreases with an increasing noise. The lower the noise, the smaller the odometry error and the frequency at which the robots find the objects source when performing neighborhood exploration. As the noise increases, the performance of each algorithm decreases. The never partitioning algorithm performs well when the noise is low and the swarm is composed of few robots. For increasing swarm sizes, the algorithms that utilize task partitioning become more advantageous. As pointed out, this is due to interference: the robots employing the never partitioning algorithm travel all the way from source to nest and interfere with each other's paths. The negative effect of interference on the algorithms increases for a decreasing odometry noise: the robots get lost few times 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, which indicates that the partitioning strategy utilized by the swarm suits the specific conditions in which the robots operate.

Figure 15 reports the partition length values selected in time by robots employing the cost-based partitioning algorithm, for $\sigma = \bar{\sigma}$. The plot reports the percentage of times each value L_i was selected by the robots. The percentages are computed across the 20 experimental runs. The sequence of plots in each row reports the data collected with swarms of different size, from top to bottom: 4, 10, and 20 robots. Each column reports the data relative to a time period of 5 hours, corresponding to a quarter of the duration of each experimental run.

Each sequence of plots shows that there is an initial phase (see plots in the first column) during which the robots sample the values L_i. In time (plots from left to right), the robots converge towards certain values of the partition length. The values selected the most by the robots in the final part of the experiment depend on the size of the swarm: the larger the swarm, the lower the partition length values that are selected by the robots (compare the 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. The results obtained by the fixed algorithms demonstrate

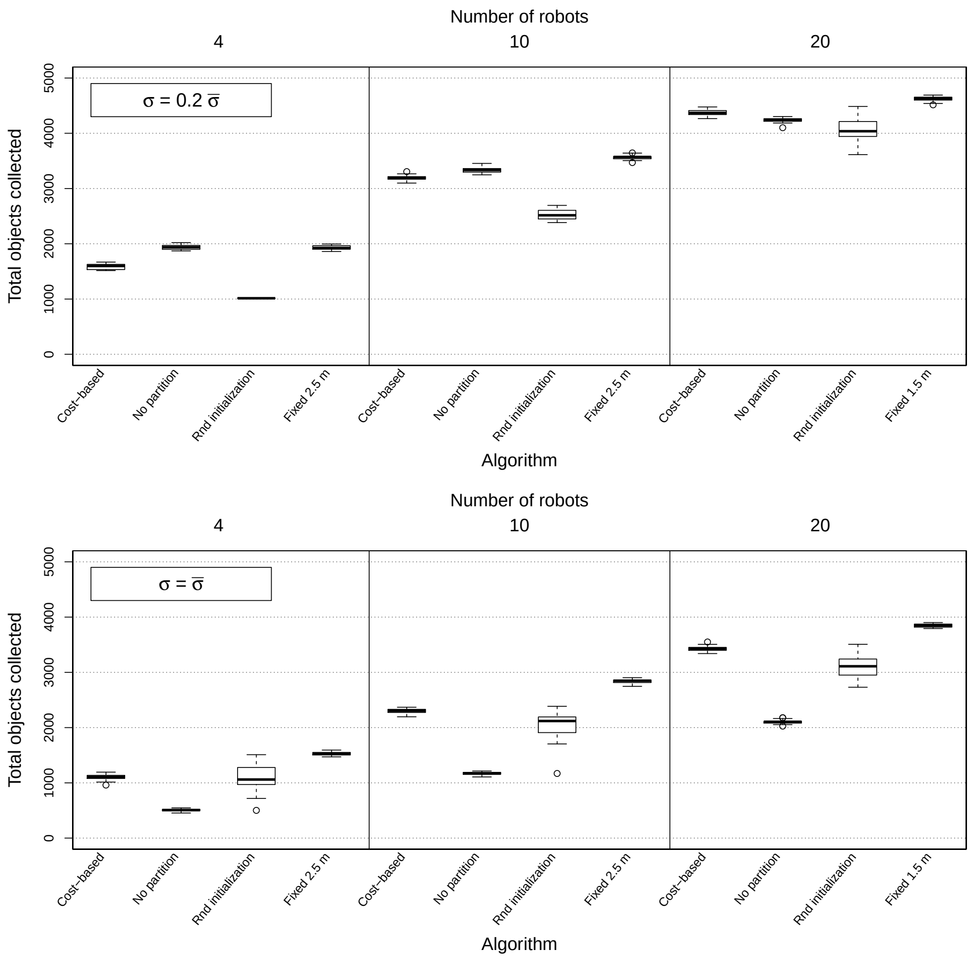


Fig. 14. Objects collected by the swarm when different partitioning algorithms are employed. Each group of bar reports data collected with a swarm of a given size (see top axis). The plot on top reports the data for a noise $\sigma = 0.2\ \bar{\sigma}$, the plot at the bottom for a noise $\sigma = \bar{\sigma}$. The fixed algorithms reported in figure are the ones performing the best in the corresponding setting. The performance of the fixed algorithm is an upper bound for the performance of a swarm that uses task partitioning.

that this is advantageous to diminish physical interference.

Analogously, Figure 16 reports the partition length values selected by the robots in time when the cost-based partitioning algorithm is employed, for $\sigma = 0.5\ \bar{\sigma}$. The plots confirm the trends observed previously: in large swarms the robots select lower values of the partition length. Comparing to the case reported in Figure 15 however, the overall choice of the robots is oriented towards higher values of the partition length. This is due to the lower noise in the odometry system, compared

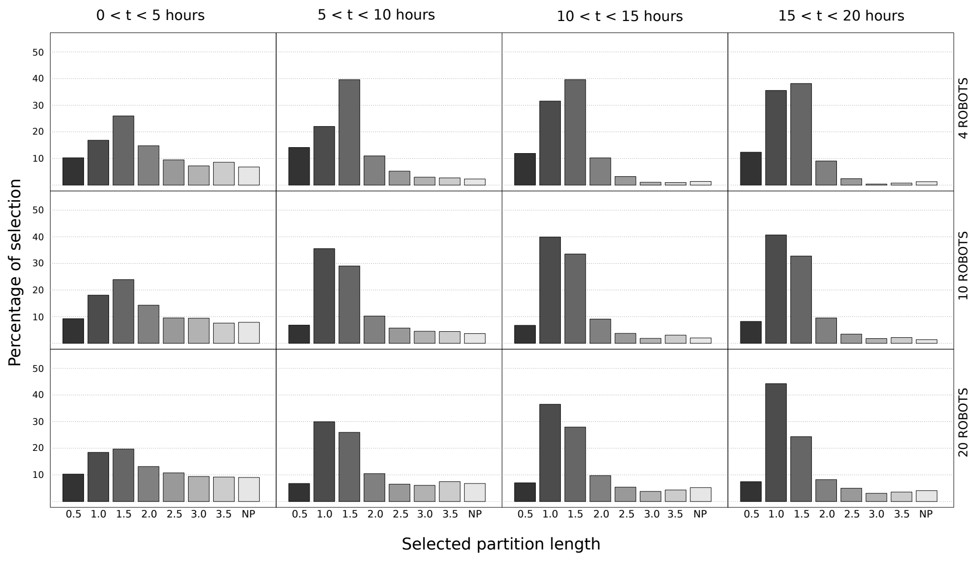


Fig. 15. Partition length values selected by the robots in time, when the cost-based partitioning algorithm is employed. The plots report data collected for $\sigma = \bar{\sigma}$. Each plot reports the percentage of selection of each value L_i in a time window of 5 hours (see labels on top).

to the previous case.

These results confirm that the robots employing the cost-based partitioning algorithm decide how to partition 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 perceived interference. Instead, they only estimate costs linked to performing sub-tasks. The strategy employed by the robots to partition the transportation task varies with the environmental conditions in a self-organized manner, due to the impact of such conditions on the cost estimates.

The plots reported in Figure 17 show the evolution of throughput in time for swarms composed of 4 (Fig. 17 a), 10 (Fig. 17 b), and 20 (Fig. 17 c) robots. The throughput is measured as the number of objects delivered to the nest in one hour. The values reported in each plot are the medians computed over 20 experimental runs, for $\sigma = \bar{\sigma}$. The throughput of the cost-based partitioning algorithm and of three reference algorithms is reported in each plot.

The results reported in Figure 17 show that the throughput of the reference algorithms rapidly stabilizes around a given value. The throughput obtained by the robots employing the cost-based partitioning algorithm, on the other hand, grows slowly in time as an effect of the convergence of the robots to a certain value of partition length (refer to the results reported in Figure 15).

We speculate that the low growth speed is due to the fact that the actions

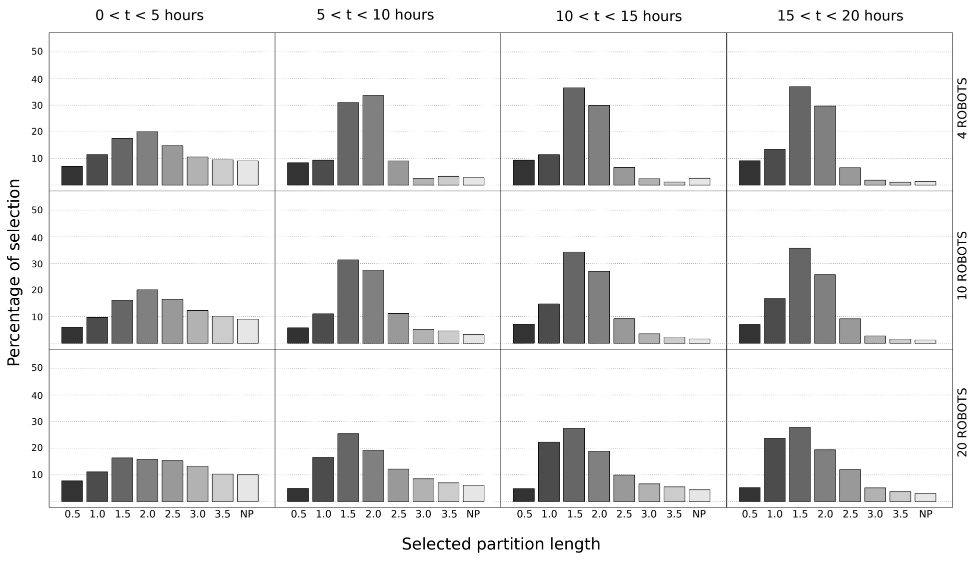


Fig. 16. Partition length values selected by the robots in time, when the cost-based partitioning algorithm is employed. The plots report data collected for $\sigma = 0.5\,\bar{\sigma}$. Each plot reports the percentage of selection of each value L_i in a time window of 5 hours (see labels on top).

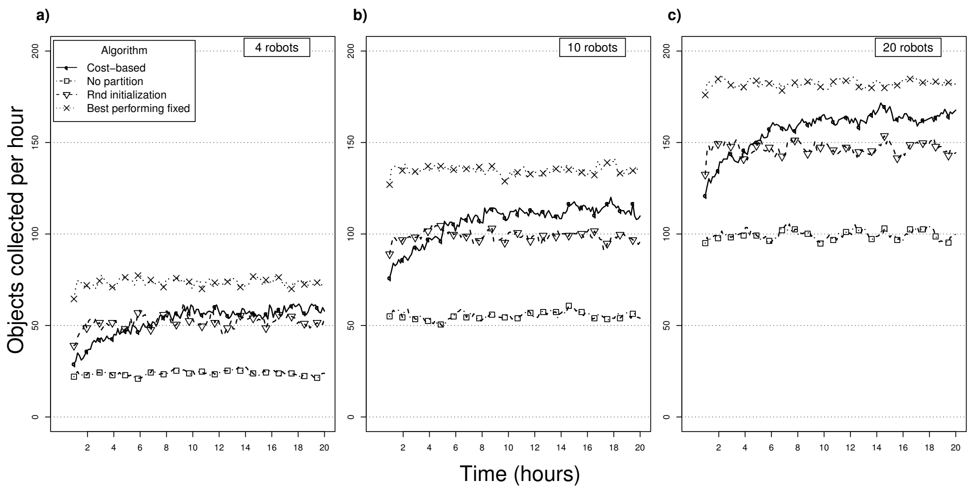


Fig. 17. Evolution in time of the throughput obtained with different algorithms. The throughput is computed as objects delivered to the nest in an hour of experiment. Each curve reports the median value, computed over 20 experimental runs.

performed by each robot influence the cost estimation process of the other robots. For example, a robot r that finds objects on the way between source and nest (i.e., not at the objects source) needs another robot r' constantly delivering objects there. If the robot r' gets lost, the robot r may not be able to find objects anymore. As a consequence, the robot r is likely to assign a high cost to the selected value of the partition length, therefore penalizing its selection for the future. This influence on each other's selection process requires the robots to sample the environment for a longer time in order to rule out the side effect of cost estimation errors and to coordinate their actions. On the contrary, the selection of the partition length done by the reference algorithms is not based upon cost estimates. Therefore, the robots do not interfere with each other's selection process and the swarm rapidly reaches a steady throughput.

To summarize, the results presented in this section confirm that task partitioning limits the negative impact of the odometry noise and of physical interference on the foraging performance. The results also show that the cost-based partitioning algorithm obtains a good performance across all the tested conditions. This is consequence of the fact that the way the overall task is partitioned by the robots using the cost-based partitioning algorithm varies in relation to the noise conditions and the swarm size. On the other hand, each of the reference algorithms performs well only in a limited sub-set of the tested conditions. We pointed out that the throughput of the cost-based partitioning algorithms grows slowly in time, compared to the reference algorithms.

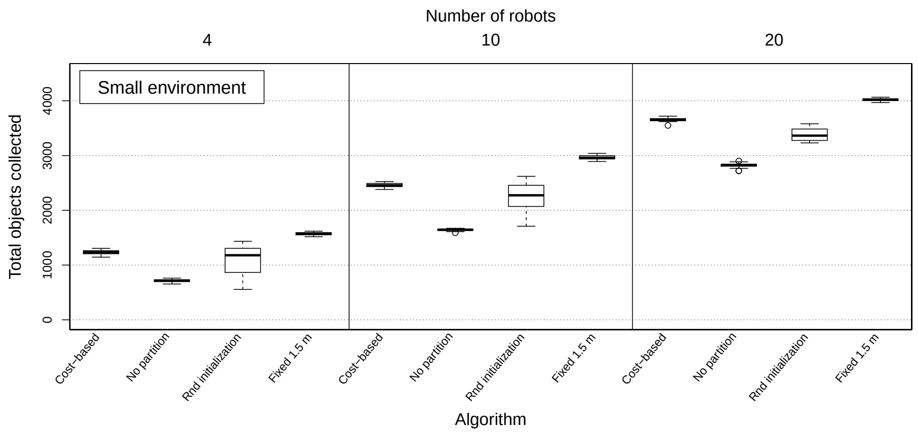
5.2. *Experimental set 2: Size of the Environment*

The goal of the second experimental set is to test the behavior of the system in environments of different size. A large environment requires more exploration, and therefore the robots take longer, on average, to locate the object source at the beginning of the experiment and in case they get lost (i.e., exploration is costly). We test two environments: a $W = 4.5\,\mathrm{m}$ by $L = 4.5\,\mathrm{m}$ environment (*small environment*) and a $W = 6.7\,\mathrm{m}$ by $L = 10.0\,\mathrm{m}$ environment (*huge environment*), for $\sigma = \bar{\sigma}$.[d] In both cases, the source-to-nest distance D is 4.0 m. The remaining experimental parameters are as reported in Table 1.

Figure 18 reports the performance of the cost-based partitioning algorithm and three reference algorithms for different swarm sizes. The top plot in Fig.18 reports the results of the experiments performed in the small environment, the bottom plot in Fig.18 reports the results of those performed in the huge environment. The results of the experiment indicate that the cost-based partitioning algorithm performs well across environments of different size.

Figure 19 reports the values of partition length selected by the robots in the final quarter of the experiment (last 5 hours), for different swarm sizes and different environments. Plots in the same row refer to the same environment: small envi-

[d]The name and the size of the environments are the same used in our previous study [Pini et al., 2012c].



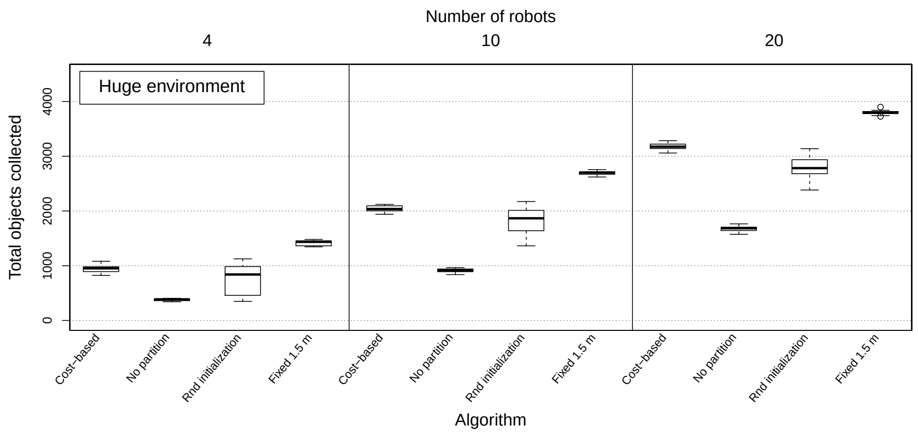


Fig. 18. Objects collected by the swarm when different partitioning algorithms are employed. Each group of bar reports data collected with a swarm of a given size (see top axis). The plot on top reports the data collected in the small environment, the plot at the bottom the data collected in the huge environment.

ronment on top and huge environment at the bottom. Plots in the same column report data for the same swarm size, from left to right: four, ten, and twenty robots. The plots show a trend in the partition length values selected by the robots. For a given swarm size, the selected values are influenced by the size of the environment: the larger the environment, the lower the partition length values selected by the robots. 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.

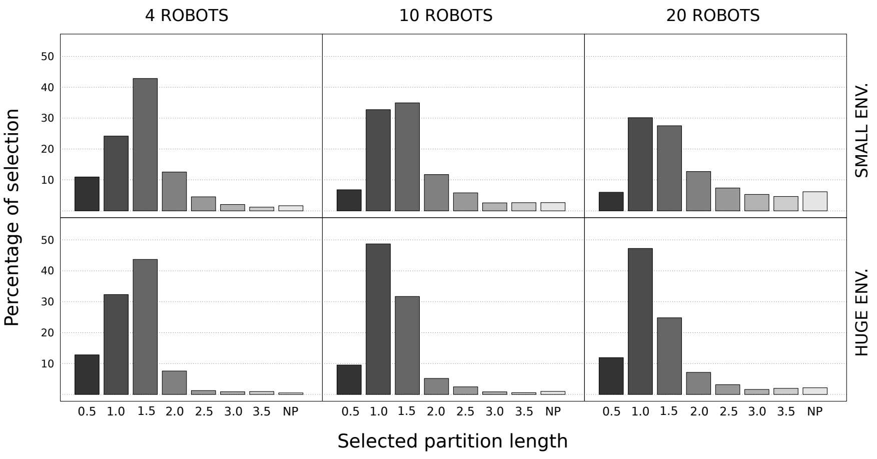


Fig. 19. Partition length values selected by the robots in time, when the cost-based partitioning algorithm is employed. The plots report data collected for $\sigma = \bar{\sigma}$. Each plot reports the percentage of selection of each value L_i in the final quarter of the experiments (last 5 hours).

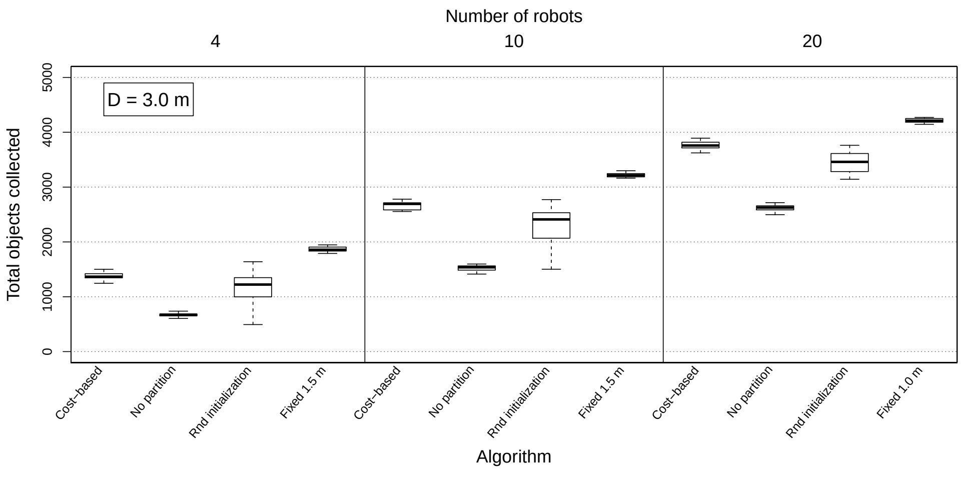
5.3. *Experimental set 3: Distance to the Source*

The goal of the set of experiments presented in this section is to study the effect of a different source-to-nest distance D (i.e., a different length of the overall task). The experiments are carried out in the huge environment ($W = 6.7$ m by $L = 10.0$ m), with $\sigma = \bar{\sigma}$. The value of the source-to-nest distance D is set to 3.0 m and 6.0 m. The values of the remaining experimental parameters are as reported in Table 1.

Figure 20 reports the performance of the cost-based partitioning algorithm and three reference algorithms for the different swarm sizes. The top plot displays the results for $D = 3.0$ m, the bottom plot for $D = 6.0$ m.

The performance obtained by a given algorithm for $D = 3.0$ m is higher than the corresponding performance for $D = 6.0$ m. As the objects must be transported for a longer distance, the throughput of the swarm is inferior. Additionally, in the case of the never partitioning algorithm, the longer distance between source and nest increases the probability that a robot gets lost, thus lowering the throughput. 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 20 shows that, for $D = 3.0$ m, the robots that use the cost-based partitioning algorithm perform well across all the values of noise and swarm sizes. This is not the case for $D = 6.0$ m: the performance of the cost-based partitioning algorithm is often close, or even inferior, to the performance of the random initialization algorithm. In the final part of the experiment the throughput of the cost-based partitioning algorithm is higher than the throughput of the random initialization algorithm. However, the throughput grows very slowly in time and this results in a



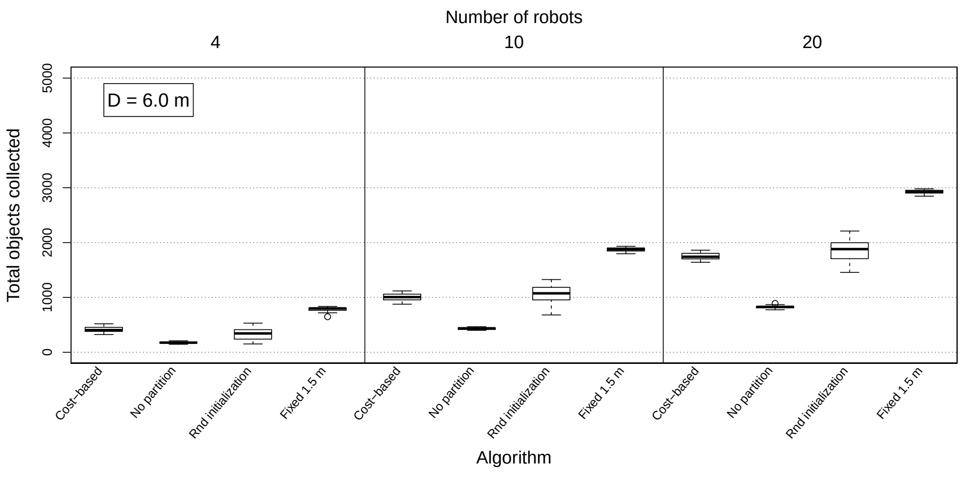


Fig. 20. Objects collected by the swarm when different partitioning algorithms are employed. Each group of bar reports data collected with a swarm of a given size (see top axis). The plot on top reports the data for $D = 3.0\,\text{m}$, the plot at the bottom for $D = 6.0\,\text{m}$.

low performance. The plot reporting the throughput of the cost-based partitioning algorithm is available with the online supplementary material [Pini et al., 2012b].

5.4. *Experimental set 4: Heterogeneity in the Robot Swarm*

The goal of the fourth experimental set is to assess whether heterogeneity in the swarm has an impact on the cost-based partitioning algorithm. I this study, we focus on heterogeneity impacting on the efficiency in performing a task. To this end, we act upon the odometry system of the robots: the accuracy of the odometry system varies from robot to robot. Therefore, different robots might be more or less successful in reaching the object source using odometry. Heterogeneity in the odometry system of the robots is a realistic assumption. In our previous work, we report results of experiments performed with real foot-bots in which we observed that the success rate in finding the objects source varies from robot to robot (refer to [Pini et al., 2012c] for more information).

Here, we simulate heterogeneity in the odometry system using different noise values across the swarm. We divide the swarm into two groups, one in which $\sigma = 0.2\bar{\sigma}$ (we will refer to robots in this group as *low-noise*), the other in which $\sigma = \bar{\sigma}$ (*default-noise* robots). We perform experiments using 10 robots, varying the percentage of low-noise robots in the swarm. We test two conditions: one in which there are 2 low-noise robots in the swarm and one in which the low-noise robots are 8. The remaining experimental parameters assume the values reported in Table 1.

Figure 21 reports the performance of the cost-based partitioning algorithm and of four reference algorithms. The plot on the left-hand side reports the case in which the low-noise robots are 2, the plot on the right-hand side the case in which the low-noise robots are 8. The results reported in Figure 21 indicate that the cost-based partitioning algorithm performs well in the tested conditions, showing that it is robust with respect to heterogeneity within the swarm.

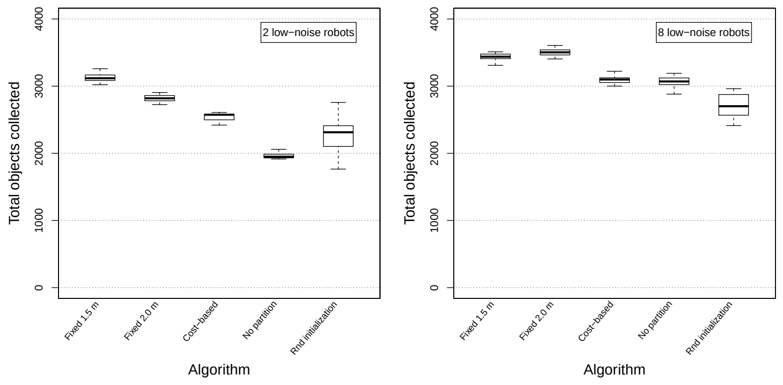


Fig. 21. Objects collected by the swarm when different partitioning algorithms are employed. The plot on the left-hand side reports the data for the case in which the low-noise robots are 2, the plot on right-hand side for the case in which their number is 8.

Figure 22 reports the partition length values selected by the robots employing the cost-based partitioning algorithm in the final quarter of experiment (last 5 hours). Each plot reports the percentage of times, computed across 20 experimental runs, each value L_i was selected by the robots. In the figure, plots on the same column refer to the same number of low-noise robots in the swarm: 2 robots in the plots on the left-hand side and 8 robots in the plots on the right-hand side. The top plots report the partition length values selected by the low-noise robots, the bottom plots the values selected by the default-noise robots.

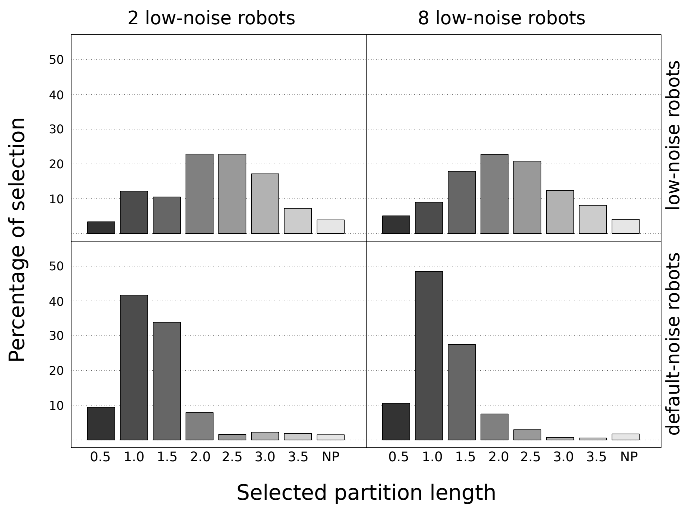


Fig. 22. Partition length values selected by the robots in time, when the cost-based partitioning algorithm is employed. Each plot reports the percentage of selection of each value L_i in the final quarter of experiment (last 5 hours). The plots in the first row report the values selected by the low-noise robots, the plots in the second row the values selected by the default-noise robots. The plots on the left-hand side report the data for the case in which the number of low-noise robots is two, the plots on the right-hand side the data for the case in which their number is eight.

The results reported in Figure 22 show that the partition lengths selected by the robots vary in relation to their odometry accuracy. The low-noise robots select higher partition length values compared to the values selected by the standard-noise robots. In other words, the way a robot partitions each task depends on the characteristics of the robot itself.

5.5. *Experimental set 5: Adaptivity to Variable Conditions*

In all the experiments presented so far, the cost-based partitioning algorithm reaches its best performance with a purely exploiting version of the ε-greedy algorithm (i.e., for $\varepsilon = 0$). This is due to the fact that, in all the experiments, we test static conditions. The only dynamics are introduced by the robots depositing objects in the environment. However, exploration may prove useful when properties of the environment vary in time. Given the way the ε-greedy works, 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 values that previously had the lowest cost associated.

Suppose, for example, that the environmental conditions initially favor low values of the partition length L_i and that the robots identified low values as a good choice. Suppose that a change occurs in the environment, that renders favorable higher values of the partition length. If the effect of the change is to reduce the cost associated to high values of partition length, without increasing the cost associated to low values, a purely exploiting version of the ε-greedy algorithm is unable to detect the change. In fact, as the robots constantly select low values of the partition length, initially identified as the best option, they never sample again high values and cannot detect that they are now preferable.

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 the objects. A longer perception diminishes the probability 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. We vary the perception range of the robots from the default value of 0.5 m to 1.2 m, when the experimental run reaches half-time. 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 cost-based partitioning algorithm, that vary with respect to the value of the parameter ε. The tested values for the parameter ε are $\{0, 0.3, 0.6\}$, corresponding to progressively increasing degrees of exploration in the algorithm. The noise is set to $\sigma = 0.5\,\bar{\sigma}$. We selected this value as it is the one that highlights the most the effects of exploration and exploitation on the cost-based partitioning algorithm.

Figure 23 reports the performance of the cost-based partitioning algorithm and of three reference algorithms for swarms composed 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 of boxes on the right-hand side of each plot reports the data collected in the second half of the experiment, after the perception range has been modified.

The results show that initially 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 the non-exploring ($\varepsilon = 0$). The other versions, which include some exploration, perform worse.

In the second half of experiment, the situation is different: the change in the perception range of the robots makes the never partitioning algorithm preferable to the fixed algorithm. However, for a swarm composed of 20 robots, the relative performance gain of the never partitioning algorithm over the fixed algorithm is limited, compared to the case of a swarm of ten robots. 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 the case of a swarm composed of ten robots, the cost-based partitioning algorithm with $\varepsilon = 0.3$ performs better that the non-exploring version. In the case of a swarm of twenty robots, the performance

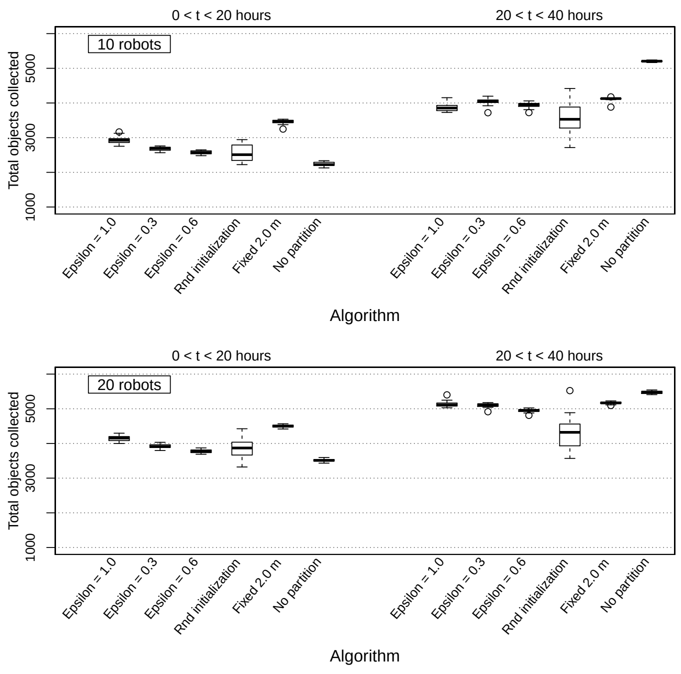


Fig. 23. Objects collected by the swarm when different partitioning algorithms are employed. In each plot, the first group of bars refers to the first half of experiment, the second group to the second half. The plot on top reports the results for a swarm of 10 robots, the plot at the bottom for a swarm of 20 robots.

reached by the two versions is comparable. However, exploration is beneficial up to a certain point: for $\varepsilon = 0.6$, performance is always inferior than for $\varepsilon = 0.3$ and therefore exploration and exploitation should be balanced carefully.

Figure 24 reports the values of partition length selected in time by a swarm of ten robots. Each row of plots corresponds to a value of ε. The plots in the first column report the frequency at which each partition length value was selected in the second quarter of the experiment (from $t = 10$ to $t = 20$ hours). The plot in the second column reports analogous data, collected in the last quarter of the experiment (from $t = 30$ to $t = 40$ hours). 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 the versions the cost-based partitioning algorithm mostly select values around 2.0 m. The effect of an increased exploration can be observed in the plots (from top to bottom): the peaks flatten and the selected values distribute more evenly. This behavior affects performance negatively, as shown in Figure 23.

The effect of exploration is also visible in the second column of plots reported in Figure 24. In case of $\varepsilon = 0.3$ and $\varepsilon = 0.6$, the peak moves from the value of 2.0 m to

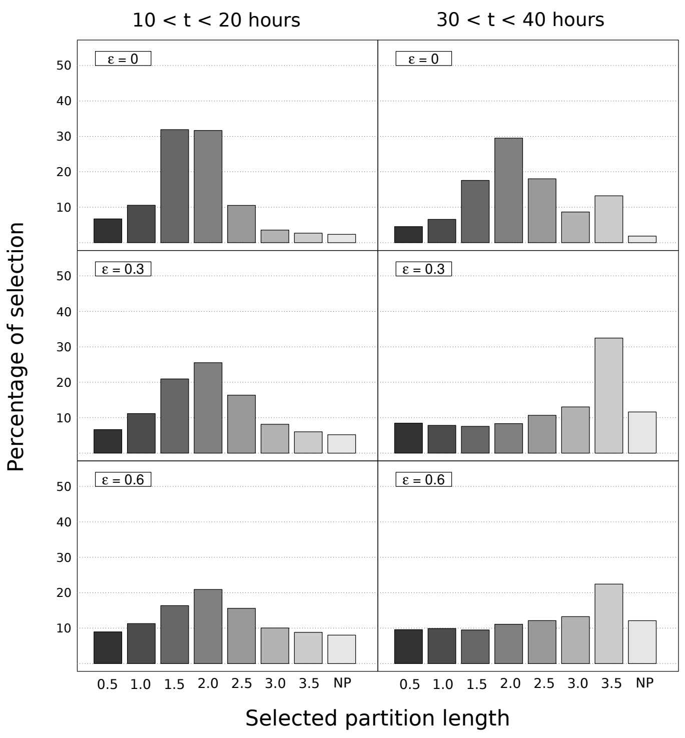


Fig. 24. Partition length values selected by the robots when different versions of the cost-based partitioning algorithm are employed. The plots in the first column report the percentage of selection of each value L_i in the second quarter of experiment. The plots in the second column analogous data collected in the final quarter. Each row of plots refer to a different version of the cost-based partitioning algorithm (increasing exploration from top to bottom).

the value of 3.5 m. This indicates that the cost-based partitioning algorithm detects the change occurring in the environment and reacts selecting a partitioning strategy more suited for the new conditions. The non-exploring version of the cost-based partitioning algorithm, on the other hand, continues to select values around 2.0 m. However the value 3.5 m is selected more often compared to the second quarter of experiment (Figure 24 top-left). This indicates that a form of exploration is present in the studied system independently of the value of ε, most likely due to stochasticity in the exploration time when the robots get lost, which introduces variations in the cost estimates. This stochasticity may not be present in other contexts and therefore explicit exploration is an option to consider, in general. The degree of exploration must be selected carefully, since in some cases exploration can hinder the performance of the system. For example, the plots for $\varepsilon = 0.6$ (Figure 24 bottom row) show that when exploration is too high, the behavior of the cost-based partitioning algorithm approaches a random behavior. A dominant choice can still be observed, but the cost-based partitioning algorithm does not exploit it efficiently and the performance is affected negatively.

6. Conclusion and Future Work

The capability of partitioning tasks into sub-tasks autonomously can result in an increased flexibility of swarm robotics systems: robots with such a capability can adapt the way tasks are performed to specific environmental conditions and goals to be attained. So far, the methods proposed in the literature to implement task partitioning are intertwined with the tasks the robots must perform. Therefore, each method can only be utilized in limited contexts and to partition specific tasks. Our long term research goal is developing a general task partitioning framework, that can be applied to a multitude of tasks and in different contexts.

In this paper, we present a research work that makes the first steps in this direction. We propose an approach that can be utilized to obtain autonomous task partitioning in a foraging scenario. The approach is based on a mapping between the amount of work a robot contributes with its sub-task and the resulting cost for performing the overall task. This mapping is implemented using a cost function, which is utilized by the robots to decide autonomously how to partition the object transportation task. The philosophy behind our approach is decoupling the process that implements task partitioning from the specific tasks performed by the robots. This decoupling is obtained through the generic concepts of cost and cost function. Costs can be used to measure the progress towards certain goals. For example, if the task is the production of items, costs can be measured as quantity of material required for producing an item, when the goal is minimizing material waste. If, on the other hand, a delivery deadline must be met, the production can be carried out at full speed regardless of material waste, and costs can be measured as time. Since the task partitioning process depends on costs, it is implicitly driven towards the goals to be attained. Moreover, if the goals or the impact of the environment on costs change, the way tasks are partitioned changes automatically in response.

The way costs are expressed can be adapted to the specific context and goals without modifying the underlying process that implements task partitioning. Therefore, we are confident that our approach can directly be applied to a multitude of contexts, by adapting the definition of the cost function to the specific case. In general, the cost function depends on (possibly unknown) environmental conditions and it can vary over time, therefore it cannot be given a priori to the robots. Instead, it must be modeled by each robot on the basis of measures taken in the environment. Consequently, our approach requires the robots to be capable of taking such measures and utilize them to estimate costs.

In this paper, we show the application of our approach to the proposed foraging scenario. The goal of the robots is to collect objects from a location of the environment, the source, and transport them to a nest location. The robots use odometry to maintain an estimate of the location of the source. In simulation-based experiments we show that task partitioning can be more or less beneficial depending on the entity of the noise on the odometry system. We compare an algorithm based on our approach to a set of reference algorithms in different contexts. We show that, using our approach, the swarm performs well across the tested conditions. The partitioning strategy employed by the swarm varies in relation to different factors: the accuracy of the odometry, the number of robots in the swarm, and the size of the environment in which they operate. The robots do not take direct measures to estimate any of these factors, yet the strategy employed to perform foraging is linked to such factors. This result indicates that our approach can be utilized without making a priori assumption about the phenomena determining the best way of utilizing task partitioning.

We perform experiments showing that the approach is robust with respect to heterogeneity within the swarm. In our experiments, heterogeneity consists in a variable accuracy of the odometry system across different robots. The results show that the partitioning strategy varies from robot to robot, in relation to the accuracy of the odometry. Therefore the approach exploits differences between the robots of the swarm, which can result in performance gains in many contexts.

We also study the trade-off between exploration ad exploitation in a setup in which the environmental conditions change in time. The results show that a certain degree of exploration is present in the system at hand, as a consequence of stochastic events happening in our scenario. However we also remark that built-in exploration can be beneficial and therefore it should be taken into account when the environmental conditions are not static.

Short term research aims to verify the applicability of the approach to other tasks than transportation, for example patrolling or construction. A long term research goal is integrating explicit communication in the studied system. We remarked that the swarm is relatively slow in converging to a certain partitioning strategy. Communication could improve this aspect of the system. For example, the robots could directly exchange information and integrate their cost estimates with the data received from other robots. This would reduce the negative impact of cost estimation errors made by the robots. Additionally communication would allow the decisions pertaining the length of the sub-tasks to be taken collectively, instead of individually, with the result of an increased coordination between robots and consequently a more responsive collective behavior.

Another direction of future work is verifying the applicability of the approach to cases in which the robots must perform heterogeneous tasks. In this work, the robots repeatedly perform the same task: transporting objects from the source to the nest. The tasks are homogeneous since their length and characteristics (e.g., odometry noise in our setup) do not vary from task to task. In general cases however, a swarm of robots might have to carry out different types of tasks at the same time. For example in the studied foraging scenario there could be multiple objects sources. Different sources correspond to tasks of a different type: the length differ from one to another. Characteristics of the environment may vary as well, for example different terrain configurations can affect the odometry estimates in a different ways. For these reasons, the best partitioning strategy varies depending on the type of task. Our approach can be applied to such situations with minor modifications: a cost function must be assigned to each task type. The main implementation issue is that the robots must be capable of discriminating tasks of different types. In our setup this can be done, for example, on the basis of the odometry estimates of the robots.

Acknowledgements.

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

References

Anderson, C. and 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., and Mondada, F. (2010). The MarXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, pages 4187–4193. IEEE Press, Piscataway, NJ.

Bovet, D. P. and Cesati, M. (2005). *Understanding the Linux Kernel.* O'Reilly Media.

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

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms, Second Edition.* MIT Press, Cambridge, MA.

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

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

Ennals, R., Sharp, R., and Mycroft, A. (2005). Task partitioning for multi-core network processors. In *Compiler Construction*, volume 3443 of *Lecture Notes in Computer Science*, pages 76–90. Springer, Berlin/Heidelberg, Germany.

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

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

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

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

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

Hart, A. G., Francis, L. W., and 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. and 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.

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

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

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

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

Parker, C. A. C. and Zhang, H. (2010). Collective unary decision-making by decentralized multiple-robot systems applied to the task-sequencing problem. *Swarm Intelligence*, 4(3):199–220.

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

Pini, G., Brutschy, A., Birattari, M., and Dorigo, M. (2009). Task partitioning in swarms of robots: reducing performance losses due to interference at shared resources. In Cetto, J. A., Filipe, J., and Ferrier, J.-L., editors, *Informatics in Control, Automation and*

Robotics, volume 85 of *LNEE*, pages 217–228. Springer, Berlin/Heidelberg, Germany.

Pini, G., Brutschy, A., Francesca, G., Dorigo, M., and Birattari, M. (2012a). Multi-armed bandit formulation of the task partitioning problem in swarm robotics. In Dorigo, M., Birattari, M., Di Caro, G. A., Doursat, R., Engelbrecht, A. P., Floreano, D., Gambardella, L. M., Groß, R., Sahin, E., Sayama, H., and Stützle, T., editors, *Swarm Intelligence, 8th International Conference, ANTS 2012*, volume 6234 of *LNCS*, pages 287–298. Springer, Berlin/Heidelberg, Germany.

Pini, G., Brutschy, A., Frison, M., Roli, A., Dorigo, M., and 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., and Birattari, M. (2012b). Autonomous task partitioning in swarms of robots: a methodology based on cost estimation – Online supplementary material. `http://iridia.ulb.ac.be/supp/IridiaSupp2012-0014/`.

Pini, G., Brutschy, A., Scheidler, A., Dorigo, M., and Birattari, M. (2012c). Task partitioning in a robot swarm: Retrieving objects by transferring them directly between sequential sub-tasks. Technical Report TR/IRIDIA/2012-010, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

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

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

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

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

Sutton, R. and Barto, A. (1998). *Reinforcement Learning, an Introduction.* MIT Press, Cambridge, MA.

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