Original paper

Adaptive Behavior

Adaptive Behavior
2016, Vol. 24(2) 87–101

DOI: 10.1177/1059712316633028
adb.sagepub.com
SAGE

Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition

Payam Zahadat and Thomas Schmickl

Abstract

In this paper, a distributed algorithm for adaptive task allocation and adaptable partitioning of a swarm into different work-groups is proposed and used in a swarm of underwater robots. The algorithm is based on local interactions of agents and is inspired by honeybee age-polyethism. It is adaptive to changes in the swarm size (workforce) and relative demands (workload) for different tasks and it limits the number of switchings of agents between different tasks enabling specialization of agents. The preliminary version of the algorithm was introduced previously. Here a fully decentralized version of the algorithm is proposed that improves the previous version by removing the need for global information. The algorithm is successfully implemented in swarms of physically-embodied underwater robots while the swarm size and the demands for the tasks change over the course of the experiments.

Keywords

Swarm intelligence, swarm robotics, adaptive division of labor, bio-inspired algorithm, social inhibition, distributed partitioning

1 Introduction

Social insects are providing inspiration sources in the field of swarm intelligence and collective robotics due to their high capability for self-regulation and self-organization. Their swarms are highly adaptive to changes in the environmental and internal conditions enabling them to survive dramatic changes and harsh conditions. One of the interesting and prominent characteristics of social insects is their capability to perform adaptive division of labor and efficient assignment of tasks to different members of the swarm. For example, colonies of honeybees (Huang & Robinson, 1992; Seeley, 1982), wasps (Torres, Montagna, Raizer, & Antonialli-Junior, 2012), termites (Crosland, Ren, & Traniello, 2010) and ants (Hölldobler & Wilson, 2008; Julian & Cahan, 1999) maintain plasticity in the structure of their colonies. That means, the colonies can adapt in an acceptable time to sudden changes in the internal status (e.g. losing many members who were taking care of a particular task), or environmental conditions (e.g. changes in the availability of food). Apart from the high adaptability and quick response of the colonies to such changes, the single members of the colonies do not need to switch between different tasks very often. Hence, such insect societies reduce the costs of task switching and allow for specialization in particular tasks.

In swarm robotics, where several relatively simple robots are expected to self-coordinate their tasks distributively, such social insect colonies are inspirational for many researchers (Navarro & Matía, 2013; Sahin, 2005). For example, in Jones and Mataric (2003); White and Helferty (2005); Yang, Zhou, and Tian (2009), a model called response-threshold reinforcement (Bonabeau, Sobkowski, Theraulaz, & Deneubourg, 1997; Theraulaz, Bonabeau, & Deneubourg, 1998) is inspired from wasps and applied in a robotic swarm. In Schmickl, Möslinger, and Crailsheim (2007) a trophallaxis-inspired strategy inspired by food exchange of honeybees and ants is applied to a swarm of robots in simulation. In Gross, Nouyan, Bonani, Mondada, and Dorigo (2008) and Labella, Dorigo, and Deneubourg (2006), models of ants-foraging behavior maintain division of labor in a group of robots to perform an object retrieval task. Adaptive division of labor

University of Graz, Austria

Corresponding author:
Payam Zahadat, Universitätsplatz 2, 8010 Graz, Austria.
Email: payam.zahadat@uni-graz.at

in object retrieval has also been used as a case study in Ferrante, Turgut, Duez-Guzmn, Dorigo, and Wenseleers (2015) where evolutionary computation finds solutions with or without task specialization depending to environmental features.

Several algorithmic models have been proposed in order to explain the mechanisms regulating task allocation and division of labor in social insects. For example, foraging-for-work (Tofts, 1993), response-threshold reinforcement (Bonabeau et al., 1997; Theraulaz et al., 1998), and common-stomach models (Karsai & Schmickl, 2011) (see Beshers & Fewell, 2001, for a detailed review of such models). A subset of the proposed models is based on a concept called social inhibition (Huang & Robinson, 1992, 1999). In the social inhibition models, presence of workers in a task leads to inhibitory effects in the behavioral development of other workers.

Here we propose a distributed algorithm called Partitioning Social Inhibition (PSI) inspired from social inhibition in age-polyethism of honeybees. A preliminary version of the algorithm was proposed previously (Zahadat, Crailsheim, & Schmickl, 2013; Zahadat, Hahshold, Thenius, Crailsheim, & Schmickl, 2015). Here we keep the simple logic of the preliminary version, reducing the shortcomings and extending the algorithm to a more distributed version. The PSI algorithm maintains division of labor and allocation of tasks to different members of a swarm. It is adaptive to changes in the swarm size and relative demands for different tasks. In addition, the number of switchings of agents between different tasks is limited enabling specialization of the members in their tasks.

In the following sections, first we briefly describe the mechanism of social inhibition in honeybees. Then the proposed algorithm is described. The simulation results of the algorithm are then presented in the next section for swarms of a large size taking care of several tasks, and the results of the simulation with different swarm sizes and communication ranges are presented. Then, the results from the implementation of the algorithm in a swarm of physical AUV (Autonomous Underwater Vehicle) robots is presented in order to show the applicability of the algorithm in real embodied systems. The results indicate a good adaptability of the swarm to changes in the number of available robots and relative demands in the presence of real-world noise, along with a low number of switchings between the tasks.[1]

2 Social inhibition in honeybees

A worker *honeybee* goes through a behavioral development during its life-time, taking different tasks in different stages of its life. In a normal situation, a young worker honeybee starts with inside-hive tasks such as cleaning the comb. Then it grows older and undertakes tasks like nursing of the brood and storing food. At some point in its life when the worker is old enough, it develops to an out-of-hive worker who performs foraging for nectar and pollen (Johnson, 2010; Robinson, 1992; Wilson, 1971). Figure 1 shows a brief illustration of the behavioral development of a worker honeybee.

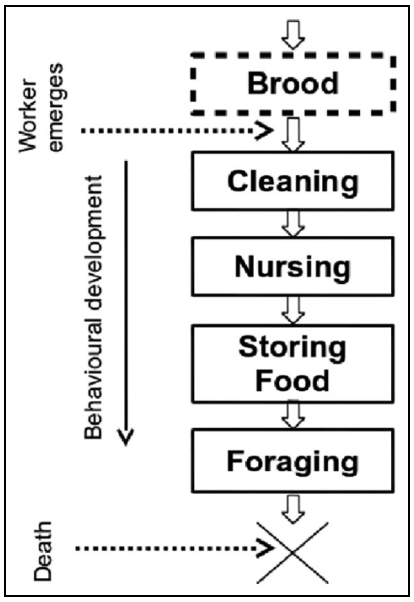

Figure 1. A brief illustration of behavioral development in a honeybee in normal situations.

According to the task the worker undertakes, a number of morphological and physiological changes occur in its body (Free & Spencer-Booth, 1959; Hrassnigg, 2005; Wilson, 1971). This mechanism of behavioral development is called *age-polyethism* (or *temporal polyethism*). In a normal situation, the tasks that the bee performs, and in turn, its so-called physiological age, are correlated to the chronological age of the bee Beshers, Robinson, and Mittenthal (1999); Crailsheim and Stabentheiner (1996); Toth, Kantarovich, Meisel, and Robinson (2005); Winston (1987). However, this correlation can be violated. The behavioral development can be delayed, accelerated, or even reversed in response to changes in the internal or environmental conditions of the colony. The adaptability of the colony to changes in the age-distribution and balance of the demands for different tasks (Huang & Robinson, 1996) is a result of the adaptability in the behavioral development. In a colony of mostly young *honeybee*s, the age in which a bee starts foraging (which is a typical out-of-hive task) is lower than in a normal colony indicating acceleration in the behavioral development. In the same way, the presence of many older bees in the colony slows down or inhibits the development of the physiological age of other bees in the colony. Another example is when the in-hive workers are removed from a colony, the development of physiological age inverts

resulting in transformation of out-of-hive workers into in-hive workers.

It has been proposed (Huang & Robinson, 1992) that the hormonal regulations in *honeybee*s driven by interactions between the workers, leads to social inhibition that explains age-polyethism and adaptability of the colony to different age distributions. The concept of social inhibition has been employed in different models, e.g., (Beshers, 2001; Naug, 1999). In this work (Naug & Gadagkar, 1999) is not used as a source of inspiration due to some technical reasons.

3 Partitioning social inhibition algorithm

The aim of PSI is to decentrally split a swarm of agents into subgroups where each subgroup is assigned to a task and the size of the subgroups are chosen according to the relative demands for the tasks. The swarm needs to be adaptive to changes in the swarm size and the relative demands for the tasks and it should prevent many switchings of the agents between different tasks.

Similar to honeybees, in PSI, tasks are considered to be ordered in a sequence such that an agent is only allowed to switch to the previous or the next task in the sequence every time it updates its decision. Each agent in the swarm contains a state variable x emulating the physiological age of a honeybee. The x variable holds a value in the range of $(x_{\min}, x_{\max})$. Similar to the honeybees that can get physiologically older or younger depending on the situations in the swarm, the x values can increase or decrease over time.

The main idea of the algorithm is to dynamically distribute the x values of the swarm members over the fixed range of $(x_{\min}, x_{\max})$ via local interactions and let the agents choose their tasks based on their x values.

The $(x_{\min}, x_{\max})$ range is split into a number of segments, each of which is associated with one of the tasks in the task sequence. An agent chooses a particular task if the value of its state variable (x) falls into the range of the segment associated with that task. Figure 2 represents an example swarm divided into four task groups based on the state variables of its agents.

The distribution of the state variables over the range is uniform only if the demands for all the tasks are equal. Otherwise, the density of the distribution is higher in the segments with higher associated task demands and lower in the segments with lower associated task demands. Figure 3(c) represents an example of the distribution where there are different demands for the tasks.

In order to implement the idea of the algorithm, a set of thresholds are defined that split the range $(x_{\min}, x_{\max})$ into the task segments. An agent chooses its task based on its x value and the defined thresholds (see Figure 2).

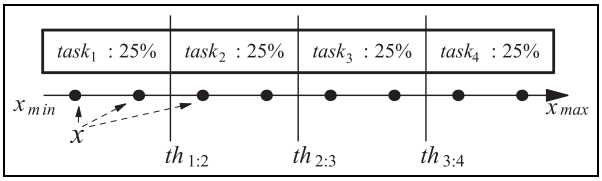

Figure 2. An example of uniform distribution of x values over the range of $(x_{\min}, x_{\max})$. Three threshold values $(th_{1:2}, th_{2:3}, th_{3:4})$ split the range into four segments associated with four different tasks. In this example, there is equal demand for all the tasks and the eight agents are divided into four groups of two agents each.

The thresholds are augmented with a lower and an upper margin. That means, an agent in $task_i$ switches to $task_{i+1}$ if its x value exceeds $th_{i:i+1} + l_u$. The agent switches to $task_{i-1}$, if its x value becomes lower than $th_{i-1:i} - l_b$. The lower and upper margins prevent the agents from instant back and forth switches between two consecutive tasks due to noise.

The aim is to divide the swarm into the task groups such that the number of agents in different groups correlates with the demands for different tasks. In the preliminary version of the algorithm (Zahadat et al., 2013), this goal was achieved by maintaining a uniform distribution of the x values in the swarm and using segments of different sizes, where the size of the segments were proportional to the demands for the tasks (See Figure 3(b) for an example). The problem with that solution is that every change in the demand of a task requires changing of all the threshold values in the agents' memories all over the swarm. This global information about the new values of the thresholds deviates from the desire for distributiveness and was the main limitation of the previous algorithm.

In the improved version of the algorithm, the problem is solved by keeping the thresholds - and consequently the size of segments - fixed (e.g. to equal segments), and instead, maintaining different densities in the segments proportional to the relative demands for the tasks (See Figure 3(c)). In order to do that, each agent keeps a value w correlated to the relative demand of its task. This value basically represents the expected relative density of agents in that task. The w values are initially set by the user based on their initial assumptions about the relative demands for the tasks (e.g. set all to 1 for equal demands). The values can be changed by the agents according to their perception of the task demands (for example, deviation from the assumed demands due to changes in the demands). The w values are used (along with other variables) to let the agents tune their x values (more details in the next subsection).

Figure 3 shows an example comparing the previous and the improved versions of the algorithm. In Figure 3(a), the agents are distributed uniformly and the thresholds are set to split the range into equal segments due to equal demand for the different tasks (both versions behave the same way in this case). Figures 3(b)

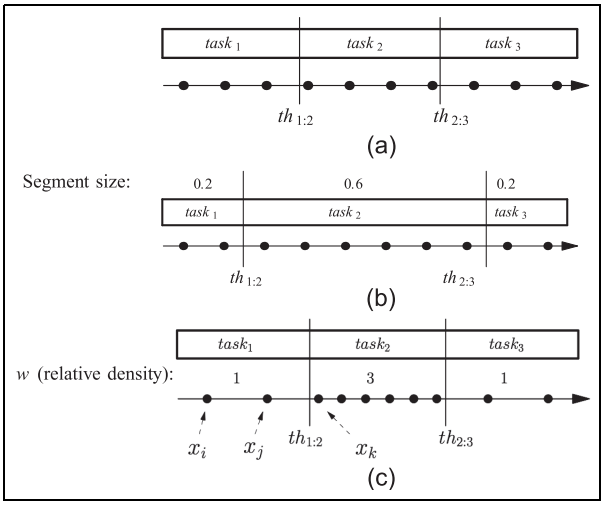

Figure 3. a) Uniform distribution of x values with equal demands (both the segment sizes and the relative densities are equal in all the tasks). b) Previous version: uniform distribution of x values with a change in relative demands and corresponding changes in the segment sizes. c) Improved version: change in relative demands and corresponding change in the density of agents in different segments.

and 3(c) depict the results of a change in the relative demands of the tasks in the previous version of the algorithm and the proposed version respectively. In Figure 3(b), the distribution of the agents stays uniform after the change in the demands from equal demands to higher demand for $task_2$. In that case, all the agents are globally informed to update their thresholds to new values according to the new demands for the tasks. Figure 3(c) depicts the new version of the algorithm where the thresholds stay constant but the changes in the demands are reflected in the w of the agents in each task and lead to the change in the relative density of agents. In this case, for example, the relative demand for $task_2$ is three times as much as $task_1$ (and $task_3$) resulting in three times as much density in $task_2$ as the density of agents in $task_1$ (and $task_3$).

By using the improved version of the algorithm, not only the adaptability to the changes in the swarm size (workforce) is maintained as in the previous version, but the local perception of the demands are also automatically reflected in the swarm by locally changing the w of the tasks within the agents. There is no need for global information to update the thresholds.

In the following, the implementation of the algorithm in the level of agents is described.

3.1 PSI algorithm in the agent level

Since the agents have no global access to the status of the swarm, the x values of the agents change via local interactions with other agents over time. In addition to x and w, every agent needs to keep its own estimations of the effective distances between its x value and the closest lower and higher x values of the other agents. The estimated effective distances, namely d^{lower} and d^{higher}, change over time and during interactions with other agents and allow the agents to dynamically update their x values and consequently their choice of tasks.

When an agent meets another one, it receives the x and w values of the other agent. The effective distance between the x values of the two agents is the distance between the two values, while the ratio between the w of the two agents is taken into account. For example, in Figure 3(c), agents i and j are in the same task (x_i and x_j are within the segment of $task_1$). Assuming that the two agents have the correct estimation of the demands for their task ($w_i = w_j = 1$), the ratio between the w of the two agents is $\frac{w_j}{w_i} = 1$ and the effective distance between x_i and x_j is simply calculated as the absolute difference between the two values, $|x_i - x_j|$. On the other hand, since the expected densities of $task_1$ and $task_2$ are different, the ratio between w_k and w_j is not 1. If the agent j tries to compute the effective distance between its x value (x_j) and x_k, it will sum the distance between x_j and $th_{1:2}$ with the distance between $th_{1:2}$ and x_k multiplied by $\frac{w_k}{w_j}$.

In a formal representation, if an agent i that performs task t meets an agent j, the effective distance between x_i and x_j is computed as follows

$$d^{\text{effective}} = |D - x_i + \frac{w_j}{w_i}(x_j - th)| \tag{1}$$

where

$$D = \begin{cases} th_{t-1:t} & \text{if } x_j < x_i \\ th_{t:t+1} & \text{otherwise} \end{cases} \tag{2}$$

According to the above equations, if both agents work in the same task, $d^{\text{effective}}$ is simply $|x_j - x_i|$.

The estimated effective distances between x_i and the closest lower and higher x values of the other agents (namely d_i^{lower} and d_i^{higher}) get updated over time and during interactions with other agents.

If an agent i meets an agent j, d_i^{lower} and d_i^{higher} are updated based on their current values and the effective distances between x_i and x_j, as follows

$$d_i^{\text{lower}} = \begin{cases} d^{effective} & \text{if } x_j < x_i \text{ and } d^{effective} < d_i^{\text{lower}} \\ d_i^{\text{lower}} & \text{otherwise} \end{cases} \tag{3}$$

$$d_i^{\text{higher}} = \begin{cases} d^{effective} & \text{if } x_i < x_j \text{ and } d^{effective} < d_i^{\text{higher}} \\ d_i^{\text{higher}} & \text{otherwise} \end{cases} \tag{4}$$

Additionally, d_i^{lower} and d_i^{higher} slowly increase in every step in order to be adaptable to changes in x values of other agents as well as changes in the environmental

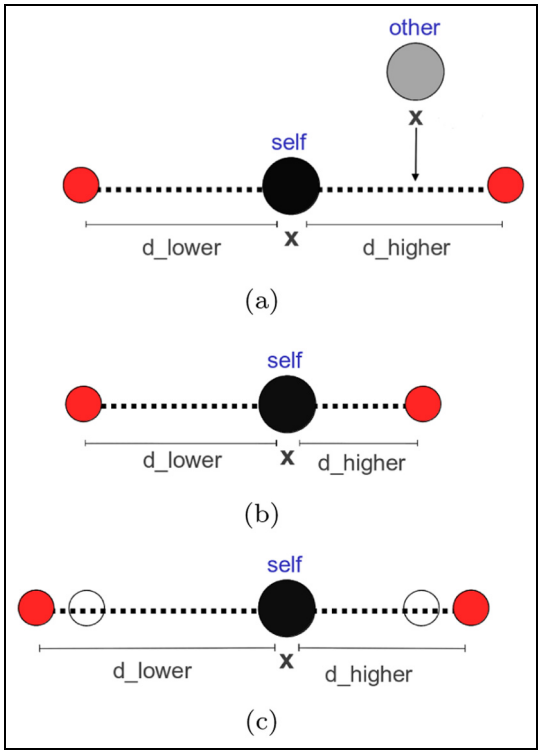

Figure 4. An example of d_{lower} and d_{higher} getting updated via interaction with another agent. For clarity of the example, the two agents in (a) are assumed to work in the same task (therefore the effective distance between the two x values is simply the absolute difference between them). (a) An agent receives data from another agent. (b) d_{higher} gets updated according to the value received from the other agent. (c) d_{higher} and d_{lower} slowly increase in every step.

conditions, i.e. changes in the workforce or task demands

$$\begin{aligned} d_i^{\text{lower}} &= d_i^{\text{lower}} + \varphi \\ d_i^{\text{higher}} &= d_i^{\text{higher}} + \varphi \end{aligned} \tag{5}$$

where φ is a small value in terms of segments sizes.

Figure 4 depicts an example update of d^{lower} and d^{higher} after receiving data from another agent.

Every agent tends to keep its x value equally distanced from the closest higher and lower x values of the others. Therefore, with every update of d^{lower} and d^{higher}, the x value is also updated, as follows

$$x = \begin{cases} x + \delta & \text{if } d^{\text{lower}} < d^{\text{higher}} \\ x - \delta & \text{if } d^{\text{lower}} > d^{\text{higher}} \\ x \pm X & \text{otherwise} \end{cases} \tag{6}$$

where δ is the *step-size* which is a constant parameter with a small value in terms of the size of task segments. In the current implementation $X \sim \delta \times U(0, 1)$.

After every update of x, an agent considers switching to the previous or the next tasks in the task sequence. For an agent in $task_t$, *new_task* is chosen as follows

$$new\_task = \begin{cases} task_{t+1} & \text{if } x > th_{t:t+1} + l_u \\ task_{t-1} & \text{if } x < th_{t-1:t} - l_b \\ task_t & \text{otherwise} \end{cases} \tag{7}$$

where $th_{t-1:t}$ and $th_{t:t+1}$ represent the threshold values between $task_{t-1}$ and $task_t$, and $task_t$ and $task_{t+1}$ respectively. l_u and l_b are the upper and lower margins for the thresholds.

Summarizing the PSI algorithm. The following actions are performed by any agent i that receives information from another agent j.

1. Update d^{lower} and d^{higher} using equation (5).
2. Update x using equation (6).
3. Update d^{lower} and d^{higher} using equations (3) and (4).
4. Update the assigned task using equation (7).

A heuristic used for updating delta-window (φ). In the robotic experiments reported in this paper, we applied the following heuristic to regulate φ based on the periods between the interactions for every agent. The value of φ is updated for an agent every time it receives information from another agent, as follows

$$\begin{aligned} \varphi &= \varphi_{\text{base}}/(T+1) \\ T &= (1-\alpha) \times T + \alpha * T_{\text{last}} \end{aligned} \tag{8}$$

where $\alpha = 0.1$, and T_{last} is the time (number of time-steps) since the last interaction occurred. φ_{base} is a constant parameter.

4 Experimenting in simulation

The proposed algorithm is preliminarily tested in an agent-based spatial simulation without embodiment.

First, two sets of experiments are performed in order to investigate the adaptability of the swarm to changes in the total number of available agents and the demands for different tasks. Then, several experiments with different swarm sizes and communication ranges are performed in order to give a better understanding of the performance of the algorithm in different setups.

The agents are located in a column with different horizontal layers associated with different tasks (similar to the robotic experiment in Figure 9) and they have a limited range of communication. The agents randomly drift around in their task layers with the average speed of 2 cm/s while every 7 time-steps is considered as one second.

All the agents start with an identical initial x value belonging to $task_1$. The range of x values is split into five equal segments by setting up the thresholds ($th_{1:2}$ to $th_{4:5}$) accordingly. In every time-step, each agent is allowed to communicate with a randomly chosen agent in its communication range. In all the following experiments, we assume that the agents can precisely estimate the relative demands for the tasks they perform.

Table 1. Parameter settings of the algorithm for simulation and robot experiments.

	Simulation	Robot
x_{min}	0	0
x_{max}	1024	$2^{14}-1$
Noise on x	0	0.5
δ	1	70
φ_{base}	0.3	210
l_u, l_b	3	140
Thresholds	Equal segments	Equal segments
Initial x	2	$thr_{1:2}-20$
Initial d_{lower}	$x-x_{min}-1$	$x-x_{min}-10$
Initial d_{higher}	$x_{max}-x-1$	$x_{max}-x-10$

All the experiments are repeated for 25 independent runs. The experimental settings of the algorithm are represented in Table 1.

The performance of the swarms are measured in the experiments by using two metrics: number of errors and number of switchings. The number of errors represent the difference between the required and actual number of agents in every task for all the tasks in total. The number of switchings is the number of unintended changes in the choice for the tasks counted over the whole swarm in a fixed period of time.

4.1 Adaptivity to changes in workforce

In the first set of experiments, the swarm is expected to take care of five tasks located in five different layers. The size of each layer is 40×40 cm^2 and the distance between the layers is 35 cm. The communication range is 50 cm. Each run continues for 70,000 time-steps.

In this set of experiments, there are equal demands for all the five tasks and the demands stay constant during the runs. The number of available agents is 30 in the beginning. The number of agents is then suddenly changed at two points of time during the runtime and the swarm is supposed to react to the changes by rearranging the agents in the tasks according to the new conditions.

The agents start in $task_1$ with initially identical x values which they regulate gradually via interactions with other agents. After some time, the x values are equally distributed in the swarm leading to an equal number of robots in each task due to the equal segmentation of the range by the task thresholds.

Figure 5(a) represents the number of agents in each task (median from 25 independent runs) and Figure 5(b) represents the total number of agents positioning in a wrong task according to the desired number of agents in the tasks.

In time-step 20,000, all the agents working in $task_3$ and $task_4$ are removed from the swarm. The swarm reacts to this deviation from uniformity of the distribution of x values by shifting the x values of the remaining agents, and ends up at a new equal number of agents (three agents) in each task.

In time-step 40,000, 12 agents are added to the swarm in $task_3$. In this case, the initial x values of the new agents are set to the average of the two thresholds separating $task_3$ from its neighboring tasks ($th_{2:3}$ and $th_{3:4}$). The swarm reacts to this change by regulating the x values for the new situation and after a while, six agents are assigned to each task.

4.2 Adaptivity to changes in demands

The adaptability of the swarm to the changes in the demands is investigated in this experiment. The swarm size is 30 and the number of tasks and spatial configurations are the same as in the previous section. The demands are equal for the five tasks in the beginning until time-step 30,000. The agents start in $task_1$ and after a while they equally distribute in the five tasks.

Figure 6(a) represents the number of agents in each task during the simulation period (median from 25 independent runs). At time-step 30,000, the demands change such that the demand for $task_2$ suddenly raises to 6 times as much as the starting demand. The new relative demands are considered as 1,6,1,1,1 for $task_1$ to $task_5$ accordingly. At time-step 50,000, the relative demands change abruptly again to 1,1,1,3,4 for $task_1$ to $task_5$ accordingly. Figure 6(b) represents the total number of agents performing a wrong task.

4.3 Number of switchings between tasks

The number of unintended switchings (switching to a wrong task) is counted in the period between 15,000 and 30,000 time-steps with the same settings as the previous experiment (Figure 6). The period is chosen since there is no external disturbance on the system in this period meaning that in an optimal case there should be no switchings between the layers. Figure 7 represents the fraction of the swarm with different numbers of switchings between the tasks, during the whole period of observation (15,000 time steps). The data are collected from 25 independent runs. As seen in the figure, around 40% of the swarm had less than 5 switchings during the period. 26% of the swarm did not have any switchings (not shown in the figure).

4.4 Investigating the performance of the algorithm in different configurations

In this set of experiments, the performance of the algorithm in terms of convergence-time and the number of unintended switchings is investigated in different setups with different swarm sizes and the range of communications.

The swarms are expected to take care of five tasks located in five different layers. In order to compare the results for different numbers of agents and

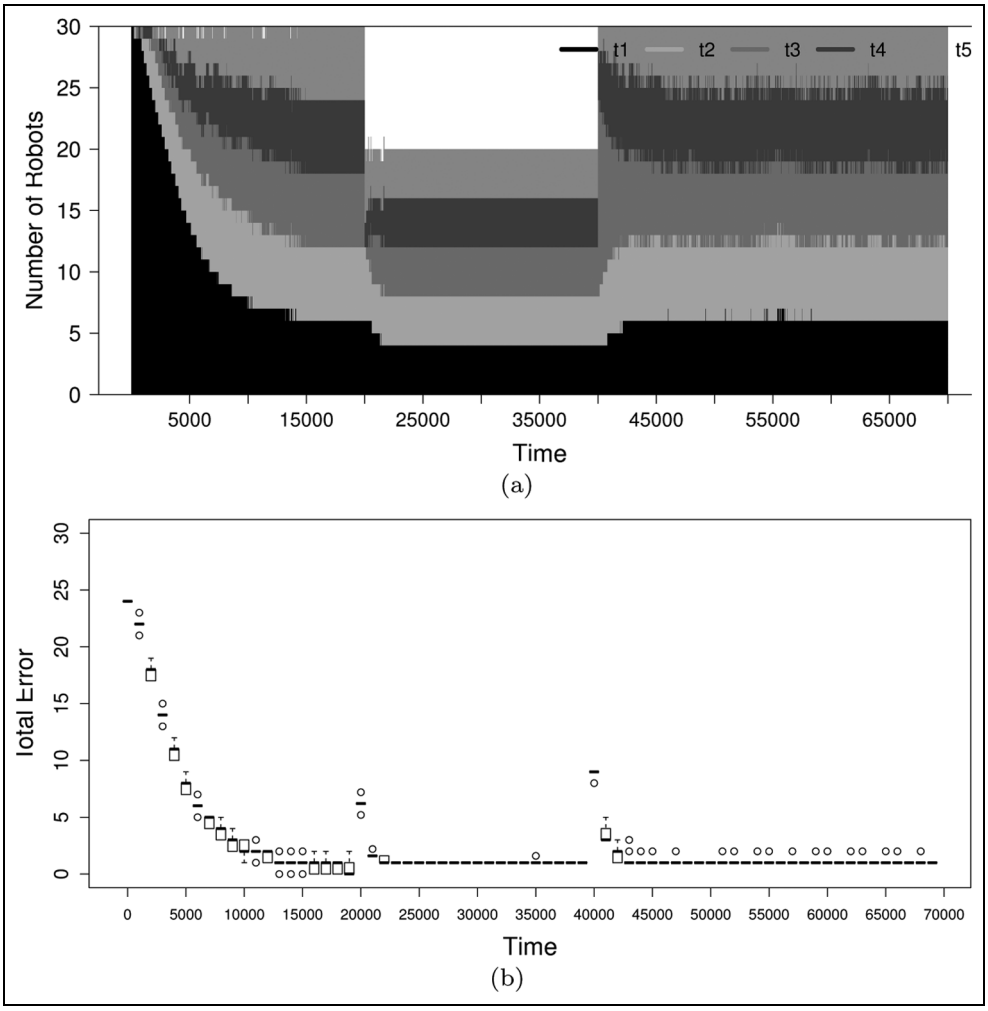

Figure 5. At the beginning, 30 agents start with the same initial x value belonging to $task_1$ (t1). The task demands are equal for all the tasks. In time-step 20,000, all the agents in t3 and t4 are removed from the experiment. In time-step 40,000, 12 agents are added to t3 (with initial x value in the middle of the range for t3). The data are collected from 25 independent runs and the median number of agents in each time-step is represented in the figure. (a) Number of robots in each task over time. (b) Total error (for the five tasks altogether) over time.

communication ranges, the size of the layers and the distance between the layers in this set of experiments are different from the experiments in the previous sections. The size of each layer is 80×80 cm^2 and the distance between the layers is 10 cm.

The experiments start with equal task demands and after a while when the swarm is equally partitioned into the five tasks, the relative demands for the tasks change to 1, 3, 1, 4, 1 for $task_1$ to $task_5$ accordingly. The convergence time is then measured as the period between the occurrence of the change of demands until there is at most 0.5% error in the number of agents in the tasks. The number of switchings of the agents between the different tasks is counted in a period of 10,000 time-steps after convergence of the swarm.

The communication range in different setups of the experiments is between 30 to 80 cm and the swarm sizes are between 10 to 60 agents. For every setup, the experiment is repeated for 25 independent times and the median of the values is depicted by a grey scale level in the figures. Figures 8(a) and 8(b) demonstrate the results. The lower the values (both the convergence time and the number of switchings), the better the performance of the swarm. The lower values are represented by brighter grey scale levels in the figures.

As seen in the figures, a very low swarm size and communication range, results in a low performance for both metrics. This is due to the fact that the swarm is too sparse and the agents may not hear from the other agents for several time-steps before they randomly get close enough to the others and are able to update their values accordingly. The increase in the population size increases the convergence time as expected. The convergence time increases with higher communication ranges where the swarm is larger than a minimum size (not too sparse). This might be due to the fact that with a higher communication range, the agents receive more diverse information leading to a longer time for tuning

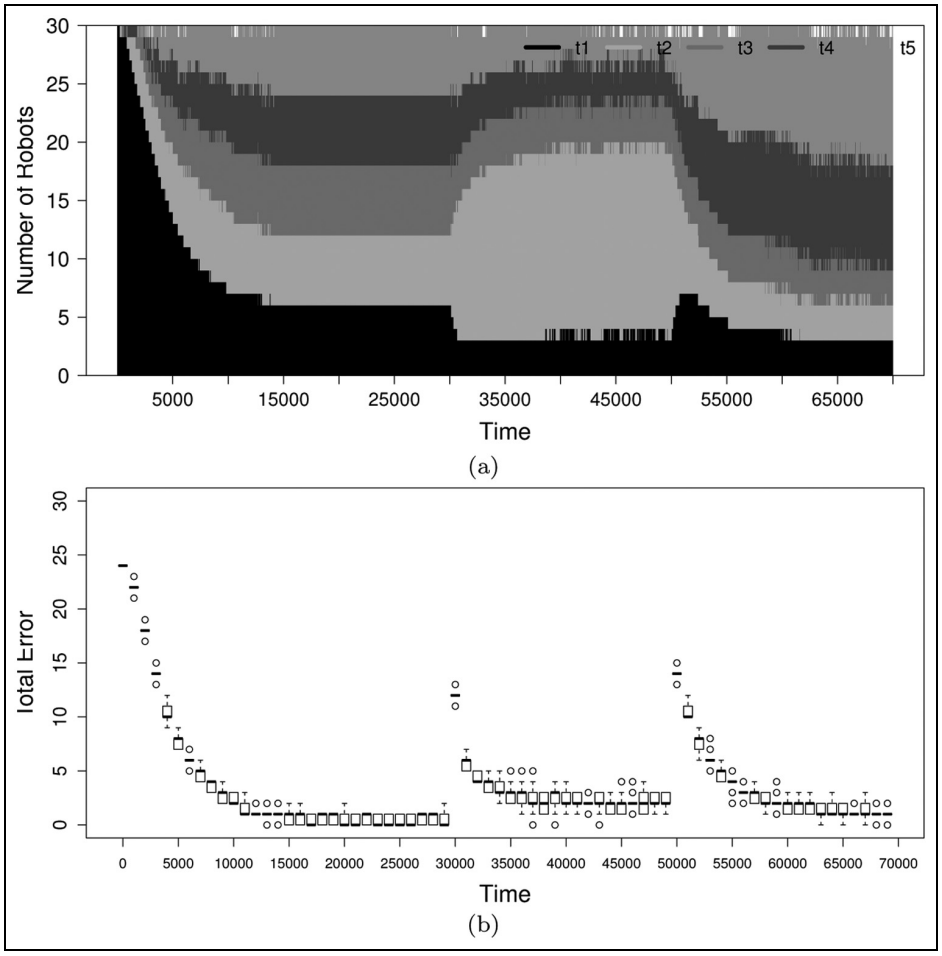

Figure 6. At the beginning, all the 30 agents start with the same initial x value belonging to $task_1$ (t1). The task demands are all equal, i.e. 1,1,1,1,1 for $t1$ to $t5$. In time-step 30,000, the demand for $task_2$ (t2) raises to 6 times that at the beginning (i.e. 1,6,1,1,1 for t1 to t5). In time-step 50,000, the demand of $task_2$ is back to the beginning while the demands for t4 and t5 raise (i.e. 1,1,1,3,4 for t1 to t6). The data are collected from 25 independent runs and the median number of robots in each time-step is represented in the figure. (a) Number of agents in each task over time. (b) Total error (for the five tasks altogether) over time.

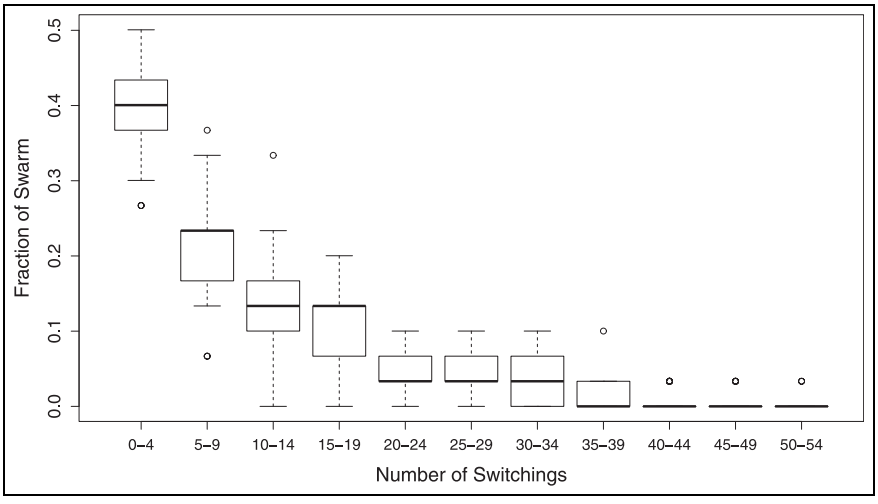

Figure 7. Number of switchings over 15,000 time-steps. Each column represents the fraction of the swarm that had a certain number of switchings over this period of time. The data are accumulated from 25 independent runs. Box-plots indicate median and quartiles, whiskers indicate the most extreme data points which are no more than 1.5 times the length of the box away from the box, circles indicate outliers.

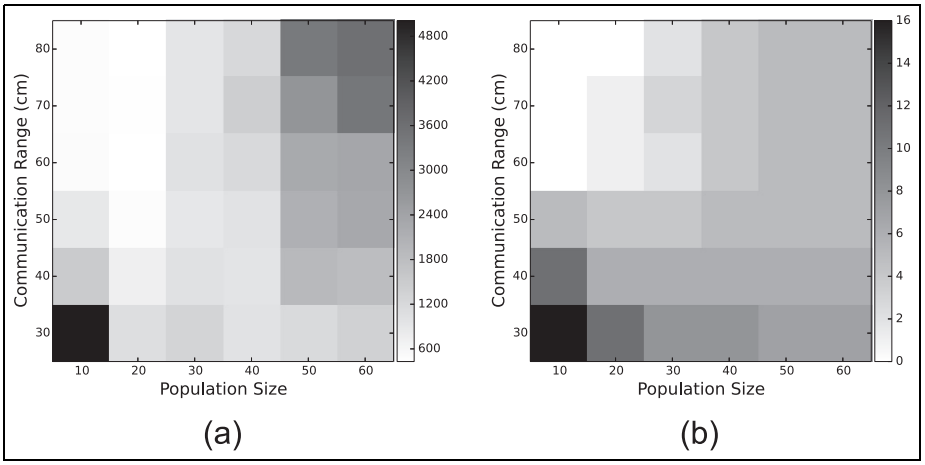

Figure 8. Convergence time (a) and the number of switchings (b) compared between different swarm sizes and communication ranges where the rest of the configuration parameters are the same.

their values. On the other hand, the converged status is more stable and makes less fluctuations in the status of the agents and therefore there are lower switchings between different tasks once the swarm is converged. Figure 8(b) also shows that the number of switchings is lower for smaller swarms once the range of communication is larger than a minimum value.

5 Experimenting with underwater robots

A swarm of "Lily" underwater AUV robots (Figure 9) developed in the project CoCoRo (CoCoRo, 2013; Dipper, 2015) was used to test the PSI algorithm in a real world platform in the presence of noise. In the following experiments, a group of up to nine robots were available in a column aquarium of size $1 \times 0.4 \times 0.4$ m^3. The task for the swarm was to split into three different sub-groups (representing sub-tasks) where every sub-group forms a separate layer (see Figure 9).

The robots have a size of about 15 cm. They use Radio Frequency (RF) for local communication under water. The RF range was about 50 cm and therefore there was no communication between the robots in the higher and the lower layers. The RF system was prone to noise. Every robot broadcasts two values detectable in the local neighborhood. The first value was a 14-bit integer representing the x value of the robot. The second part indicated the local estimation of the demand in the layer. All the robots were initialized with an identical x and run the proposed algorithm as summarized in Algorithm 1. The robotic experimental settings and the algorithmic parameters are presented in Table 2 and Table 1 respectively.

In the following experiments, first we investigate if the algorithm is capable of distributing the robots in the layers equally adaptive to the changes in the number of available robots. Then we investigate the adaptivity of the swarm to changes in the demand for the different layers.

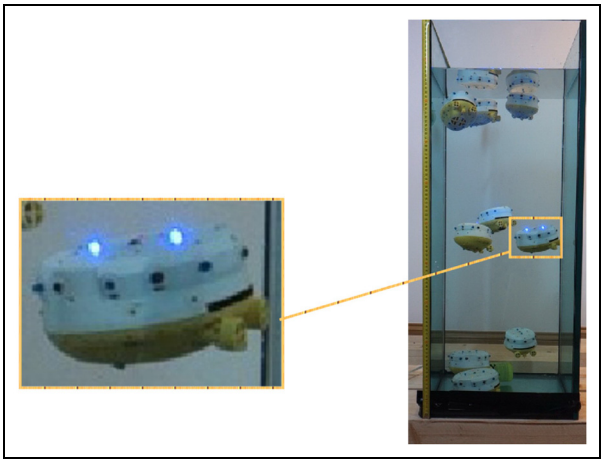

Figure 9. A screen-shot of the experiment with nine robots equally distributed in three layers (right) and a Lily robot (left) with its peripheral BL sensors and LEDs (black circles). Every third black dot on the periphery of the robot indicates a BL LED.

An increase in the demand in a layer is emulated by using a special robot in that layer hanging on a rope. The special robot pulses blue light (BL) on its peripheral LEDs indicating a source of interest in the layer. This way, the layer is marked as a layer with a higher demand for robots. The BL pulsing is detectable by the robots in the neighborhood. If a robot detects the BL pulsing, it also pulses on its peripheral BL LEDs (Light Emitter Diodes) and this way the other robots in the same layer can be informed about the interestingness of the layer. To a robot, detection of a BL pulse means a high workload in the layer in that point of time.

5.1 *Adaptivity to changes in workforce: Adding and removing robots in the swarm*

In the first part of the experiment, nine robots were available. The robots were put in the water one after

Algorithm 1 Robots' operational loop.

while *true* **do**
  if data available from other robots *__then__
    update d^{lower} and d^{higher} by equations (3) and (4)
  end if
  update x value by equation (6)
  increase d^{lower} and d^{higher} by equation (5)
  broadcast data (x, w)
  choose a layer based on x value by equation (7)
  if new layer is different from the current layer **then**
    change the target depth to the new layer
  end if
  if detect the high demand signal **then**
    set the w to a high value
    emit BL pulse (high demand signal)
  else
    set the w to a low value
end if
end while
**at most, one set of data can be accepted in every cycle due to limitations in the robot's operation.*

Table 2. Robotic experimental settings.

Parameter	Value
Length of an AUV	15 cm
Surface size of operation area	40 × 40 cm^2
Distance between the layers	~35 cm
Range of RF communication	~50 cm
Range of BL communication	~50 cm
Opening angle of BL sensors	120°
Level of noise	Unknown
Robot's cycle length	150 ms

the other. All the robots were started with an identical initial x value (Table 1) belonging to the top layer (L1). We observed the number of robots in each layer for 1025 seconds. Every time a robot passes the half distance between two neighboring layers, it is considered to be in the new layer.

Figure 10 represents the number of robots in the water over the period of experiment and the number of robots in every layer. After adding or removing a robot, the swarm rearranges to a new configuration close to the status where the number of robots in all layers are equal. In the beginning, the robots are added to the swarm one after the other until six robots are in the water. The swarm forms a three-layer configuration with two robots in each layer. We leave the swarm for ˜270 seconds during which no switching of the robots between the layers occur. After this period, three more robots are added to the swarm resulting in switching of some of the robots to new layers such that there are three robots in each layer. In the period between the first appearance of the new configuration and before making a new change in the swarm size (at ~690 seconds), three unintended switchings (switching to a wrong layer) occur. In second ~690, all three robots in the top layer (L1) are removed from the water almost at the same time. The swarm then rearranges such that each layer has two robots. This configuration stays until a new change in the swarm size is enforced by the experimenter. In second 950, the two robots in the top layer (L1) and one robot from the middle layer (L2) are removed. The three remaining robots relocate such that every layer contains a single robot. Table 3 represents the average density of robots in every layer over the

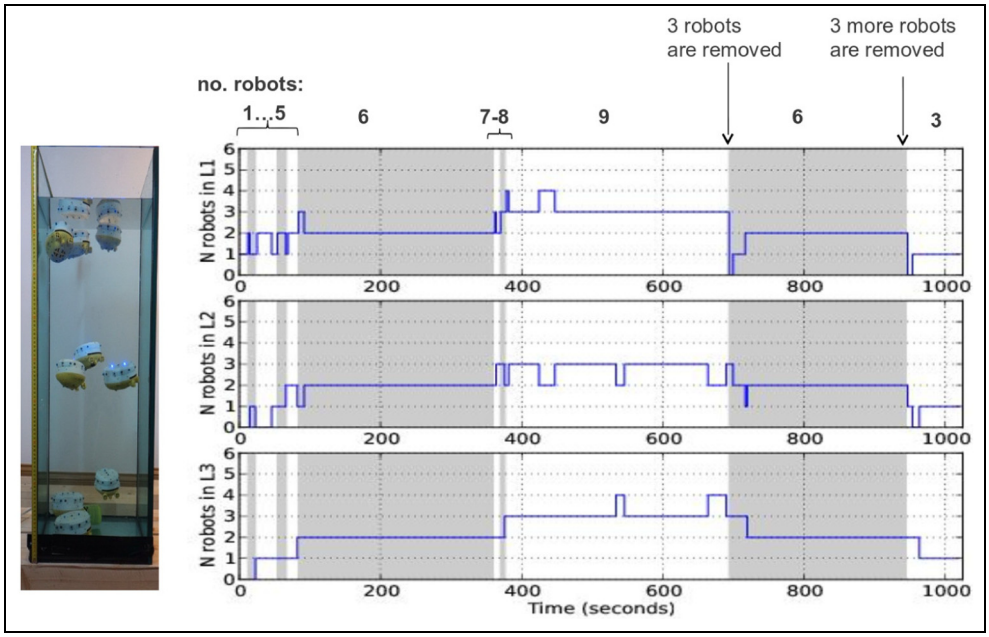

Figure 10. Nine robot experiment: the number of robots in each layer is represented over the period of the experiment.

Table 3. Average robot density in each layer (L1,L2,L3) for different setups (median of several repetitions), and the number of unintended switchings over the period of each experiment.

Setup	Total number of robots	L1	L2	L3	Number of switchings	Measurement period (s)
No interestingness	9	3.07	2.81	3.12	3	315
No interestingness	6	2	2	2	0	1128
Interestingness in L1	6	4.38	0.66	0.97	3.5	128
Interestingness in L2	6	1.0	3.94	1.01	0.5	127.5
Interestingness in L3	6	0.96	0.72	4.33	3.5	176.5
Interestingness in L1 & L3	7	2.89	0.87	3.23	14	1087

experimental period and the number of unintended switchings between the layers.

5.2 Adaptivity to changes in the demands: Adding and removing interestingness to layers

In this set of experiments, we investigated the swarm behavior when the relative demand for the robots in different layers changes. For that, we first put six robots in the water and wait until the robots are evenly distributed in the three layers. A source of interest is then added to the top layer meaning that the demand is raised in the top layer (L1). The source of interest is removed after 128 seconds. We leave the swarm to again rearrange into the even distribution. Then the source of interest is added to the middle layer (L2) and stays there for another ~128 seconds. Then the source of interest is removed from the middle layer and is added to the bottom layer (L3). The source of interest is removed after 176 seconds.

The time needed for rearrangement of the swarm to the new configurations according to the changes in the relative demands is depicted in Figure 11. Data is accumulated from six independent repetitions of the experiment. As it can be seen in the figure, the required time for rearrangement of the swarm when the source of interest is added or removed from the middle layer (L2) is lower than the other two layers. This is intuitive due to the fact that the rearrangement in this case means movement of the robots from (or to) the immediate neighboring layers in contrast to the cases where L1 or L3 contain the source of interest and the information needed to be propagated to the farther layers. Figure 12 demonstrates the average density of the layers when one of the layers contains a source of interest. Figure 13 represents the number of unintended switchings while one of the layers is more interesting. As seen in Figures 12 and 13, when the middle layer contains the source of interest, the behavior of the swarm is more stable and precise. This is due to the fact that the density of the x values of the robots is higher in the segment of the task with higher demand. That means that there is a lower distance between the x values in the interesting layer and therefore a higher sensitivity to errors. When the interesting layer is at one of the ends (L1 or L3), the fluctuations can happen only in one direction. When the middle layer (L2) contains the source of interest, the fluctuations can happen in two directions resulting in a more relaxed arrangement of the values and less errors in the number of agents.

In another experiment we marked both the top and bottom layers as the interesting (high demand) layers. Seven robots were available in this case. Figure 14 represents the number of robots in each layer over ~1087 seconds of the experiment. The average density of the layers over the whole experiment and the number of unintended switchings are represented in Table 3.

As it is shown in Figure 14, most of the time, three robots are present in the high-demand layers (top and bottom layers) and one robot in the low-demand layer (middle layer). Once in a while, the robot in the middle layer switches to one of the other layers. The same robot or another robot will then fill in the empty layer after a short period of time. In some cases when the middle layer is empty (e.g. after about 200 seconds), one robot from the top and one robot from the bottom layer come to the middle layer almost at the same time trying to fill the empty layer. After some time, one of the robots will move back to the layer with less robots. This is due to the fact that there is no communication between the non-neighboring layers and therefore both layers attempt to fill the empty layer without knowing about the other layer.

6 Discussion

In this paper, a distributed algorithm inspired from social inhibition in honeybees is proposed for partitioning a swarm and division of labor between the swarm members. The improved version of Partitioning Social Inhibition, enhances its previous version (Zahadat et al., 2013, 2015) by reducing the need for global information and therefore making the algorithm more decentralized. The algorithm follows the simple logic of distributing state variables of the swarm members in a way that there is a gradient of values in a fixed range distributed in the swarm. Based on this gradient, the agents decide on the task they should perform. The

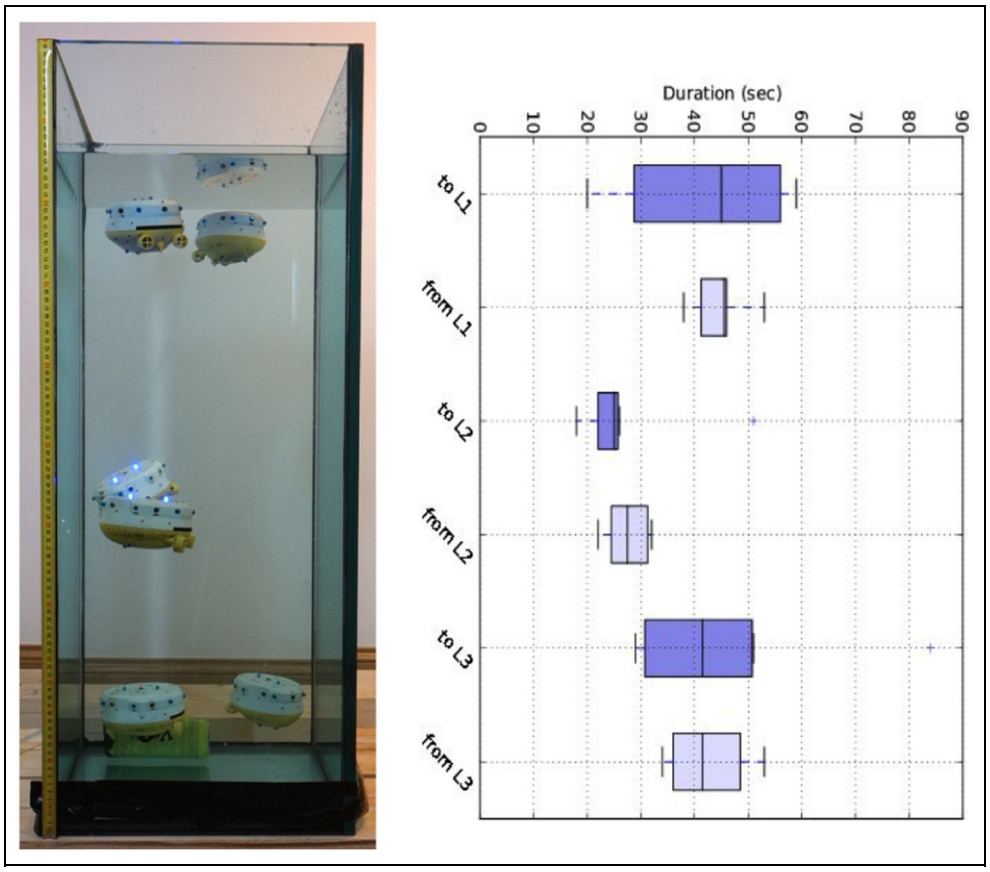

Figure 11. Six robot experiment: the time needed for rearrangement of the swarm to the new configurations according to the changes of the relative demands is represented with the box-plots. The data are accumulated from six independent experiments.

distribution of the state variable is done via local interactions and occurs dynamically during the run time. Therefore, the swarm is adaptive to changes in the swarm size as well as changes in the relative demands for different tasks. In addition, the algorithm enables specialization by maintaining low switchings of the agents between the tasks.

The algorithm is investigated in simulation and successfully implemented in a swarm of physical robots in the presence of communication noise and blind spots. Different scenarios are investigated including sudden changes in the workforce (size) and the demands for different tasks. The performance of the algorithm in terms of convergence-time and number of switchings is investigated for different swarm sizes and communication ranges in an otherwise fixed setup in simulation.

In the real robotic experiments in this paper, the demands for different tasks are emulated by the presence or absence of a source of interest. If a robot detects a source of interest in its task region (layer) directly or via another robot, it perceives it as a high demand for the task. No detection of the source of interest means a low demand for a robot. In real systems in practice, the estimation of the agents on the demand of the tasks can be done in different ways according to the nature of the task. For example, an agent can estimate the demand for its task by measuring its busy-time (workload). However, the better the agents perceive the demand for their local tasks, the more efficiently the PSI algorithm performs.

In addition, the thresholds that split the range of the state variable into a number of segments associated to the tasks can be chosen based on the relaxed configuration of the system if such a configuration is defined for the system.

By a relax configuration, we mean a configuration which happens more often in a normal situation of the swarm (without disturbance). For example, if the swarm is initially planned to work with a size of 30 robots performing four tasks while the first and the third tasks need twice as many robots as the other two tasks, the thresholds can be set such that the size of the segments associated to the first and third tasks are twice as big as the other two segments. In that case, a uniform distribution of the state variables is enough for keeping the swarm appropriately partitioned according to the demands for the tasks. This is similar to the honeybees in the normal condition of a colony when there is no stress or dramatic change happening. In case of such changes, the relative demands change accordingly and the colony survives the changes by changing the distribution of workers according to the new conditions. In honeybees, the proper thresholds might have evolved over time.

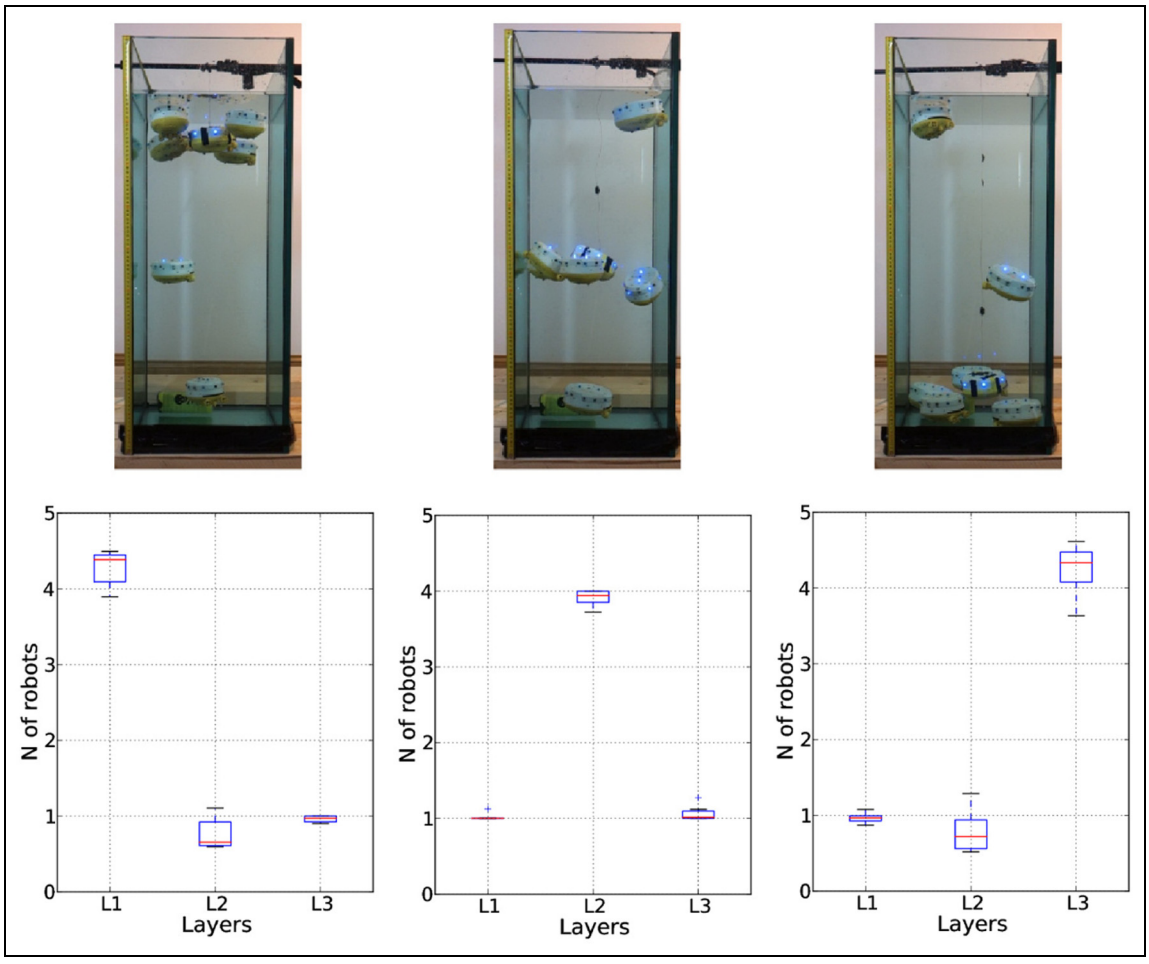

Figure 12. Six robot experiment: the average density of the layers while one of the layers contains a source of interest. The data are accumulated from six independent experiments.

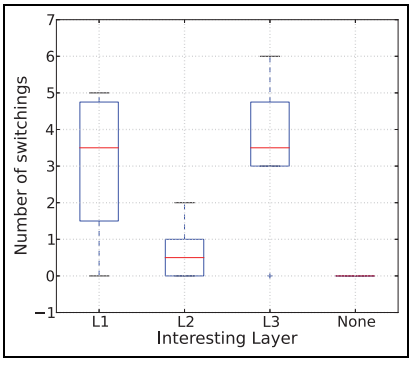

Figure 13. Six robot experiment: the number of unintended switchings while one of the layers is more interesting. The data is accumulated from six independent experiments.

The improved PSI algorithm has some similarities with other algorithms in the field, such as foraging-for-work (FFW) (Tofts, 1993), while it also has some basic differences. For example, in FFW, tasks are spatially arranged in a sequence and agents wander around and seek a task to perform. The spatial presence and physical movement of the agents in the task areas lead to the distribution of the agents relative to the task demands. In PSI, instead of physical spatial positioning of the agents, the physiological age (x) of the agents is virtually distributed relative to the task demands and leads to a physical distribution of the agents in different areas. From a biological point of view, FFW does not consider physiological differences between bees to drive them to perform different tasks. On the contrary, in PSI, a physiological age (x) is defined for the agents in an analogy to the physiological and morphological characteristics of the agents. There are also other methods such as stochastic policies as discussed in Berman, Halasz, Hsieh, and Kumar (2009). That method does not assume sequential ordering of the tasks. Instead, it uses a directed graph that defines interconnection topology of possible switchings between the tasks. Unlike PSI that aims for low task switchings, the stochastic policy method works based on persistent transitions between tasks while the transition rates are computed according to the demand for different tasks.

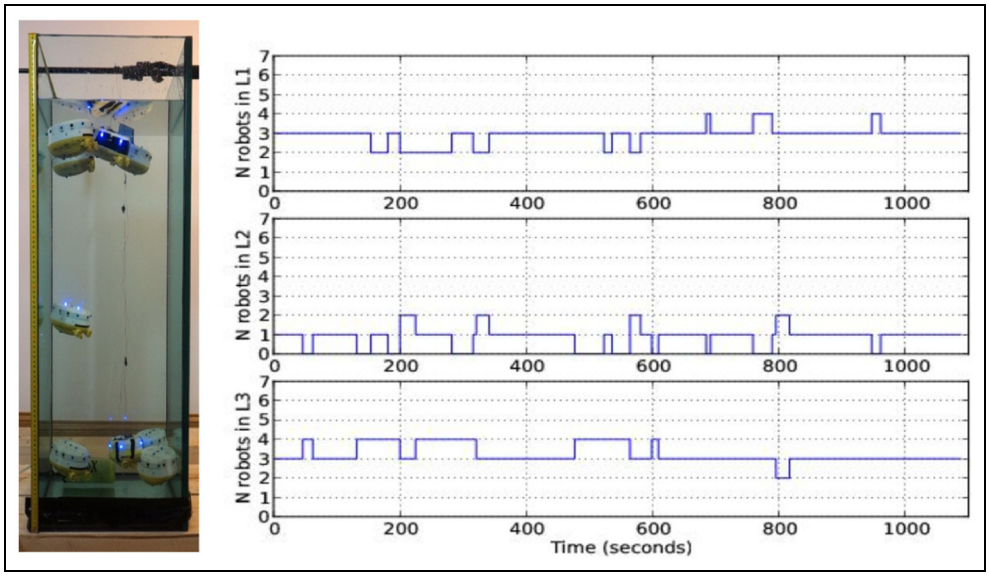

Figure 14. Seven robot experiment: the number of robots in each layer over the period of the experiment. In this experiment, the top and the bottom layers are marked as interesting layers.

The low switching rate of PSI leads to lower time and energy costs in the swarm, and enables the agents to specialize in their tasks.

In the future, we will study the behavior of the algorithm in more practical physical scenarios where the robots actually fulfill some real work.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by EU-ICT project 'CoCoRo' no. 270382 and EU-H2020 project 'subCULTron' no. 640967.

Note

1. A short video that briefly describes the basic idea of the algorithm and experiments is shown at https://youtu.be/_C7fFRSAW_E

References

Berman, S., Halasz, A., Hsieh, M. A., & Kumar, V. (2009). Optimized stochastic policies for task allocation in swarms of robots. *Robotics, IEEE Transactions on*, *25*(4), 927–937.

Beshers, S., & Fewell, J. (2001). Models of division of labor in social insects. *Annual Review of Entomology*, *46*, 413–440.

Beshers, S., Huang, Z., Oono, Y., & Robinson, G. (2001). Social inhibition and the regulation of temporal polyethism in honey bees. *Journal of Theoretical Biology*, *213*(3), 461–479.

Beshers, S., Robinson, G., & Mittenthal, J. (1999). Response thresholds and division of labor in social insects. In C. Detrain, J. Deneubourg, & J. Pasteels (Eds.), *Information processing in social insects* (p. 115–139). Basel: Birkhauser.

Bonabeau, E., Sobkowski, A., Theraulaz, G., & Deneubourg, J.-L. (1997). Adaptive task allocation inspired by a model of division of labor in social insects. In *Biocomputing and emergent computation: Proceedings of bcec97* (pp. 36–45). Skovde, Sweden: World Scientific Press.

CoCoRo. (2013). *Project website*. (Retrieved from http://cocoro.uni-graz.at)

Crailsheim, K., & Stabentheiner, A. (1996). Diurnal behavioural differences in forager and nurse honey bees (apis mellifera carnica pollm). *Apidologie*, *27*(4), 235–244.

Crosland, M. W. J., Ren, S. X., & Traniello, J. F. A. (2010). Division of labour among workers in the termite, reticulitermes fukienensis (isoptera: Rhinotermitidae). *Ethology*, *104*, 57–67.

Dipper, T. (2015). Einsatz von elektrischen Feldern zur Lokalisierung von autonomen Unterwasserschwarmrobotern. (in: BV-Forschungsberichte), University of Stuttgart.

Ferrante, E., Turgut, A. E., Duez-Guzmn, E. A., Dorigo, M., & Wenseleers, T. (2015). Evolution of self-organized task specialization in robot swarms. *PLoS Computational Biology*, *11*(8), e1004273.

Free, J. B., & Spencer-Booth, Y. (1959). The longevity of worker honeybees. *Proceedings of the Royal Entomological Society of London. Series A, General Entomology*, *34* (10–12), 141–150.

Gross, R., Nouyan, S., Bonani, M., Mondada, F., & Dorigo, M. (2008). Division of labour in self-organised groups. In *Sab* (p. 426–436). Berlin: Springer.

Hölldobler, B., & Wilson, E. (2008). *The superorganism: The beauty, elegance, and strangeness of insect societies*. New York, USA: W. W. Norton and Company.

Hrassnigg, N. (2005). Differences in drone and worker physiology in honeybees (apis mellifera). *Apidologie*, *36*(2), 255–277.

Huang, Z. Y., & Robinson, G. E. (1992). Honeybee colony integration: workerworker interactions mediate hormonally regulated plasticity in division of labor. *Proceedings of the National Academy of Sciences*, *89*(24), 11726-9.

Huang, Z. Y., & Robinson, G. E. (1996). Regulation of honey bee division of labor by colony age demography. *Behavioral Ecology and Sociobiology*, *39*(3), 147–158.

Huang, Z.-Y., & Robinson, G. (1999). Social control of division of labor in honey bee colonies. In C. Detrain, J. Deneubourg, & J. Pasteels (Eds.), *Information processing in social insects* (p. 165–186). Basel: Birkhauser.

Johnson, B. (2010). Division of labor in honeybees: form, function, and proximate mechanisms. *Behavioral Ecology and Sociobiology*, *64*(3), 305–316.

Jones, C., & Mataric, M. J. (2003). Adaptive division of labor in large-scale minimalist multi-robot systems. In *Iros* (p. 1969–1974). Las Vegas, NV: IEEE.

Julian, G. E., & Cahan, S. (1999). Undertaking specialization in the desert leafcutter ant acromyrmex versicolor. *Animal Behaviour*, *58*(2), 437–442.

Karsai, I., & Schmickl, T. (2011). Regulation of task partitioning by a "common stomach": A model of nest construction in social wasps. *Behavioral Ecology*, *22*, 819–830.

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

Naug, D., & Gadagkar, R. (1999). Flexible division of labor mediated by social interactions in an insect colony a simulation model. *Journal of Theoretical Biology*, *197*, 123–133.

Navarro, I., & Matía, F. (2013). An introduction to swarm robotics. *ISRN Robotics*, *2013*, 10, 608164.

Robinson, G. E. (1992). Regulation of division of labor in insect societies. *Annual Reviews Entomology*, *37*, 637–65.

Sahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In *Lecture notes in computer science* (Vol. 3342, p. 10–20). Berlin, Germany: Springer-Verlag.

Schmickl, T., Möslinger, C., & Crailsheim, K. (2007). Collective perception in a robot swarm. In E. Şahin, W. M. Spears, & A. F. T. Winfield (Eds.), *Swarm robotics - second sab 2006 international workshop* (*Vol. 4433*). Heidelberg/Berlin, Germany: Springer-Verlag.

Seeley, T. D. (1982). Adaptive significance of the age polyethism schedule in honeybee colonies. *Behavioral Ecology and Sociobiology*, *11*, 287–293.

Theraulaz, G., Bonabeau, E., & Deneubourg, J.-L. (1998). Response threshold reinforcement and division of labour in insect societies. In *Proceedings of the Royal Society of London* (*Vol. 265*, p. 327–332). London: The Royal Society.

Tofts, C. (1993). Algorithms for task allocation in ants. *Bulletin of Mathematical Biology*, *55*, 891–918.

Torres, V., Montagna, T., Raizer, J., & Antonialli-Junior, W. (2012). Division of labor in colonies of the eusocial wasp, mischocyttarus consimilis. *Journal of Insect Science*, *12*, 21.

Toth, A. L., Kantarovich, S., Meisel, A. F., & Robinson, G. E. (2005). Nutritional status influences socially regulated foraging ontogeny in honey bees. *The Journal of Experimental Biology*, *208*, 4641–4649.

White, T., & Helferty, J. (2005). Emergent team formation: Applying division of labour principles to robot soccer. *Engineering Self-Organising Systems*, 180–194.

Wilson, E. O. (1971). *The insect societies*. Cambridge, MA: Belknap Press of Harvard University Press.

Winston, M. (1987). *The biology of the honey bee*. Cambridge, MA: Harvard University Press.

Yang, Y., Zhou, C., & Tian, Y. (2009). Swarm robots task allocation based on response threshold model. In G. S. Gupta, & S. C. Mukhopadhyay (Eds.), *Icara* (p. 171–176). Wellington: IEEE.

Zahadat, P., Crailsheim, K., & Schmickl, T. (2013). Social inhibition manages division of labour in artificial swarm systems. In P. Lio, O. Miglino, G. Nicosia, S. Nolfi, & M. Pavone (Eds.), *12th european conference on artificial life (ecal 2013)* (p. 609–616). Cambridge, MA: MIT Press.

Zahadat, P., Hahshold, S., Thenius, R., Crailsheim, K., & Schmickl, T. (2015). From honeybees to robots and back: Division of labor based on partitioning social inhibition. *Bioinspiration & Biomimetics*, *10*(6), 066005.

About the Authors

Payam Zahadat received her PhD in Artificial Intelligence in 2011 from Shiraz University, Iran. She has been a research assistant at the Modular Robotics Lab, University of Southern Denmark in 2009-2010 and since 2011 she has been a post doctoral researcher at the Department of Zoology at the Karl-Franzes-University, Graz, Austria. Her research interests include both theoretical studies and applications in the field of swarm/multi-modular robotics, 'decentralised control and self-organization, regulatory networks, evolutionary algorithms and evolutionary robotics. She was/is involved in several EU-funded projects florarobotica, ASSISIbf, CoCoRo, SYMBRION, REPLICATOR, and Locomorph.

Thomas Schmickl is professor at the Department of Zoology at the Karl-Franzes-University, Graz, Austria, where he is supervising the Artificial Life Lab, which he founded in 2007. He was in 2012 the Basler Chair of Excellence at the East Tennessee State University (ETSU), Johnson City, TN, USA. Besides his research activities in the fields of zoology, biological/ecological modeling, bio-inspired robotics (swarm robotics, modular robotics, neural networks, artificial hormone systems, evolutionary robotics) he teaches also at the Department of ''Environmental System Sciences'' at the URBI faculty at Karl-Franzes-University Graz. He also acted as a lecturer in multi-agent modeling at the course of study ''Industrial simulation'' at the University of Applied Sciences in St.Poelten, Austria for more than 10 years. He was/is partner in the international projects I-SWARM, SYMBRION, REPLICATOR, FloraRobotica and he was/is leading scientist and coordinator of the EU grants CoCoRo, ASSISIbf and subCULTron.