Decentralized Communications for Self-regulated Division of Labour in Robot Society

Md Omar Faruque Sarker and Torbjørn S. Dahl
Robotic Intelligence Lab, Newport Business School
University of Wales, Newport, Allt-yr-yn Campus
Allt-yr-yn Avenue, Newport, NP205XR, UK
Mdomarfaruque.Sarker@newport.ac.uk
Torbjorn.Dahl@newport.ac.uk

***Abstract*— Distributed local communication is one of the essential means by which various social insects achieve their self-regulatory division of labour (SRDL). Unlike centralized static communication, this communication mode enables individuals to respond to local changes quickly and it generally produces steady-state convergence of SRDL in social insects. However, realizing this kind of communication in a distributed multi-robot system (MRS) is not as straight forward as a centralized one. From a robot controller's point of view, it is not easy to determine how often or how much dynamic peer-to-peer (P2P) communication is needed to maintain system's convergence of SRDL. To deal with this issue, in this paper, we first present a model for dynamic local communication and its implementation on a MRS with two different communication radii. Then we compare this system with our baseline centralized communication based MRS in terms of convergence of SRDL, communication load, robot task specializations and their motions. Results from these experiments suggest us that similar or better convergence of SRDL can be obtained by setting a smaller P2P communication radius where a robot locally exchanges signals with a minimum number of its nearby peers. Our experiments used 16 e-puck robots in a 2m X 2m area.**

I. INTRODUCTION

Self-regulated division of labour (SRDL) is one of the key research issues in the filed of multi-agent and multi-robot systems (MRS). Inspired from biological and various other social systems, robotic researchers have achieved SRDL in MRS using various forms of communications [**?**]. Two basic forms of communications are: 1) direct or explicit communication and 2) indirect or implicit communication. Direct communication is an intentional communicative act of message passing that aims at one or more particular receiver(s) [12]. It typically exchanges information through physical signals. In contrast, indirect communication, sometimes termed as *stigmergic* in biological literature, happens as a form of modifying the environment (e.g., pheromone dropping by ants) [1]. In ordinary sense, this is an observed behaviour and many researchers call it as *no communication* [2]. In order to avoid ambiguity, by the term *communication*, we always refer to direct communication. In this paper, we also confine our discussion on SRDL within the context of direct communication only.

In order to pursue a SRDL, robots can receive information from a centralised source [10] or from their local peers [11]. In [6], we reported a steady-state convergence of SRDL in a practical MRS using a centralized information source. This centralized communication system is easy to implement and it simplifies the overall design of a robot controller. However this system has disadvantage of a single point of failure and it is not scalable. The increased number of robots and tasks cause inevitable increase in communication load and transmission delay and as a consequence, the overall system performance degrades. On the other hand, uncontrolled reception of information from decentralized or local sources is also not free from drawbacks. If a robot exchanges signals with all other robots (hereafter called as *peers*), it might get the global view of the system quickly and can select an optimal or near optimal task. This can produce a great improvement in overall performance of some types of tasks e.g., in area coverage [9]. But this is also not practical and scalable for a typically large MRS due to the limited communication and computation capabilities of robots and limited available communication bandwidth.

A potential alternate solution to this problem can be obtained by decreasing the number of message recipient peers on the basis of a local communication radius (r_{comm}). This means that robots are allowed to communicate only with those peers who are physically located within a preset distance. When this strategy is used for sharing task information among peers, SRDL can be more robust and efficient. This concept was validated by simulation in [11]. But they did not give us any insight for choosing the value of r_{comm}. In this paper, we have presented a set of experimental results of SRDL in our MRS with two values of r_{comm}: 0.5m and 1m. In case of the former one, we call the peers located within CR as *nearby peers*. Similarly in case of latter one, we call the peers located within CR as *distant peers*. Along with a practical insight for selecting r_{comm} value, various other design issues have been tackled in this paper. The recursion-free design of local communication channels is achieved by a dynamic publish/subscribe model of communication. This has been validated by using a state-of-the-art D-Bus [1] inter-process communication (IPC)ctechnology in Linux.

Our contributions from this study are as follows. We present

[1] http://dbus.freedesktop.org/doc/dbus-specification.html

a dynamic publish/subscribe local model of communication that achieves similar or better SRDL than its centralized counterpart. The reduction of robot motion is half in this case and this tells us about the level of impact this local model can make on the energy efficiency of a MRS. The reduction in communication load in local model is also significant. Unlike a fixed communication frequency, the dynamic variation in communication frequency produced by by our local model suggests us that local communication model should be preferred to a centralized one when robustness and scalability of SRDL of a MRS is an important issue.

Rest of the paper is organized as follows. Section II presents our local communication model that allows us to implement a inter-disciplinary generic model of SRDL as a multi-robot task allocation (MTRA) mechanism. Section IV introduces our implementation of MRTA including the interactions between the hardware, software and communication modules. Section V presents the design of our experiments including specific parameters and observables. Section VI discusses our experimental results. Section VII discusses related background works. Section VII discusses related background works and section VIII draws conclusions.

II. Attactive Field Model (AFM)

Our model of self-regulated DoL is based on AFM. It provides us a generic framework for implementing self-regulatory DoL in robots. Here we briefly describe how this model gives our robots self-regulatory DoL behaviours, particularly task-specialization, concurrency, flexibility and robustness. Interested readers should consult [3] for a general overview and and [6] a robotic implementation of AFM.

Let us consider a manufacturing shop floor scenario where N number of mobile robots are required to attend to M number of shop tasks spread over a fixed area A. Let these tasks be represented by a set of small rectangular boxes resembling to manufacturing machines. Let R_1, R_2 R_n be the set of all robots and J_1, J_2 J_m be the set of all tasks. Each task j has an associated task-urgency ϕ_j that indicates its relative importance over time. If a robot attends to a task j in xth time-step, value of ϕ_j will decreases by a small amount δ_ϕ in (x+1)th time-step. On the other hand, if a task has not been served by any robot in xth time-step, ϕ_j will increase by another small amount in (x+1)th time-step. In order to complete a shop task J_1, a robot R_1 needs to reach within a fixed boundary D_j1 of J_1. If a robot completes a task j we say that it learns about it and this will increase robot's likelihood of selecting that task in next step. We call this variable affinity of a robot to that task as its sensitization k_j . If a robot does not do a task j for some time, we say that it forgets about j and k_j has been decreased.

According AFM, all robots will establish attractive fields to all tasks due to the presence of a system-wide continuous flow of information. The strength of these attractive fields called stimulus will vary according to the distances between robots and tasks, task-urgencies and corresponding sensitizations of robots. This is encoded in Eq. 1.

$$S_j^i = tanh\{\frac{k_j^i}{d+\delta}\phi_j\} \quad (1) \qquad\qquad P_j^i = \frac{S_j^i}{\sum_j S_j^i} \quad (2)$$

Eqn 1 says that the stimuli of a robot i to a particular task j, (S_j^i) depends on robot's spatial distance d to j, level of sensitization to that task (k_j^i) and perceived urgency of that task (ϕ_j). We use a vary small value δ in 1 to prevent division by zero. The probability of selecting each task has been determined by a probabilistic method outlined in Eq. 2 . AFM suggests concurrency of a self-regulatory system by specifying at least two task options: 1) doing a task and 2) doing no task. In robots, the latter can be be treated as random walking. So in any time-step a robot will choose from M+1 tasks. Let T_a be the allocated time to accomplish a task. If R_1 can enter inside the task boundary within T_a time it waits there until T_a elapsed. Otherwise it will select a different task.

III. Our Locality-based P2P Communication Model

A. Characteristics

Our Locality-based P2P Communication (LPC) model relies upon the local P2P communications among robots. Here there is no centralized server to disseminate information but each robot can communicate to its nearby peers within a certain communication radius, r_{comm}. Here by r_{comm}, we assume that within this distance robots can 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 its r_{comm}. As shown in Fig. 1, local communication can also give robots similar task information as in centralized communication mode. It shows that 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 task 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 many ways: such as, by using their sensory perception (e.g., camera), by doing local P2P communication and so on. In Fig. reffig:lcm we have shown that robots exchange both task-info and self-status signals to peers.

We characterize our communication model in terms of three fundamental issues: 1) message content (*what to communicate*) 2) communication frequency (*when to communicate*) and 3) target recipients (*with whom to communicate*) [8]. In a typical MRS, message content can be categorized into two types: 1) state of each individual robot and 2) target task (goal) information [7]. The latter can also be subdivided into two types: 1) an individual robot's target task information 2) information of all available tasks found in the system. Regarding the first issue, our communication model is open. Robots can communicate with their peers with any kind of message. Our model addresses the last

two issues very specifically. Robots communicate only when they meet their peers within a certain communication radius (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 (PSC) model by introducing the concept of dynamic subscription. In a traditional PSC 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 it neither necessary to create any custom subject namespace (e.g., [8]) nor to hard-code information in each robot controller about the knowledge of their potential peers. Subscription is done automatically based on their respective r_{comm}.

B. Implementation Algorithm

Our LPC model has three major aspects: 1) local sensing of peers (and optionally tasks) 2) listening signals from peers and 3) emitting signals to peers. Here we present a typical implementation. Let N be the set of robots and at time step q, a robot i that can receive $h_{i,q}$ information by listening to its incoming channel L_i. Let M be the set of tasks and each task j has an associated information H_j. It encodes the necessary properties of tasks, such as their locations, urgencies etc. Each task j also has a task perception radius r_{task} such that if a robot comes within this radius at time step q it can perceive current value of $H_{j,q}$. Let at time step q, robot i has its own task information $G_{i,q}$ that has been perceived and listened from its peers. Let r_{comm} be the communication radius of each robot. Let at time step q, $P^i_{p,q}$ be a set of peer robots of i that are within r^i_{comm}. Let $E^i_{p,q}$ be its active signal emission channels. Algorithm 1 shows implementation of our proposed P2P dynamic communication.

Algorithm 1: Locality based Dynamic P2P Communication

Initialization:
$robotid \leftarrow id$
$r_{comm} \leftarrow r_1$
$r_{task} \leftarrow r_2$
$pose[id] \leftarrow (0,0,0)$
$G[id],\ P[id],\ L[id],\ E[\,] \leftarrow 0$
Loop:
$pose[id] \leftarrow (x,y,\theta)$
if $pose[id] \in U(pose[k],\ r^k_{task}), (k = 0,1,.,M-1)$ **then**
    $G[id] \leftarrow G[id]\ \cup\ H_k$
end if
if $pose[id] \in V(pose[k],\ r^k_{comm}), (k = 0,1,.,N-1, k \neq id)$ **then**
    $P[id] \leftarrow P[id]\ \cup\ k$
    $h_k \leftarrow W(E[k],\ L[id])$
    $G[id] \leftarrow G[id]\ \cup\ h_k$
end if
for all $k \in P[id], (k = 0,1,.,N-1, k \neq id)$ **do**
    $W(E[id],\ L[k]) \leftarrow G[id]$
end for
$P[id] \leftarrow 0$
Loop again

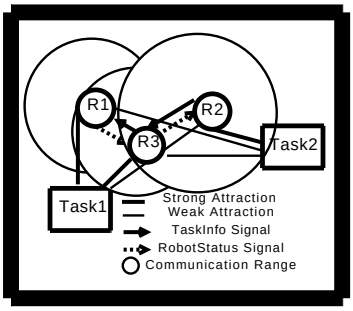

Fig. 1. Local communication model

From Algorithm 1, we see a robot controller is initialized with a specific robot-id and default values of r_{comm} and r_{task}. We assumed that these values are same for all robots and for all tasks. Initially a robot has no information about tasks and it neither has not listened anything from any peer nor it transmitted any information. Upon initialization, robot determines it current pose and evaluates a function $U(pose,\ r_{task})$ that helps it to perceiving a nearby task. This is not strictly necessary as this information can be available from alternate sources. In second step, robot senses its nearby peers by evaluating $V(pose,\ r_{comm})$ and start filling the list of peers P by their id. The signal exchange with a peer is denoted by a communication function $W(emitter,\ listener)$. So by listening it receives task information h and aggregates it with own task info G. In last stage, robot emits its task information to the peers from list P. Finally it removes id of all the current peers and repeat this loop.

IV. Implementation

We have developed a system where up to 40 E-puck robots [5] can operate together according to the generic rules of the AFM. As shown in Fig. 2 (right), our software system consists of a multi-robot tracking system, a task perception assistant (TPA) and robot controller clients (RCC). Here at first we have presented the design of our communication system. Then we have discussed about our specific implementation.

A. Our communication system

As shown in Fig. 1, RCCs disseminate task information to each other by *TaskInfo* D-Bus signal. The contents of task information are physical locations of tasks and their urgencies. However as our robots are incapable of sensing

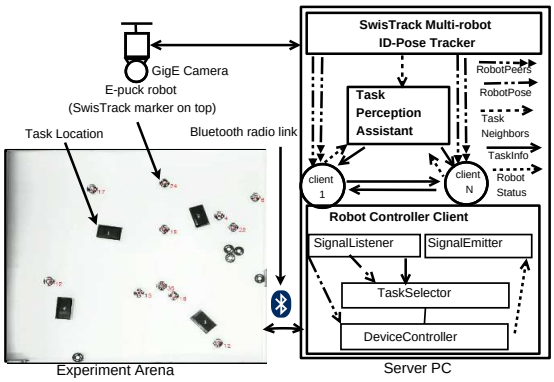

Fig. 2. Hardware and software setup

task directly. So it relies on TPA for actual task location and urgency. When a robot i comes within r_{task} of a task j, SwisTrack reports it to TPA (by *TaskNeighbors* signal) and TPA then gives it current task information of that task H_j to i (by *TaskInfo* signal). TPA has another interface for catching feedback signals from robots. The *RobotStatus* signal can be used to inform TPA about a robot's current task id, its device status and so on. TPA uses this information to update relevant part of task information such as, task-urgency. This up-to-date information is encoded in next TaskInfo signal.

In Fig. 1 an initial configuration of our communication system has been presented. Upon receiving an initial Task-Info signal from TPA, robot R_3 has shown strong attraction towards *Task*1 and robot R_2 has shown strong attraction toward *Task*2. However R_1 receives information about both tasks through its peer R_3 and shows strong attraction towrads Task1. These can be inferred from Eq. 1 that says if the initial task urgencies and sensitizations for all tasks are same, a robot will strongly be attracted towards a task that is relatively closer to it.

B. Our current implementation

In order to track all robots real-time we have used Swis-Track [4], a state of the art open-source, multi-agent tracking system, with a16-megapixel overhead GigE camera. This set-up gives us the position, heading and id of each of the robots at a frequency of 1. The interaction of the hardware and software of our system is illustrated in Fig. 2.

For IPC we have used D-Bus technology. We have developed an IPC component for SwisTrack (hereafter called as *SwisTrack D-Bus Server*) that can broadcast id and pose of all robots in real-time over our server's D-Bus interface.

Apart from SwisTrack, we have implemented two major software modules: *Task Perception Assistant (TPA)* and *Robot Controller Client (RCC)*. They are developed in Python with its state of the art *Multiprocessing* [2] module. This python module simplifies our need to manage data sharing and synchronization among different sub-processes. As shown in Fig. 2, RCC consists of four sub-processes. *SignalListener* and *SignalEmitter*, interface with SwisTrack D-Bus Server and TPA. *TaskSelector* implements AFM guidelines for task selection . *DeviceController* moves a robot to a target task. Bluetooth radio link is used as a communication medium between a RCC and a corresponding E-puck robot.

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

TABLE I
EXPERIMENTAL PARAMETERS

Parameter	Value
Total number of robots (N)	16
Total number of tasks (M)	4
Experiment area (A)	4 m^2
Intial task urgency (Φ_{INIT})	0.5
Task urgency increase rate ($\Delta\phi_{INC}$)	0.005
Task urgency decrease rate ($\Delta\phi_{DEC}$)	0.0025
Intial sensitization (K_{INIT})	0.1
Sensitization increase rate (Δk_{INC})	0.03
Sensitization decrease rate (Δk_{DEC})	0.01
A very small distance (δ)	0.000001
Task info update interval (ΔTS_u)	5s
Task info signal emission interval (ΔTS_e)	2.5s
Robot's task time-out interval (ΔRT_{to})	10s

V. EXPERIMENT DESIGN

In this section, we have described the design of parameters and observables of our experiments. These experiments are designed to validate AFM by testing the occurrence of convergent MRTA. Table I lists a set of essential parameters of our experiments.

A. Parameters

We intend to have a setup that is relatively complex, i.e., a high number of robots and tasks in a large area, but with a high probability of convergence. The following criteria shows our rationale behind our selected parameters.

- Robots should be capable of moving at a reasonably high speed (e.g., $\geq 4cm/s$) without interfering to each other very much.
- Tasks density should be as least one task per square meter.
- Bluetooth communication should be as dedicated and stable as possible.
- No task should be left unattended completely for a long time (e.g., $\geq 300s$).

When many Bluetooth devices talk to a single Bluetooth adapter, communication delays become very frequent due to the fact that each device gets a guaranteed turn to communicate [13]. After some initial testing, we found a stable server configuration with 8 Bluetooth adapters, i.e., one Bluetooth adapter is used to communicate with two robots. This limits us to set the total number of robots to 16. Also we found that after about 35-40 minutes from the start of our experiments some of robots fail to get the access to their designated Bluetooth adapters. So we limit the length of our experiments to 40 minutes. We expect that this limitation

would be removed by distributing Bluetooth adapters among multiple server PCs.

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. We have found that this ratio reasonable in order to allow the robots to move at a speed of 7.5 mm/sec without much interference to each other. We randomly placed four 18.5 cm x 11.5 cm rectangular boxes as a shop task in our experiment arena. They were about one meter apart from each other.

The initial values of task urgencies can be set to any value as long as they are same for all tasks. We choose a limit of 0 and 1, where 0 means no urgency and 1 means maximum urgency. Same applies to sensitisation as well, where 0 means no sensitisation and 1 means maximum sensitisation. We choose a default sensitization value of 0.1 for all tasks. Our rationale behind selecting task urgency and sensitisation change rates can be found in [6].

B. Observables

We have defined a set of observables to benchmark our implementation. They are briefly explained here.

Changes in task-urgencies ($\Delta\Phi$): In our experiments, urgency of each task in each step has been logged. From the above design of task urgency, we can see that if a task is not served by any robot for 100 consecutive steps (500s), urgency of that task will reach from 0.5 to its maximum value 1.0. On the other hand, if a task is served by only one robot for 200 consecutive steps (1000s) urgency of that task will be 0. But in real experiment, it is more likely that more than one robot will serve a task. So urgency of a task will decrease $\Delta\phi_{DEC}$ times number of working robots on that task (based on AFM guidelines [3]). The overall changes in task urgencies will show the convergence behaviour of our system.

Changes in robot sensitizations (ΔK): According AFM, as robots will do tasks they will specialize on each task by increasing or decreasing sensitizations (learning and forgetting). From our above design, we can see that if a robot starts doing a task with an initial sensitization of 0.1 and it repeatedly does it for 30 consecutive steps, we will be able to say that it has learnt it completely.

Changes in robot motions (ΔU): As we might guess that initially the task urgencies will be relatively higher for all tasks so robots will need to do a lot of movements by switching from one tasks to another. But as the system convergences overall robot motions will be decreased. In order to observe this phenomenon we log the pose of robots in each time step.

D-Bus Signals emitted by Task server (S_f): In order to measure the communication load on our system and to benchmark Task server's D-Bus signalling performance we are also interested to log P2P TaskInfo D-Bus signals. Since the emission of signals happens asynchronously it is more likely that the overall communication load on the system will vary over time.

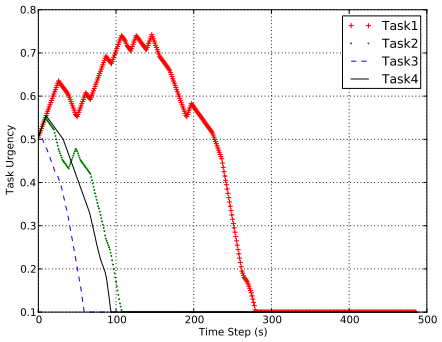

Fig. 3. Task urgencies observed at TaskServer in local mode r_{comm}=0.5m

VI. Results and Discussions

In this section we have presented our experimental results. We ran those experiments for about 40 minutes and averaged them from three iterations. For comparison purposes, here we present some of the results of our baseline experiments in centralized communication mode. Details can be found in [6].

Fig. 3 shows the dynamic changes in task urgencies. In order to describe our system's dynamic behaviour holistically we analyse the changes in task urgencies over time. Let $\phi_{j,q}$ be the urgency of a task j at q^{th} step. In $(q+1)^{th}$ step, we can find the change of urgency of task j :

$$\delta\phi_{j,q+1} = (\phi_{j,q+1} - \phi_{j,q}) \tag{3}$$

So we can calculate the sum of changes in urgencies of all tasks at $(q+1)^{th}$ step:

$$\Delta\Phi_{j,q+1} = \sum_{j=1}^{M} \delta\phi_{j,q+1} \tag{4}$$

Fig. 4 plots this sum of changes of task urgencies by a dashed line. If we consider the absolute change over a window w of time in the following equation we can describe the overall changes of our systems in both positive and negative directions.

$$\Delta\Phi_{jw,q+1} = \sum_{j=0}^{w-1} \left|\Delta\Phi_{q+j}\right| \tag{5}$$

In order to find convergence in DoL we have calculated the sum of absolute changes in task urgencies over a window of 2 consecutive steps (100s). This is plotted in solid line in Fig. 4. Note that we scale down the time steps of this plot by aggregating the values of 10 consecutive steps (50s) of Fig. 3 into a single step value. From Fig. 3 we can see that initially the sum of changes of task urgencies are towards negative direction. This implies that tasks are being served by a high number of robots. When the task urgencies stabilize near zero the fluctuations in urgencies become minimum. Since robots chose tasks stochastically, there will always be a small changes in task urgencies. A potential convergence point is shown in Fig. 4 by considering the persistence

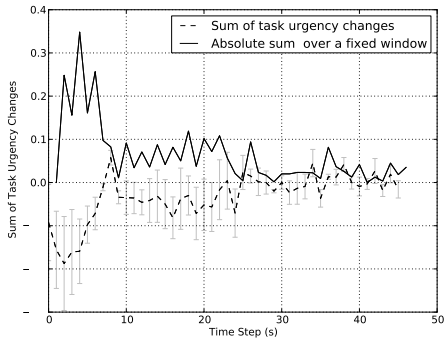

Fig. 4. Convergence of task urgencies in centralized comm. mode

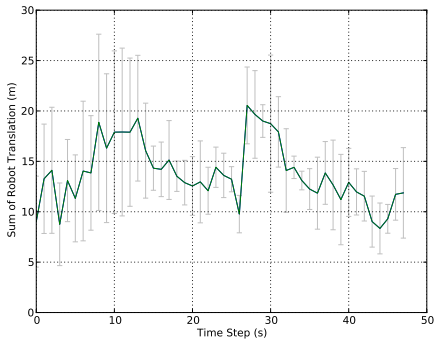

Fig. 5. Sum of translations of all robots in centralized comm. mode

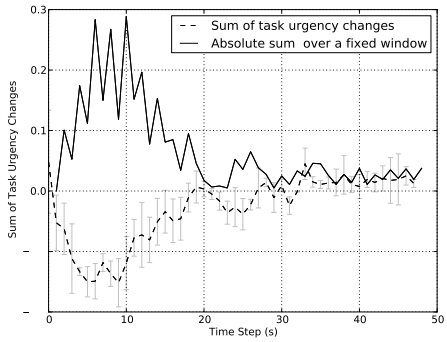

Fig. 6. Convergence of task urgencies in local mode r_{comm}=0.5m

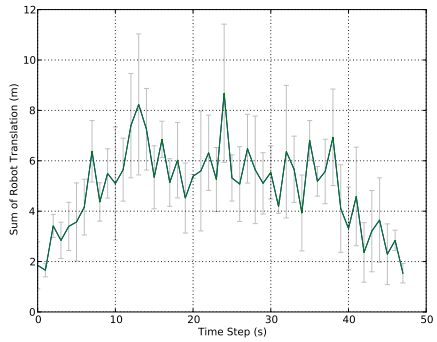

Fig. 7. Sum of translations of all robots in local mode r_{comm}=0.5m

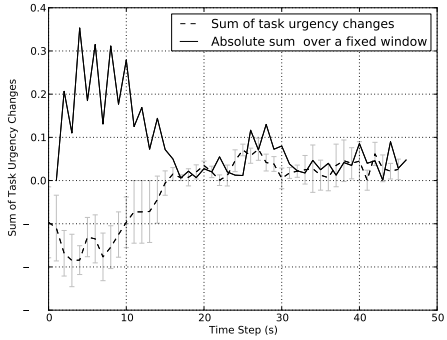

Fig. 8. Convergence of task urgencies in local mode r_{comm}=1m

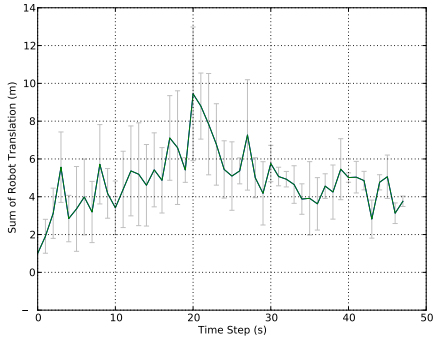

Fig. 9. Sum of translations of all robots in local mode r_{comm}=1m

existence of the value of $\Delta\Phi_{jw,q+1}$ below a threshold 0.1. This convergence happens near step 23 or after 1150s from the beginning of our experiments. This implies that from this point of time and onwards, changes of our system's behaviour remain under a small threshold value.

Using this same criteria, we find that in local communication experiment convergence happens after step 14 in case of r_{comm}=0.5m (Fig. 6) and after step 29 in case of r_{comm}=1m (Fig. 8).

We have aggregated the changes in translation motion of all robots over time. 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.

$$\Delta U_{q+1} = \sum_{i=1}^{N} \delta u_{i,q+1} \tag{6}$$

The translation result from centralized communication experiment is plotted in Fig. 5. In this plot we can see that robot translations also vary over varying task requirements of tasks. But it fails to show a consistence behaviour like previous plots. The translation result from local communication experiment is plotted in Fig. 7 and Fig. 9. The reduction of

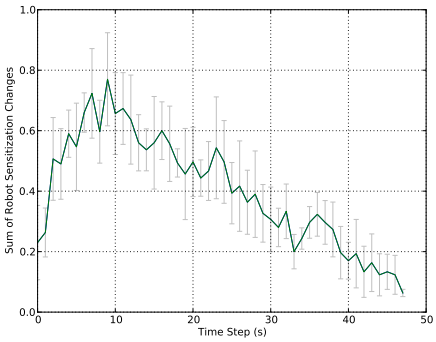

Fig. 10. Changes in sensitizations of all robots in local mode r_{comm}=0.5m

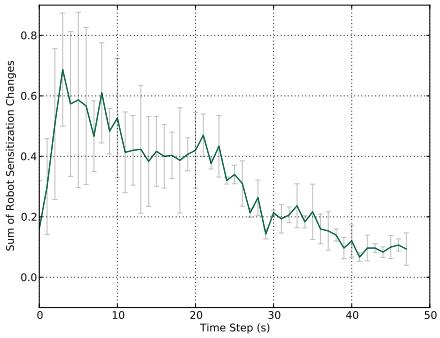

Fig. 11. Changes in sensitizations of all robots in local mode r_{comm}=1m

robot translation is significant in both local communication experiments.

Similar to Eq. 4, we can calculate the absolute sum of changes in sensitizations by all robots in the following equation.

$$\Delta K_{j,q+1} = \sum_{j=1}^{M} \left|\Delta k_{j,q+1}\right| \tag{7}$$

This values of ΔK from local experiments are plotted in Fig. 10 and Fig. 11. They show that the overall rate of learning and forgetting decrease over time. It is a consequence of the gradually increased task specialization of robots.

As an example of P2P signal reception of a robot, Fig. 12 and Fig. 13 show the number of received signals by Robot12 in two local experiments. It states the relative difference of peers over time in two local cases. The overall P2P taskinformation signals of both modes are plotted in Fig. 14 and Fig. 15. As an example of task specialization of a robot we plotted sensitization of Robot9 in Fig. 16. It shows that this robot has specialized in Task1. The continuous learning of Task1 happens from step 23 to step 59 where it has learned this task completely. This behaviour was found common in all robots with varying level of sensitizations. Hence we get the linear decrease of ΔK in sensitization plots. However, the changes in motion of this robot plotted in Fig. 17 is not stable due to the fact that robots frequently avoid dynamic obstacles and select random-walking.

VII. Related Works

VIII. Conclusions and Future Works

References

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

[2] Labella, T. H.; Dorigo, M. and Deneubourg, J. Division of labor in a group of robots inspired by ants' foraging behavior ACM Trans. Auton. Adapt. Syst., ACM, 2006, 1, 4-25

[3] Arcaute, E.; Christensen, K.; Sendova-Franks, A.; Dahl, T.; Espinosa, A. and Jensen, H. J. : Division of labour in ant colonies in terms of attractive fields. Ecological Complexity, Elsevier (2008)

[4] Lochmatter T., Roduit P., Cianci C., Correll N., Jacot J., and Martinoli A.: SwisTrack - A Flexible Open Source Tracking Software for Multi-Agent Systems. In Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS 2008), 4004-4010, IEEE (2008)

[5] Cianci, C.; Raemy, X.; Pugh, J. and Martinoli, A. Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics Swarm robotics: second SAB 2006 international workshop, Rome, Italy, September 30-October 1, 2006: revised selected papers, 2007, 103

[6] M. O. F. Sarker and T. S. Dahl, Robotic Validation of an Interdisciplinary Generic Model of Self-regulated Division of Labour in Social Systems, ANTS 2010, 7th International Conference on Swarm Intelligence (submitted, available upon request).

[7] Balch, T. and Arkin, R. Communication in reactive multiagent robotic systems Autonomous Robots, Springer, 1994, 1, 27-52

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

[9] Rutishauser, S.; Correll, N. and Martinoli, A. Collaborative coverage using a swarm of networked miniature robots Robotics and Autonomous Systems, Elsevier, 2009, 57, 517-525

[10] Krieger, M. and Billeter, J. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots Robotics and Autonomous Systems, Citeseer, 2000, 30, 65-84

[11] Agassounon, W. and Martinoli, A. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3, 2002, 1090-1097

[12] Mataric, M. Using communication to reduce locality in distributed multiagent learning Journal of experimental and theoretical artificial intelligence, 1998, 10, 357-369

[13] Huang, A.S. and Rudolph, L.: Bluetooth essentials for programmers. Cambridge University Press (2007).

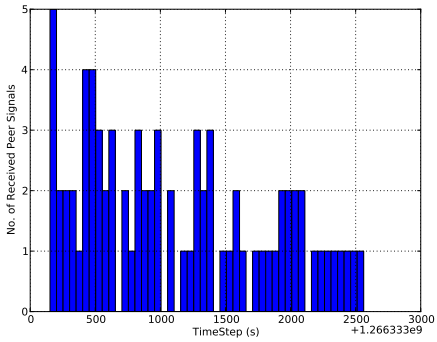

Fig. 12. Number of peer signals caught by Robot12 in local mode r_{comm}=0.5m

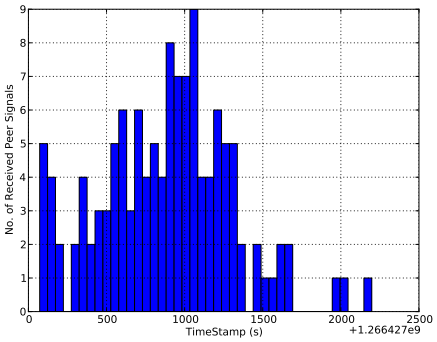

Fig. 13. Number of peer signals caught by Robot12 in local mode r_{comm}=1m

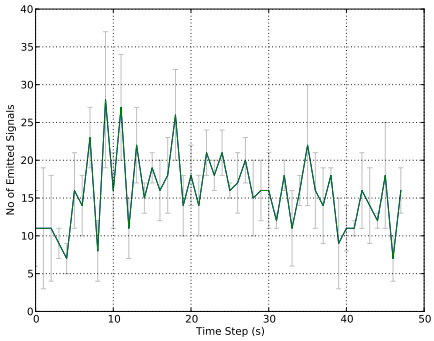

Fig. 14. Local peers' frequency of task information signalling in local mode r_{comm}=0.5m

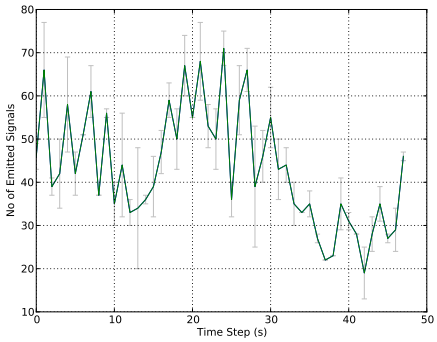

Fig. 15. Local peers' frequency of task information signalling in local mode r_{comm}=1m

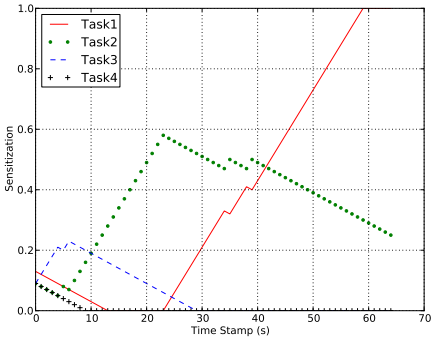

Fig. 16. Task specialization of Robot12 in local mode r_{comm}=0.5m

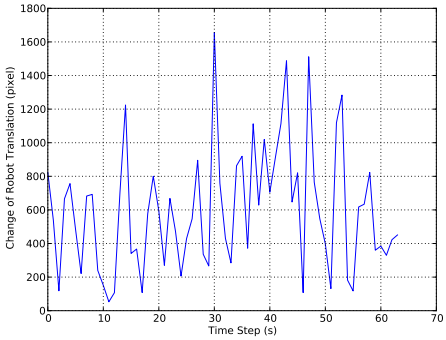

Fig. 17. Changes in translation of Robot12 in local mode r_{comm}=0.5m