Self-organized Task Partitioning in a Swarm of Robots

Marco Frison, Nam-Luc Tran, Nadir Baiboun, Arne Brutschy, Giovanni Pini, Andrea Roli, Marco Dorigo, and Mauro Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2010-016

31 May 2010

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/2010-016

Self-organized Task Partitioning in a Swarm of Robots

Marco Frison[1,2], Nam-Luc Tran[1], Nadir Baiboun[1,3], Arne Brutschy[1], Giovanni Pini[1], Andrea Roli[2,1], Marco Dorigo[1], and Mauro Birattari[1]

[1] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
[2] Universitá di Bologna, Bologna, Italy
[3] ECAM, Institut Supérieur Industriel, Brussels, Belgium
mfrison85@gmail.com, namltran@ulb.ac.be, nadir_ecam@hotmail.com, arne.brutschy@ulb.ac.be, gpini@ulb.ac.be, andrea.roli@unibo.it, mdorigo@ulb.ac.be, mbiro@ulb.ac.be

Abstract. In this work, we propose a method for self-organized adaptive task partitioning in a swarm of robots. Task partitioning refers to the decomposition of a task into less complex subtasks, which can then be tackled separately. Task partitioning can be observed in many species of social animals, where it provides several benefits for the group. Self-organized task partitioning in artificial swarm systems is currently not widely studied, although it has clear advantages in large groups. We propose a fully decentralized adaptive method that allows a swarm of robots to autonomously decide whether to partition a task into two sequential subtasks or not. The method is tested on a simulated foraging problem. We study the method's performance in two different environments. In one environment the performance of the system is optimal when the foraging task is partitioned, in the other case when it is not. We show that by employing the method proposed in this paper, a swarm of autonomous robots can reach optimal performance in both environments.

1 Introduction

Many animal species are able to partition complex tasks into simpler subtasks. The act of dividing a task into simpler subtasks that can be tackled by different workers is usually referred to as *task partitioning* [1].

Although task partitioning may have associated costs, for example because of work transfer between subtasks, there are many situations in which partitioning is advantageous. Benefits of task partitioning include, for example, a reduction of interference between individuals, an improved exploitation of the heterogeneity of the individuals, and an improved transport efficiency [2].

Humans widely exploit the advantages of task partitioning in everyday activities. Through centuries, humans have developed complex social rules to achieve cooperation. These include planning, roles and work-flows. Ancient romans realized the importance of partitioning and they codified it in their military principle *divide et impera* (also known as divide and conquer), which became an axiom in many political [3] and sociological theories [4].

Examples of task partitioning can also be observed in social insects. A widely studied case is the foraging task in ants and bees. In foraging, a group of individuals has to collect and transport material to their nest. Foraging involves many different phases and partitioning can occur simultaneously in many of them. Typical phases where task partitioning can occur are the exploration of the environment and the preparation of raw materials [1]. Examples of task partitioning are the harvesting of leaves by the leaf-cutter ants [2], the excavation of nest chambers in *Pogomomyrmex*, and the fungus garden removal in *Atta* [5].

Also in swarm robotics there are situations in which it is convenient to partition a task into subtasks. Advantages include increased performance at group level, stimulated specialization, and parallel task execution. In most of the cases, task partitioning is done a priori and the focus is on the problem of allocating individuals to subtasks in a way that maximizes efficiency. However, in many cases, task partitioning cannot be done a priori because the relevant information on the environment is not available. We consider self-organized task partitioning as a suitable approach in these cases.

In this work, we focus on the case in which a task is partitioned into subtasks that are sequentially interdependent. We propose a simple method, based on individuals' perception and decisions, that allows a swarm of autonomous robots to decide whether to partition a foraging task into subtasks. We test the method with a swarm of simulated robots in two different environmental conditions.

The rest of the paper is organized as follows. In Section 2 we describe the problem and we review related works. In Section 3 we propose an adaptive method that we tested with simulated robots. In Section 4 we provide a description of the experimental framework we consider. In Section 5 we report and discuss the results. Finally, in Section 6 we summarize the contribution of this work and present some directions for future research.

2 Problem Description and Related Works

We study a swarm of robots that has to autonomously decide whether to partition a task into subtasks. Our focus is on situations in which a task is partitioned into sequential subtasks: the subtasks have to be executed in a certain sequence, in order to complete the global task once [6].

In these cases, we can identify *tasks interfaces* where one task ends and another begins. Through tasks interfaces, individuals working on one of the subtasks can interact, either directly or indirectly, with individuals working on other subtasks.

An example of sequential task partitioning, observable in nature, is the foraging activity in *Atta* leaf cutting ants. The sequential interdependency between tasks stems from the fact that each leaf has to be cut from a tree before it can be transported to the nest. Each individual can choose whether to perform both the cutting and transporting subtasks, or to specialize in one subtask only.

Hart at al. [2] described the strategy employed by *Atta* ants: some individuals work on the tree, cutting and dropping leaves to the ground, while the rest of the swarm gathers and transports these leaves to the nest. Here the advantage of partitioning comes from the fact that the energy cost to climb the tree has to be paid only once by those individuals working as leaf cutters. Disadvantages come from the fact that energy has to be spent to search for leaves on the ground. The task interface can be identified, in this case, as the area where the leaves land. Such areas are usually referred to as *caches*, and facilitate *indirect* transfer of material between individuals. Anderson and Ratnieks described how foragers of different ant species partition the leaf transport task by using caches [7].

Partitioning along foraging trails using *direct* transfer of material between individuals can be observed in other ant species and in other social insects. In the case of direct transfer, the benefit of partitioning can come from the fact that material weight can be matched with the strength of the transporter [8]. Akre et al. observed task partitioning within *Vespula* nectar foraging [9]. Anderson and Ratnieks studied partitioned foraging in honeybees species, showing that the larger the swarm, the higher the performance [10].

Robotic swarms often face situations similar to those of their natural counterparts. However, despite its importance, few works have been devoted to task partitioning in swarm robotics. Notable exceptions are the works of Fontan and Matarić as well as Shell and Matarić on *bucket-brigading* [11,12]. In these works, robots have to gather objects in an environment. Each robot operates in a limited area and drops the object it carries when it reaches the boundaries of its working area. This process leads to objects being passed across working areas until they reach the nest. Lein and Vaughan proposed an extension to this work, in which the size of the robots' working areas is adapted dynamically [13]. Pini et al. showed that the loss of performance due to interference, can be reduced by partitioning the global task into subtasks [14]. To the best of our knowledge, self-organized task partitioning in terms of adaptive task decomposition has never been investigated.

3 The Method

The method we propose allows a swarm of robots to adaptively decide whether to partition a task into sequential subtasks or not. A decision is made by each individual: In case a robot decides to employ task partitioning, it works only on one of the subtasks. In case a robot decides not to employ task partitioning, it performs the whole sequence of subtasks. The method is fully distributed and does not require explicit communication between individuals. The swarm organizes in a way that maximizes the overall efficiency of the group, regardless of the specific environment. Efficiency is defined as the throughput of the system.

In the method proposed, each individual infers whether task partitioning is advantageous or not on the basis of its waiting time at tasks interfaces. We define the probability p that a robot has to employ task partitioning as:

$$p = 1 - \frac{1}{1 + e^{-\theta(w(k))}} \ , \tag{1}$$

with θ being:

$$\theta\big(w(k)\big) = \frac{w(k)}{s} - d \ , \tag{2}$$

and $w(k)$ being the weighted average waiting time at task interfaces after k visits, which is calculated as follows:

$$w(k) = (1-\alpha)w(k-1) + \alpha w_M \ . \tag{3}$$

In Equation 2, s and d are a scale and a delay factor, respectively. In Equation 3, $\alpha \in (0, 1]$, is a weight factor that influences the responsiveness to changes: higher values lead to a readily responsive behavior. The value of these parameters can be determined empirically. The variable w_M is the *measured waiting time* at task interface and ranges in $[0, w_{MAX})$. The upper limit w_{MAX} ensures that robots eventually renounce to employ task partitioning when their waiting time becomes too high. Each time a robot completes a subtask, it decides whether to employ task partitioning for the next task execution, or not.

4 Experimental Setup

The purpose of the experiments described in this section is to show the validity of the method described in Section 3. To illustrate the approach we have chosen a foraging problem as a testbed. It is a canonical choice in collective robotics as it is easy to model and its applications are numerous [15].

In the experiments, the global task can be described as harvesting objects from a single source and storing them in the nest. The global task can be partitioned into two subtasks, referred to as *harvesting* and *storing*, respectively. Partitioning enables the subdivision of the environment into two areas, linked by a task interface as defined in Section 2. The task interface is represented by a cache that can be used by the robots to exchange objects. As the cache has a limited capacity, robots that decide to use it may have to wait. The waiting time is defined as the delay between the moment when a robot decides to use the cache, either for picking up or dropping objects, and the moment when this effectively becomes possible. It is also possible to avoid the cache by using a separate corridor, which links directly the source and the nest.

Each robot has to autonomously decide whether to partition the foraging task, by using the cache; or not to partition it, by using the corridor. The swarm can also employ a mixed strategy in which some individuals use the cache and others use the corridor. Robots have no notion of the global performance of the swarm. In no case explicit communication is used. Figure 1 shows a simplified state diagram that represents the behavior of each individual.

4.1 Simulation Tools

All the results presented in the paper are obtained using the ARGoS simulation framework [16]. ARGoS is a discrete-time, physics-based simulation environment

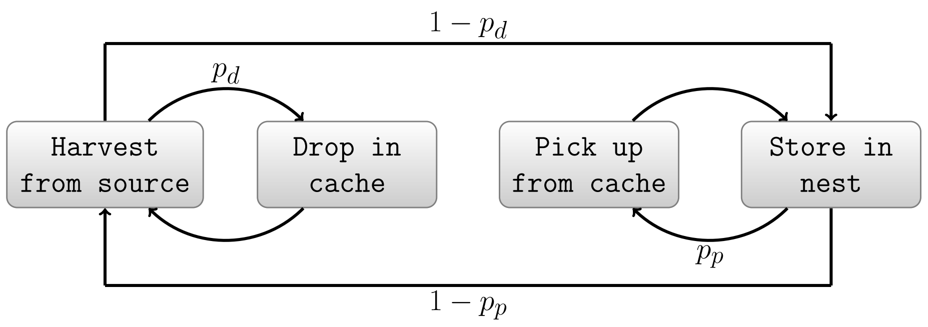

Fig. 1: Simplified state machine representing the behavior of each individual. Probabilities p_d and p_p are both defined using Equation 1 as described in Section 3. The variable p_d represents the probability of using the cache to drop an object. The variable p_p represents the probability of picking up an object from the cache. The states Avoid obstacles and Navigate have been omitted for clarity.

that allows one to simulate experiments at different levels of detail. For the experiments presented in this paper, it is sufficient to simulate kinematics in a bi-dimensional space. A common control interface provides transparent access to real and simulated hardware, allowing the same controller to run also on the real robots without modifications.

The robots we use in this research are the e-pucks[4]. The e-puck has been designed with the purpose of being both a research and an educational tool for universities [17]. ARGoS simulates the whole set of sensors and actuators of the e-puck. In our experiments we use the wheel actuators, the 8 IR sensors for light and proximity detection, the VGA camera, and the ground sensors.

4.2 Harvesting Abstraction

As the e-pucks are not capable of grasping objects, we developed a device to simulate this process [18]. Figure 2 shows a schematic representation of the device, called an *array*[5]. It consists of a variable number of slots, named *booths*. Each booth is equipped with two RGB LEDs that can be detected by the robots through their color camera. A light barrier can detect the presence of a robot within each booth. Reactions to this event, such as changes in LEDs color, are programmable.

In the experiments presented in the paper, a green booth, either in the source or in the cache, means that an object is available there. Analogously, a blue booth means that an object can be dropped there. By using this abstraction, when a robot enters a booth in which the LEDs are lit up in green, we consider that the robot picks up an object from that booth. When a robot enters a booth in

[4] http://www.e-puck.org/
[5] The array is under development and a working prototype is currently available.

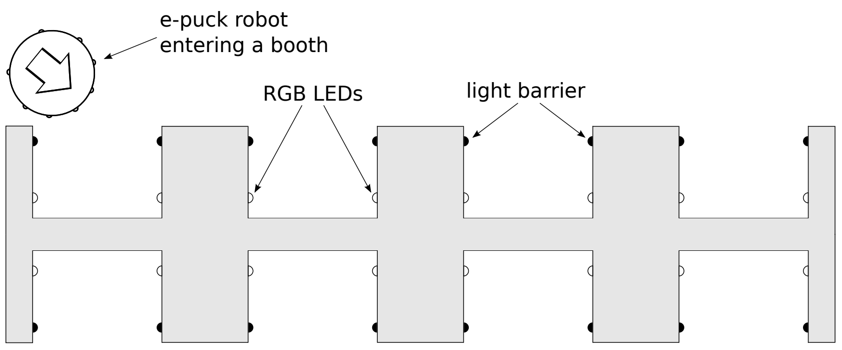

Fig. 2: Schematics of an array of booths with four booths on each side. The light barriers, represented by black semicircles, detect when a robot enters in the corresponding booth. LEDs, used to signal available pick up or drop sites, are represented by blank semicircles.

which the LEDs are lit up in blue, we consider that the robot drops an object in that booth. In both cases, when the booth perceives the presence of the robot, it reacts by turning the LEDs red, until the robot has left. Once the robot has left, the booths behave in a different way, depending whether they are source, nest or cache booths. In the case of the source, the booth turns green, to signal the availability of a new object to harvest. In the case of the nest, the booth turns blue, to signal that the corresponding store spot is available again. In case of the cache, if the robot leaves after picking up an object, the booth, previously green, turns off and the corresponding booth on the other side turns blue signaling that the spot is now available again for dropping an object. Conversely, if the robot leaves after dropping an object, the booth, previously blue, turns off and the corresponding booth on the other side turns green signaling that an object is available for being picked up. This simple logic allows us to simulate object transfer through the cache, as well as harvest from the source and store in the nest.

4.3 Environments

We run the experiments in two different environments, named *short-corridor* environment and *long-corridor* environment (see Figures 3a and 3b).

In both the environments, the nest array is located on the right-hand side, while the source array is located on the left-hand side of a rectangular arena. Both the nest and the source arrays have four booths, all on one side. The cache array is located between the nest and the source and has three booths on each side. Therefore, the cache has a limited capacity, which is determined by the number of booths on each side. Different ground colors allow the robots to recognize on which side of the cache they are.

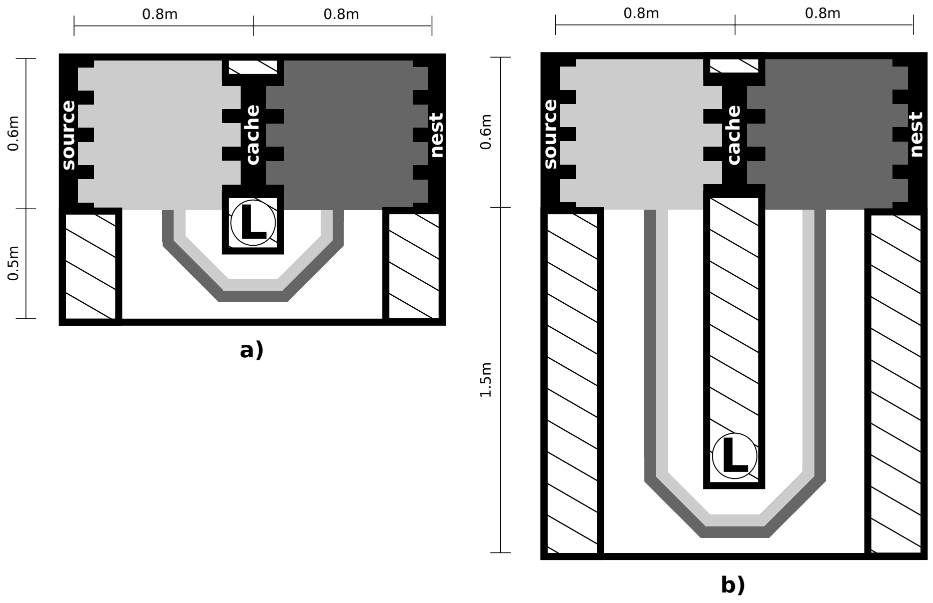

Fig. 3: Representation of the **a)** *short-corridor* and the **b)** *long-corridor* environments used in the experiments. Nest and source arrays have both four booths on one side. The cache array has three booths for each side. Different ground colors help the robots to distinguish between different parts of the environment and to navigate through the corridor that connects the two areas. The light source, used as landmark for navigating in the corridor, is marked with "L".

Although the cache array cannot be crossed by robots, a corridor links the two areas and allows the robots to harvest/store objects without using the cache array (i.e., without partitioning the foraging task). A light source and two colored trails help the robots to navigate through the corridor. Both the environments are 1.6 m wide, but they differ in the corridor's length: in the short-corridor environment the corridor is 1.5 m long, while in the long-corridor environment it is 3.5 m long. Both the use of the cache and the use of the corridor entail costs. The cache can be seen as a shortcut between source and nest: robots cannot cross the cache, but its use can make material transfer faster. However, the cache can also become a bottleneck as the decision of using it can lead to delays if it is busy when dropping objects, or empty when picking them up. The decision of using the corridor imposes a cost due to the time spent traveling through it. Thus, the transfer cost varies with cache and group size while the travel cost varies with corridor length. As we keep the size of the cache array constant in our experiments, the corridor length determines the relative cost between partitioning and not partitioning. In the long-corridor environment, the use of the cache is expected to be preferable. On the other hand, in the short-corridor environment, the cost imposed by the corridor length is low and should lead to the decision not to use the cache.

4.4 Experimental Settings

We run two different sets of experiments: in a first set of experiments we are interested in assessing the performance of the adaptive method that we propose, while in the second set of experiments we aim at evaluating its scalability. In the first set of experiments the robotic swarm is composed of twelve e-pucks, each controlled by the same finite state machine depicted in Figure 1. In both the environments, we compare the adaptive method described in Section 3 to two strategies which always partition ($p_d = p_p = 1$) or never partition ($p_d = p_p = 0$). These are used as reference strategies for evaluating both the performance of the proposed method and its capability of adapting to different environmental conditions. The values of the parameters have been determined empirically and fixed to $s = 20$, $d = 5$, $\alpha = 0.75$, $w_{MAX} = 15\,\text{s}$. The duration of each experimental run is 150 simulated minutes. For each experimental condition we run 30 repetitions, varying the seed of the random numbers generator. At the beginning of each experiment the robots are randomly positioned in the right side of the environment, where the nest array is located. As the average waiting time $w(k)$ is initially equal to zero, all the robots start with probabilities $p_d = p_p \approx 1$: from Equations 1 and 2, with $d = 5$; when $\theta\big(w(k)\big) = 0$, $p = 0.993$.

In the second set of experiments we compare the adaptive method to the reference strategies in the short-corridor environment. In this case the size of the swarm varies in the interval $[4, 60]$. For each condition we run 10 randomly seeded experiments. The remaining parameters of the experiment are the same as described for the first set of experiments.

5 Results and Discussion

As we keep the capacity of the cache array constant, it is the length of the corridor that determines which behavior maximizes the throughput of the system. Partitioning allows the robots to avoid the corridor but, in order to exploit efficiently the cache, the swarm has to organize and to work on both of its sides. Additionally, as pointed out in Section 4, the robots might have to wait in order to use the cache.

In the short-corridor environment, the cost of using the cache is higher than the cost of using the corridor. In this case the robots should decide for a non-partitioning strategy, without exchanging objects at the cache. Conversely, in the long-corridor environment the time required to travel along the corridor is high, and partitioning the task by using the cache is expected to be more efficient.

The graphs in Figure 4 show the average throughput of a swarm of twelve robots in the two environments. Throughput is measured as the number of objects retrieved per minute. The adaptive method is compared with the two reference strategies in which the robots never/always use the cache. As expected,

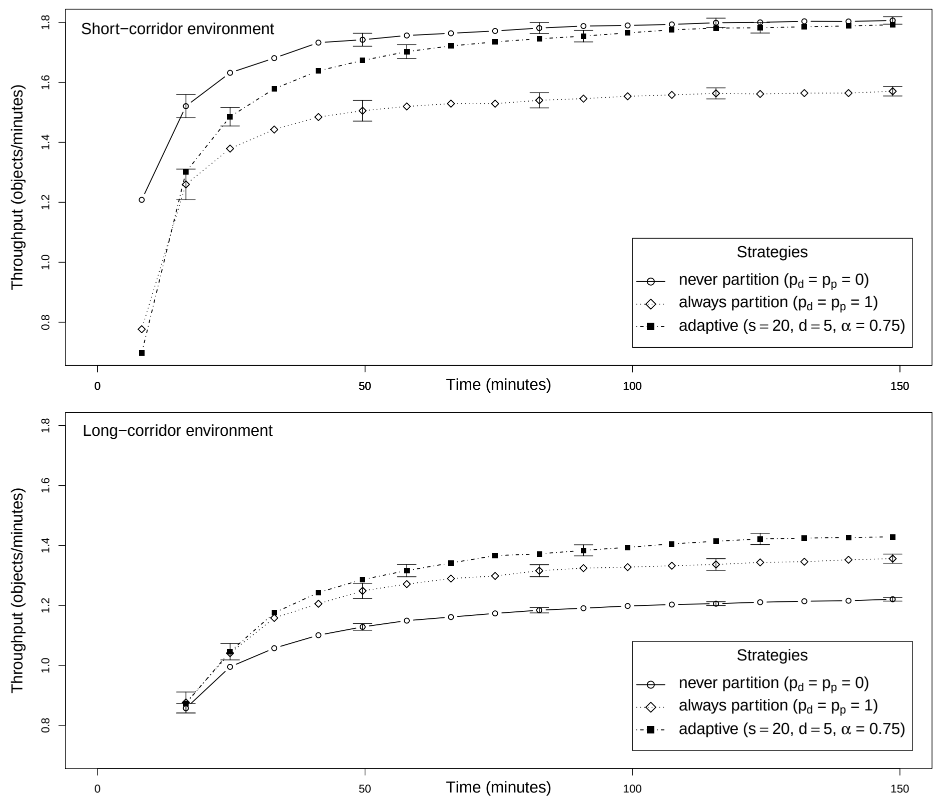

Fig. 4: Average throughput in the short-corridor (top) and long-corridor (bottom) environment for different strategies. Time is given in simulated minutes. Throughput is measured as objects retrieved per minute. Parameters for the adaptive behavior are set to $s = 20$, $d = 5$, $\alpha = 0.75$, $w_{MAX} = 15\,\text{s}$. Parameter values have been obtained empirically. Each experiment has been repeated 30 times, varying the seed of the random number generator. The bars around the single data points represent the confidence interval of 95% on the observed mean.

each of these reference strategies performs well only in one environment: the strategy that never uses the cache performs better in the short-corridor environment, while the strategy that always use the cache performs better in the long-corridor environment. On the other hand, the adaptive method we propose shows good performance in both environments.

Concerning the long-corridor environment, we assumed that the best strategy was to always use the cache and to avoid the corridor. However, Figure 4(bottom) shows that the adaptive method proposed improves over this fixed strategy. To better understand this behavior, we empirically determined the fixed-probability strategy yielding the highest throughput for each environment. This analysis revealed that the best strategy in the long-corridor environment is to use the

Table 1: Average throughput at the end of the experiment for a swarm of 12 robots. Results are reported for different partitioning strategies in the short-corridor and long-corridor environments. The values in parenthesis indicate the 95% confidence interval on the value of the mean.

	Never partition	Always partition	Fixed ($p_d, p_p = 0.8$)	Adaptive
Short-corridor	1.81 (±0.012)	1.57 (±0.015)	1.67 (±0.016)	1.79 (±0.017)
Long-corridor	1.22 (±0.006)	1.36 (±0.015)	1.40 (±0.013)	1.43 (±0.017)

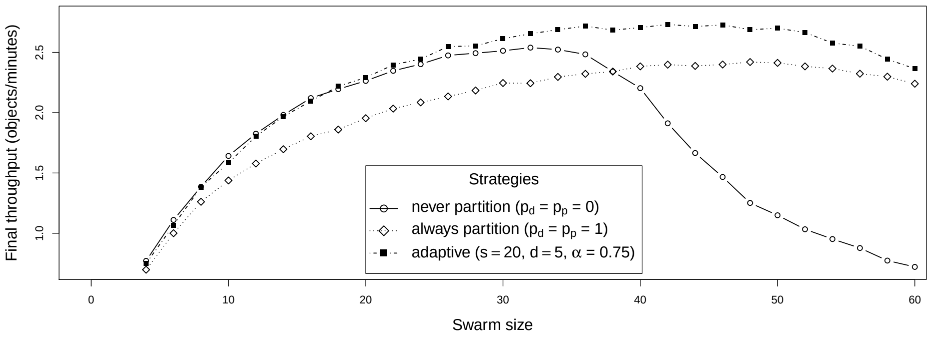

Fig. 5: Impact of the swarm size in the short-corridor environment. The value of the throughput, measured as number of objects retrieved per minute, is the average value reached by the swarm at the end of the experiment. Each experimental run lasts 150 simulated minutes. For each experimental condition we run 10 repetitions, varying the seed of the random number generator.

cache with a probability around 80%. In the short corridor environment a non-partitioning strategy is preferable.

Table 1 reports the average throughput obtained at the end of the run for each strategy in each environment. The results reported in the table show that fixed-probability strategies perform well only in one of the two environments. Our adaptive method, on the other hand, reaches good performance in both the short-corridor and the long-corridor environment. These results confirm that the method we propose allows a swarm of robots to take a decision concerning whether to partition a task into sequential subtasks or not.

The graph in Figure 5 shows the results of the second set of experiments, in which we focus on scalability. In this case we compare different strategies for different swarm sizes in the short-corridor environment. As discussed previously, in the short-corridor environment the optimal strategy is to always use the corridor. It can be observed that for small swarm sizes this strategy performs well. However, the performance of this strategy drops drastically when the number of robots increases. The reason for this degradation is the increasing interference between the robots, which increases the cost of using the corridor. The partitioned strategy does not suffer from steep drops of performance. However, the throughput is considerably lower for smaller group sizes, as the waiting time at the cache becomes dominant. The adaptive method performs well across all the studied swarm sizes, finding a good balance between the robots that use the cache and those that use the corridor.

6 Conclusions

In this research we investigated the self-organized task partitioning problem in a swarm of robots. In particular we have proposed an adaptive method to tackle the task partitioning problem with a simple strategy based on individual's perception of each subtask performance. In the method proposed, the subtask performance is estimated by each robot by measuring its waiting time at task interfaces. Results show that the adaptive method we propose reaches the best performance in the two environments we considered, employing task partitioning only in those cases in which the benefits of partitioning overcome its costs. The study of the impact of the group size reveals that the method scales well with the swarm size. Future work will concern the study of self-organized task partitioning in multi-foraging problems in environments with several caches. In addition, we are interested in studying the application of the method proposed to cases in which partitioning happens through direct material transfer.

Acknowledgements. This work was partially supported by the virtual Swarmanoid project funded by the Fund for Scientific Research F.R.S.–FNRS of Belgium's French Community. Marco Dorigo and Mauro Birattari acknowledge support from the F.R.S.–FNRS, of which they are a research director and a research associate, respectively. Marco Frison acknowledges support from "Seconda Facoltà di Ingegneria", Alma Mater Studiorum, Università di Bologna. Andrea Roli acknowledges support from the "Brains (Back) to Brussels" 2009 programme, founded by IRSIB – Institut d'encouragement de la Recherche Scientifique et de l'Innovation de Bruxelles.

References

1. Ratnieks, F.L.W., Anderson, C.: Task partitioning in insect societies. Insectes Sociaux **46**(2) (1999) 95–108
2. Hart, A.G., Anderson, C., Ratnieks, F.L.W.: Task partitioning in leafcutting ants. Acta Ethologica **5** (2002) 1–11
3. Traiano, B.: La Bilancia Politica di Tutte le Opere di Traiano Boccalini, Part 2-3. Kessinger Publishing, Whitefish, Montana, USA (1678)
4. Kant, E.: Perpetual Peace: A Philosophical Sketch. Hackett Publishing Company, Indianapolis, Indiana, USA (1795)
5. Wagner, D., Brown, M.J.F., Broun, P., Cuevas, W., Moses, L.E., Chao, D.L., Gordon, D.M.: Task-related differences in the cuticular hydrocarbon composition of harvester ants, *Pogonomyrmex barbatus.* J Chem Ecol **24** (1998) 2021–2037

6. Brutschy, A.: Task allocation in swarm robotics. Towards a method for self-organized allocation to complex tasks. Technical Report TR/IRIDIA/2009-007, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2009)
7. Anderson, C., Ratnieks, F.L.W.: Task partitioning in insect societies: novel situations. Insectes Sociaux **47** (2000) 198–199
8. Anderson, C., Jadin, J.L.V.: The adaptive benefit of leaf transfer in *Atta colombica*. Insectes Sociaux **48** (2001) 404– 405
9. Akre, R.D., Garnett, W.B., MacDonald, J.F., Greene, A., Landolt, P.: Behavior and colony development of *Vespula pensylvanica* and *Vespula atropilosa* (hymenoptera: Vespidae). Journal of the Kansas Entomological Society **49** (1976) 63–84
10. Anderson, C., Ratnieks, F.L.W.: Task Partitioning in Insect Societies. I. Effect of colony size on queueing delay and colony ergonomic efficiency. The American naturalist **154**(5) (1999) 521–535
11. Fontan, M.S., Matarić, M.J.: A study of territoriality: The role of critical mass in adaptive task division. In: From Animals to Animats 4: Proceedings of the Fourth International Conference of Simulation of Adaptive Behavior, MIT Press, Cambridge, MA (1996) 553–561
12. Shell, D.J., Matarić, M.J.: On foraging strategies for large-scale multi-robot systems. In: Proceedings of the 19th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2006) 2717–2723
13. Lein, A., Vaughan, R.: Adaptive multi-robot bucket brigade foraging. In Bullock, S., Noble, J., Watson, R., Bedau, M.A., eds.: Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, MIT Press, Cambridge, MA (2008) 337–342
14. Pini, G., Brutschy, A., Birattari, M., Dorigo, M.: Interference reduction through task partitioning in a robotic swarm. In Filipe, J., Andrade-Cetto, J., Ferrier, J.L., eds.: Sixth International Conference on Informatics in Control, Automation and Robotics – ICINCO 2009. INSTICC Press, Setùbal, Portugal (2009) 52–59
15. Dorigo, M., Sahin, E.: Guest editorial. Special issue: Swarm robotics. Autonomous Robots **17**(2–3) (2004) 111–113
16. Pinciroli, C.: Object retrieval by a swarm of ground based robots driven by aerial robots. Technical Report TR/IRIDIA/2007-025, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2007)
17. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, Portugal, IPCB: Instituto Politècnico de Castelo Branco (2009) 59–65
18. Brutschy, A., Pini, G., Baiboun, N., Decugnière, A., Birattari, M.: The IRIDIA TAM: A device for task abstraction for the e-puck robot. Technical Report TR/IRIDIA/2010-015, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2010)