1

1

Self-organized Multi-robot Task Allocation using Attractive Filed Model: A Taxonomy and Comparison of Centralized and Local Communication

Md Omar Faruque Sarker and Torbjørn S. Dahl

***Abstract*—This paper proposes to solve multi-robot task-allocation (MRTA) problems using a set of previously published generic rules for self-organised division of labour derived from the observation of ant, human and robotic social systems. The concrete form of these rules, the *attractive filed model* (AFM), are sufficiently abstraction to accommodate different sensing and communication models. We have validated the effectiveness of AFM for MRTA using two bio-inspired communication and sensing strategies: *global sensing - no communication* and *local sensing - local communication*. The former is realized using a centralized communication system and the latter is emulated as a peer-to-peer local communication scheme. Both strategies are evaluated using 16 e-puck robots.**

I. Introduction

MULTI-ROBOT systems can provide high levels of performance, fault-tolerance and robustness in complex and distributed tasks through parallelism and redundancy [1], [2]. *But how can we allocate the tasks among robots?* Traditionally, this issue has been identified as *multi-robot task allocation* (MRTA) [3], analogous to *division of labour* (DOL) in biological and human social systems [4].

MRTA is generally defined as the problem of assigning tasks in an appropriate time to the appropriate robots considering the changes in task-requirements, team-performance and the environment [5]. The complexities of the *distributed* MRTA problem arise from the fact that there is no central planner or coordinator for task assignments, and in a large multi-robot system, robots are generally limited to local sensing, communication and to interaction.

Within the context of the Engineering and Physical Sciences Research Council (EPSRC) project, 'Defying the Rules: How Self-regulatory Systems Work', we have presented the attractive field model (AFM) [12], a model of self-regulated DOL informed by studies of self-regulated task-allocation in ant and human social systems. AFM suggests four generic rules to explain self-regulation in those social systems. These four rules are: *continuous flow of information*, *concurrency*, *learning* and *forgetting*. All of them are explained below. Primarily, these rules enable the derivation of local control laws for regulating an individual's task-allocation behaviour in a way that facilitates DOL in the entire group.

In biological social systems, communications among the group members, as well as the perception of tasks, are two key components of self-organized DOL. In robotics, existing self-organized task-allocation methods rely heavily upon local sensing and local communication between individuals. AFM however, differs significantly in this point. Being a highly abstract model, it avoids a strong dependence on any specific communication strategy such as the local communications and interactions

M. O. F. Sarker and T. S. Dahl is with Cognitive Robotics Research Centre, University of Wales, Newport, Newport Business School, Allt-yr-yn Campus Newport, NP20 5DA, United Kingdom. e-mail: Mdomarfaruque.Sarker@newport.ac.uk

found in many existing approaches to MRTA. AFM uses the abstraction of attractive fields or continuous flows of information from tasks to agents. This flow of information can be realized by through either centralized or decentralized communication strategies under explicit and implicit communication modes.

In order to enable continuous flow of information in our multi-robot system, we have implemented two types of sensing and communication strategies inspired by the self-regulated DOL found in two types of social wasps: *polistes* and *polybia* [13]. Depending on the group size, these species follow different strategies for communication and sensing of tasks. Polistes wasps are called *independent founders*. In this species reproductive females establish colonies alone or in small groups (in the order of 10^2), but independent of any sterile workers. Polybia wasps, on the other hand, are called *swarm founders* where a swarm of workers and queens initiate colonies consisting of several hundreds to millions of individuals.

The most notable difference in the organization of work in these two social wasps is that independent founders do not rely on any cooperative task performance while swarm founders interact with eachother locally to accomplish their tasks. The work mode of independent founders can be considered as *global sensing - no communication (GSNC)* where the individuals sense the task requirements throughout a small colony and do these tasks without communicating with each other. The work mode of the swarm founders, on the other hand, can be treated as *local sensing - local communication (LSLC)* where the individuals can only sense tasks locally due to the large colony size and can communicate locally to exchange information, e.g. task-requirements (although their exact communication mechanism is unknown). In the study presented here, we have compare the performance of these two sensing and communication strategies for self-regulated DOL in robots under AFM.

The main contributions of this paper are as follows. This study compares the performance of two bio-inspired sensing and communication strategies in producing self-regulated MRTA. A flexible multi-robot control architecture has been developed adopting D-Bus inter-process communication technology. A novel taxonomy of MRTA solutions has been derived based on organization of task-allocation, interaction and communication.

This paper has been organized as follows. Section 2 reviews the related literature on MRTA. Section 3 presents the new taxonomy of MRTA solutions. Section 4 presents AFM based task-allocation solution for multi-robot systems. Section 5 describes our centralized and local communication schemes used in this study. Section 6 and Section 7 present the design and implementation of our experiments respectively. Section 8 describes our results. Section 9 discusses the performance of self-regulated MRTA under both centralized and local communication strategies. Section 10 draws conclusions and presents possibilities for future work.

II. Related work

Since the 90s, many robot control architectures have been designed with thed sole purpose of addressing the MRTA issue from different perspectives. Based on the high-level design of those solutions, we have classified them into two categories: i) *predefined* or *explicit* task-allocation and ii) *bio-inspired self-organized* task-allocation.

Early research on explicit task-allocation was dominated by explicit cooperation [6], use of dynamic role assignment [7] and market-based bidding approach [8]. When taking these approaches, robots perform explicit task-allocation activities, commonly including communication with other group members for negotiating tasks. These approaches are intuitive and comparatively straight forward to design and implement and can be analysed formally. However, these approaches typically work well only when the number of robots are small (≤ 10) [9].

Task performance in self-organized approaches relies on the collective behaviours resulting from the local interactions of many simple and mostly homogeneous or interchangeable agents. Robots choose their tasks independently using general principles of self-organization such as: positive and negative

feedback, multiple interactions among entities and their environment, and randomness or amplification of fluctuations [20]. Moreover. interaction among individuals and their environment can be modulated by stigmergic, local and global communications. Among many variants of self-organized task-allocation mechanisms, the most common type is threshold-based task-allocation [21]. In this approach, a robot's decision to select a particular task depends largely on its perception of a stimulus, e.g., the demand for a task to be performed, and its corresponding response threshold for that task.

Under the deterministic response-threshold approach, each robot has an activation threshold for each task that needs to be performed. It continuously perceives or monitors the stimulus of all tasks that reflect the relative urgencies of tasks. When a particular task-stimulus exceeds a predefined threshold the robot starts working on that task, typically reducing thie related stimuli. When the task-stimuli falls below the fixed threshold the robot abandons that task. This type of approach has been effectively applied in foraging [11], [22] and aggregation [23]. The fixed response threshold can initially be same for all robots [24] or they can be different according robot capabilities or the configuration of the system [22]. Adaptive response-threshold models change the thresholds over time. A robot's response threshold for a given task is often decreased as a result of the robot performing that task. This enables a robot to select that particular task more frequently or in other words specialise on that task [21], [23].

Unlike the deterministic approach, where robots respond preditably, e.g., to the task stimulus that is the furthest above its threshold, probabilistic or stochastic approaches offer a selection process based-on a probability distribution over the tasks. In this case, all tasks commonly have at least a small, non-zero probability of being chosen. This random element typically prevents starvation of low=stimulus tasks.

Bio-inspired, self-organized MRTA mechanisms rely to a much smaller degree on modelling the environment, tasks or robot capabilities. Most existing research in this area considers a single global task e.g. foraging, area cleaning and box-pushing with the focus of the work being the design of individual robot controllers that can accomplish the task [5]. More research is needed to study the performance of the self-organized approach in handling several different tasks.

Both of the above task-allocation approaches expose their relative strengths and weaknesses when they are put under real-time experiments with variable number of robots and tasks. In an arbitrary event handling domain, a comparison between self-organized and predefined market-based task-allocation was reported by Kalra and Martinoli [25]. They found that explicit task-allocation was more efficient when the required information could be captured accurately. The threshold-based approach offered similar quality of allocation at a fraction of cost in noisy environment.

Gerkey and Mataric have presented a study [5] that compared the complexity and performance of key architectures, including ALLIANCE [14], the Broadcast of local eligibillity (BLE) arcgutecture [26], M+ [27], MURDOCH [28], First piece auctions [29] and Dynamic role assignment [7]. All of these rely on explicit task-allocation mechanisms. The computational and communication requirements of these MRTA systems were expressed as a function of the number of robots and tasks. Although the study did not explicitly measure the scalability of the architectures, it clearly showed that many explicit task-allocation mechanisms fail to scale well in environments where the number of robots and tasks are large, given limited overall communication bandwidth and limited processing power for individual robots.

III. A TAXONOMY OF MRTA

In order to categorize both explicit and self-organized MRTA approaches we propose three distinct dimensions: i) organization of task-allocation, ii) degree of interaction and iii) degree of communication. Fig. **??** depicts these dimensions. This taxonomy can be used to measure the complexities involved in various kinds of MRTA problems and to facilcilitated the design of appropriate systems form handling them. In Fig. **??**, X axis repre-

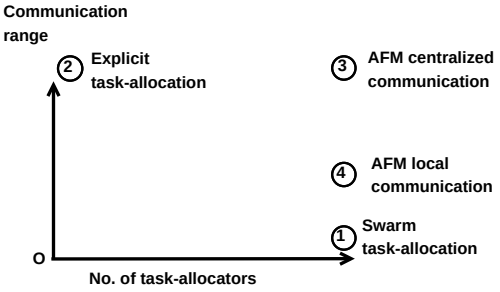


Fig. 1: Classification of MRTA solutions based on task-allocation and communication strategies

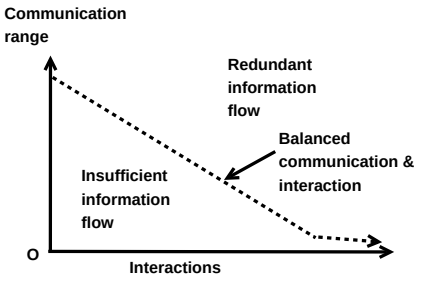


Fig. 2: Information flow caused by different levels of communication and interaction

sents the number of active nodes that provides the task-allocation to the group. For example, in any predefined task-allocation approach, we can use one external centralized entity or one of the robots (aka leader) to manage the task-allocation. In many predefined methods, e.g. in market-based systems, multiple nodes can act as mediators or task-allocators. Under predefined task-allocation approach, a small number of robots can have fully distributed task-allocation where each robot acts as an independent task-allocator (e.g. as in ALLIANCE architecture). Most of the self-organized task-allocation methods are fully distributed, i.e. each robot selects its tasks independently and without the help of a leader. However, robots might be dependent on other entities for getting the latest *information* about tasks. Recent studies on swarm-robotic flocking by Celikkanat *et al.* [30] show that a swarm can be guided to a target by a few informed individuals or leaders while the remaining robots remain self-organied. Task-allocation in a swarm of robots through one central entity may be rare since one of the major motivations behind swarm robotic systems is to make more robust by avoiding single points of failure.

IV. Multi-robot task-allocation using AFM

The AFM has been developed through a collaborative between the partners on the 'Defying the Rules: How Self-Regulated Social; Systems Work' project. Experiments on ants colonies *Temnothorax albipennis* studied the collective performance of ants during brood-sorting and nest construction after emigration to a new site. The model is also inspired by observational data from the self-organized development of the infrastructure of an *eco-village* by an open community of volunteers. These studies helped us to formalize the a set of necessary and sufficient requirements for self-organisation in a social systems. We also developed a concrete model of self-organisation based on task-allocation. This model was validated by deploying it as a robot controller mechanism. In this section, we describe the generic AFM and how it embodies the requirements for self-organization. We also provide an interpretation of AFM in multi-robot systems.

A. The Attractive Field Model

Inspired from the DOL in ants, humans and robots, we have proposed the following necessary and sufficient set of four requirements for self-regulation in social systems.

Requirement 1: Concurrence The simultaneous presence of several options is necessary in order to

meaningfully say that the system has organised into a recognisable structure. In task-allocation terms the minimum requirement is a single task as well as the option of not performing any task.

Requirement 2: Continuous flow of information Self-organised social systems establish a flow of information over the period of time when self-organisation can be defined. The task information provides the basis on which the agents self-organise by enabling them to perceive tasks and receive feedback on system performance.

Requirement 3: Sensitization The system must have a way of representing the structure produced by self-organisation, in terms of MRTA, which tasks the robots are allocated. One of the simplest ways of representing this information is an individual preference parameter for each task-robot combination. A system where each robot has different levels of preference or *sensitivity* to the available tasks, can be said to have to embody a distinct organisation through differentiation.

Requirement 4: Forgetting When a system self-organises by repeated increases in individual sensitisation levels, it is also necessary, in order to avoid saturation, to have a mechanism by which the sensitisation levels are reduced or *forgotten*. In addition to avoiding the situation where a structure produced by self-organisation is eroded by an eventual increase of sensitisation values to a given maximum, forgetting also allows flexibility in the system, in that the structure can change as certain tasks become important and other tasks become less so. This effect can be achieved by mechanisms such as a slow general decay of sensitisation values or explicit negative feedback.

Building on the requirements for self-organised social systems, AFM formalises these requirements in terms of the relationships between properties of individual agents and of the system as a whole [12]. AFM is a bipartite network, i.e. there are two different types of nodes. One set of nodes describes the sources of the attractive fields, the tasks, and the other set describes the agents. Edges only exist between different types of nodes and they encode the strength of the atrractive field as perceived by the agent. There are no edges between agent nodes. All

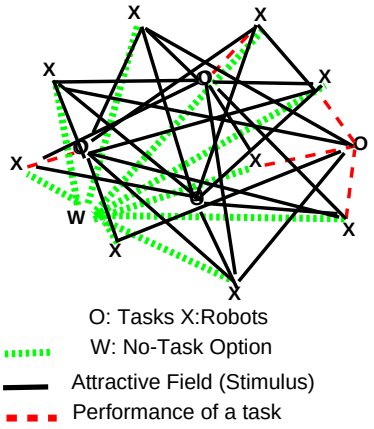


Fig. 3: The attractive filed model (AFM)

communication is considered part of the attractive fields. There is also a permanent field representing the *no-task* option of not working in any of the available tasks. This option is modelled as a random walk. The model is presented graphically in Fig. 3. The elements depicted are:

1) Source nodes (o) are tasks to be allocated to agents
2) Agent nodes (x) e.g., ants, humans, or robots
3) Black solid edges represent the attractive fields and correspond to an agent's perceived stimuli from each task.
4) Green edges represent the attractive field of the ever present no-task option, reresented as a particular task (w).
5) The red lines are not edges, but represent how each agent is allocated to a single task at any point in time.

The edges of the AFM network are weighted and the value of this weight describes the strength of the stimulus as perceived by the agent. In a spatial representation of the model, the strength of the field depends on the physical distance of the agent to the source. In information-based models, the distance can represent an agent's level of understanding of that task. The strength of a field is increased through the sensitisation of the agent through experience with performing the task. This elements is not depicted explicitly in Figure 3 but is represented

in the weights of the edges. In Figure 3), the nodes have arbitrary positions. Even though the distance is physical in this case, it need not be. When the model is applied to other domains, the distance can represent the accessibility of information or the time the information takes to reach the agent. In summary, from the above diagram of the network, we can see that each of the agents is connected to each of the tasks. This means that even if an agent is currently involved in a task, the probability that it stops doing it in order to pursue a different task, or to random walk, is always non-zero.

AFM assumed a repeated task selection by indicidual agents. The probability of an agent chosing to perform a task is proportional to the sttrength of the task's attractive field, as given by Equation 1.

$$P_j^i = \frac{S_j^i}{\sum_{j=0}^{J} S_j^i} \quad where, \quad S_0^i = S_{RW}^i \tag{1}$$

Equation 1 states that the probability of an agent, i, selecting a task, j, is proportional to the stimulus, S_j^i, perceived from that task, with the sum of all the task stimuli normalised to 1.

The strength of an attractive field varies according to how sensitive the agent is to that task, k_j^i, the distance between the task and the agent, d_{ij}, and the *urgency*, ϕ_j of the task. In order to give a clear edge to each field, its value is modulated by the hyperbolic tangent function, $tanh$. Equation 2 formalises this part of AFM.

$$S_j^i = tanh\{\frac{k_j^i}{d_{ij} + \delta}\phi_j\} \tag{2}$$

Eqation 2, used small constant δ, called *delta distance*, to avoid division by zero, in the case when a robot has reached to a task.

Equation 3 shows how AFM handles the the no-task, or random walk, option. The strength of the stimuli of the random walk task depends on the strengths of the fields real tasks. In particular, when the other tasks have a low overall level of sensitisation, i.e., relatively weak fields, the strength of the random walk field if relatively high. On the other hand, when the agent is highly sensitised, the strength of the random walk field becomes relatively low. We use J to denote the number of real tasks. AFM effectively considers random walking as an ever present additional task. Thus the total number of tasks becomes $J + 1$.

$$S_{RW}^i = tanh\left\{1 - \frac{\sum_{j=1}^{J} S_j^i}{J+1}\right\} \tag{3}$$

A task j has an associated urgency ϕ_j indicating its relative importance over time. If an agent attends a task j in time step t, the value of ϕ_j will decrease by an amount $\delta_{\phi_{INC}}$ in the time-step $t+1$. On the other hand, if a task has not been served by any of the agents in time-step t, ϕ_j will increase by a different amount, $\delta_{\phi_{DEC}}$ in time-step $t+1$. This behaviour is formalised in Equations 4 and 5.

$$If\ the\ task\ is\ not\ being\ done: \ \phi_{j,t+1} \rightarrow \phi_{j,t} + \delta_{\phi_{INC}} \tag{4}$$

$$If\ the\ task\ is\ being\ done: \ \phi_{j,t+1} \rightarrow \phi_{j,t} - n\,\delta_{\phi_{DEC}} \tag{5}$$

Equation 4 refers to a case where no agent attends to task j and Equation 5 to the case where n agens are concurrently performing task j.

In order to complete a task, an agent needs to be within a fixed distance of that task. When an agent performs a task, it learns about it and this will increases the probability of that agent selecting that task in the future. This is done by increasing its sensitization to the task by a fixed amount, k_{INC}. The variable affinity of an agent, i, to a task, j, is called its *sensitization* to that task and is denoted k_j^i. If an agent, i, does not do a task j, k_j^i is decreased by a different fixed amount, k_{DEC}. This behaviour is formalised in Equations 6 and 7.

$$If\ task\ is\ done: \ k_j^i \rightarrow k_j^i + k_{INC} \tag{6}$$

$$If\ task\ is\ not\ done: \ k_j^i \rightarrow k_j^i - k_{DEC} \tag{7}$$

B. A Robotic Interpretation of AFM

The interpretation of AFM in a multi-robot system follows the above mentioned generic interpretation. Each robot is modelled as an agent and each task is modellled as a spatial location. The robots repeatedly select tasks and if the robot is outside a fixed task boundry, it navigates towards the task. If the robot is within the task boundry it remains there until the end of the time step when a new (or the

same) task is selected. The distance between a task and a robot is simply the physical distance and the sensitivities are recorded as specific values on each robot. The urgency values of the tasks are calculated based on the number of robots attending each task and the updated urgency values are communicated to the robots.

The sensing of the distance between the taks and robots as well as the communication of urgency values are non-trivial in a robotic systen. Both the sensing and communication can be done either locally by the individual robots or centrally, through an overhead camera and a global communication network. This article presents work on exploring the effects of different sensing and communication models on the performance of MRTA systems.

We have designed a set of manufacturing shop-floor scenario experiments for validating the effectiveness of our AFM in producing self-regulated MRTA. In this section, we have described our manufacturing shop-floor scenario and our task-allocation solution for robot-controllers.

C. A manufacturing shop-floor scenario

By extending our interpretation of AFM in multi-robot system, we can set-up manufacturing shop-floor scenario. Here, each task represents a manufacturing machine that is capable of producing goods from raw materials, but they also require constant maintenance works for stable operations. Let W_j be a finite number of material parts that can be loaded into a machine j in the beginning of its production process and in each time-step, ω_j units of material parts can be processed ($\omega_j \ll W_j$). So let Ω_j^p be the initial production workload of j which is simply: W_j/ω_j unit.

We assume that all machines are identical. In each time step, each machine always requires a minimum threshold number of robots, called hereafter as *minimum robots per machine (μ)*, to meet its constant maintenance work-load, Ω_j^m unit. However, if μ or more robots are present in a machine for production purpose, we assume that, no extra robot is required to do its maintenance work separately. These robots, along with their production jobs, can do necessary maintenance works concurrently. For the sake of simplicity, here we consider μ = 1.

Now let us fit the above production and maintenance work-loads and task performance of robots into a unit task-urgency scale. Let us divide our manufacturing operation into two subsequent stages: 1) *production and maintenance mode* (PMM), and 2) *maintenance only mode* (MOM). Initially a machine starts working in PMM and does production and maintenance works concurrently. When there is no production work left, then it enters into MOM. Fig. 4 illustrates this scenario for a single machine.

Under both modes, let α_j be the amount of workload occurs in a unit time-step if no robot serves a task and it corresponds to a fixed task-urgency $\Delta\phi_{INC}$. On the other hand, let us assume that in each time-step, a robot, i, can decrease a constant workload β_i by doing some maintenance work along with doing any available production work. This corresponds to a negative task urgency: $-\Delta\phi_{DEC}$. So, at the beginning of production process, task-urgency, occurred in a machine due to its production work-loads, can be encoded by Eq. 8.

$$\Phi_{j,INIT}^{PMM} = \Omega_j^p \times \Delta\phi_{INC} + \phi_j^{m0} \tag{8}$$

where ϕ_j^{m0} represents the task-urgency due to any initial maintenance work-load of j. Now if no robot attends to serve a machine, each time-step a constant maintenance workload of α_j^m will be added to j and that will increase its task-urgency by $\Delta\phi_{INC}$. So, if k time steps passes without any production work being done, task urgency at k^{th} time-step will follow Eq. 9.

$$\Phi_{j,k}^{PMM} = \Phi_{j,INIT}^{PMM} + k \times \Delta\phi_{INC} \tag{9}$$

However, if a robot attends to a machine and does some production works from it, there would be no extra maintenance work as we have assumed that μ = 1. Rather, the task-urgency on this machine will decrease by $\Delta\phi_{DEC}$ amount. If ν_k robots work on a machine simultaneously at time-step k, this decrease will be: $\nu_k \times \Delta\phi_{DEC}$. So in such cases, task-urgency in $(k+1)^{th}$ time-step can be represented by:

$$\Phi_{j,k+1}^{PMM} = \Phi_{j,k}^{PMM} - \nu_k \times \Delta\phi_{DEC} \tag{10}$$

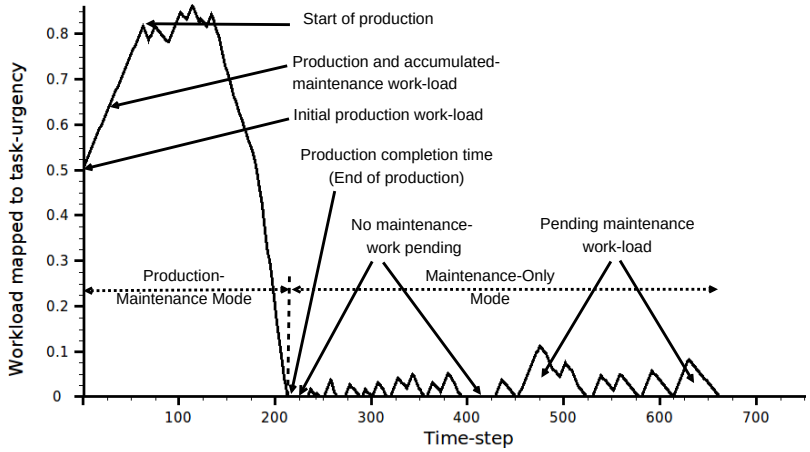


Fig. 4: A manufacturing shop-floor production and maintenance cycle

At a particular machine j, once $\Phi_{j,k}^{PMM}$ reaches to zero, we can say that there is no more production work left and this time-step k can give us the *production completion time* of j, T_j^{PMM}. Average production time-steps of a shop-floor with M machines can be calculated by the following simple equation.

$$T_{avg}^{PMM} = \frac{1}{M}\sum_{j=1}^{M} T_j^{PMM} \tag{11}$$

T_{avg}^{PMM} can be compared with the minimum number of time-steps necessary to finish production works, T_{min}^{PMM}. This can only happen in an ideal case where all robots work for production without any random walking or failure. We can get T_{min}^{PMM} from the total amount of work load and maximum possible inputs from all robots. If there are M machines and N robots, each machine has Φ_{INIT}^{PMM} task-urgency, and each time-step robots can decrease N $\times$ $\Delta\phi_{DEC}$ task-urgencies, then the theoretical T_{min}^{PMM} can be found from the following Eq. 12.

$$T_{min}^{PMM} = \frac{M \times \Phi_{INIT}^{PMM}}{N \times \Delta\phi_{DEC}} \tag{12}$$

$$\zeta_{avg}^{PMM} = \frac{T_{avg}^{PMM} - T_{min}^{PMM}}{T_{min}^{PMM}} \tag{13}$$

Thus we can define ζ_{avg}^{PMM}, average production completion delay (APCD) by following Eq. 13:

When a machine enters into MOM, only μ robots are required to do its maintenance works in each time step. So, in such cases, if no robot serves a machine, the growth of task-urgency will follow Eq. 9. However, if ν_k robots are serving this machine at a particular time-step k^{th} , task-urgency at $(k+1)^{th}$ time-step can be represented by:

$$\Phi_{j,k+1}^{MOM} = \Phi_{j,k}^{MOM} - (\nu_k - \mu) \times \Delta\phi_{DEC} \tag{14}$$

By considering $\mu = 1$, Eq. 14 will reduces to Eq. 10. Here, $\Phi_{j,k+1}^{MOM}$ will correspond to the *pending maintenance work-load* of a particular machine at a given time. This happens due to the random task switching of robots with a no-task option (random-walking). Interestingly PMW will indicate the robustness of this system since higher PMW value will indicate the delay in attending maintenance works by robots. We can find the *average pending maintenance work-load* (APMW) per time-step per machine, χ_j^{MOM} (Eq. 15) and average PMW per machine per time-step, χ_{avg}^{MOM} (Eq. 16).

$$\chi_j^{MOM} = \frac{1}{K}\sum_{k=1}^{K} \Phi_{j,k}^{MOM} \tag{15}$$

$$\chi_{avg}^{MOM} = \frac{1}{M}\sum_{j=1}^{M} \chi_j^{MOM} \tag{16}$$

D. Robot-controller algorithms

In our task-allocation algorithm, for each task, we extract its pose, urgency, sensitisation from input data. *CalculateTaskDistance()* function calculates the Euclidian distance between a task and a the robot by taking the current robot-pose and static task-pose as the input. These values are enough to get a task's stimuli based on Eq. 2. We get the random-walk task's stimuli from Eq. 3.

Algorithm 1: Self-regulated task-allocation by AFM

Input: $AllTaskInfo, RobotPose, TaskRecords, DeltaDistance$
Output: $SelectedTaskID$
comment: Stage 1: Get sum of stimulus of all tasks
$TotalTasks \leftarrow$ Total number of tasks $\in AllTaskInfo$
$TaskStimuliSum \leftarrow 0$
for all $Task \in AllTaskInfo$ **do**
  $TaskPose \leftarrow$ Task-position $\in Task$
  $TaskUrgency \leftarrow$ Task-urgency $\in Task$
  $TaskSensitization \leftarrow$ Task-sensitization $\in Task$
  $DistToTask \leftarrow$ **CalculateTaskDistance(** $RobotPose, TaskPose$**)**
  $TaskStimuli \leftarrow$ **CalculateTaskStimuli(** $DistToTask, TaskSensitization, TaskUrgency, DeltaDistance$**)**
  $TaskStimuliSum \leftarrow TaskStimuliSum + TaskStimuli$
  $TaskRecords \leftarrow$ **UpdateTaskRecords(** $TaskStimuli, DistToTask, TaskUrgency, TaskSensitization$**)**
end for
$RandomWalkStimuli \leftarrow$ **CalculateRandomWalkStimuli(** $TotalTasks, TaskStimuliSum$**)**
$AllStimuliSum \leftarrow TaskStimuliSum + RandomWalkStimuli$
comment: Stage 2: Find probability of doing each task
$TaskID \leftarrow 0$
while $TaskID \leq TotalTasks$ **do**
  $TaskStimuli \leftarrow$ Task-stimuli $\in TaskRecords(TaskID)$
  $TaskProbability \leftarrow$ **GetTaskProbability(** $TaskStimuli, AllStimuliSum$**)**
  $TaskProbabilityRange \leftarrow$ **ConvertTaskProbabilityIntoRange(** $TaskProbability$**)**
  $TaskRecords \leftarrow$ **UpdateTaskRecords(**$TaskProbability$**)**
end while
comment: Stage 3: Draw a random-number to match Task-ID
$RandomNum \leftarrow$ **GetRandomNumber(**$0,$ **Max(**$TaskProbabilityRange$**))**
while $TaskID \leq TotalTasks$ **do**
  $RangeStart \leftarrow$ **Min(**$TaskProbabilityRange(TaskID)$**)**
  $RangeEnd \leftarrow$ **Max(**$TaskProbabilityRange(TaskID)$**)**
  if $RandomNum \geq RangeStart$ && $RandomNum \leq RangeEnd$ **then**
    $SelectedTaskID \leftarrow TaskID$
  end if
end while

In the second stage of our task-allocation algorithm, we find the probability of each task (including random-walk) based on Eq. 1. Then this probability value of each task, between 0 and 1, is rounded to the closest two-digit fractions and multiplied by 100 and put into a linear scale. Then we use a random-number generator that draws a random-number between 0 and the highest value of task-probability range. Then this random number is compared against the task probability range of each task. Under this probabilistic method, the task with larger probability range has a higher chance to be selected, but low probability tasks can also be selected time-to-time.

Algorithm 2: Robot's learning and forgetting of tasks

Input: $TaskRecords, LeranRate, ForgetRate, SelectedTaskID$
Output: Updated $TaskSensitisation \in TaskRecords$
for all $Task \in TaskRecords$ **do**
  $TaskID \leftarrow$ ID $\in Task$
  $TaskSensitization \leftarrow$ Sensitisation $\in Task$
  if $TaskID \equiv SelectedTaskID$ **then**
    $TaskSensitization \leftarrow$ **Max(**$1, (TaskSensitization + LeanRate)$**)**
  else
    $TaskSensitization \leftarrow$ **Min(**$0, (TaskSensitization - ForgetRate)$**)**
  end if
end for

Algorithm 2 lists the pseudo-code for updating (learning and forgetting) the robot's task-sensitization values. Along with previously mentioned *TaskRecords*, and *SelectedTaskID*, it takes

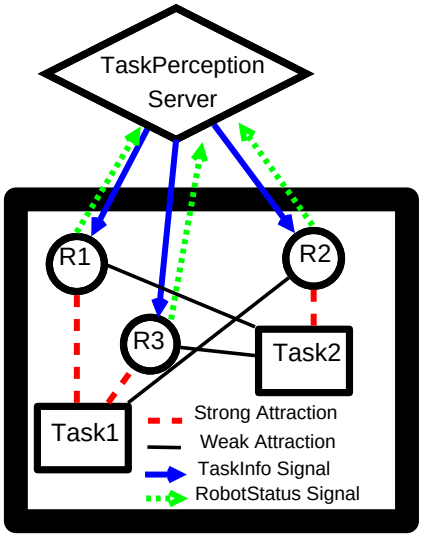


Fig. 5: A centralized communication scheme

two other inputs: *LeranRate* (Δk_{INC}) and *ForgetRate* (Δk_{INC}) as outlined in Eq. 6 and 7 respectively. In addition to that it also keep the value of task-sensitization between the fixed limit of 0 and 1.

V. Communication Models

A. Centralized communication model

AFM relies upon a system-wide continuous flow of information which can be realized using any suitable communication model. A simple centralized communication scheme is outlined in Fig. 5. In this model we have used bi-directional signal-based exchange of communication messages between a centralized *task perception server* (TPS) and a *robot-controller client* (RCC). The main role of TPS is to send up-to-date task-information to RCCs. This task-information mainly contains the location and urgency of all tasks which is used by the RCCs for running their task-allocation algorithm. The urgency value of each task is dynamically updated by TPS after receiving the status signals from the working robots of that particular task. Fig. 5 shows how three robots are attracted to two different tasks and their communications with TPS. Here although the robots are selecting task independently based-on the strength of their attractive fields to different tasks, they are depended on the TPS for task-information.

We can characterize our communication model in terms of three fundamental issues of communication [31].

1) Message content: *what to communicate?*
2) Communication frequency: *when to communicate?*
3) Target message recipients: *with whom to communicate?*

AFM suggests the communication of task-urgencies among robots. This communication helps the robots to gain information that can be treated as "global sensing". However in this model robots do not communicate among themselves. Hence this model can be approximated as the GSNC strategy. Since in order to run the task-allocation algorithm robot-controllers need the distance information we also include the task position information in the message. Our centralized communication model is open to include any further information, such as time-stamp, in the message payload. In this centralized communication model the frequency of signal emission depends on several issues, e.g. the rate at which the environment is changing, the bandwidth of communication medium. In case of time-extended tasks, robots can receive information less frequently and the bandwidth usage can be kept minimum. However under a fast changing environment relatively more bandwidth will be required. Finally the centralized communication model spread the attractive fields of all tasks globally by broadcasting information to all robots.

The algorithm that updates the task-urgencies in TPS is given here.

Algorithm 3: Task-urgency update rules based on AFM

Input: $AllTaskInfo, AllTaskWorkers, ThresholdWorkers,$
$DeltaUrgencyINC, DeltaUrgencyDEC$
Output: Updated $TaskUrgencies \in AllTaskInfo$
for all $(Task, Workers) \in (AllTaskInfo, AllTaskWorkers)$ **do**
  $TaskUrgency \leftarrow$ Task-urgency $\in Task$
  $TaskWorkers \leftarrow$ Number of workers $\in Workers$
  if $TaskWorkers < ThresholdWorkers$ **then**

8: $TaskUrgency \leftarrow$ **Max(**$1, (TaskUgerncy + DeltaUrgencyINC)$**)**
9: **else**
10: $TaskUrgency \leftarrow$ **Min(**$0, (TaskUgerncy - TaskWorkers \times DeltaUrgencyDEC)$**)**
12: **end if**
13: **end for**

The above algorithm relies on the previously mentioned task-status signals from the working robots. It sorts out the worker robots' robot-id according to their task-id and use the worker count to determine whether the corresponding task-urgency needs any change according the rules of AFM. For example, here we have shown that if the number of working robots exceeds a pre-set threshold task urgency has been updated based on Eq. 4 and 5. Moreover, in order to satisfy our interpretation of AFM, the task-urgency values are kept within a fixed range 0 and 1.

B. Local P2P communication model (LPCM)

In most swarm robotic research local communication is considered as the one of the most critical components of the swarms where the global behaviours emerges from the local interactions among the individuals and their environment. In this study, we have used the concepts of pheromone active-space of ants to realize our simple LSLC scheme. Ants use various chemical pheromones with different active spaces (or communication ranges) to communicate different messages with their group members [32]. Ants sitting near the source of this pheromone sense and respond quicker than others who wander in far distances. Thus both communication and sensing occurs within a small communication range[1]. We have used this concept of communication range or locality in our LPCM. A suitable range (or radius) of communication and sensing can be set at design time based on the capabilities of robots [23]. Alternately they can also be varied dynamically over time depending on the cost of communication and sensing, e.g. density of peers, ambient noise in the communication channels, or even by aiming for maximizing information spread [33]. In this study, we have followed the former approach as our robots do not have the precise hardware to dynamically vary their communication and sensing ranges. Below we describe the general characteristics and implementation algorithm and strategy of LPCM. Our LPCM relies on the local P2P communications among robots. we have assumed that robots can communicate to its nearby peers within a certain communication radius, r_{comm} and they can sense tasks within another radius r_{task}. They exchange communication signals reliably without any significant loss of information. A robot R_1 is a *peer* of robot R_2, if spatial distance between R_1 and R_2 is less than this r_{comm}. Similarly, when a robot comes within this r_{task} of a task, it can sense the status of this task. Although the communication and sensing range can be different based on robot capabilities, we have considered them same for the sake of simplicity of our implementation.

Local communication can also give robots similar task information as in centralized communication. In this case, it is not necessary for each robot to communicate with every other robot to get information on all tasks. Since robots can random walk and explore the environment we assume that for a reasonably high robot-to-space density, all tasks will be known to all robots after an initial exploration period. In order to update the urgency of a task, robots can estimate the number of robots working on a task in two ways: by either using their sensory perception (e.g. on-board camera) or doing local P2P communication with others.

Similar to our centralized communication model, we can characterize our local communication model in terms of message content, communication frequency and target recipients [31]. Regarding the issue of message content, our local communication model is open. Robots can communicate with their peers with any kind of message. Our local model addresses the last two issues very specifically. Robots communicate only when they meet their peers within a certain communication radius

[1]Although, generally communication and sensing are two different issues, however within the context of our self-regulated MRTA, we have broadly viewed sensing as the part of communication process, either implicitly via environment, or explicitly via local peers.

(r_{comm}). Although in case of an environment where robots move relatively faster the peer relationships can also be changed dynamically. But this can be manipulated by setting the signal frequency and robot to space density to somewhat reasonably higher value.

In terms of target recipients, our model differs from a traditional publish/subscribe communication model by introducing the concept of dynamic subscription. In a traditional publish/subscribe communication model, subscription of messages happens prior to the actual message transmission. In that case prior knowledge about the subjects of a system is necessary. But in our model this is not necessary as long as all robots uses a common addressing convention for naming their incoming signal channels. In this way, when a robot meets with another robot it can infer the address of this peer robot's channel name by using a shared rule. A robot is thus always listening to its own channel for receiving messages from its potential peers or message publishers. On the other side, upon recognizing a peer, a robot sends a message to this particular peer. So here neither it is necessary to create any custom subject name-space [31] nor we need to hard-code information in each robot controller about the knowledge of their potential peers *a priori*. **Algorithm 4: Locality based P2P Communication Model**

Input: $RobotID, r_{comm}, r_{task}, TaskInfoDB, RobotPose,$
$ListeningBuffer, EmissionBuffer$
Output: updated $TaskInfoDB$
comment: Perception of a task, if the robot is within r_{task}
$TaskPose \leftarrow$ Estimate/get pose of a task within r_{task}
$TaskUrgency \leftarrow$ Estimate/get urgency of a task within r_{task}
$TaskInfo \leftarrow (TaskPose, TaskUrgency)$
$TaskInfoDB \leftarrow$ **UpdateTaskInfoDB(**$TaskInfo$**)**
comment: Listening *LocalTaskInfo* signal(s) from a peer
$RobotPeers \leftarrow$ Identify/get a list of peers within r_{comm}
for all peer $\in RobotPeers$ **do**
  if (caught a *LocalTaskInfo* signal from nearby peer) **then**
    $ListeningBuffer \leftarrow$ *LocalTaskInfo* of peer
    $TaskInfoDB \leftarrow$ **UpdateTaskInfoDB(**$ListeningBuffer$**)**
  end if
end for
comment: Emitting own *LocalTaskInfo* signal to peers
$EmissionBuffer \leftarrow TaskInfoDB$
for all peer $\in RobotPeers$ **do**
  EmitLocalTaskInfoSignal(peer, $EmissionBuffer$ **)**
end for
$RobotPeers \leftarrow 0$

Within the context of our self-regulated MRTA experiments under AFM, we have formalized the following three aspects of LPCM into an implementation algorithm.

1) Sensing the presence of the local peers and task, if any.
2) Listening to the task-information signals from local peers.
3) Emitting local task-information signals to local peers.

From Algorithm 4, we see that a robot controller is initialized with its specific $RobotID$ and default values of r_{comm} and r_{task} that correspond to the robot's communication and sensing ranges respectively. We have assumed that these values are same for all robots and for all tasks. Initially a robot has no information about tasks, i.e. $TaskInfoDB$, $ListeningBuffer$ and $EmissionBuffer$ all are empty. Upon sensing a task, robot determines the task's pose and urgency of that task. This is not strictly necessary as this information can also be available from alternate sources, e.g. via communicating with a TPS which is discussed later.

Robot controller then executes the **UpdateTaskInfoDB()** that modified the robot's information about the corresponding task. In second step, robot senses its nearby peers located within r_{comm}. The potential *LocalTaskInfo* signal reception from a peer again triggers the **UpdateTaskInfoDB()** function. In the last step, robot emits its own task information as a *LocalTaskInfo* signal to its peers.

VI. Experimental design

The overall aim of our MRTA experiments is to compare the various properties of self-regulated MRTA under both GSNC and LSLC strategies. These experiments are grouped into four series labelled using the following letters: *A, B, C* and *D*. Series A and B experiments are done using a centralized communication system under GSNC strategy and Series C and D experiments are carried out using an emulated local P2P communication under LSLC stategty. Below we describe the observables, parameters and implementation of our experiments.

A. Observables

Plasticity: Self-regulated MRTA is often characterised by the plasticity and task-specialization, in both macroscopic and microscopic levels. Within our manufacturing shop-floor context, plasticity refers to the collective ability of the robots to switch from doing no-task option (random-walking) to doing a task (or vice-versa) depending on the work-load present in the system. Here we expect to see that most of the robots would be able to engage in tasks when there would be high workloads (or task-urgencies) during PMM. Similarity, when there would be low workload in case of MOM, only a few robots would do the task, rest of them would either be idle (not doing any task) or perform a random-walk. The changes of task-urgencies and the ratio of robots engaged in tasks can be good metrics to observe plasticity in MRTA.

Task-specialization: Self-regulated MRTA is generally accompanied with task-specializations of agents. That means that few robots will be more active than others. From the interpretation of AFM, we can see that after doing a task a few times, a robot will soon be sensitized to it. Therefore, from the raw log of task-sensitization of robots, we can be able to find the pattern of task-sensitization of robots per task basis.

Quality of task-performance: As discussed in Sec. IV-C we can measure the quality of MRTA from the APCD. It first calculates the ideal minimum production time and then finds the delay in production process from the actual production completion data. Thus this will indicate how much more time is spent in the production process due to the self-regulation of robots in this distributed task-allocation scheme. In order to calculate APCD, we can find the production completion time for each task from the raw log of task-urgency and make an average from them.

Robustness: In order to see if our system can respond to the gradually increasing workloads, we can measure APMW within the context of our manufacturing shop-floor scenario. This can show the robustness of our system. When a task is not being served by any robot for some time we can see that its urgency will rise and robots will respond to this dynamic demand. For measuring APMW we need only the task-urgency data.

Flexibility: From the design of AFM, we know that robots that are not doing a task will be desensitized to it or forget that task. So at an overall low work-load (or task urgency), less robots will do the tasks and hence less robots will have the opportunity to learn tasks. From the shop-floor work-load data, we can confirm the presence of flexibility in MRTA.

Energy-efficiency: In order to characterize the energy-efficiency in MRTA we can log the pose data of each robot that can give us the total translations occurred by all robots in our experiments. This can give us a rough indication of energy-usage by our robots.

Information flow: Since AFM requires a system-wide continuous flow of information, we can measure the communication load to bench-mark our implementation of communication system. This bench-mark data can be used to compare among various communication strategies. Here we can measure how much task-related information, i.e. task-urgency, location etc. are sent to the robots at each time step. This amount of information or communication load can be constant or variable depending on the design of the communication system.

Scalability: In order to see the effects of scaling on MRTA, we have designed two group of experiments. Series A corresponds to a small group where

TABLE I: Experimental parameters of Series A & B experiments

Parameter	Series A | Series B
Total number of robots (N)	8 | 16
Total number of tasks (M)	2 | 4
Experiment area (A)	2 m^2 | 4 m^2
Initial production work load/machine (Ω_j^p)	100 unit
Task urgency increase rate ($\Delta\phi_{INC}$)	0.005
Task urgency decrease rate ($\Delta\phi_{DEC}$)	0.0025
Initial sensitization (K_{INIT})	0.1
Sensitization increase rate (Δk_{INC})	0.03
Sensitization decrease rate (Δk_{DEC})	0.01

we have used 8 robots, 2 tasks under an arena of 2 m^2. We have doubled these numbers in Series B, C and D, i.e. 16 robots, 4 tasks under an arena of 4 m^2. This proportional design can give us a valuable insight about the effects of scaling on self-regulated MRTA.

Thus, in order to observe the above properties of self-regulated MRTA , we have designed our experiments to record the following observables in each time-step.

1) Task-urgency of each task (ϕ).
2) Number of robots engaged in each task.
3) Task-sensitizations (k) of robots.
4) Pose data of robots.
5) Communication of task-information message among TPS and RCCs.

B. Parameters

Table I lists a set of essential parameters of our GSNC startegy based experiments (Series A and B). We intend to have a set-up that is relatively complex, i.e., with a high number of robots and tasks in a large area. The diameter of the marker of our e-puck robot is 0.08m. So, if we put 4 robots in an area of one square meter, this will give us a robot-occupied-space to free-space ratio of about 1:49 per square meter. This ratio is reasonable in order to allow the robots to move at a speed of 5 cm/sec without causing much interference to each other.

The initial values of task urgencies correspond to 100 units of production work-load without any maintenance work-load as outlined in Eq. 8. We choose a limit of 0 and 1, where 0 means no urgency and 1 means maximum urgency. Same rule applies to sensitisation, where 0 means no sensitisation and 1 means maximum sensitisation. This also implies that if sensitization is 0, task has been forgotten completely. On the other hand, if sensitization is 1, the task has been learnt completely. We choose a initial sensitization value of 0.1 for all tasks. The following relationships are maintained for selecting task-urgency and sensitization parameters.

$$\Delta\phi_{INC} = \frac{\Delta\phi_{DEC} \times N}{2 \times M} \tag{17}$$

$$\Delta k_{DEC} = \frac{\Delta k_{INC}}{M - 1} \tag{18}$$

Eq. 17 establishes the fact that task urgency will increase at a higher rate than that of its decrease. As we do not like to keep a task left unattended for a long time we choose a higher rate of increase of task urgency. This difference is set on the basis of our assumption that at least half of the expected number of robots (ratio of number of robots to tasks) would be available to work on a task. So they would produce similar types of increase and decrease behaviours in task urgencies.

Eq. 18 suggests that the learning will happen much faster than the forgetting. The difference in these two rates is based on the fact that faster leaning gives a robot more chances to select a task in next time-step and thus it becomes more specialized on it.

Our LSLC startegy based experiments (Series C and D) follow the similar design of experimental parameters and observables of Series B experiments as outlined above. Table II highlights the major parameters of these experiments. For the sake of simplicity, the communication and sensing range values are kept same in both series of experiments. Series D uses larger ranges that enables our robot to capture more task information at a time. That is

TABLE II: Experimental parameters of Series C & D experiments

Parameter	Series C | Series D
task-perception range (r_{task})	$0.5m$ | $1m$
Communication range (r_{comm})	$0.5m$ | $1m$
Total number of robots (N)	16
Total number of tasks (M)	4
Experiment area (A)	4 m^2

similar to "global sensing and global communication" of information by the robots, since in a large group of robots, it is not often practical that a robot communicates with all other robots. On the other hand, Series C uses the half of the range values of Series D. This enables our robots to capture less information, that mimics a close LSLC strategy.

VII. Implementation

Ideally, AFM can be implementation as a complete distributed task-allocation system where each agent selects its own task based on its own external perception about task-urgencies (i.e. attractive fields), distances from tasks and internal task-sensitisation records. Such an implementation requires powerful robots with sophisticated sensors (camera, laser etc.) and sufficient computation and communication capabilities. In that case, robots can keep task-urgency information up-to-date through suitable local communication schemes with their peers who can monitor the tasks. By using suitable navigation and mapping modules, they can also accurately calculate the distances from tasks and navigate to tasks autonomously. Moreover, they also require necessary hardware to do the actual task, e.g. gripper for picking up objects. However, in this study, we are particularly interested to find the suitable communication schemes that can effectively spread the attractive fields (task-urgencies) among robots. So we have simplified the complexities of a full-fledged implementation by using a centralized communication system that effectively makes up the limitations of our e-puck robots. For example, our robots are not capable of sensing a task to estimate its urgencies, instead our centralized TPS broadcast task information, e.g. task-urgencies, locations etc. to robots in certain time intervals.

In our current implementation, instead of doing any real work with powerful robots, we emulate a mock manufacturing shop-floor scenario that requires the robot only to travel among tasks. As our robots do not have on-board CPU, they need a host PC to control them using BTCom Bluetooth communication protocol over the wireless link [34]. Thus our host-PC runs one RCC for each physical robot. These RCCs also rely upon SwisTrack multi-robot tracking system for updating their real-time pose. So although our MRTA solution is distributed by design, we primarily used a centralized approach to implement it due to the limitations of our robots and the convenience of implementation. Table **??** list the D-Bus communication interfaces among our software components. Fig. **??** outlines the placements of various software components of our multi-robot control architecture based on their functional characteristics and processing requirements [35].

1) RCC Modules: As shown in Fig. 6, each e-puck robot is controlled by a corresponding RCC. A RCC sends commands to the robot's firmware using BTCom protocol. A RCC consists of several Python modules. Each module represents a sub-process under a main process that ties all of them together by using Python's Multiprocessing module[2]. Below we have described briefly the implementation of these Python modules. All code are publicly accessible through GitHub repository[3]. In the implementation of RCC, we have employed a data and event management process called `DataManager` that acts as a data warehouse and event management centre for RCC. Table **??** list the major data structures and corresponding event channels of `DataManager`. Here `mRobotID` represents the e-puck robot's marker ID (converted to decimal number from the binary code) which uniquely identifies a robot from others. We have used Python *dictionary* type data structure for all other data structures that are managed by the Python Multiprocessing's *Manager()* object. `DataManager` object also runs

[2]http://docs.python.org/library/multiprocessing.html

[3]git://@github.com/roboshepherd/EpuckCentralizedClient.git, *hash:*7e3af8902e64db3fa59e

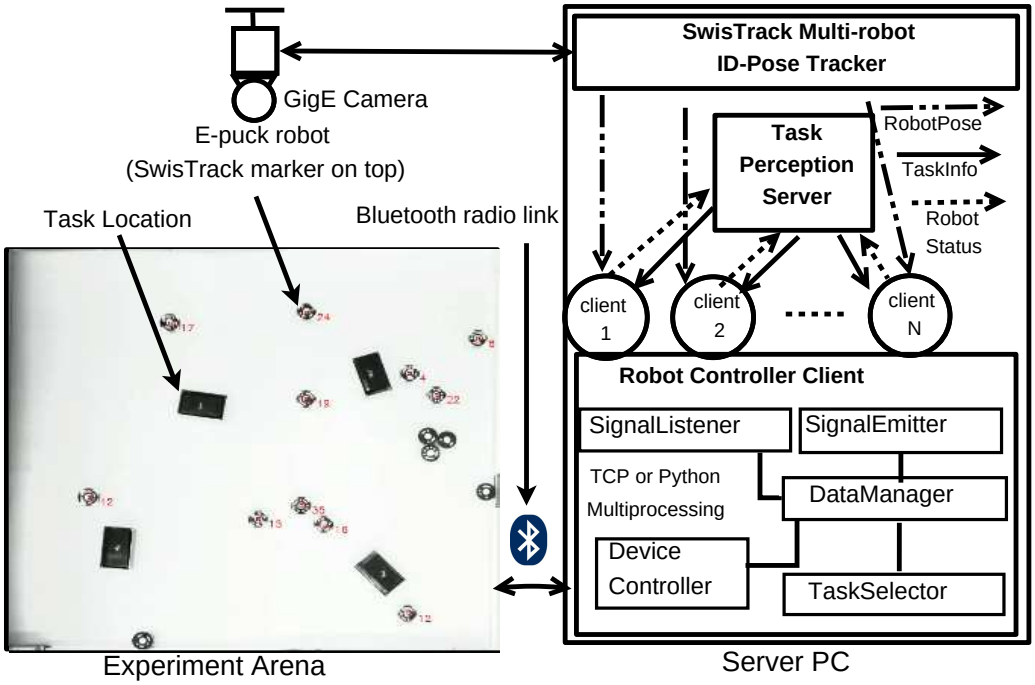


Fig. 6: Hardware and software setup for series A & B experiments

a tiny TCP/IP server process powered by the Multiprocessing's *RemoteManager()* interface. So, if necessary, any other process can access all of the DataManger's data structures and event channels remotely by instantiating a proxy client that connects to this TCP/IP server process.

We have employed two D-Bus communication modules in RCC: `SignalListener` and `SignalEmitter`, that are responsible for listening and emitting D-Bus signals respectively. `SignalListener`, has subscribed to D-Bus session bus for listening to D-Bus `RobotPose` and `TaskInfo` signals. `SignalEmitter` makes use of Python *sched* module that enables it to check periodically `mSelectedTaskStarted` event and emit a robot's task status containing signal, `RobotStatus`. `TaskSelector` works in the application layer of our multi-robot control architecture. It plugs the AFM algorithms (Sec. IV-D) into RCC to select task based-on `DataManger`'s event notifications.

The final code that moves a robot to desired task location or make random-walk resides in `DeviceController` module. When `TaskSelector` sets `mSelectedTaskAvailable` event `DeviceController` sub-process checks the Bluetooth link-connectivity with physical robot. In case of successful task start-up it sets `mSelectedTaskStarted` event that, in turn, triggers `SignalEmitter` to emit a D-Bus signal publishing the robot's currently engaged task. The full state switching policies of `DeviceController` is illustrated in Fig. **??**. Externally we can observe two distinct physical state of a robot: it is either idle or in motion (separated by dotted line). These two physical states are mapped into several logical steps as described below.

DeviceUnavailable: Periodically `DeviceController` checks whether the device has connected to host-PC and changes its

state to *DeviceAvailable* (i.e device connected). We can see that from every other state, device can come to this unavailable state (shown by dotted arrow) when the link is lost e.g. in case of robot's low battery or any other disconnection event (i.e device disconnected).

DeviceAvailable: `DeviceController` stays in this state until `Data-Manager`'s mSelected-TaskAvailable event fires (Device connected, no task selected). Then it switches to either *Move-ToTask* or *RandomWalk* state depending on the selected task. `DeviceController` returns to this state from *RandomWalk*, *MoveToTask* or *At-Task* states after a pre-set task time-out period has elapsed. In that case it triggers the `DataManager`'s `mTaskTimedOut` event.

RandomWalk: Robot can random walk in two cases: 1) when `TaskSelector` selects this random-walk task or 2) when the robot can not get its pose information for a moment. The latter helps the pose tracker to recover the robot-pose from a crowd of robots. Robot continues to do random-walk until it's task time-out period elapses or it's pose remains unavailable from the pose tracker (i.e. random-walk pending).

MoveToTask: In this state, robot takes step-by-step navigation approach to reach a task boundary. Until the robot reaches the task boundary (Away from task), in every navigation-step it adjusts its heading to task based on both robot and task pose information and then makes a fixed-time translation movement. In worse cases, when time-out happens early before reaching a task `DeviceController` switches from this state to *DeviceAvailable* state. However, if the same task is selected immediately, it is more likely that it will go to *AtTask* state in next few steps.

AtTask: In this state `DeviceController` discovers itself neat the task and normally until task time-out happens (i.e. task pending) it stays at this state.

In order to realize LPCM strategy, in the implementation of RCC we have added three new D-Bus signal interfaces. They are briefly specified in Table **??**. Our base implementation of RCC's task-allocation (i.e. `TaskSelector`) is almost unchanged in this local implementation. The only thing we need to extend is the D-Bus signal reception and emission interfaces that catches the above signals and put the signal payload in `DataManager`. For the sake of simplicity, we have merges the perceived `TaskInfo` with communicated `LocalTaskInfo` into one so that `TaskAllocator` always gets all available task information. We have not made any major change in other modules of RCC, i.e. `DeviceController`. The full Python implementation of this RCC is available for download from the GitHub repository[4].

2) TPS modules: In our GSNC strategy based implementation of TPS periodically broadcasts `TaskInfo` signals to all robots using similar D-Bus signal emitting and listening modules. Task information has been updated by a separate Python module following the TPS algorithm described before. In the local version, TPS only emits `TaskInfo` signal when a robot is within a task's perception range, r_{task} based-on the `TaskNeighbour` signal from SwisTrack. For this additional interface, we have extended D-Bus `SignalListener` to catch this additional D-Bus signal. No major changes are done in the other parts of TPS implementation. The full Python implementation of TPS is available for download from the authors' GitHub centralized[5] and distribued [6] repositories.

VIII. Results

In this section we have presented our experimental results. We ran those experiments for about 40 minutes duration. They are averaged over five iterations for Series A and B experiments and three iterations for Series C and D experiments.

A. Shop-floor work-load history

In our experiments we have defined shop-floor work-load in terms of task urgencies. For example, Eq. 8 shows how we have calculated initial

[4]http://github.com/roboshepherd/EpuckDistributedClient
[5]http://github.com/roboshepherd/CentralizedTaskServer
[6]http://github.com/roboshepherd/DitributedTaskServer

production work-load of our manufacturing shop-floor scenario. Fig. 11 show the dynamic changes in task-urgencies for the single iteration of each experiment. The fluctuations in these plots are resulted from the different levels of task-performance of our robots. In case of Series D, we can see that an unattended task, *Task4*, was not served by any robot for a long period and later it was picked up by some of the robots. In order to measure the task-related work-loads on our system we have summed up the changes in all task-urgencies over time. We call this as *shop-floor work-load history* and formalized as follows. Let $\phi_{j,q}$ be the urgency of a task j at q^{th} step and $\phi_{j,q+1}$ be the task urgency of $(q+1)^{th}$ step. We can calculate the sum of changes in urgencies of all M tasks at $(q+1)^{th}$ step:

$$\Delta\Phi_{j,q+1} = \sum_{j=1}^{M}(\phi_{j,q+1} - \phi_{j,q}) \qquad (19)$$

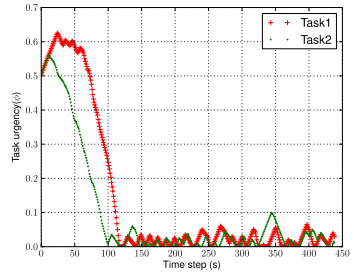
Fig. 12 shows the dynamic shop-floor workload for all four series of experiments. From these plots, we can see that initially the sum of changes of task urgencies (shop-floor workload) is going towards negative direction. This implies that tasks are being served by a high number of robots.

B. Ratio of active workers

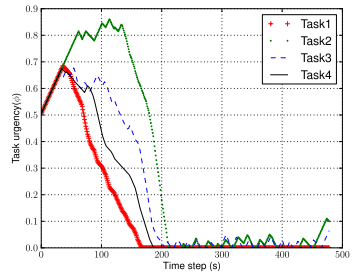
From Fig. 13, we can see that in production stage, when work-load is high, many robots are active in tasks. Here active workers ratio is the ratio of those robots that work on tasks to the total number of robots N of a particular experiment. Here we can see that this ratio varies according to the shop-floor work-load changes.

C. Shop-task performance

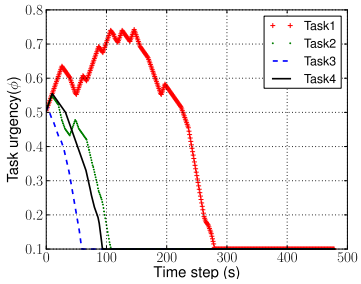
In our manufacturing shop-floor scenario, we have calculated the APCD and APMW as shown in Fig. **??**. For Series A we have got average production completion time 111 time-steps (555s) where sample size is (5 x 2) = 10 tasks, SD = 10 time-steps (50s). According to Eq. 12, our theoretical minimum production completion time is 50 time-steps (250s) assuming the non-stop task performance of all 8 robots with an initial task



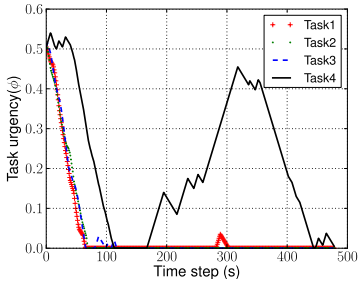
(a) Series A



(b) Series B

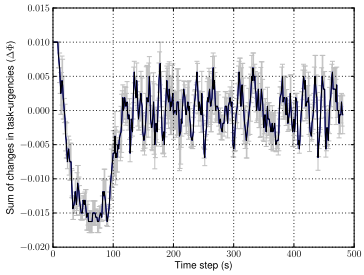


(c) Series C

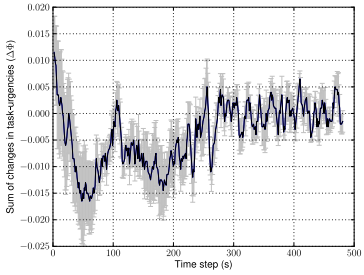


(d) Series D

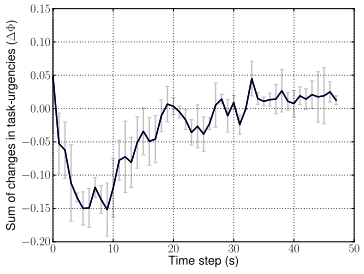
Fig. 11: Changes in task-urgencies.



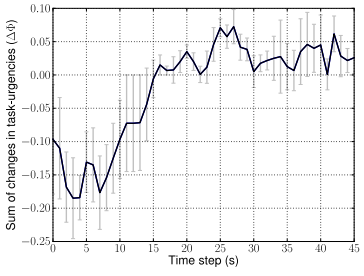
(a) Series A



(b) Series B

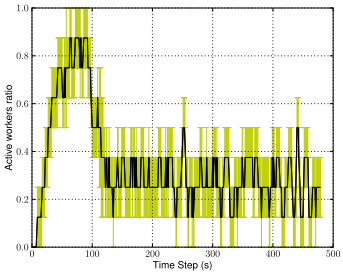


(c) Series C

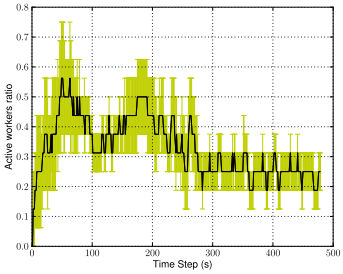


(d) Series D

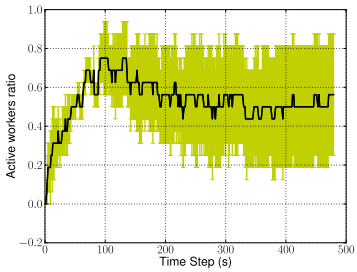
Fig. 12: Shop-floor workload change history.



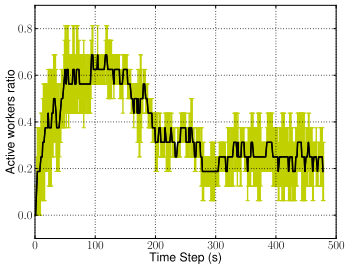
(a) Series A



(b) Series B



(c) Series C



(d) Series D

Fig. 13: Self-organized allocation of robots.

urgency of 0.5 for 2 tasks and task urgency decrease rate $\Delta\Phi_{DEC}$ = 0.0025 per robot per time-step. Hence, Eq. 13 gives us APCD, ζ = 1.22 which means that in Series A experiments, it took 1.22 times more time (305s) than the estimated minimum production completion time (250s). For Series B, we have got average production completion time 165 time-steps (825s) where sample size is (5 x 4) = 20 tasks, SD = 72 time-steps (360s). Hence, Eq. 13 gives us APCD, ζ = 2.3. For Series C we have got average production completion time 121 time-steps (605s) with SD = 36 time-steps (180s). For Series D, average production completion time is 123 time-steps (615s) with SD = 40 time-steps (200s). According to Eq. 12, our theoretical minimum production completion time is 50 time-steps (250s). The values of APCD are as follows. For Series C, ζ = 1.42 and for Series D, ζ = 1.46. For both series of experiments APCD values are very close.

For APMW, Series A experiments give us an average time length of 369 time-steps (1845s). In this period we have calculated APMW and it is 1 time-step with SD = 1 time-step (5s) and $\Delta\Phi_{INC}$ = 0.005 per task per time-step. This shows a very low APMW (χ = 0.000235) implying a very high robustness of our system. For Series B experiments, from the average 315 time-steps (1575s) maintenance activity of our robots per experiment run, we have got APMW, χ = 0.012756 which corresponds to the pending work of 3 time-steps (15s) where SD = 13 time-steps (65s). This also tells us the robust task performance of our robots which can return to an abandoned task within a minute or so. The APMW of Series C experiments give us an average time length of 359 time-steps (1795s). In this period we calculated APMW and it is 5 time-steps with SD = 17 time-steps and χ = 0.023420. For Series D experiments, from the average 357 time-steps (1575s) of maintenance activity of our robots per experiment run, we have got APMW, χ = 0.005359 which corresponds to the pending work of 2 time-steps (10s) where SD = 7 time-steps.

TABLE V: Peak task-sensitization values of robots in a Series A experiment.

Robot ID	Maximum k	At time-step (q)	Task
1	0.54	64	Task1
4	0.32	14	„
5	0.27	11	„
3	0.47	63	Task2
2	0.46	64	„
6	0.20	10	„
7	0.18	4	„
8	0.15	3	„

D. Task specializations

We have measured the task-specialization of the robots based-on their peak value of sensitization. This maximum value represents how long a robot has repeatedly been selecting a particular task. Since tasks are homogeneous we have considered the maximum sensitization value of a robot among all tasks during an experiment run. This value is then averaged for all robots using the following equation.

$$K^G_{avg} = \frac{1}{N}\sum_{i=1}^{N}\max_{j=1}^{M}\left(k^i_{j,q}\right) \tag{20}$$

If a robot r_i has the peak sensitization value k^i_j on task j ($j \in M$) at q^{th} time-step, Eq. 20 calculates the average of the peak task-specialization values of all robots for a certain iteration of our experiments. We have also averaged the time-step values (q) taken to reach those peak values for all robots using the following equation.

$$Q^G_{avg} = \frac{1}{N}\sum_{i=1}^{N} q^i_{k=k_{max}} \tag{21}$$

In Eq. 21, $q^i_{k=k_{max}}$ represents the time-step of robot r_i where its sensitization value k reaches the peak k_{max} as discussed above. By averaging this peak time-step values of all robots we can have an overall idea of how many task-execution cycles are spent to reach the maximum task-specialization value K^G_{avg}. Table V and Table **??** show the peak sensitization values of Series A and Series B experiments respectively. Based on Eq. 20 and Eq. 21, we have got the peak task-sensitization K^G_{avg}

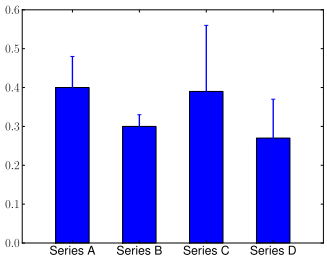
TABLE VII: Peak task-sensitization values of robots in a Series C experiment.

Robot ID	Maximum k	At time-step (q)	Task
16	1.00	52	Task1
12	0.58	24	,,
1	0.16	2	,,
13	0.13	1	,,
9	1.00	41	Task2
3	0.55	23	,,
19	0.24	6	,,
35	0.24	6	,,
15	0.13	1	,,
33	0.13	2	,,
6	0.61	29	Task3
31	0.59	35	,,
5	0.13	1	,,
17	0.36	10	Task4
20	0.09	1	,,
22	0.09	1	,,

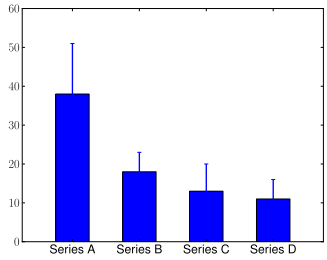
TABLE VIII: Peak task-sensitization values of robots in a Series D experiment.

Robot ID	Maximum k	At time-step (q)	Task
19	1.00	34	Task1
35	1.00	60	,,
24	0.41	16	,,
12	0.20	6	,,
1	1.00	41	Task2
31	1.00	47	,,
22	0.40	10	,,
9	0.38	12	,,
15	0.16	2	,,
33	0.13	1	,,
5	0.92	70	Task3
13	0.74	36	,,
17	0.76	62	Task4
6	0.22	4	,,
3	0.13	2	,,
16	0.13	1	,,

values: 0.40 (SD=0.08) and 0.30 (SD=0.03), and their respective time-step Q^G_{avg} values: 38 (SD=13) and 18 (SD=5) time-step. They are shown in Fig. 15. Here we can see that the robots in Series A had higher chances of task-specialization than that of Series B experiments. Fig. 16 shows us the task specialization of five robots on *Task3* in a particular run of Series B experiment. This shows us how some of the robots can specialize (learn) and de-specialize (forget) tasks over time.



(a) Task-sensitization (K)



(b) Time-steps (Q)

Fig. 15: Overall task-specialization of robot groups based on Peak task-sensitization values.

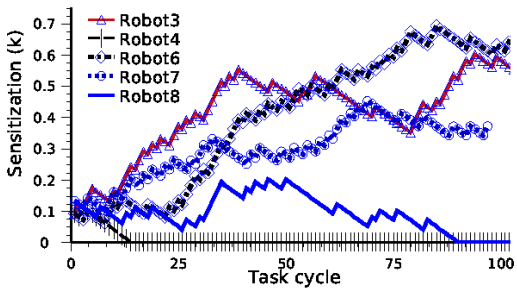


Fig. 16: Task specialization on Task3 for a Series B experiment.

E. Robot motions

We have aggregated the changes in translation motion of all robots over time based on their travelled distances in Euclidean metric. Let $u_{i,q}$ and $u_{i,q+1}$ be the translations of a robot i in two consecutive steps. If the difference between these two translations be δu_i, we can find the sum of changes of translations of all robots in $(q+1)^{th}$ step using the following equation.

$$\Delta U_{q+1} = \sum_{i=1}^{N} \delta u_{i,q+1} \tag{22}$$

The sum of translations of robots from our experiments are plotted in Fig. 17. In this plot we can see that robot translations also vary over varying task requirements of tasks.

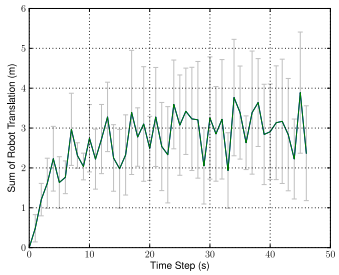
F. Communication load

Fig. 18 (a) and (b) shows the number of received `TaskInfo` signals by each robot in Series A and Series B experiments. Since the duration of each time-step is 50s long and TPS emits signal in every 2.5s, there is an average of 20 signals in each time-step. The overall P2P task information signals of both of these local modes are plotted in Fig. 18 (c) and (d). As an example of P2P signal reception of a robot, Fig. **??** show the number of received signals by *Robot12* in two local experiments. It states the relative difference of peers over time in two different cases.

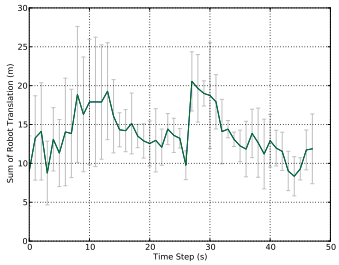
IX. Discussions

A. Self-regulated MRTA

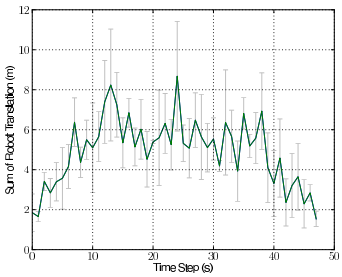
From our experimental results, we have noted several aspects of self-regulated MRTA that expose the effectiveness of AFM. As we have pointed out that this self-regulated MRTA, as observed in biological and human social systems, needs to satisfy several important characteristics, particularly plasticity and task-specialization. In addition to satisfying those basic qualities, AFM has demonstrated many other aspects. Our self-regulated robots, driven by AFM, effectively handle the dynamic work-load in our manufacturing shop-floor. They can dynamically support the need to



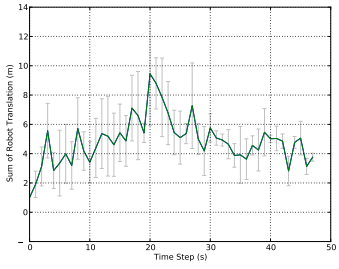
(a) Series A



(b) Series B

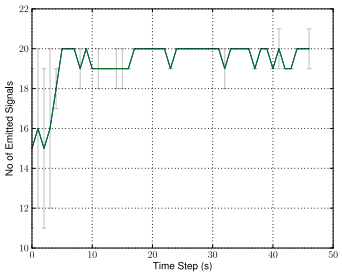


(c) Series C

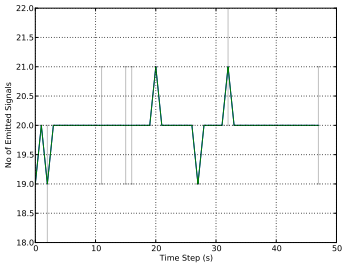


(d) Series D

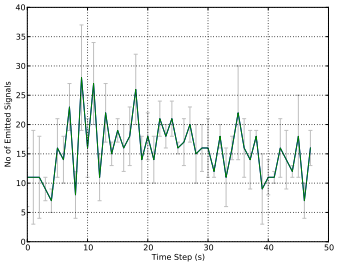
Fig. 17: Sum of the translations of robots



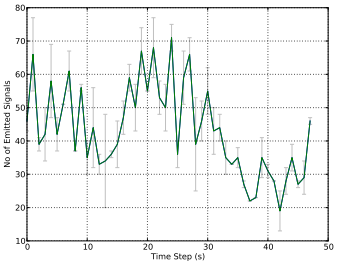
(a) Series A



(b) Series B



(c) Series C



(d) Series D

Fig. 18: Frequency of `TaskInfo` signalling of (a) and (b) by TPS and (c) and (d) by local peers.

work on demanding tasks, if there any. The variations of active worker ratio shows this.

From the self-organized worker allocations of AFM, it is clear to us that although in larger system (Series B) the degree of variations of active-worker ratio can show us significantly unpredictable patterns, nevertheless the self-regulated rules drive the robots to respond to the dynamic needs of the system. This means that AFM can sufficiently produce the plasticity of DOL in order to meet the dynamic work-load of the system.

B. Learning and forgetting

From the individual and group-level task-specialization, we can see that robots can maintain both task-specialization and flexibility. In a self-organized system, it is very common that only a few individuals specialize on tasks and others generally do not. From two sample data sets provided in Table V and Table **??**, we can see that in particular runs of Series A and Series B experiments, task-sensitization values of only 2-3 robots reach above the group-level average score. Thus in both types of experiments, robots exhibit similar task-specialization behaviours.

C. Concurrency and robustness

As a consequence of fewer robots specializing in tasks, we can also see that robots can concurrently consider different tasks without being biased to a particular task all the time. Our experiments also show us the robust MRTA as in case of both high and low work-loads present in the system. This is evident from the manufacturing shop-floor task performance during PMM and MOM. For example, in case of Series B experiments APMW was 13 time-steps (65s) which corresponded to pending work-load of 0.065 unit for a single robot. Thus, on an average, before the work-load exceeded by about 13 percent of initial work-load, robots were able to respond to a task.

D. Communication load

In GSNC strategy based experiments we used a centralized communication system (source of attractive fields) that serves the robots with necessary task-perception information. Our centralized communication system has the advantage of minimising the communication load and the disadvantage of a single point of failure as well as a single point of load. Under LSLC strategy based implementation, we present how the task perception can be decentralized by P2P communications among RCCs.

E. Scaling-up

We have observed the effect of scaling-up the robot team size. The system size of Series B is double of that of Series A in terms of robots, tasks and experiment arena. Keeping a fixed ratio of robot-to-task and task-to-arena we have intended to see the scaling effects in our experiments. Here we see that both systems can show sufficient self-regulated DOL, but task-performance of both systems varies significantly. For example, the value of APCD in Series B is higher by 1.08. This means that performance is decreased in Series B experiments despite having the resources in same proportion in both systems. This occurs partly due to the greater stochastic effects found in task-allocation in a larger system, e.g. presence of more tasks produce higher stochastic behaviours in robot's task selection.

Similarly we can see that in larger system robots have less chances to specialize on tasks, as the Series B experiments show us that the overall average task-specialization of the group K^G_{avg} is lower by 0.10 and it lasts for significantly less time (the difference of Q^G_{avg} of both systems is 20 time-steps). Thus, in a large group, robots are more likely to switch among tasks more frequently and this produces more translation motions which cost more energy (e.g. battery power) in task-performance.

F. Comparisons between GSNC and LSLC strategies

Results from Series C and Series D experiments show us many similarities and differences with respect to the results of Series A and Series B experiments. Both Series C and Series D experiments show similar APCD values: 1.42 and 1.46 respectively, which are significantly less than Series B experiment result (APCD = 2.3) and are close to

TABLE IX: Sum of translations of robots in our experiments.

Series	Average translation (m)	SD
A	2.631	0.804
B	13.882	3.099
C	4.907	1.678
D	4.854	1.592

Series A experiment result (APCD = 1.22). This means that for large group, task-performance is efficient under LSLC strategy (Series C and Series D) comparing with their GSNC counterpart (Series B).

Besides, in terms of task-specialization, the overall task-specialization of group in Series C (K^G_{avg} = 0.4) is closer to that of Series A experiments (K^G_{avg} = 0.39) and interestingly, the value of Series D (K^G_{avg} = 0.27) is much closer to that of Series B experiments (K^G_{avg} = 0.30). So task-specialization in large group under LSLC strategy shows higher performance than their GSNC counter part. Besides task-specialization happens much faster under LSLC strategy as we can see that the average time to reach peak sensitization values of the group, Q^G_{avg} in Series C is lower than that of Series A values by 25 time-steps.

From the robot motion profiles found in all four series of experiments, we have found that under LSLC strategy, robot translations have been reduced significantly. Table IX summarizes the average translations done by robots in all four series of experiments. From this table we can see than Series C and Series D show about 2.8 times less translation than that in Series B experiments. The translation of 16 robots in Series C and Series D experiments are approximately double (1.89 times) than that of Series A experiments with 8 robots. Thus the energy-efficiency under LSLC strategy seems to be higher than that under GSNC strategy.

From the above results we can see that large group of robots achieve better MRTA under LSLC strategy. The local sensing of tasks prevents them to attend a far-reaching task which may be more common under global sensing strategy. However, as we have seen in Fig. 11 some tasks can be left unattended for a long period of time due to the failure to discover it by any robot. For that reason we see that the values of APMW is slightly higher under LSLC strategy. But this trade-off is worth as LSLC strategy provides superior self-regulated MRTA in terms of task-performance, task-specialization and energy-efficiency.

X. Conclusion and future works

The benefits of deploying a large number of robots in dynamic multi-tasking environment can not be fully realized without adopting an effective communication and sensing strategy for a given multi-robot system. This study has focused on comparing two bio-inspired communication and sensing strategies in producing self-regulated MRTA by an interdisciplinary model of DOL, AFM. Under the GSNC strategy, AFM has produced the desired self-regulated MRTA among a group of 8 and 16 robots. This gives us the evidence that AFM can successfully solve the MRTA issue of a complex multi-tasking environment like a manufacturing shop-floor. Under the LSLC strategy, AFM can also produce the desired self-regulated MRTA for 16 robots with different communication and sensing ranges.

From our comparative results, we note that for large group of robots, degradation in task-performance and task-specialization of robots are likely to occur under GSNC strategy that relies upon a centralized communication system. Thus GSNC strategy can give us better performance when the number of tasks and robots are relatively small. This confirms us the assertions made by some biologists that self-regulated DOL among small group of individuals can happen without any significant amount of local communications and interactions. However, our findings suggest that task-specialization can still be beneficial among the individuals of a small group which contradicts the claim that small groups only posses the generalist workers, but not the specialists.

On the other hand, LSLC strategy is more suitable for large group of individuals that are likely to be unable to perform global sensing and global communications with all individuals of the group.

The design of communication and sensing range has still remained as a critical research issue. However, our results suggest that the idea of maximizing information gain is not appropriate under a stochastic task-allocation process. Here more information tends to cause more task-switching behaviours that lowers the level of task-specialization of the group. This might not be the case under a deterministic task-allocation scheme where more information may lead to better and optimum allocations in a small group of individuals. Nevertheless, despite having the limited communication and sensing range, LSLC strategy has helped the robots to produce comparatively better task-allocation with increased task-specialization and significantly reduced motions or savings in energy consumption.

REFERENCES

[1] R. C. Arkin, *Behavior-based robotics*. Cambridge, Mass. : MIT Press, c1998., 1998, includes bibliographical references (p. [445]-476) and indexes.

[2] L. E. Parker and F. Tang, "Building multirobot coalitions through automated task solution synthesis," *Proceedings of the IEEE*, vol. 94, pp. 1289–1305, 2006.

[3] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, p. 939, 2004.

[4] A. B. Sendova-Franks and N. R. Franks, "Self-assembly, self-organization and division of labour'," *Philosophical Transactions of the Royal Society of London B*, vol. 354, pp. 1395–1405, 1999.

[5] B. Gerkey and M. Mataric, "Multi-robot task allocation: analyzing the complexity and optimality of key architectures," *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 3, 2003.

[6] L. E. Parker, "Distributed intelligence: Overview of the field and its application in multi-robot systems," *Journal of Physical Agents, special issue on multi-robot systems*, vol. vol. 2, no. no. 2, pp. 5–14, 2008.

[7] L. Chaimowicz, M. F. M. Campos, and V. Kumar, "Dynamic role assignment for cooperative robots," *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1, 2002.

[8] M. B. Dias, R. M. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, pp. 1257–1270, 2006.

[9] K. Lerman, C. Jones, A. Galstyan, and M. J. Mataric, "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, p. 225, 2006.

[10] C. R. Kube and H. Zhang, "Collective robotics: From social insects to robots," *Adaptive Behavior*, vol. 2, p. 189, 1993.

[11] W. Liu, A. F. T. Winfield, J. Sa, J. Chen, and L. Dou, "Towards energy optimization: Emergent task allocation in a swarm of foraging robots," *Adaptive Behavior*, vol. 15, no. 3, pp. 289–305, 2007.

[12] E. Arcaute, K. Christensen, A. Sendova-Franks, T. Dahl, A. Espinosa, and H. J. Jensen, "Division of labour in ant colonies in terms of attractive fields," *Ecol. Complexity*, 2008.

[13] R. Jeanne, "Group size, productivity, and information flow in social wasps," *Information processing in social insects*, pp. 3–30, 1999.

[14] L. E. Parker, "Alliance: an architecture for fault tolerant multirobot cooperation," *Robotics and Automation, IEEE Transactions on*, vol. 14, pp. 220–240, 1998.

[15] W. Shen, D. H. Norrie, and J.-P. Barthes, *Multi-agent systems for concurrent intelligent design and manufacturing*. London: Taylor & Francis, 2001.

[16] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *Artificial Intelligence*, vol. 110, pp. 241–273, 1999.

[17] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Transactions on Robotics*, vol. 20, no. 5, pp. 865–875, 2004.

[18] G. Pereira, V. Kumar, and M. Campos, "Decentralized algorithms for multirobot manipulation via caging," *Algorithmic Foundations of Robotics V*, pp. 257–274, 2003.

[19] T. Dahl, M. Mataric, and G. Sukhatme, "Distributed multi-robot task allocation through vacancy chains," *Autonomous Robots*, 2004.

[20] S. Camazine, N. Franks, J. Sneyd, E. Bonabeau, J. Deneubourg, and G. Theraula, *Self-organization in biological systems*. Princeton, N.J. : Princeton University Press, c2001., 2001, what is self-organization? – How self-organization works – Characteristics of self-organizing systems – Alternatives to self-organization –.

[21] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford University Press, 1999.

[22] M. Krieger and J. Billeter, "The call of duty: Self-organised task allocation in a population of up to twelve mobile robots," *Robotics and Autonomous Systems*, vol. 30, no. 1-2, pp. 65–84, 2000.

[23] W. Agassounon and A. Martinoli, "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, 2002, p. 1097.

[24] C. Jones and M. Mataric, "Adaptive division of labor in large-scale minimalist multi-robot systems," in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings*, 2003.

[25] N. Kalra and A. Martinoli, "A comparative study of market-based and threshold-based task allocation," *Distributed Autonomous Robotic Systems 7*, pp. 91–101, 2007.

[26] B. B. Werger and M. J. Mataric, "Broadcast of local eligibility for multi-target observation," *Distributed Autonomous Robotic Systems*, vol. 4, p. 347356, 2001.

[27] S. Botelho, R. Alami, and T. LAAS-CNRS, "M+: a scheme for multi-robot cooperation through negotiated taskallo-

cation and achievement," *Proceedings IEEE International Conference on Robotics and Automation*, vol. 2, 1999.

[28] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE Transaction on Robotics and Automation*, vol. 18, 2002.

[29] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3, 2002.

[30] H. Çelikkanat, A. Turgut, and E. Şahin, "Guiding a robot flock via informed robots," *Distributed Autonomous Robotic Systems 8*, pp. 215–225, 2008.

[31] B. Gerkey and M. Mataric, "Principled communication for dynamic multi-robot task allocation," *Experimental Robotics VII*, pp. 353–362, 2001.

[32] B. Holldobler and E. Wilson, *The ants*. Belknap Press, 1990.

[33] E. Yoshida and T. Arai, "Performance analysis of local communication by cooperating mobile robots," *IEICE Transactions on Communications*, vol. 83, no. 5, pp. 1048–1059, 2000.

[34] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proc. of the 9th Conf. on Autonomous Robot Systems and Competitions*, 2009.

[35] M. Sarker and T. Dahl, "Flexible Communication in Multi-robotic Control System Using HEAD: Hybrid Event-driven Architecture on D-Bus," in *In Proc. of the UKACC International Conference on Control 2010 (CONTROL 2010), to appear*, 2010.