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

Adaptive Foraging for Simulated and Real Robotic Swarms: The dynamical response threshold approach

Eduardo Castello · Tomoyuki Yamamoto · Fabio Dalla Libera · Wenguo Liu · Alan F. T. Winfield · Yutaka Nakamura · Hiroshi Ishiguro

Received: date / Accepted: date

Abstract Developing self-organized swarm systems capable of adapting to environmental changes as well as to dynamic situations is a complex challenge. An efficient labour division model, with the ability to regulate the distribution of work among swarm robots, is an important element of this kind of system. This paper extends the popular Response Threshold Model (RTM) and proposes a new Adaptive Response Threshold Model (ARTM). Experiments were carried out in simulation and in real-robot scenarios with the aim of studying the performance of this new adaptive model. Results presented in this paper verify that the extended approach improves on the adaptability of previous systems. For example, by reducing collision duration among robots in foraging missions, our approach helps small swarms of robots to adapt more efficiently to changing environments, thus increasing their self-sustainability (survival rate). Finally, we propose a minimal version of ARTM, which is derived from the conclusions obtained through real-robot and simulation results.

Keywords Adaptive Foraging, Cooperative Behaviour, Autonomous Systems

Eduardo Castello, Tomoyuki Yamamoto, Fabio Dalla Libera, Yutaka Nakamura and Hiroshi Ishiguro
Graduate School of Engineering Science
Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka, 560-8531, Japan.
Tel & Fax.: +81-6-6850-6360
E-mail: {eduardo.castello,yamamoto,fabio.dl}@irl.sys.es.osaka-u.ac.jp
E-mail: nakamura@is.sys.es.osaka-u.ac.jp
E-mail: ishiguro@sys.es.osaka-u.ac.jp

Tomoyuki Yamamoto is also with
CiNet, National Institute of Information and Communications Technology,
1-4 Yamadaoka Suita, 565-0871, Osaka, Japan.
Tel.: +81-80-9098-3252

Wenguo Liu and Alan F. T. Winfield
BRL (Bristol Robotics Lab),
University of the West of England, Bristol, UK.
Tel.: +44 117 328 2644
E-mail: alan.winfield@uwe.ac.uk

1 Introduction

Division of labour is one of the fundamental problems that must be solved for swarm or multi-agent systems. In multi-agent systems a straightforward approach is centralised workload distribution: a main controller collects all required data, such as robots' positions and sensor readings, then allocates scheduled tasks to the most suitable robots (Keshmiri and Payandeh, 2011; Liu and Kroll, 2012) in order to achieve the goal. Despite the simplicity of this approach and recent breakthroughs within indoor environments (D'Andrea, 2012), there are several well-known challenges related to the accuracy and availability of global information; precise information is not always possible to obtain, especially within large domains, for instance outdoor (Reggente and Lilienthal, 2009) or unstructured scenarios (Chitta et al., 2012). In addition, the robustness of such systems might be compromised when a problem or failure affects the central controller, which could lead to a halt in overall system operation. Therefore, increasing attention has been paid to swarm systems that do not rely on centralised controllers and treat division of labour in a distributed fashion.

Searching for a technique that could address the problems of centralised controllers led to research in which robots coordinated their actions through intensive communication and negotiation (Jin et al., 1994; Jun et al., 1999). These models were mainly applied in small teams of robots due to bandwidth limited communication (Parker, 1994; Gerkey and Matarić, 2000). As the number of robots increases within the swarm, the design complexity of the controller also increases (Stergiopoulos and Tzes, 2011; Escobedo et al., 2014) due to high communication bandwidth requirements or to the limited computational power of the simple robots normally used in robotic swarm studies. Therefore, a new approach to labour division and task allocation, which improves upon centralised controllers in terms of robustness and at the same time avoids the complexity issues of classical distributed models, is needed for the next generation of swarm robotics systems.

The most prominent work within task allocation and division of labour research involves foraging tasks (regarded as one of the main benchmark problems of swarm robotics (Winfield, 2009)), with very limited (Lerman et al., 2006; Liu and Winfield, 2010) or no communication among the robots (Yongming et al., 2010), making systems extremely scalable. Within the broad field of robot foraging, the topic of sustainable foraging has gained popularity among roboticists in the last decade (Song and Vaughan, 2013; Ashikaga et al., 2007). However, previous approaches are mostly centralised and too sensitive to specific food distributions (Song and Vaughan, 2013). Moreover, they do not consider the requirements of satisfying an external system with changing needs (Ashikaga et al., 2007), as needed for promising real-world applications for robotic swarms such as agriculture or mining. This paper aims to fill the gap existing in the field.

Numerous studies (Yang et al., 2009; Yongming et al., 2010; Bailong et al., 2008) have applied simple algorithms based on Fixed Response Threshold Models (FRTM) (Bonabeau et al., 1998) to robotic foraging swarms. These models have shown similar behaviours to the mechanisms that regulate labour in insect colonies such as worker specialisation (Labella et al., 2006) or hierarchical differentiation (Theraulaz et al., 1990).

Despite its simplicity, FRTM is still very interesting because of the ability to adapt to changing situations (Yongming et al., 2010; Bailong et al., 2008) and

different scenarios (Bonabeau et al., 1998; Yang et al., 2009). The fixed response threshold (θ) is an internal FRTM variable that determines the tendency of a robot to respond to a given stimulus in the environment and perform an associated task. In foraging scenarios, θ has a great influence on the robot's behaviour, because it determines when the robot goes foraging (i.e., to expend energy looking for food within the environment), or rest (and conserve energy by staying in the nest) in case its work is not needed at that specific moment (Labella et al., 2006; Krieger et al., 2000). On one hand, prior work in the field has shown that a high value of θ can decrease interference among robots and therefore increase the system's performance (Zhang and Zeng, 2012), while on the other hand lower values of θ can beneficially affect the adaptation in response to abrupt changes in the availability of food in the environment (Castello et al., 2013). Tuning and optimising θ according to different environmental conditions in order to make a robot swarm adapt to dynamical situations is therefore not straightforward.

This paper analyses the results of simulation and real-robot experiments using our proposed algorithm: the Adaptive Response Threshold Model (ARTM), with the purpose of confirming the adaptability of ARTM in stochastic environments such as the real-robot scenarios within Section 2.3. This is motivated by the fact that experiments with real hardware always suffer from stochasticity given by noisy sensor readings or unpredictable interference among robots.

Preliminary simulation results obtained with ARTM were presented in Castello et al. (2013) and Castello et al. (2014). In addition to the more extensive results included in this paper, we demonstrate that ARTM can achieve adaptive division of labour in a small size robot swarm in an efficacious and robust way. It is also the purpose of this paper to show that simple modifications (e.g., dynamic calculation of θ) of already known algorithms such as FRTM, can usefully improve upon the original model and lead to simpler and more adaptive systems which can be efficient both in simulated and real-robot scenarios.

Previous works in the fields of threshold-based division of labour and adaptive foraging show fundamental differences with respect to the model presented in this paper.

Firstly, Agassounon and Martinoli (2002) propose a private variable threshold worker allocation algorithm (PrVT), which averages a robot's successful searching times in order to compute an optimal searching time-out threshold in static clustering missions. Unlike our approach, the algorithm proposed in Agassounon and Martinoli (2002) requires robots to gradually store additional information such as previous successful searching times. This information is then used to re-calibrate the threshold parameter. This approach might be a limitation for swarm systems that need to be deployed in long-term missions and are composed of relatively simple robots due to memory constraints. Furthermore, the parameters are modified only when a search mission is successfully completed. This episodic type of update may be insufficient if the environment is dynamic and changes abruptly during the mission. In contrast, our approach proposes an ongoing adaptation process, which takes place during robot idle time — the wait action — making the robot swarm more reactive against those changes.

Secondly, unlike Lee and Kim (2014), our approach does not require memory based lists in order to adapt robots' thresholds over time. This point is particularly important, since it has been proved that the size of lists requires optimisation in order to cope with certain dynamical situations (Lerman et al., 2006).

Thirdly, even though a well-known model such as FRTM is used as the basis for many works in the field, the analysis of parameters relevant to adaptation such as the slope of response curves has not been addressed in previous literature, with the exception of Kanakia et al. (2014). In Kanakia et al. (2014), certain predefined values for parameters such as the response threshold — θ — or the slope of the response curve — n — were analysed for a specific swarm collaborative mission. Unlike our approach, Kanakia et al. (2014) do not propose a model in which the dynamical adaptation of these parameters takes place in a reactive manner given the environmental conditions.

Several reference works have conducted similar foraging experiments (Krieger and Billeter, 2000; Song and Vaughan, 2013; Theraulaz et al., 1990) to the ones presented in the present research. However, they all assumed fixed values for the stimulus variables (such as constant consumption rates (Beshers and Fewell, 2001)) or incremental success/failure rates (Labella et al., 2006), where adaptation to environmental conditions might take a long time. In contrast, our proposed model is tested under different values for the swarm's stimulus (time-changing consumption rates) with the purpose of proving its use in time-critical foraging.

Finally, few of the aforementioned works, with the notable exceptions of Labella et al. (2006) and Krieger and Billeter (2000), present data obtained from real-robot experiments in order to corroborate findings discovered in simulation. This paper aims to provide a clear comparison between simulation and real-robot results.

2 Methods

2.1 Adaptive Response Threshold Model (ARTM)

The Adaptive Response Threshold Model (ARTM), first described in Castello et al. (2013), is an extension of the classical Fixed Response Threshold Model (FRTM), in which the response threshold (θ) is calculated dynamically instead of remaining fixed. ARTM, like other response threshold methods, strongly relies on the concept of stimulus:

$$S_{(t)} = F_{(t_0)} - F_{(t)} \tag{1}$$

Equation (1) provides the definition of stimulus ($S_{(t)}$) used throughout this research, in which $F_{(t)}$ corresponds to the amount of food within the swarm's nest (regarded as the robot's headquarters in every foraging mission) at time t, and $F_{(t_0)}$ to the amount of food found in the nest at the beginning of the foraging mission. $F_{(t_0)}$ is regarded as the optimal level of food robots should maintain at the nest. $F_{(t_0)}$ is arbitrarily chosen at the beginning of each experimental run. The response curve P_f (i.e., the probability of leaving the nest area to start foraging) is defined as:

$$P_f = \begin{cases} 0 & \text{if } S_{(t)} \leq 0 \\ \frac{{S_{(t)}}^n}{{S_{(t)}}^n + \theta^n} & \text{if } S_{(t)} > 0 \end{cases} \tag{2}$$

In addition to the stimulus variable ($S_{(t)}$) and the threshold parameter (θ), the variable n is shown in equation (2). n determines the slope of the response curve [1]. Diversity of n values is regarded as one of the main forms of emergent behaviour within swarms of robots using Response Threshold Models (RTM) (Kanakia et al., 2014), since robots with the same θ threshold but different n values might produce different responses given the same stimulus $S_{(t)}$. Typically n is randomly generated from a predefined range and remains fixed for the whole experiment.

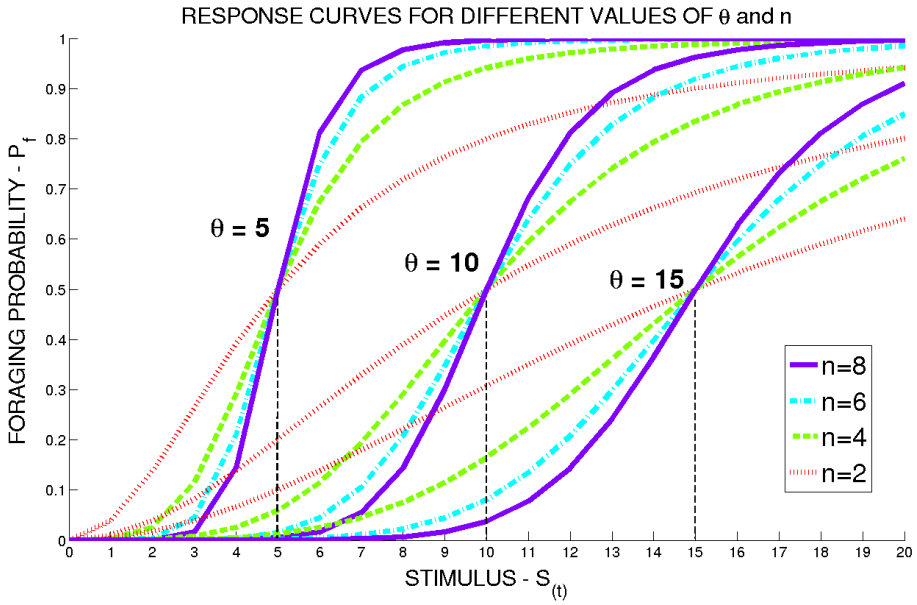


Fig. 1 Response curves for several θ and n values. θ represents the location of the 0.5 point for each corresponding response curve. n represents the slope of each corresponding response curve

Fig. 1 shows different response curves for $\theta = 5$, $\theta = 10$ and $\theta = 15$ as well as several values of n. Fig. 1 suggests that robots with a smaller θ tend to react faster to small stimulus ($S_{(t)}$) values. Conversely, robots with a high θ will not be so likely to react to small stimulus values and will have a tendency to remain out of action for longer. As it was explained previously, n represents the slope of each corresponding response curve. Robots with a smaller n will have a "flatter" response curve, showing moderate progressions of their P_f values as $S_{(t)}$ increases. In contrast, robots with a higher n will have a "steeper" curve that will abruptly increase their probability of going foraging as $S_{(t)}$ increases.

[1] The response curve corresponds to the plot of the response probability P_f as a function of the stimulus $S_{(t)}$.

2.2 ARTM foraging framework

Fig. 2 shows a general overview of the ARTM foraging framework. The major difference with previous fixed response threshold models is the Discrete Attractor Selection Model (DASM), represented by a rectangular box with dashed boundaries. DASM is the algorithm first introduced in Castello et al. (2013), which dynamically calculates the θ threshold according to different environmental conditions. Once the process of updating θ is over, robots are able to calculate their response curve (defined in equation (2)) according to the new conditions and decide whether to begin the foraging process or wait, in standby mode, near the nest.

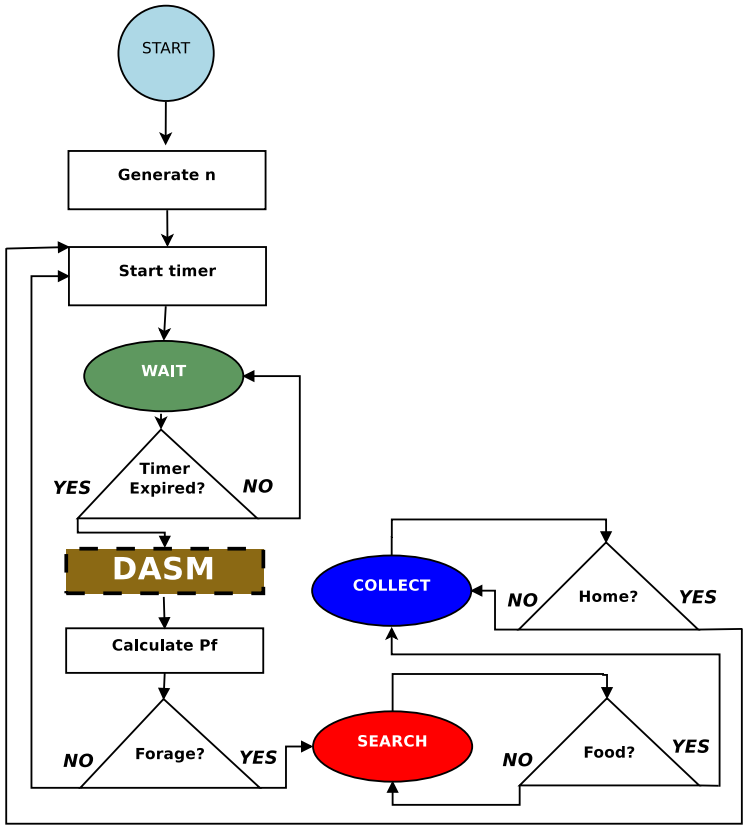


Fig. 2 Flow chart of the Adaptive Response Threshold Model (ARTM) framework

The ARTM algorithm in Fig. 2 relies on 3 basic actions:

– *Wait*: At the start of each experiment all robots run the *Wait* action, which could be regarded as a type of standby mode. As can be seen in Fig. 2, at the start of the experiment each robot randomly generates a value for n from a predefined range in order to calculate P_f and then starts a timer. When the timer expires, the robot senses the current amount of stimulus at that

particular moment ($S_{(t)}$) and calculates P_f. If P_f is high, the transition to the *Search* action is likely to occur, if not, the robot starts another wait cycle. Every robot needs to perform the *Wait* action near the nest since once its timer expires, it needs to sense the current amount of food at the nest ($F_{(t)}$) in order to calculate the new stimulus ($S_{(t)}$) value.
- *Search*: The robot performs a random walk searching for food tokens. If the robot detects a food token within its sensor range, it executes the *Collect* action, otherwise it continues searching. During the execution of this action the robot is able to detect obstacles such as walls or other robots and avoid them.
- *Collect*: Once the robot has moved close to the food token and grabbed it, it moves to the swarm's nest and deposits the food token inside it. After placing the food token in the nest the *Wait* action is triggered again.

2.2.1 Discrete Attractor Selection Model (DASM)

The Attractor Selection Model (ASM) is one of the simplest ways of describing biological fluctuations (Kashiwagi et al., 2006). Despite its simplicity, this model can provide adaptation capabilities in unknown and dynamical environments.

In previous research works (Shimizu et al., 2011; Nurzaman et al., 2008), ASM has been formalised by one of the most popular equations in biology, namely the Langevin equation (3):

$$\frac{dx}{dt} = -\nabla U(x)A + \varepsilon \tag{3}$$

Equation (3) is composed of three main elements. x represents the state of the system. x is calculated after multiplying a variable called activity (A) by a potential field ($U(x)$) with several attractors (local minima). A describes how well suited the current state x is to the current environment. A is designed in a way that a high value is given when x is well suited and a low value when x is not suited to the environment. Therefore, A drives the behaviour of the whole model. For instance, when A becomes large (x is suited for the current environment), the term $-\nabla U(x)$ becomes dominant in equation (3) and a transition between attractors is not likely to occur. However, if A is small, the noise parameter ε becomes dominant in equation (3) starting a random search procedure that looks for a more suitable attractor.

As its name suggests, the Discrete Attractor Selection Model (DASM) is the discrete version of the above explained ASM. DASM is used throughout this paper because of its simplicity and low computational cost which makes it ideal for swarm robotics research.

Parameter	Description
W	Set of states
$w_{(t_0)}$	Initial state of the system
$\Sigma_{(t)}$	System's input
δ	State transition function, where $\delta : p(w_{(t+1)}, w_{(t)}, \Sigma_{(t)})$

Table 1 DASM's parameter table

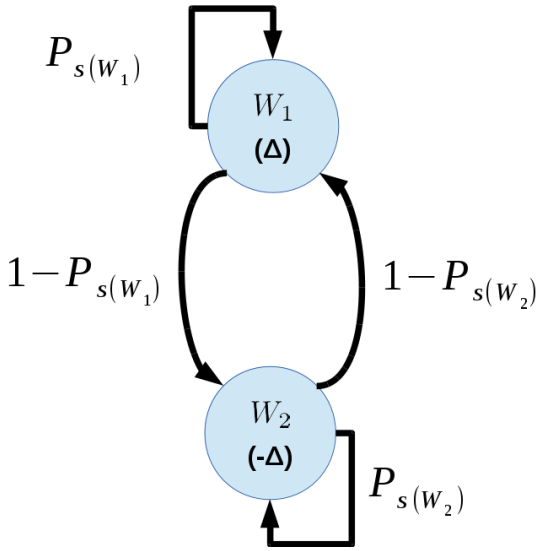


Fig. 3 Overview of DASM. DASM is a PFSM (Probabilistic Finite State Machine) in which each system state — W_1 and W_2 — is bound to two probability values. P_s determines the probability of remaining in the current state of the system ($w_{(t)}$). This probability is updated based on equation (4). The complement $1 - P_s$ determines the probability of changing state in the next time step.

Rule Number	DASM Input	Behavioural Rules		DASM Output
	$\Sigma_{(t)}$	$w_{(t)}$	$S_{(t-1)} \geq S_{(t)}$	θ
I	1	Δ	true	$\uparrow$
II	0	$-\Delta$	true	$\uparrow$
III	0	Δ	false	$\downarrow$
IV	1	$-\Delta$	false	$\downarrow$

Table 2 DASM's input and output table according to several behavioural rules. Rules I and II make a robot increase its θ threshold. Rules III and IV trigger a decrement in a robot's θ threshold.

DASM is defined by the tuple $(W, w_{(t_0)}, \Sigma_{(t)}, \delta)$ whose parameters are given in table 1. Fig. 3 shows a typical DASM's structure composed of a two state set $W = \{W_1, W_2\}$. W_1 and W_2 represent a constant value Δ chosen at the start of the experiment. DASM's current state ($w_{(t)}$) is in charge of increasing or decreasing the θ threshold parameter by the value Δ at every time step, and therefore make it fluctuate. In this paper, W_1 is assigned to Δ and W_2 is assigned to $-\Delta$. The initial state of the system ($w_{(t_0)}$) is randomly chosen from these two states at the start of each experiment. $w_{(t_0)}$ and $w_{(t)}$ could be regarded as two "pointers", which always point to one of the states in W at certain points in time: $w_{(t_0)}$ in $t = 0$ and $w_{(t)}$ at time step t. Therefore, the expression $w_{(t)} = \Delta$ means that the current state of the system at time step t has the value of Δ, which analogously means that the current state of the system is bound to W_1.

DASM's input variable is $\Sigma_{(t)} = \{0, 1\}$. The behavioural rules behind this variable are shown in table 2. $\Sigma_{(t)}$ is set to 1 in two cases. In the first case (see line corresponding to rule I), when there is a reduction or no change in the level of the

stimulus variable between the current time step ($S_{(t)}$) and the previous time step ($S_{(t-1)}$), and DASM's current state ($w_{(t)}$) is assigned to Δ. The reason behind this behavioural rule is that if robots are capable of increasing their θ threshold ($w_{(t)} = \Delta$), because more food is being placed within the nest ($S_{(t-1)} \geq S_{(t)}$), then the θ threshold should be increased. As a consequence of this, robots tend to keep running the *Wait* action.

In the second case (see line corresponding to rule IV), when there is an increase in the stimulus variable between $S_{(t)}$ and $S_{(t-1)}$ and $w_{(t)}$ is assigned to $-\Delta$. The reason behind this rule is that if robots are capable of decreasing their θ threshold ($w_{(t)} = -\Delta$) and less food is being placed within the nest ($S_{(t-1)} \leq S_{(t)}$), then the θ threshold should be decreased. $\Sigma_{(t)}$ is bound to 0 in case neither of these conditions are met (table 2 rule II, and table 2, rule III). The mapping of $\Sigma_{(t)}$ represents how well suited the current state of each robot's DASM ($w_{(t)}$) is to the condition of the food dynamics ($S_{(t-1)} \geq S_{(t)}$) at the swarm's nest.

As shown in Fig. 3, every W state is connected to two probability values. $P_{s_{(w_{(t)})}}$ represents the probability of remaining in the current state for one more time step. On the other hand, $(1 - P_{s_{(w_{(t)})}})$ represents the probability of swapping state.

The state transition function p involves the calculation of P_s. The p function is defined in equation (4):

$$p(w_{(t+1)}, w_{(t)}, \Sigma_{(t)}) = P_{s_{(w_{(t)})}} = \begin{cases} P_{s_{(w_{(t-1)})}} + \alpha, & \text{if } \Sigma_{(t)} = 1 \\ P_{s_{(w_{(t-1)})}} - \alpha, & \text{if } \Sigma_{(t)} = 0 \end{cases} \tag{4}$$

Equation (4) shows how, in case the system's input variable $\Sigma_{(t)}$ is bound to 1, a predefined constant value α is added to the previous probability $P_{s_{(w_{(t-1)})}}$ of staying in the current state, resulting in a higher probability $P_{s_{(w_{(t)})}}$ of staying in the current state. In contrast, if $\Sigma_{(t)}$ is bound to 0, α will decrease $P_{s_{(w_{(t-1)})}}$, resulting in a lower $P_{s_{(w_{(t)})}}$ for the next time step.

After the state transition function is calculated for the current time step ($p(w_{(t)})$), a random number with uniform distribution is generated in the range [0,1], and compared to the updated transition function. Once a new state is selected for the current time step ($w_{(t)}$), equation (5) is used:

$$\theta_{(t)} = \theta_{(t-1)} + w_{(t)} \tag{5}$$

Equation (5), which could be regarded as an update rule, is used to obtain the value of the threshold parameter $\theta_{(t)}$.

DASM provides a decentralised way to update each robot's θ threshold, exploiting features only present in the environment (contact with the associated stimulus $S_{(t)}$ in the nest). No direct communication with a centralised controller is required in order to assist robots to adapt to the environment.

Finally, with the purpose of making a comparison between ASM (equation (3)) and its discrete version (DASM), W could be compared to the set of attractors (local minima) within the potential field $U(x)$, while $\Sigma_{(t)}$ plays the same role as the activity value (A) in equation (3), and δ (and its probabilistic transition function p) is analogous to the noise distribution ε.

2.2.2 ARTM's typical behaviour

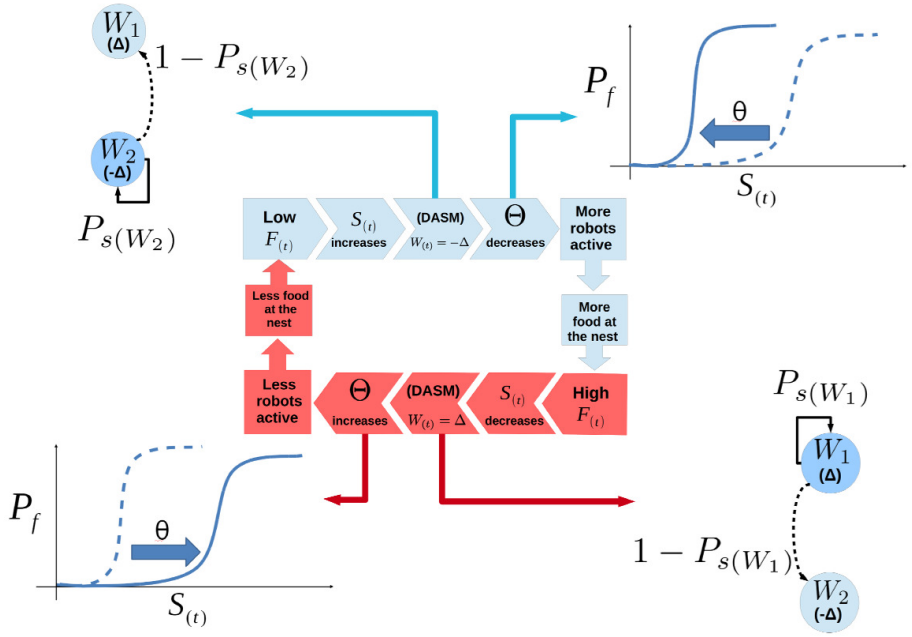


Fig. 4 The Swarm Foraging Process using ARTM. When the food level ($F_{(t)}$) at the swarm's nest is low, the stimulus ($S_{(t)}$) value will eventually increase, biasing the DASM to decrease the θ threshold and therefore making more robots go foraging. Due to the increased number of working robots, the amount of food ($F_{(t)}$) at the swarm's nest will increase, making the stimulus ($S_{(t)}$) values decrease, therefore DASM will tend to increase the θ threshold, causing fewer robots to go foraging in the next running cycle.

Fig. 4 shows the behaviour of a foraging swarm using the ARTM process. This diagram describes the chain of events that occur once we employ ARTM as a division of labour mechanism. For instance, if the amount of food ($F_{(t)}$) at the swarm's nest becomes smaller, due to the action of the consumption rate (C_{rate}) (an agent that extracts food from the swarm's nest, simulating an external system), the stimulus ($S_{(t)}$) value gradually increases. This effect causes the DASM to decrease the θ threshold and therefore more robots to go foraging in the following time steps. Because of the increased number of working robots, the amount of food ($F_{(t)}$) at the swarm's nest tends to increase, causing the system's stimulus ($S_{(t)}$) values to become smaller. This effect biases DASM to increase the θ threshold, therefore causing fewer robots to go foraging in the next running cycle. Fig. 4 presents a decentralised feedback-based approach, since it only relies on local information and no single entity is choosing θ for each robot.

2.3 Real-robot experiments

The benefits of using dynamical threshold models (such as ARTM), in terms of increase of the survivability and adaptability of swarm systems in changing environments, was previously shown only by computer simulations in Castello et al. (2013) and Castello et al. (2014). This paper aims at confirming those simulation results in a new set of experiments conducted using real robots.

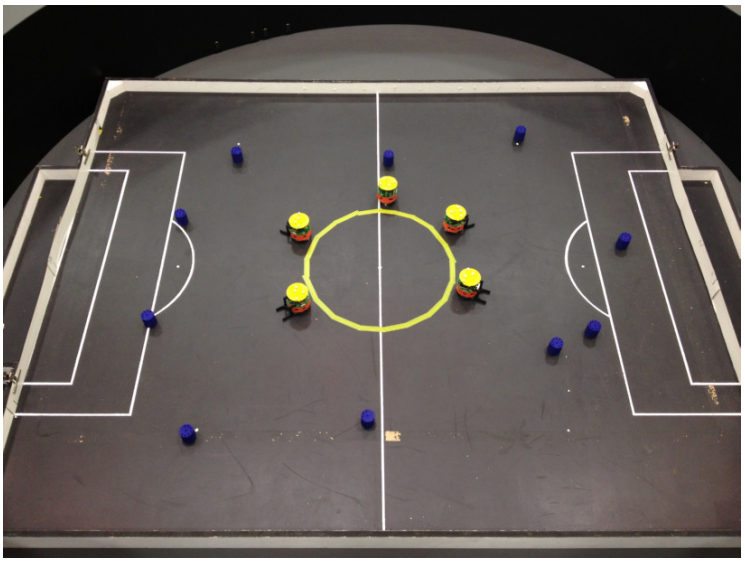
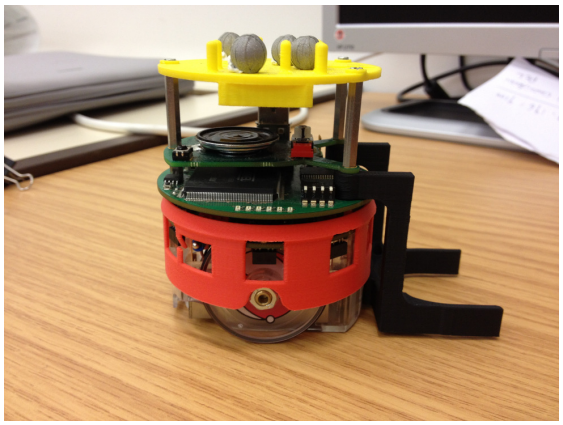
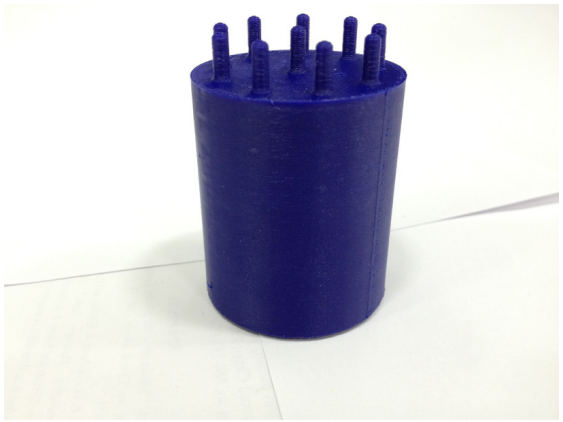


Fig. 5 Photo of the arena used in order to conduct the real-robot experiments. In the picture, five e-puck robots (in *Wait* action) gather in the middle of the experiment space, surrounding the nest (circle located at the center), and 10 food tokens are scattered in the arena. Note that the other markings in the experimental arena have no significance

Fig. 5 shows a snapshot of the arena where several e-puck (Mondada et al., 2009) robots (identical to the one shown in Fig. 6(a)) were placed. The e-puck platform is a well-known open hardware testbed extensively used for swarm experiments (Chen et al., 2012; Cianci et al., 2007). Besides the main processing unit, which was enhanced by the addition of a Linux extension board (Liu and Winfield, 2011) and the differential wheel drive that makes the robot move, each e-puck robot is equipped with the following sensors: 8 infrared proximity sensors for detecting obstacles, 1 colour camera with 640 × 480 pixel resolution, 1 3D accelerometer, 3 microphones and 1 loudspeaker. In addition to this hardware configuration each e-puck robot is equipped with a 3D printed passive gripper especially designed for foraging experiments. The environment was designed as a rectangular area of 2.0 x 1.5 square meters with a circular area (50 cm in diameter) at the centre which represents the swarm's nest. Solid dark cylinders (such as the one shown in Fig. 6(b)) within the arena represent food tokens. Following the example of simulation experiments described and conducted in Castello et al. (2013) and Castello et al. (2014), a certain number of food tokens (D) were randomly but evenly positioned at the beginning of each run.



(a) E-puck robot with a Linux Expansion Board and a passive 3D printed gripper



(b) 3D printed food token

Fig. 6 (a) Photo of the e-puck robot used in the experiments. In addition to the expansion Linux board (Liu and Winfield, 2011) used for controlling the robot, a 3D printed passive gripper was fitted to each robot in order to bring food tokens distributed within the arena to the swarm nest. (b) Food token used in the foraging experiments. Food tokens are 3D printed cylinders with the following dimensions: 8 cm tall and 2 cm in diameter. The small pips on top of the food token are for reflective markers to track the tokens. However, the location of food tokens was not tracked during these experiments.

Fig. 7 outlines the main elements of the experimental setup and its data flows. Every time an e-puck robot detects a food token with its embedded camera, it aligns with it and grabs it with its passive gripper. In order to carry that food token to the nest, the e-puck robot needs to know the position of the nest. This information is provided using a Robot Operating System (ROS) (Quigley et al., 2009) instance which sends that specific robot its own position and orientation as well as the nest's position at a frequency of 1 Hz through a ROS topic; in this way we provide robots with a virtual nest sensor. Once the food token is within the gripper's jaws, the e-puck robot rotates until it aligns with the nest's centre.

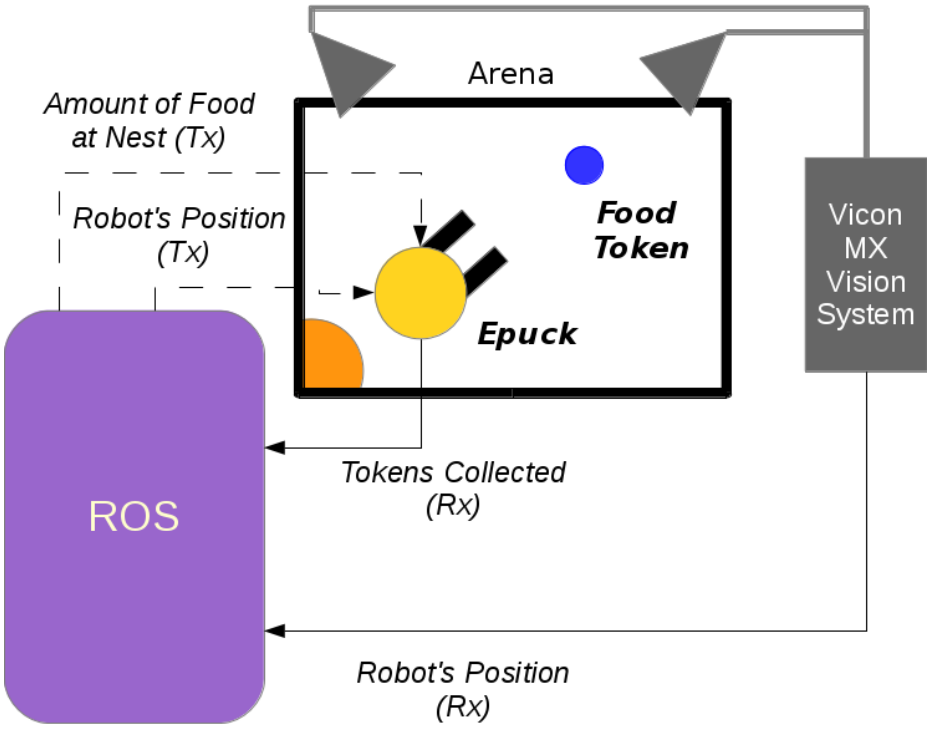


Fig. 7 Main data flows of the real-robot experiment setup. The use of the Vicon MX tracking system and ROS infrastructure were two important features of the experimental setup, needed to provide the robots with virtual sensing of the nest and amount of food collected. Tx refers to ROS topics that were used in order to send information from the ROS main instance into the robots. Rx refers to ROS topics used in order to receive information from the environment and robots into the ROS system.

After this, the robot moves in a straight line until the edge of the nest is reached; at that point the robot reverses, leaving the food token within the nest limits and re-executing the *Wait* action. As soon as a food token is deposited within the nest, it is then manually re-allocated to a new random location within the arena. In this way, we are able to simulate random re-appearance of food within the arena while maintaining the amount of available food constant. Each robot's position is fed into the ROS system through the camera-based tracking system VICON MX (by Vicon Motion Systems Ltd).

Finally, each e-puck robot sends, through another ROS topic, the information regarding the number of food tokens collected. An external ROS package calculates the total amount of food at the nest at any time, accounting for the consumption rate, and sends it to robots that are in *Wait* action near the nest; again this provides robots with virtual sensing of the amount of food in the nest. It is assumed that all robots executing the *Wait* action during the experiments outlined in Sections 2.3 and 2.4 are able to detect the exact amount of food at the nest. During the real-robot experiments, due to the limited number of physical food tokens, the

amount of food that robots detect at the nest is not represented by the number of physical food tokens within the central circle (nest) of the arena.

It is important to note that despite using an external infrastructure for the real-robot experiments, such as ROS shown in Fig. 7, the response threshold models studied in this paper do not internally rely on any globalised transmitted information. This external architecture is only a convenient means to provide robots with virtual sensors, and thus the same sensor information as in our previous simulation experiments (Castello et al., 2013, 2014). Therefore, these models can be regarded as decentralised solutions to foraging processes.

2.4 Simulation experiments

In order to complement and extend the real-robot experiments outlined in Section 2.3, we also conducted several simulation tests with similar conditions as those described above. As in Castello et al. (2013) and Castello et al. (2014), all simulation tests were conducted using the widely used open-source multi-robot simulation library STAGE (Gerkey et al., 2003). Although STAGE does not simulate physics (and in particular dynamics), we argue that the low-speed operation of real-robot experiments allows us to make direct comparison of results obtained from real-robot and simulation experiments.

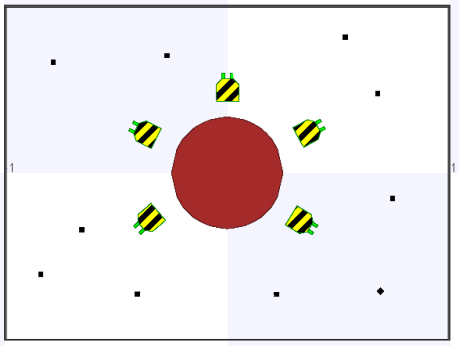
Fig. 8(a) is a screenshot of the simulation arena. The environment was setup as a rectangular area of 2×1.5 square meters with a circle (50 cm in diameter) at the centre, which represents the swarm's nest. Solid dark rectangles within the arena represent food tokens which were also randomly positioned at the beginning of each simulation test. The polygons close to the swarm's nest represent the simulated robots used in these experiments. Fig. 8(b) depicts a typical simulation test in which a team of robots conduct a foraging mission. The three different actions, explained within Section 2.2, were implemented within the simulated robot units. The robot's color indicates the action being executed. Light coloured robots (*Search* action) carry out a random walk until they sense a food token. Analogously to real-robot experiments, once a food token is detected, dark coloured robots (*Collect* action) collect the token and deliver it to the swarm's nest. Finally, striped coloured robots (*Wait* action) remain in standby near the nest.

Fig. 9 shows a closer view of the simulated robot structure and capabilities. Each robot is composed of a differential steering drive model mounted under a polygonal chassis with the following measures: 10 cm $\times$ 12.5 cm $\times$ 7 cm. These measures correlate with the e-puck robots with the 3D printed gripper fitted. Each simulated robot has 7 on-board IR sensors $(S_0, ..., S_6)$ at different positions.

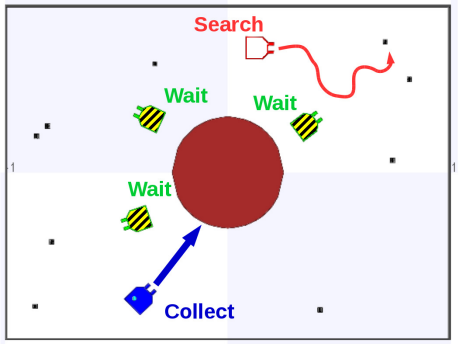
2.5 Performance measures

One of the most widely used performance indexes within swarm and multi-agent division of labour tests is the average number of working robots (N_v) during the mission:

$$N_v = \sum_{i=1}^{N_r} t_i / T_M \tag{6}$$



(a) Arena's Screenshot



(b) robot swarm in the field

Fig. 8 (a) Screenshot of the foraging arena in the extended simulation tests. In it, various robots (running the *Wait* action) gather around the nest (solid color circle) located at the center of the simulated area. (b) Screenshot of several robots during a typical foraging mission. Light coloured robots (*Search* action) perform a random walk until a food token is found. Dark coloured robots (*Collect* action) approach the swarm's nest in order to deposit an already gathered food token. Lastly, striped coloured robots (*Wait* action) remain in standby near the nest. Each one of the robots' actions is explained in greater detail in Section 2.2.

where N_r denotes the total number of robots, t_i denotes the total time when robot i is working during the mission, and T_M denotes the duration of the mission. It is important to remark that smaller values of N_v imply that the mission was conducted using a smaller number of robots and therefore less energy.

In this research, a working robot is defined as a robot that is not running the *Wait* action. In such case, a robot is consuming energy either collecting food tokens, or doing a random search looking for them. Smaller values of N_v indicate that the mission was conducted using less energy and therefore carried out more efficiently.

Moreover, the average deviation of food at the nest (V_f) is also introduced as a performance measure. V_f is defined as:

$$V_f = \sum_{t=0}^{T_M} \frac{|F_{(t_0)} - F_{(t)}|}{T_M} \tag{7}$$

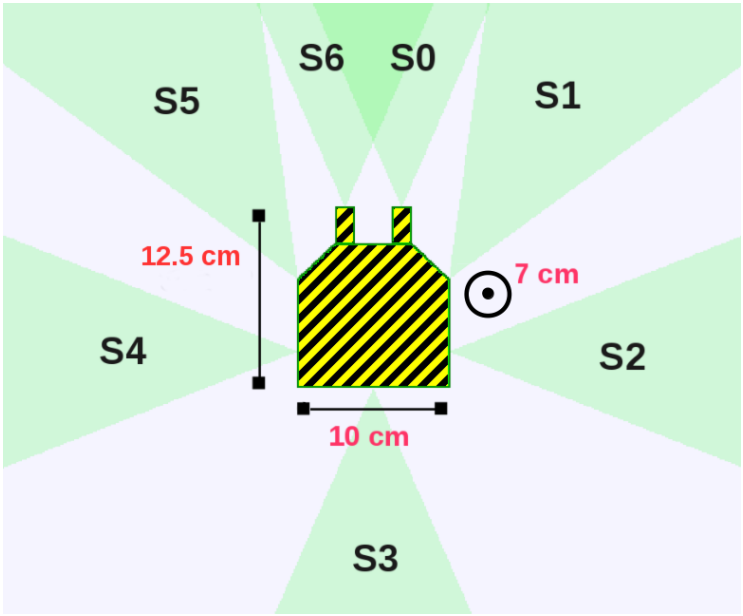


Fig. 9 Diagram of the simulated robot shape and sensor layout. Robot's size and features correlate with commercial models for swarm and multi-agent systems research such as the e-puck platform used in the real-robot experiments.

In (7), $F_{(t)}$ corresponds to the food level within the swarm's nest at time t, and $F_{(t_0)}$ to the food level in the nest at the beginning of the foraging mission. T_M denotes the duration of the mission. Smaller values of V_f imply the foraging mission was conducted maintaining food levels at the nest closer to the optimal value $F_{(t_0)}$, making the system more adaptive to the effects of external consumption rates (C_{rate}). The V_f measure takes into account not only the difference between lower food levels at the nest compared to the optimal value (food scarcity), but also food levels that exceed the optimal value (food overload). V_f was designed in order to provide a suitable measure for possible future realisations of swarms' nests such as electric power storage or mining collection hubs in which exceeding the capacity of the system might lead to waste.

Lastly, S_{rate} — the survival percentage of the system — is also considered as a performance measure:

$$S_{rate} = \sum_{i=1}^{N} \frac{S_i}{N} 100 \tag{8}$$

$$S_i = \begin{cases} 1 & \text{if } F_{(t)} \geq 0 \\ 0 & \text{if } F_{(t)} < 0 \end{cases} \tag{9}$$

where S_i is equal to 1 in case the amount of food at the swarm's nest remains positive ($F_{(t)} \geq 0$) for the whole duration of experiment i and bound to 0 otherwise; and N denotes the total number of experiments conducted. Greater values of S_{rate} indicate that robots were able to maintain positive food levels ($F_{(t)} \geq 0$) at the

nest, and therefore survive, during the full-length of experiments in a larger number of experiments.

3 Results

3.1 Real-robot results

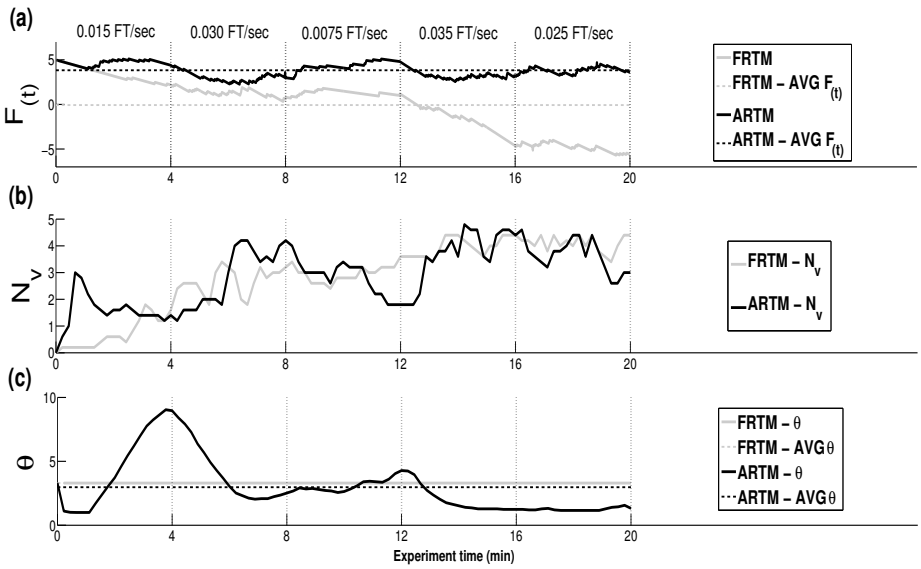


Fig. 10 Average results for five real-robot experimental runs. (a) Amount of food robots could maintain at the nest for each different value of C_{rate} applied during the experiments. (b) N_v throughout the real-robot experiment time. (c) Fluctuation of θ throughout experiments.

A collection of experiments were carried out in order to analyse the behaviour of both (FRTM and ARTM) models in real-robot scenarios. The following parameters were used in all real-robot experimental runs: $N_r = 5$, $D = 10$, $F_{(t_0)} = 5$. The threshold parameter θ was initialised to 3.3 (following the simulation experiments conducted in Castello et al. (2013) and Castello et al. (2014)) for FRTM as well as for ARTM ($\theta_{(t_0)} = 3.3$). In addition, $\Delta = 1$ and $\alpha = 0.25$ were used as DASM parameters. During ARTM execution, θ was bound to the range ($1 \leq \theta \leq 10$). This integer n was randomly generated within a predefined range ($2 \leq n \leq 9$) and remained fixed for the whole experiment for both methods. In addition, robots' waiting timer was set to 1 second. Finally, each experimental run lasted 1200 seconds (20 minutes) and was divided into five stages — each 240 seconds long — in which a different consumption rate (C_{rate}) against the nest reserves was applied.

Averaged results for five real-robot experimental runs are shown in Fig. 10. Fig. 10 (a) presents the different food levels robots were able to maintain at the nest

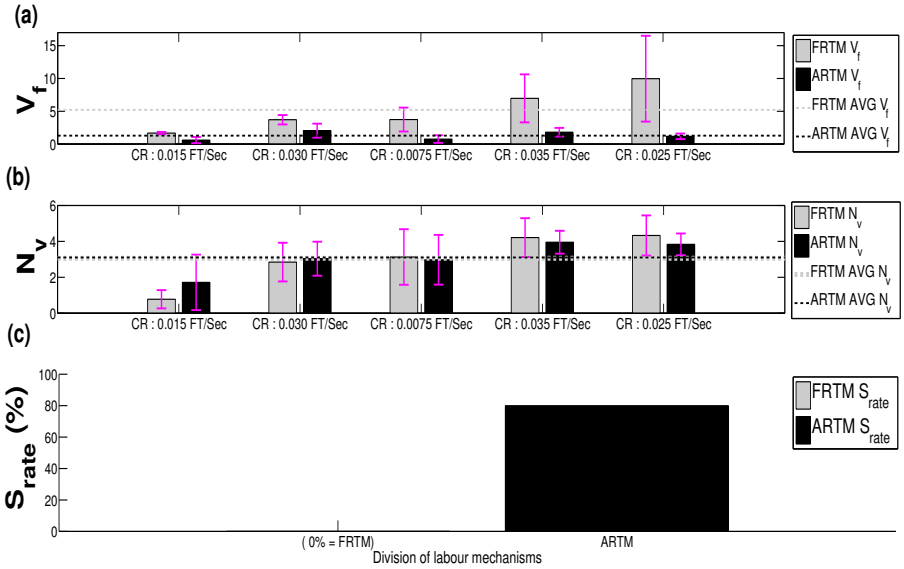


Fig. 11 Average results and standard deviations for five real-robot experiments using the same parameter set as in Fig. 10. (a) V_f — average deviation of food — and its standard deviation (error bars). (b) N_v — average number of working robots — and its standard deviation (error bars) during the five stages of the real-robot experiments. (c) S_{rate} — survival rate — of the whole system.

for the five different C_{rate} (0.015 food tokens/sec, 0.030 food tokens/sec, 0.0075 food tokens/sec, 0.035 food tokens/sec and 0.025 food tokens/sec) applied for both methods. As reported in Fig. 10 (a), ARTM could maintain a food level close to its initial value ($F_{(t_0)} = 5$) and higher than FRTM's level.

As reported in Fig. 10 (b), ARTM was able to activate more robots within the initial stages of the experiments, such as the first (0.015 food tokens/sec) and second (0.03 food tokens/sec) stages in order to balance the difference between the current amount of food at the swarm nest ($F_{(t)}$) and its initial value ($F_{(t_0)}$). In stages where C_{rate} was not so high, such as the third one (0.0075 food tokens/sec), ARTM was able to activate fewer robots than FRTM in response to the lower demands of the system, therefore optimising the swarm's resources.

Fig. 10 (c) shows that within the first minutes of the real-robot runs, θ was decreased in order to neutralise the increasing $S_{(t)}$, then suddenly increased due to the very low or even absent $S_{(t)}$ because of the closeness to the $F_{(t_0)}$ value, making rules I and II within table 2 more predominant. When the second stage (0.03 food tokens/sec) began, θ was decreased because of the increasing $S_{(t)}$ due to the high C_{rate}. In contrast, in the third stage of the experiments (0.0075 food tokens/sec) θ was gradually increased due to a lower C_{rate}. A similar behavior of θ can be observed in the following stages of the experimental runs.

Fig. 11 presents the results of all performance measures defined in Section 2.5. Fig. 11 (a) and (b) are split into five different stages, each one representing the 5 different levels of C_{rate} used during the real-robot runs. In each one of these stages

a comparison between both methods' values and their standard deviations (error bars) is given.

Fig. 11 (a) shows the average deviation of food at the nest V_f for the whole set of experimental runs. In all stages, V_f was higher for FRTM than for ARTM, showing that ARTM has stronger adaptive capabilities as it maintains the food level ($F_{(t)}$) at the swarm nest closer to its initial value ($F_{(t_0)}$). In addition, standard deviation bars follow an increasing pattern for FRTM as the experiment progresses, showing that FRTM produces very different food deviation values. In contrast, they remain relatively small for the ARTM in all experimental stages. This phenomenon is related to the different behavior against collisions shown by both division of labour mechanisms, which will be explained in detail in Section 3.2.

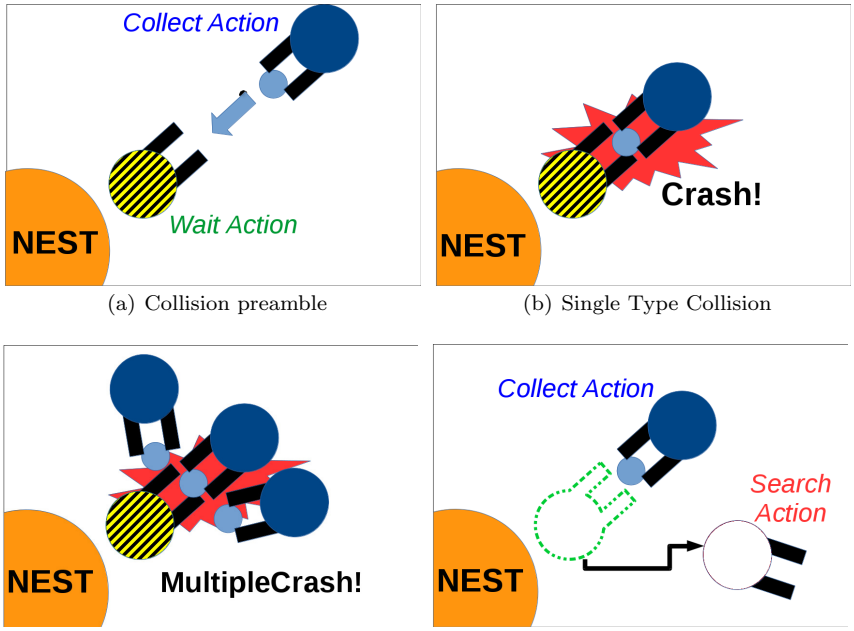
Fig. 11 (b) shows the average number of working robots N_v for the whole set of experimental runs. Overall, both methods resulted in a very similar averaged N_v value (depicted with grey and black dashed lines) for the whole experiment set, and therefore showed that no significant difference is seen in the number of robots and energy employed by both methods.

Finally, Fig. 11 (c) shows the swarm survival rate S_{rate} for the whole set of experimental runs. ARTM's survival rate (80%) was considerably higher than FRTM's (0%). This proves that a method that can dynamically modify its θ threshold can also increase the system's sustainability and thus its ability to cope with real-robot noisy environments.

Figs. 10 and 11 confirmed the results presented during the preliminary simulation stage of this research (Castello et al., 2013, 2014). However, FRTM performed worse (resulting in 0% S_{rate}) than the 24% reported in our previous works (Castello et al., 2014). One of the reasons for this low performance might be the fact that FRTM is not able to efficiently adapt to issues that arise in experiments with real hardware, such as noisy sensor readings or collisions among robots. These conjectures are analysed in detail within Section 3.2 below. Confirming the results reported in Castello et al. (2014), ARTM was able to obtain a very small average deviation of food at the nest (V_f) for the whole experiment, as well as to maintain a very similar average number of working robots (N_v) compared to FRTM. Therefore, even though both division of labour mechanisms use similar amounts of energy (N_v), they produce significantly different results in terms of food deviation (V_f) and survival rate (S_{rate}).

3.2 Response to collision

Communication is one of the factors behind collision avoidance (Yared et al., 2007; Liu et al., 2003). Therefore, in systems in which there is no explicit communication between robots, collisions are particularly problematic. In our experiments, situations in which robots near the nest unintentionally block the path of incoming robots trying to deliver food tokens to the nest (Fig. 12(a)) were especially frequent. For instance, collisions take place when robots in the *Wait* action have a very low n (very flat response curve), which makes them less likely to go foraging and therefore more likely to block the path of other robots moving towards the nest (Fig. 12(b)). Another example is when the C_{rate} of the system is low and a high percentage of the robots tend to remain idle at the nest, blocking the path of the few working robots. The collision problem is especially serious when the



(a) Collision preamble (b) Single Type Collision

(c) Multiple Type Collision (d) Collision Resolution

Fig. 12 Typical collision cases that occurs in real-robot experiments. (a) A collecting robot goes towards the nest in order to deliver some food, while a waiting robot is blocking its path to the nest. (b) A single collision blocks a collecting robot. The waiting robot is not affected since it is inactive. (c) Several robots collide with each other causing a "multiple collision". (d) Collisions are typically resolved when the waiting robot involved is activated for foraging.

swarm's nest is running out of food (high $S_{(t)}$) and smooth and efficient foraging is required to restock the nest.

For this reason we considered it important to analyse the duration and number of robots' collisions close to the nest (1 m from the nest's center).

It is important to note that neither ARTM or FRTM have a clear strategy for how to conduct proper collision avoidance (there is no path planning or communications among robots in Fig. 2). However, our results suggest that even though ARTM has no influence on the number of robot collisions during experiments (Fig. 13) it is better able to compensate for their duration (Fig. 14). Latter sections of Fig. 14 show a pronounced collision duration time for FRTM while ARTM shows a shorter collision duration time, resulting in a more efficient system.

The explanation for this phenomenon is that even though ARTM *per se* does not know if robots have collided near the nest, as the result of θ adaptation, cases in which $F_{(t)}$ is low at the nest result in θ being decreased accordingly. Robots in the *Wait* action are then more likely to go foraging, therefore unblocking the path faster (Fig. 12(d)). This leads to a large difference in the total time each robot is stuck and unproductive during the whole foraging process (32 seconds average for ARTM vs. 187 seconds average for FRTM).

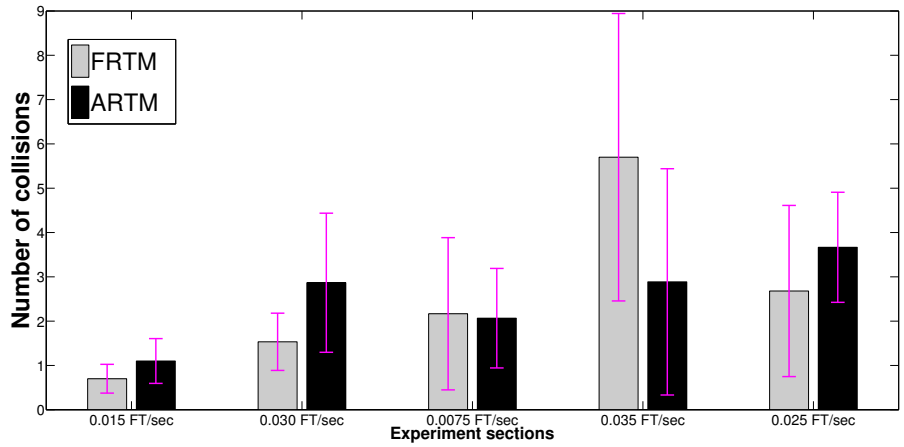


Fig. 13 Number of collisions in real-robot experiments. The figure shows the average number of collisions and its standard deviation for each section of the real-robot experiments. No pattern can be observed in the figure.

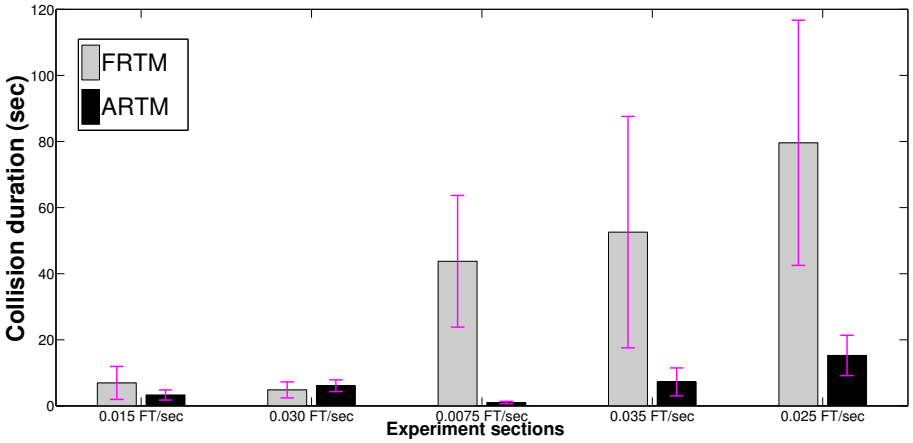


Fig. 14 Collision duration in real-robot experiments. The figure shows the average time needed to resolve one collision in each experimental stage of the real-robot experiments and its standard deviation. ARTM can resolve collisions in less time, making the system more adaptive, and hence better for real-robot application.

One of the worst-case scenarios involving collisions is when multiple robots collide near the nest (Fig. 12(c)). This case is especially important, since it can disable a large number of robots, making it extremely difficult for the swarm to meet the demands of the different C_{rate} applied — as a consequence the swarm arrives earlier at the condition of no food in the nest ($S_i = 0$).

Fig. 15 shows the number of multiple collisions (three or more robots involved) in real-robot experiments. Despite the fact that neither FRTM nor ARTM have control over the number of these types of collision, FRTM shows an increasing col-

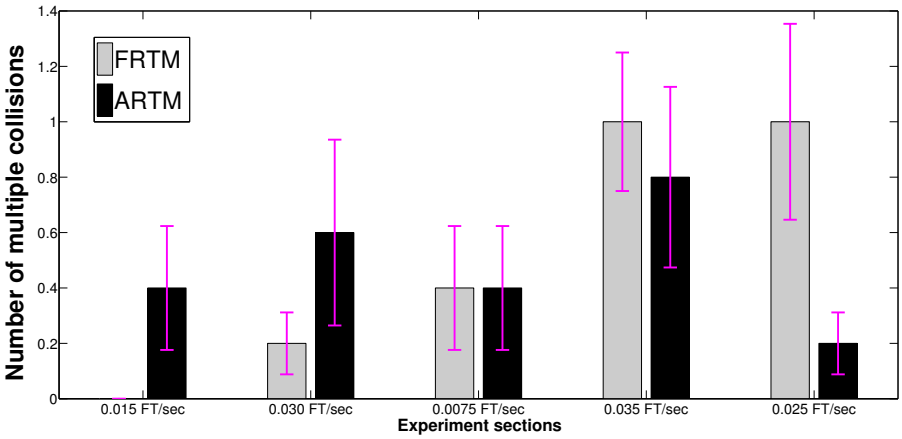


Fig. 15 Number of multiple collisions in real-robot experiments. The figure shows the average number of multiple collisions and its standard deviation for each section of the real-robot experiments. No pattern can be observed in the figure.

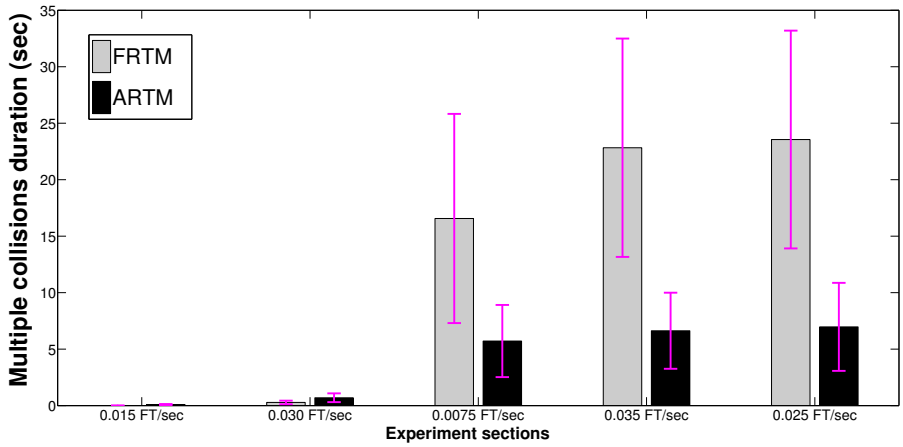


Fig. 16 Multiple collision duration in real-robot experiments. The figure shows the average time needed to resolve one multiple collision. ARTM can resolve multiple collisions in less time, making the system more adaptive to the real-robot environment.

lision duration towards latter stages of the experiments. This increase in collision duration implies that once a multiple collision has started (especially within the stages with a high C_{rate}) it is very difficult to resolve. In contrast, ARTM's tendency fluctuates, not having a clear pattern during the set of experiments shown in this paper. This phenomenon suggests that ARTM is able to resolve collisions in less time than FRTM, which produces a progressive increase in the collision duration (from the 3rd section onwards) as shown in Fig. 16.

3.3 Extended simulation analysis

3.3.1 Distribution of n

In order to understand more deeply the reasons behind ARTM's performance and the poor results of FRTM, we decided to study the distribution of the n variable, which, as described before, is randomly generated for each robot from a predefined range.

As given in equation (2) and Fig. 1, n determines the slope of the response curve P_f for each robot. Usually, n is randomly generated for each robot using a uniform distribution from a predefined range of positive integers ($2 \leq n \leq 9$ in this research) in order to make robots have different response curves and therefore react differently to the same stimulus $S_{(t)}$.

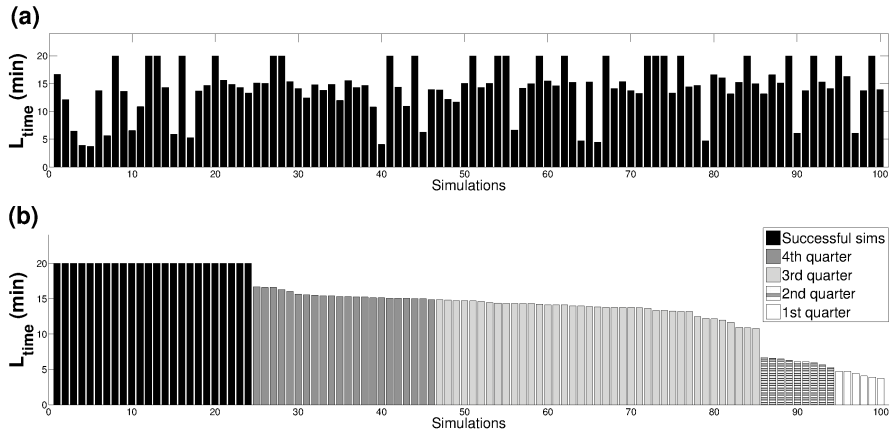
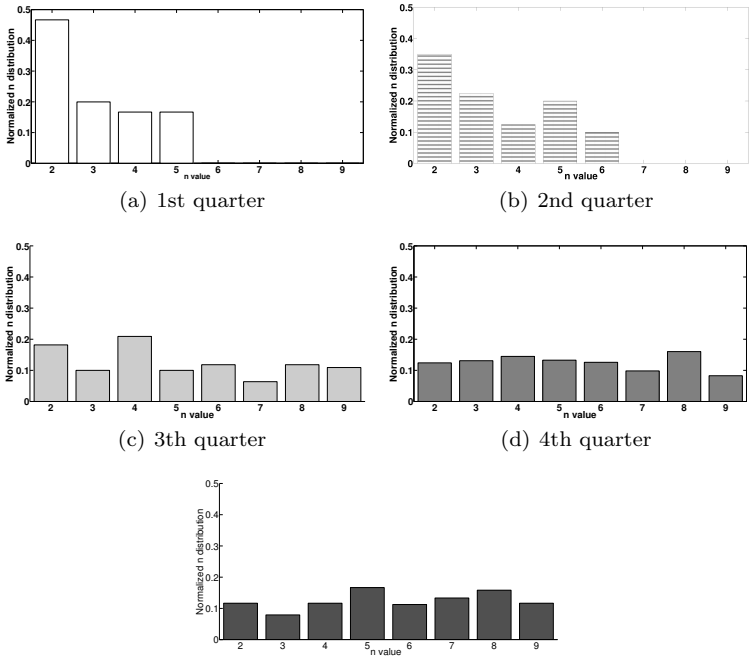


Fig. 17 (a) Living time (L_{time}) of the 100 simulations using FRTM as division of labour mechanism and conducted using the parameters described in Section 2.4. (b) The same simulation experiments as in (a) ordered by living time in a descending manner.

Fig. 17 (a) shows the living time (L_{time}) of 100 simulations using FRTM as division of labour mechanism and conducted with the same parameters described in Section 2.4. L_{time} is defined as the point in time when robots were not able to maintain a positive food level ($F_{(t)} \geq 0$) in the swarm nest, causing that specific simulation to "die" ($S_i = 0$). In this figure, successful simulations ($S_i = 1$) have a L_{time} value equal to the total simulation time (20 minutes for these experiments).

Fig. 17 (b) shows simulations' living time ordered in a descending manner. Different patterns in the figure indicate the clusters in which the simulation "died", where white, striped grey, grey and dark grey consecutively represent the first, second, third and fourth quarters of the simulation time. Simulations that were able to survive during the whole simulation time ($S_i = 1$) have a L_{time} value equal to the total simulation time and are depicted in black.



(a) 1st quarter (b) 2nd quarter

(c) 3th quarter (d) 4th quarter

(e) Successful simulations

Fig. 18 Normalised n distribution for the simulations that (a) "died" within the first 5 minutes of the simulation time; (b) "died" within the second 5 minutes of the simulation time; (c) "died" within the third 5 minutes of the simulation time; (d) "died" within the fourth 5 minutes of the simulation time; (e) that remained "alive" during the whole simulation time.

Fig. 18(a) to Fig. 18(d) show the different normalised histograms[2] for n for each of the quarters shown in Fig. 17 (b). From the histograms we can conclude that simulations which "died" earlier have a strong tendency of robots with low n ($2 \leq n \leq 4$). This effect makes more robots have a "flatter" response curve causing a delay in their foraging activity.

As we explore the n distribution of other quarters, the n distribution turns out to be more uniform with a slight tendency towards higher numbers (Fig. 18(d)).

Fig. 19 shows that there are substantial differences between the histograms of successful (Fig. 18(e)) and early dying simulations (Fig. 18(a) and Fig. 18(b)), which suggests that low n value distributions give poor survivability due to their weak response to $S_{(t)}$.

Therefore, we decided to test the survival capabilities of restricted values of n. Fig. 20 confirms that lowering the upper boundary of the n range has an effect on the S_{rate} of FRTM, achieving only a 17% S_{rate} when n is restricted to 2. However, restricted values of n do not affect ARTM, which can maintain a remarkably high S_{rate} in all cases.

Results presented in Fig. 20 suggest that the dynamic calculation of θ gives the possibility of overcoming a lower and potentially "inconvenient" n distribution.

[2] Sum of the heights equal to 1

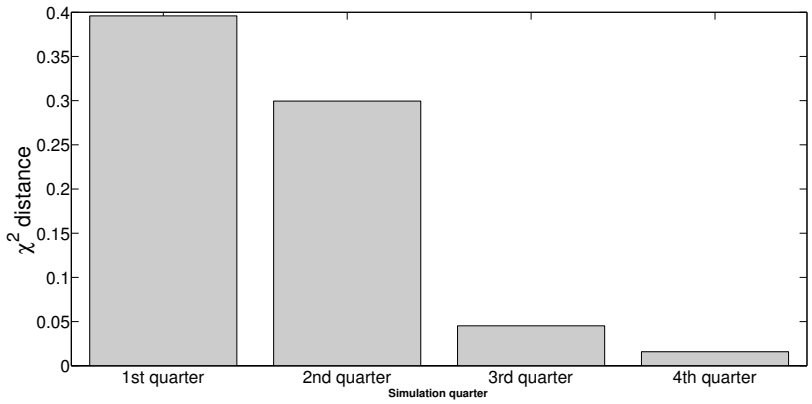


Fig. 19 Each bar represents the χ^2 distance between the histogram of successful simulations (Fig. 18(e)) and the correspondent simulation quarter (Fig. 18(a)-(d)). The χ^2 distance increases when lower quarters are compared, suggesting that certain n configurations have a strong impact on FRTM's survival capabilities.

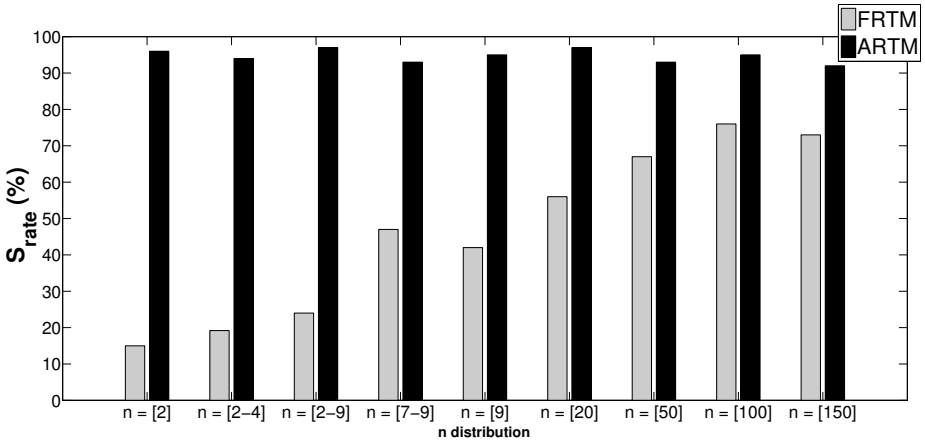


Fig. 20 ARTM and FRTM survival rates (S_{rate}) measured after 100 simulations runs for each experimental condition. FRTM decreases its performance when n is restricted to ranges with lower values, while ARTM maintains its performance regardless of the used n range.

However, if we increase the value of n in order to make robots more responsive, while we keep θ fixed, FRTM can only reach a 70% S_{rate}. Given these results, it is possible to outline a simpler method for ARTM in which n is fixed or even non-existent.

4 SARTM - Simple Adaptive Response Threshold Model

Figs. 20 and 18 show that the distribution of n has a major impact on the survival rate of swarms in the case of FRTM. However, even when we restrict the n distribution in both directions (lower and upper), ARTM can still adjust the θ threshold accordingly and preserve a high S_{rate}. This phenomenon suggests that a simpler

model, in which the effects of inconvenient n distributions are avoided, is possible if the θ threshold is adjusted properly. In SARTM (Simple Adaptive Response Threshold Model), P_f is defined as a simple step function on the θ threshold and the n variable is not present:

$$P_f = \begin{cases} 0 & \text{if } S_{(t)} < \theta \\ 1 & \text{if } S_{(t)} \geq \theta \end{cases} \tag{10}$$

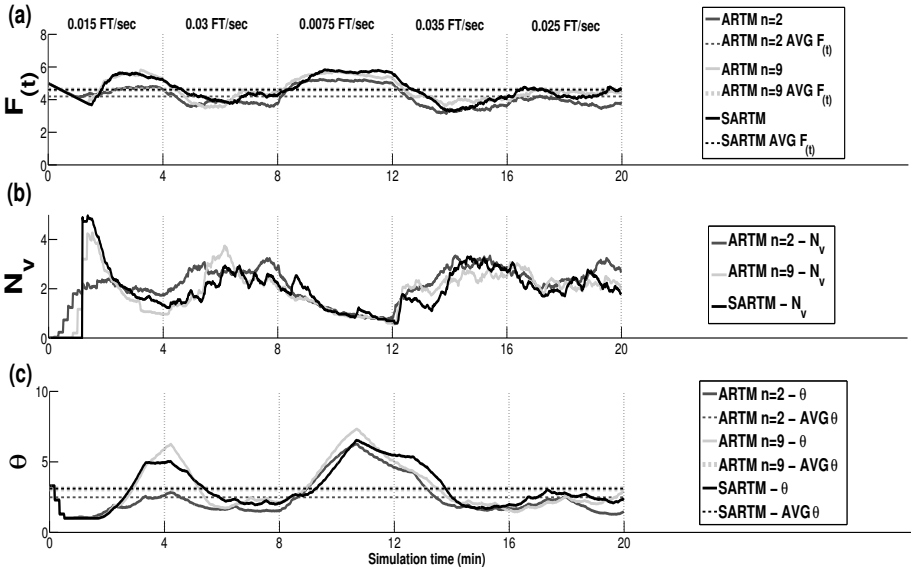


Fig. 21 Average results for 30 simulation tests. (a) Amount of food robots could maintain at the nest for each different value ofC_{rate} applied during the simulation experiments. (b) N_v throughout the simulation experiment time. (c) Fluctuation of θ during the experiments.

In order to compare the performance of SARTM — defined in equation 10 — we conducted 30 extra simulation experiments comparing SARTM against ARTM when n is fixed to the lower ($n = 2$) and upper ($n = 9$) boundaries of the range used through the paper. Fig. 21 and Fig. 22 show the averaged results of 30 simulations conducted with the parameters explained in Section 2.4. Fig. 22 (a) and 22 (b) show that SARTM can activate the same low number of working robots (N_v) as the best version of ARTM (ARTM $n = 9$) as well as achieve the same low average deviation of food (V_f).

Finally, Fig. 22 (c) shows S_{rate} results for the three compared models. SARTM results in a very similar S_{rate} level (95%) compared to both versions of ARTM: 93% for $n = 2$ and 96% for $n = 9$. These results demonstrate that the dynamic calculation of the θ threshold could produce comparable results to ARTM based methods with a simpler response function such as the step function proposed in equation (10).

It is important to remark that even though P_f is equal to 1 according to equation (10), not all robots are activated at the same time when SARTM is used.

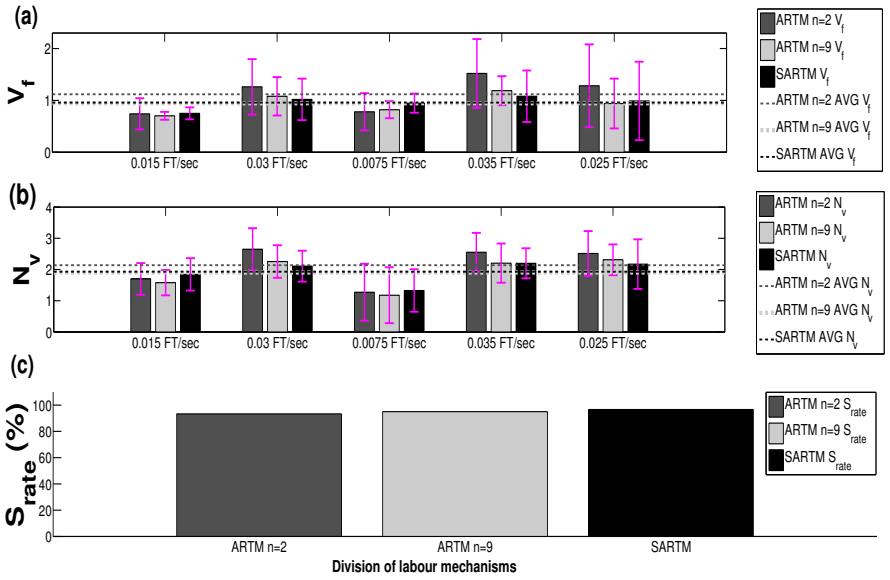


Fig. 22 Average results and standard deviations for 30 simulation tests using the same parameter set as in Fig. 21. (a) V_f — average deviation of food — and its standard deviation (error bars). (b) N_v — average number of working robots — and its standard deviation (error bars) during the five stages of the simulation experiments. (c) S_{rate} — survival rate — of the whole system.

The *Wait* action (Fig. 2) guarantees that robots are triggered at different times, since by the time a robot's waiting timer expires, other working robots might have already lowered the $S_{(t)}$ value, fulfilling the $(S_{(t)} < \theta)$ condition according to equation (10) and thus inhibiting the activation $(P_f = 0)$ of that specific robot.

4.1 Robustness against sensor noise

A supplementary set of simulation experiments was carried out in order to analyse the behaviour of SARTM and FRTM in the presence of faulty sensor readings. Previous sections were useful in order to explain the fundamental differences regarding both methods and their performance levels. However, all these sections assumed that robots were able to sense, explicitly and without errors, the exact amount of food $F_{(t)}$ at the nest in order to calculate their corresponding response curves.

In more realistic scenarios, robot sensors are likely to suffer from noisy or faulty readings, leading to incorrect estimates of the environment and hence unexpected robot behaviours. $F_{(t)}$ then becomes: $F_{(t)} = F_{(t)} + g(\mu, \sigma)$, where $g(\mu, \sigma)$ is a random value following a normal distribution perceived by the robots while executing the *Wait action* near the nest. μ is the mean of the normal distribution and σ its standard deviation. In order to test the robustness of both methods against noise, we assigned σ a relatively high value — around 1.5 times higher than the highest consumption rate — C_{rate} — value used in these experiments: 0.035 food

tokens/sec. Fig. 23 and Fig. 24 show the results obtained after conducting 20 simulations for each model, with the same parameter settings introduced in Section 3 and with a noise distribution characterised by $\mu = 0$ and $\sigma = 0.05$.

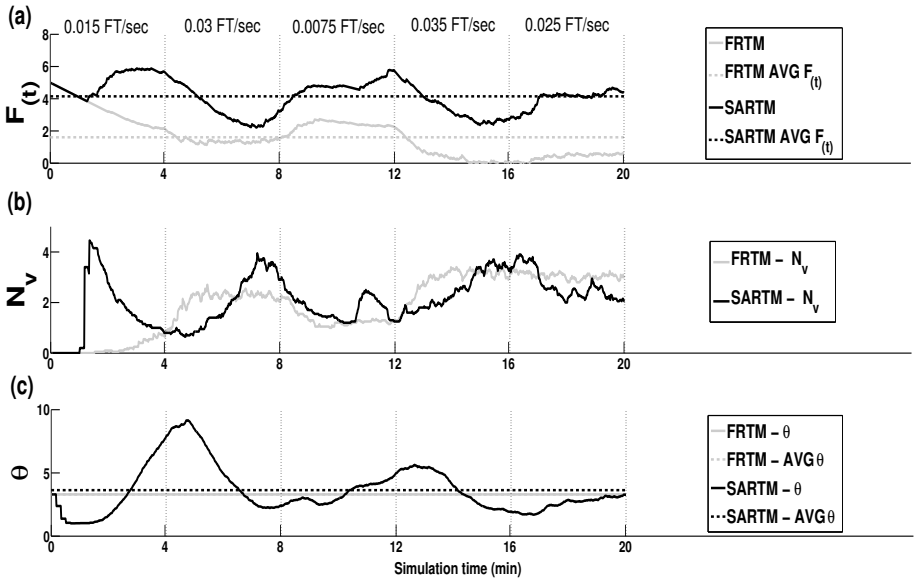


Fig. 23 Average results over 20 simulation runs in which a Gaussian noise component — $g(\mu, \sigma)$ — has been added to the perceived food levels by robots — $F_{(t)}$. (a) Amount of food robots could maintain at the nest for each different value of C_{rate} applied during the experiments. (b) N_v throughout the simulation experiment time. (c) Fluctuation of θ throughout experiments.

According to Fig. 23 (a) an adaptive response threshold model such as SARTM is still able to maintain higher food levels at the nest than FRTM and therefore improve the swarm's survival rate, see Fig. 24 (c). Even though these results correlate to already presented results in which perfect readings were available, one of the main differences with previous experiments is the increase of the average value of the θ threshold for SARTM. In Fig. 21 (c) — where no noise was involved — the average θ value for SARTM was 3.15, while in Fig. 23 (c) the average value of θ was able to reach 3.6. This effect suggests that the addition of noise in the food sensor readings might increase the average θ threshold values, especially if σ is larger than the C_{rate} currently active at the nest, since the effect of noise might cancel the action of the current C_{rate}. As explained previously, the adjustment of the θ threshold has a major impact on the S_{rate} values achieved by adaptive response threshold algorithms (Fig. 20). For this reason, we decided to measure the survival rates — S_{rate} — of robot swarms using SARTM and different noise distributions.

In Fig. 25, two bar groups can be seen. Each bar represents the survival rate achieved over 20 simulation runs. The first group represents simulations conducted with FRTM as division of labour mechanism and the latter group with SARTM. In each group, the first bar represents simulations with $\mu = 0$ and $\sigma = 0.05$, the

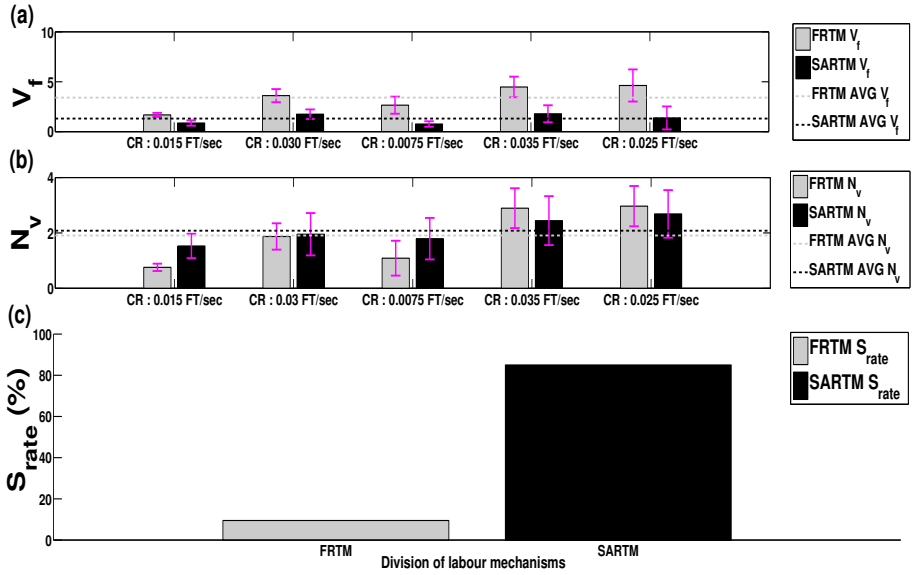


Fig. 24 Average results and standard deviations for 20 simulation experiments using the same parameter set as in Fig. 23. (a) V_f — average deviation of food — and its standard deviation values (error bars). (b) N_v — average number of working robots — and its standard deviation values (error bars) during the five stages of the simulation experiments. (c) S_{rate} — survival rate — of the whole system.

second $\mu = 0$ and $\sigma = 0.10$ and the third $\mu = 0$ and $\sigma = 0.25$. As can be seen in Fig. 25, FRTM shows low levels of survivability in all cases, and the increasing amount of noise does not produce a significant change in the bars' progression. This is possible due to the fixed threshold nature of FTRM, since noise is only able to affect the calculation of the stimulus — $S_{(t)}$ — value. However, the SARTM group shows a clear reduction in the survival rate — S_{rate} — levels as we increase the noise level. This is due to the fact that increasing the amount of noise in food readings cancels and even counter-balances the effect of the current C_{rate}. Therefore, as we increase the amount of noise in a robot's food sensing capabilities, the average θ threshold also tends to increase. Fig 25 shows that as we increase the noise magnitude in the food sensor readings, we obtain a higher average θ threshold.

As explained in previous sections, models such as ARTM and SARTM can adapt more easily to changes in the environment and dynamical situations than their fixed thresholds counterparts. However, the addition of noise in the stimulus sensing — in this case food levels at the nest — might lead not only to incorrect P_f values but also to noisy θ threshold values affecting the efficiency of the adaptive algorithms proposed.

5 Discussion

Although FRTM and ARTM were tested under the same conditions, there are clear differences in their ability to adapt to the environmental conditions in real-

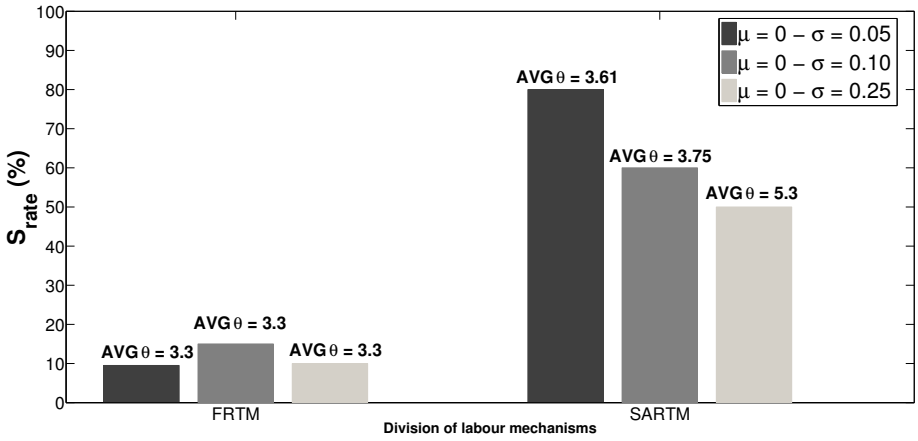


Fig. 25 ARTM and FRTM survival rates (S_{rate}) measured after 20 simulations runs for each experimental condition. SARTM decreases its survivability as the food sensor noise is increased. The addition of noise distributions with higher σ produce an increment in the average θ threshold.

robot experiments. Firstly, there is a difference in the average level of food each method can maintain at the swarm nest (shown with discontinued lines in Fig. 10 (a)). Fig. 10 (a) clearly shows that our proposed method is able to adapt better to different consumption rates and therefore maintain food values at the nest closer to its initial level $F_{(t_0)}$.

Secondly, the optimised division of labour achieved by ARTM leads to a clear improvement on the survival rate (S_{rate}): our proposed method is able to obtain (80%) compared to FRTM (0%) in real-robot experiments.

Moreover, the fact that ARTM can influence the collision duration (32 seconds average for ARTM vs. 187 seconds average for FRTM) gives an additional and unexpected advantage, especially under real-robot conditions, for ARTM against FRTM.

Goldberg and Matarić (1997) show that having a homogeneous population of robots allows reducing the average task execution time, which is a crucial element in time-critical foraging (i.e when robots' power is time-limited or the items to be collected must be disposed of as soon as possible). However, this brings as a drawback an increase in the collision time (called interference time in Goldberg and Matarić (1997)). In this paper, we show that the effect of this kind of drawback can be reduced by simple modification of the original FRTM algorithm. ARTM demonstrates adaptation and good performance even in real-robots environments. In other words, we saw an example of this automatic adaptation for the specific case of collision duration (Fig. 14 and Fig. 16).

In addition, by identifying the relevance of the n distribution in Response Threshold Models we are able to spot and explain the effects of n within the foraging process for these kind of models. Based on the results depicted in Fig. 18, and in particular in Fig. 18(a), the performance of a robot swarm can be compromised if the values of n assigned to the robots are too homogeneous, as becomes very clear observing the capacity of the system of keeping a positive

amount of food in the nest (Fig. 20). This phenomenon is clearly more likely to occur for a low number of robots. Indeed, by the law of large numbers, as the swarm size increases the probability of obtaining uneven distributions of n decreases, and thus performance improvement with increasing number of robots can be expected. This paper uncovers and proposes the possibility of creating a simplified labour division model, in which the consequences of an "inconvenient" n distribution — even for very small swarm sizes — could be avoided.

SARTM opens a new path towards achieving important capabilities for bio-inspired robotics such as adaptation to dynamic environments and self-sustainability. These capabilities are believed to have a crucial role in future applications of swarm and multi-agent systems, such as smart manufacturing (Bolmsjö et al., 2012), infrastructure maintenance (Morozovsky and Bewley, 2013), or medical applications using micro-scale robots (Suter et al., 2013).

On the other hand, the results presented in Section 4.1 show the dependency of SARTM to a certain quality of sensor readings in order to perceive the robot's stimulus. Even though adaptive methods such as SARTM can keep good survivability levels in the presence of "moderate" noise distributions, their survival rates decrease as the error in their readings becomes larger. SARTM and ARTM do not only rely on stimulus values to calculate response curves such as fixed methods do, but also on the fine adjustment of their θ thresholds, which makes them especially vulnerable to this kind of issue. A possible solution to this problem might be to include a Kalman filter or other form of filtering system in order to cancel the stimulus reading noise and make robots using ARTM/SARTM more robust against these kind of situations.

6 Conclusions

This paper presented a division of labour algorithm for a simple foraging task that acts to maintain a target amount of food at the nest despite consumption rates that vary over time. It has been shown to be efficacious in achieving adaptive workload distribution for a small size robot swarm. Real-robot and simulation experiments confirm that ARTM is able to adapt the number of active robots, increasing the system's survival rate (S_{rate}) in highly dynamic foraging missions. Moreover, it was shown that ARTM reduces a common problem of real-robot experiments in swarm robotics, i.e., the duration of collisions among robots. Finally, SARTM, a simplified version of ARTM, was proposed in order to improve the adaptation and emergent capabilities of robotic swarms in which the response threshold is calculated dynamically. The simplicity of this new algorithm makes it a good candidate as a basis of more advanced algorithms. As future work, we will focus on an improved version of SARTM, which can better cope with noisy sensor readings. This extended new model is intended to be employed in multi-nest scenarios within bigger arenas where different swarm sizes and food densities will be studied.

Acknowledgements All real-robot experiments described in the research were conducted within the Swarm Robotics Group at BRL (Bristol Robotics Laboratory). This research was supported by "Program for Leading Graduate Schools" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

Agassounon W., Martinoli A. (2002). Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3, ACM, New York, NY, USA, AAMAS '02, pp. 1090–1097, DOI 10.1145/545056.545077

Ashikaga M., Kikuchi M., Hiraguchi T., Sakura M., Anonuma H., Ota J. (2007). Foraging task of multiple mobile robots in a dynamic environment using adaptive behavior in crickets. Journal of Robotics and Mechatronics 19(4):446–473

Bailong L., Rubo Z., Changting S. (2008). Response threshold model of aggregation in a swarm: A theoretical and simulative comparison. In IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence), pp. 1103–1109, DOI 10.1109/CEC.2008.4630934

Beshers S. N., Fewell J. H. (2001). Models of division of labor in social insects. Annual Review of Entomology 46(1):413–440, DOI 10.1146/annurev.ento.46.1.413

Bolmsjö G., Danielsson F., Svensson B. (2012). Collaborative robots to support flexible operation in a manufacturing system. In Proceedings of the Flexible Automation and Intelligent Manufacturing, FAIM 2012, Tampere University of Technology, Department of Production Engineering, pp. 531–538

Bonabeau E., Theraulaz G., Deneubourg J.-L. (1998). Fixed response thresholds and the regulation of division of labor in insect societies. Bulletin of Mathematical Biology 60(4):753–807, DOI 10.1006/bulm.1998.0041

Castello E., Yamamoto T., Nakamura Y., Ishiguro H. (2013). Task allocation for a robotic swarm based on an adaptive response threshold model. In 2013 13th International Conference on Control, Automation and Systems (ICCAS), IEEE, pp. 259–266, DOI 10.1109/ICCAS.2013.6703905

Castello E., Yamamoto T., Nakamura Y., Ishiguro H. (2014). Foraging optimization in swarm robotic systems based on an adaptive response threshold model. Advanced Robotics 28(20):1343–1356, DOI 10.1080/01691864.2014.939104

Chen J., Gauci M., Price M. J., Groß R. (2012). Segregation in swarms of e-puck robots based on the Brazil nut effect. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '12, pp. 163–170

Chitta S., Jones E., Ciocarlie M., Hsiao K. (2012). Mobile manipulation in unstructured environments: Perception, planning, and execution. IEEE Robotics Automation Magazine, 19(2):58–71, DOI 10.1109/MRA.2012.2191995

Cianci C. M., Raemy X., Pugh J., Martinoli A. (2007). Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics. In Proceedings of the 2nd International Conference on Swarm Robotics, Springer-Verlag, Berlin, Heidelberg, SAB'06, pp. 103–115

D'Andrea R. (2012). Guest editorial: A revolution in the warehouse: A retrospective on Kiva systems and the grand challenges ahead. IEEE Transactions on Automation Science and Engineering, 9(4):638–639, DOI 10.1109/TASE.2012.2214676

Escobedo R., Muro C., Spector L., Coppinger R. P. (2014). Group size, individual role differentiation and effectiveness of cooperation in a homogeneous group of

hunters. Journal of The Royal Society Interface 11(95), DOI 10.1098/rsif.2014.0204

Gerkey B. P., Matarić M. J. (2000). Murdoch: Publish/subscribe task allocation for heterogeneous agents. In Proceedings of the Fourth International Conference on Autonomous Agents, ACM, New York, NY, USA, AGENTS '00, pp. 203–204, DOI 10.1145/336595.337369

Gerkey B. P., Vaughan R. T., Howard A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In Proceedings of the 11th International Conference on Advanced Robotics, IEEE, pp. 317–323

Goldberg D., Matarić M. (1997). Interference as a tool for designing and evaluating multi-robot controllers. In Proceedings, AAAI-97, AAAI Press, pp. 637–642

Jin K., Liang P., Beni G. (1994). Stability of synchronized distributed control of discrete swarm structures. In 1994 IEEE International Conference on Robotics and Automation, 1994., pp. 1033–1038 vol.2, DOI 10.1109/ROBOT.1994.351221

Jun J.-H., Lee D.-W., Sim K.-B. (1999). Realization of cooperative strategies and swarm behavior in distributed autonomous robotic systems using artificial immune system. In 1999 IEEE International Conference on Systems, Man, and Cybernetics, IEEE SMC '99 Conference Proceedings, vol 6, pp. 614–619 vol.6, DOI 10.1109/ICSMC.1999.816622

Kanakia A., Klingner J., Correll N. (2014). A response threshold sigmoid function model for swarm robot collaboration. Proceedings of International Symposium on Distributed Autonomous Robotic Systems (DARS), Springer Tracts on Advanced Robotics, Daejeon, Korea (In press)

Kashiwagi A., Urabe I., Kaneko K., Yomo T. (2006). Adaptive response of a gene network to environmental changes by fitness-induced attractor selection. PloS one 1(1):e49

Keshmiri S., Payandeh S. (2011). A centralized framework to multi-robots formation control: Theory and application. In Proceedings o the CARE@AI 2009 and CARE@IAT 2010 International Conference on Collaborative Agents - Research and Development, Springer-Verlag, Berlin, Heidelberg, CARE@AI'09/CARE@IAT'10, pp. 85–98

Krieger M. J., Billeter J.-B. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. Robotics and Autonomous Systems 30(12):65–84, DOI 10.1016/S0921-8890(99)00065-2

Krieger M. J. B., Billeter J.-B., Keller L. (2000). Ant-like task allocation and recruitment in cooperative robots. Nature 406(6799):992–995

Labella T. H., Dorigo M., Deneubourg J.-L. (2006). Division of labor in a group of robots inspired by ants' foraging behavior. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 1(1):4–25, DOI 10.1145/1152934.1152936

Lee W., Kim D. (2014). Adaptive divison of labor in multi-robot system with minimum task switching. In ALIFE 14: Proceedings of the Fourteenth International Conference on the Systhesis and Simulation of Living Systems, MIT Press, pp. 750–756, DOI 10.7551/978-0-262-32621-6-ch120

Lerman K., Jones C., Galstyan A., Matarić M. J. (2006). Analysis of dynamic task allocation in multi-robot systems. The International Journal of Robotics Research 25(3):225–241, DOI 10.1177/0278364906063426

Liu C., Kroll A. (2012). A centralized multi-robot task allocation for industrial plant inspection by using A* and genetic algorithms. In Proceedings of the 11th International Conference on Artificial Intelligence and Soft Computing -

Volume Part II, Springer-Verlag, Berlin, Heidelberg, ICAISC'12, pp. 466–474, DOI 10.1007/978-3-642-29350-456

Liu W., Winfield A. F. T. (2010). Modeling and optimization of adaptive foraging in swarm robotic systems. The International Journal of Robotics Research 29(14):1743–1760

Liu W., Winfield A. F. T. (2011). Open-hardware e-puck linux extension board for experimental swarm robotics research. Microprocessors and Microsystems - Embedded Hardware Design 35(1):60–67

Liu Y., Passino K., Polycarpou M. (2003). Stability analysis of m-dimensional asynchronous swarms with a fixed communication topology. IEEE Transactions on Automatic Control 48(1):76–95, DOI 10.1109/TAC.2002.806657

Mondada F., Bonani M., Raemy X., Pugh J., Cianci C., Klaptocz A., Magnenat S., christophe Zufferey J., Floreano D., Martinoli A. (2009). The e-puck, a robot designed for education in engineering. In Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, IEEE, pp. 59–65

Morozovsky N., Bewley T. (2013). Skysweeper: A low dof, dynamic high wire robot. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ, pp. 2339–2344

Nurzaman S., Matsumoto Y., Nakamura Y., Koizumi S., Ishiguro H. (2008). Yuragi-based adaptive searching behavior in mobile robot: From bacterial chemotaxis to levy walk. In International Conference on Robotics and Biomimetics, 2008. ROBIO 2008, IEEE, pp. 806–811, DOI 10.1109/ROBIO.2009.4913103

Parker L. (1994). Alliance: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems. 'Advanced Robotic Systems and the Real World', IROS '94., IEEE/RSJ/GI, vol 2, pp. 776–783 vol.2, DOI 10.1109/IROS.1994.407550

Quigley M., Conley K., Gerkey B., Faust J., Foote T. B., Leibs J., Wheeler R., Ng A. Y. (2009). ROS: An open-source robot operating system. In Proc. Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA), IEEE

Reggente M., Lilienthal A. (2009). Using local wind information for gas distribution mapping in outdoor environments with a mobile robot. In IEEE Sensors 2009, pp. 1715–1720, DOI 10.1109/ICSENS.2009.5398498

Shimizu Y., Tsuru S., Ito Y., Ying B.-W., Yomo T. (2011). Stochastic switching induced adaptation in a starved escherichia coli population. PLoS ONE 6(9):e23,953

Song Z., Vaughan R. (2013). Sustainable robot foraging: Adaptive fine-grained multi-robot task allocation for maximum sustainable yield of biological resources. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ, pp. 3309–3316, DOI 10.1109/IROS.2013.6696827

Stergiopoulos Y., Tzes A. (2011). Decentralized swarm coordination: A combined coverage/connectivity approach. Journal of Intelligent & Robotic Systems 64(3-4):603–623, DOI 10.1007/s10846-010-9537-1

Suter M., Zhang L., Siringil E., Peters C., Luehmann T., Ergeneman O., Peyer K., Nelson B., Hierold C. (2013). Superparamagnetic microrobots: fabrication by two-photon polymerization and biocompatibility. Biomedical Microdevices 15(6):997–1003

Theraulaz G., Goss S., Gervet J., Deneubourg J.-L. (1990). Task differentiation in polistes wasp colonies: A model for self-organizing groups of robots. In Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats, MIT Press, Cambridge, MA, USA, pp. 346–355

Winfield A. F. (2009). Foraging robots. In Encyclopedia of Complexity and System Science, Springer, pp. 3682–3700

Yang Y., Zhou C., Tian Y. (2009). Swarm robots task allocation based on response threshold model. In 4th International Conference on Autonomous Robots and Agents, 2009. ICARA 2009., IEEE, pp. 171–176, DOI 10.1109/ICARA.2000.4803959

Yared R., Defago X., Wiesmann M. (2007). Collision prevention using group communication for asynchronous cooperative mobile robots. In 21st International Conference on Advanced Information Networking and Applications, 2007. AINA '07., IEEE, pp. 244–249, DOI 10.1109/AINA.2007.44

Yongming Y., Xihui C., Qingjun L., Yantao T. (2010). Swarm robots task allocation based on local communication. In 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE), IEEE, vol 5, pp. 415–418, DOI 10.1109/CMCE.2010.5609944

Zhang G.-Y., Zeng J.-C. (2012). A discrete stochastic process for analysis of terrain coverage algorithm based on wasp swarm. In 2012 International Conference on Computing, Measurement, Control and Sensor Network (CMCSN), IEEE, pp. 191–194, DOI 10.1109/CMCSN.2012.49