On a Multi-agent Distributed Asynchronous Intelligence-Sharing and Learning Framework

Shashi Shekhar Jha(✉) and Shivashankar B. Nair

Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati, Guwahati 781039, India
{j.shashi,sbnair}@iitg.ernet.in

Abstract. The current digital era is flooded with devices having high processing and networking capabilities. Sharing of information, learning and adaptation in such highly distributed systems can greatly enhance their performance and utility. However, achieving the same in the presence of asynchronous entities is a complex affair. Multi-agent system paradigms possess intrinsic similarities with these distributed systems and thus provide a fitting platform to solve the problems within. Traditional approaches to efficient information sharing and learning among autonomous agents in distributed environments incur high communication overheads. Non-conventional tactics based on social insect colonies provide natural solutions for transfer of social information in highly distributed and dense populations. This paper portrays a framework to achieve distributed and asynchronous sharing of intelligence and consequent learning among the entities of a networked distributed system. This framework couples localized communication with the available multi-agent technologies to realize asynchronous intelligence-sharing and learning. The framework takes in a user-defined objective together with a learning algorithm as inputs and facilitates cooperative learning among the agents using the mechanisms embedded within. The proposed framework has been implemented using *Typhon* agent framework over a LAN. The results obtained from the experiments performed using both static and dynamic LANs, substantiate the applicability of the proposed framework in real distributed mobile computing environments.

Keywords: Multi-agent learning · Distributed intelligence · Mobile agents · Typhon · Emulation

1 Introduction

The drastic increase in the number of computational entities or devices in our surroundings has propelled research towards devising decentralized and distributed approaches for solving complex real-world problems. New paradigms, such as Cyber-Physical Systems (CPS) [39] and the Internet-of-Things (IoT) [2], wherein researchers attempt to evolve large-scale intelligent and autonomous applications, require a high degree of co-ordination among the numerous small

N.T. Nguyen (Ed.): Transactions on CCI XVIII, LNCS 9240, pp. 166–200, 2015.
DOI: 10.1007/978-3-662-48145-5_9

scale computing units that comprise these systems. The scale of the system and the computational and communication related complexities restrict the use of traditional centralized approaches to achieve such co-ordination. Though these approaches provide a single-point control of the overall system, they are less flexible in terms of reconfigurability and failures. Researchers have thus digressed to conceive non-traditional paradigms [28,41] to develop such distributed and autonomous large scale systems.

One of the approaches to solve problems that lie in distributed environments is the multi-agent system paradigm. Multi-agent systems [12] focus on the collective behaviours of agents and the complexities springing from their interactions. They form a fitting platform to realize truly distributed solutions in their most natural forms. Learning and sharing of information in multi-agent systems is a complex task both conceptually and technically [40] as it involves multiple learners wherein each agent tries to learn and adapt concurrently and in conjunction with the others. Research in the domain of multi-agent learning [1] has largely bifurcated into two streams - those that use reinforcement learning and the non-conventional evolutionary learning paradigms [36]. While in the former, the focus is to learn and evolve the value functions associated with each state and related actions, the latter tries to evolve behaviours. There are various aspects that need to be addressed while devising multi-agent based strategies. These include homogeneous versus heterogeneous agent teams, co-operation, restricted communication, credit assignments, etc.

Communication amongst agents plays an important role for the success of any multi-agent system. Efficient communication is the foundation for effective coordination, information distribution and learning from one another. Nonetheless, an unrestricted and torrential communication essentially reduces a multi-agent system to a single agent system [42]. Further, such communication facilities are not pragmatic in real-world systems. Hence, communication among agents in a multi-agent system needs to be selective and judiciously restricted while at the same time facilitating learning and co-operation [16]. Many multi-agent frameworks disregard this objective or rather neglect communication complexities to simplify the process of information sharing and learning [3,6]. The problem becomes more complex when, in a large distributed network, asynchronous agents need to discover other similar agents with which sharing can be performed.

Multi-agent based approaches which also use the mobility of agents termed as *Mobile* agents [20,37] can be considered as a compelling paradigm to realize truly distributed yet intelligent systems. Apart from mobility, these agents have various distinguishing features such as adaptability, autonomy, on-site computation and are distributed and pervasive. They have been proved to be more efficient than the traditional point-to-point communication models [35,38]. Although the mobile agent technology is still shaping up, recent advancements in small handheld devices, miniature computers with high processing capabilities and the advent of emerging fields such as CPS and IoT have propelled the research towards the development of mobile agent based systems [8]. Since these agents migrate throughout the network to sense and process data from and at different

nodes in a real distributed environment, the information they possess can vary with time. This form of information is akin to the level of experience that different persons possess while working in the same or different environments. Sharing of such experiences among the individuals could result in the growth of the overall knowledge of the group [13]. Though, researchers have used mobile agents based cooperation in a myriad of applications [22,32,35], this paper puts forward a concentrated effort on collaborative learning and information sharing among the entities of a distributed environment that utilizes the intrinsic characteristics of mobility and local execution capability of such agents.

In this paper, we present a framework for sharing intelligence and mutual learning among a set of location-unaware spatially segregated networked entities or nodes of a distributed environment; all of which have the same objective. The static agents are resident within each of the entities while the mobile ones facilitate the exchange of information and localized sharing over a network. The framework focuses on the *generic* mechanisms required to be embedded within individual agents so as to result in the overall convergence of the entire agent population, towards their common goal. The proposed framework is fully distributed in the sense that neither the agents (both static and mobile) possess any knowledge about the overall number of such agents present in the network nor do they possess any location information about other agents. Further the sharing of information among the agents is completely asynchronous and local. The major contributions of this work include:

- A multi-agent framework for distributed intelligence-sharing and learning.
- Modalities for local sharing and exchange of information.
- Dynamics for facilitating agent migration, inter-agent interactions and asynchronous executions.

The succeeding sections discuss our motivation and explore the idea of a multi-agent distributed intelligence-sharing and learning framework. A formal description of the proposed framework and the related dynamics have also been provided. The latter sections present the results along with the related discussions and conclusions arrived at.

2 Motivation

Social insect colonies provide the most natural examples of large scale multi-agent systems. These complex and self-organizing societies function based on very simple processes of information transfer between the individuals, thus providing an ideal perspective to understand the mechanisms of social learning [30]. Social insects invest considerable effort in passing on learned information to others in their group or swarm. The value of information obtained from others depends on the context [10]. Such social learning systems are often flexible enough to ensure that individuals rely on social information only when individual learning does not suffice. In an insect colony, information learned by an individual is not actually broadcasted to its peers or mediated through a supervised or

centralized control. On the contrary, the information flows through local interactions among individuals e.g. *Trophallaxis* in bees [29] and *Tandem running* in ants [14]. It has been demonstrated that bees learn associations between floral scent and nectar rewards during trophallatic interactions, just as they would if they were to sample the flowers themselves [10]. These local interactions among the individuals within an insect colony add up to eventually alter the behaviour of the entire colony and converge their searches to the location having maximum availability of nectar.

These complex yet versatile systems of insect colonies have been a source of inspiration for many a researcher in various fields of engineering and science [43]. In the context of learning in multi-agent systems, social insect-colony based models have been of notable interest [9,21]. In this paper, our focus is to exploit the use of social interactions among *nomadic* agents so as to facilitate asynchronous sharing and co-operative learning among the entities of a distributed environment. Exchange of information among these *mobile* agents, populating a network of nodes, takes place locally within a node, as and when they meet other such agents within. These local interactions tend to increase the quantum of knowledge they possess. The mobile agents use this extra knowledge gained through local interactions that are spread spatially across the network, for enhancing their self-centric learning thus reducing their individual as also overall search spaces. This learned information is thus provided to the static agents resident within the entities of the distributed environment, which evaluate the new information and provide valuable feedback for further enhancement.

For illustration, imagine a scenario wherein multiple mobile robots are trying to learn a single objective function such as solving a Rubik's cube, wrapping gifts, assembling a chair, etc. As can be observed, these tasks require a specific sequence of actions to be performed in a specified manner to achieve the desired objective. Since, there are multiple learning robots in this scenario, sharing of their individual experiences can enhance the performance of the whole system and also reduce the time the robots take to achieve the goal. However, sharing information in such a setting wherein the robots are dynamic entities is not trivial. Drawing inspiration from the social insects, one of the possible ways to share information could be by making a robot move to the vicinity of another and share information locally. The robots can gather such locally shared information over time and then use sophisticated learning tools to create a new plan of action. The robots can evaluate this new plan by executing them individually and consequently find the amount of progress they have made in achieving the goal. Repeating such a process would eventually lead all the robots towards the convergence of their shared objective. The point to be noted here is that the *mobility* of the robots contributes to the spreading of the information in the environment. However, mobility in robots is a costly affair in terms of both energy and actuation. The problem may become more critical if we think of a large network of robots wherein a few of them are only trying to learn and share a common objective.

In another scenario, assume a large Campus Area Network (CAN) populated by different kinds of devices as its nodes forming an IoT. These devices may include several PCs, projectors, air conditioners, sensors, Wi-Fi routers, printers, etc. In such a setting, imagine that a set of devices are required to find/learn to use a specific set of parameters (such as resolution (dpi), mode, paper-type, toner density, etc. in case of a set of printers) so as to optimize their life-time and utility. It is also possible that the devices need to adapt their settings based on their make and model. The simplest or naïve solution will be to package each device with an algorithm or program which always tries to figure out the optimized settings based on the user-feedback it receives on its own current settings. Assume that if the parameters are not good enough, the user changes these settings to suit her/his needs. This could be used as a feedback for the algorithm embedded within the devices. The problem with this approach is that each device (say printer) would try to solve the same problem repeatedly and hence there would be wastage of power, paper and cartridges. The life-time of the printer would also go down due to wear and tear during this laborious learning phase. A smarter way would be to share the locally learned information among the printers as in the case with mobile robots. This will not only reduce the wastage but also boost convergence time since multiple printers would be collaborating to achieve the same goal. However, it is not essential that these printers know the location of other such printers of their kind on the network. Further with no physical mobility and sophisticated programs on-board to handle communication complexities, learning optimal settings by a heterogeneous set of networked location-unaware printers, autonomously in the CAN, becomes a challenging task.

In the multi-agent based approach portrayed herein, we try to leverage the mobility based local sharing model of the mobile robots to alleviate the challenges discussed above. While a static agent manages local tasks at the physical learning entity i.e. a robot or a printer, its mobile counterparts (mobile agents) provide the much needed mobility of all the learned information. Imagine the network of robots supports a framework with all agent based functionalities as proposed in [18]. The authors in [18] describe the methodologies how a mobile agent based framework for a network of heterogeneous devices can be conceived. Let us further assume that the static agents reside within each of the networked robots. These agents manage local information and configuration of a robot such as preserving feedback, executing an action, etc. A set of mobile agents embedded with a learning algorithm suited for such an application could be released into the network of robots. These mobile agents then forage for the robots which are trying to learn within the network and facilitate the exchange of information locally with static agents hosted within each robot. Further, these agents can search the network to share their information with other such mobile agents asynchronously. When a mobile agent encounters other such agents at the various nodes in the network (not essentially the targeted robots) it exchanges information on the newly learned aspects of the solution to the problem. As a result of sharing each mobile agent comes up with a new set of actions/plan as per its

learning algorithm. This new learned information is then provided to the static agent hosted within the robots which in turn executes the new plan or solution and provides its feedback to the mobile agent. The latter continues its sojourn in the network of robots after receiving the feedback. The process of learning and sharing continues till eventually all such mobile agents agree on the same set of actions indicating convergence. It may be noted that the job of collaboratively learning the optimal set of actions is achieved by the set of robots using local communications of migrating mobile agents. Similar applications such as learning optimal printer settings, finding unique genome sequences among different databases distributed across a large network, etc., can be envisaged using such multi-agent based approaches.

Although the above mentioned approach may seem trivial, it puts forward many interesting challenges that are crucial in the realization of an asynchronous and distributed intelligence-sharing and learning framework. These include:

1. The parameters both Input and Output such as number of mobile agents, learning algorithm, etc. that are essential to regulate the functioning of the framework.
2. The dynamics associated with the different parameters involved within such as agent migration, on-node execution, etc.
3. The duration for which a mobile agent should search for other mobile agents so as to share or receive information.
4. The mechanisms for the movement and exchange of learned information.
5. The formulation of conditions that will subsequently trigger the execution of the learning algorithm using the newly collected information.

In the succeeding sections, we present the proposed multi-agent based approach to model the above mentioned challenges followed by the framework in detail.

3 Proposed Framework

This section discusses the parameters, the inputs and the mechanisms required to realize the proposed multi-agent framework for distributed and asynchronous sharing of intelligence along with the formalism to model all the processes and interactions used within.

3.1 System Model

The system under consideration is modelled based on the following:

- W is an undirected connected network, where $W = (N, E)$ wherein nodes are location-unaware.
- N is a set of nodes such that $N = \{n_i | i <= C_1\}$, where C_1 is the total number of nodes in the network W, $i, C_1 \in I$ where I is a set of positive integers.
- E is a set of links such that $E = \{e_i | i <= C_2\}$, where C_2 is the total number of links in the network W, $i, C_2 \in I$.

- A is a set of static agents resident on each of the nodes such that $A = \{a_i | \forall a_i \exists n_i, n_i \in N\}$, $i \in I$.
- M is a finite set of autonomous mobile agents such that $M = \{m_i | i <= K\}$, where K is the total number of mobile agents in the system and $i, K \in I$.
- P is an application dependent user-defined learning problem whose solution (G) is to be found.
- G is a goal (objective function) provided by user.
- L is a learning algorithm provided by the user based on the underlying application which is carried by each mobile agent as its payload.
- $\eta(.)$ is a function (set of actions) that a static agent can execute in any order with or without repetitions to achieve the goal G.

A mobile agent $m_i \in M$, which is by itself an autonomous program, is capable of migrating from a node n_i to another node n_j if there exists a link e_i between (n_i, n_j), where $n_i, n_j \in N$ and $e_i \in E$ within W. Each node n_i hosts an agent framework such as [31] that is capable of managing all agent related functionalities. Each node n_i also maintains a queue (Qn_i) of mobile agents present within node n_i. The queue, Qn_i, at a node n_i is defined as

$$Qn_i = \{q_j | q_j \in M, j <= \Gamma, \Gamma < K\}, Qn_i \subset M,$$

Γ is the length of queue, $j, \Gamma \in I$.

It is apparent that if $|Qn_i| = \Gamma$, no mobile agent can enter the node n_i. Each node is a uniprocessing entity thus all the operations within a node are executed sequentially. Figure 1 shows a typical network along with the mobile and static agents, the agent framework, the learning problem and the respective queues within.

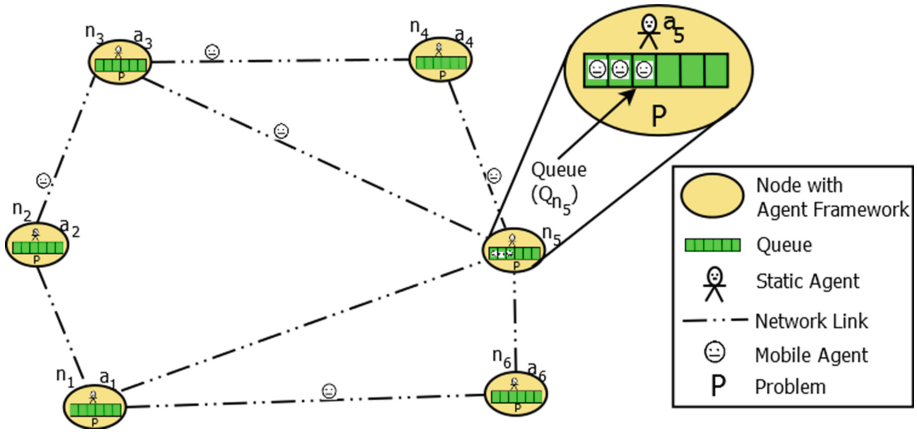


Fig. 1. A network of nodes having the agent framework, the static agent, the learning problem and the queue within. The mobile agents are shown to be either migrating from one node to another or resident within the queue of a node.

In the proposed framework, the mobile agents carry the learned information within themselves, share it with other such agents during their sojourn in the network. They also assimilate the newly gathered information to discover newer paths towards the goal G and execute the same at a node using the help of the static agents within. They collect the feedback and enrich their learned information and once again set out to collect more information from other such agents. Both, the mobile and static agents, co-exist in the network and share and execute to eventually ensure convergence.

3.2 Inter-agent Interactions

The interaction among the agents forms a crucial component of the proposed framework. As can be observed, there can be two kinds of possible agent interactions within the framework namely mobile-to-static agent interaction and mobile-to-mobile agent interaction. Since the network under consideration is distributed having location-unaware nodes, the static-to-static agent interaction is not possible under this framework.

Mobile-to-Static Agent Interaction. Mobile and static agents interact with each other in the following ways:

(a) En-queue: A mobile agent m_j can request a static agent a_i on the node n_i to execute an *en-queue* operation so as to enter Qn_i to effect its migration from another node n_k to n_i.
(b) De-queue: A mobile agent m_j can request the static agent a_i on the node n_i to execute a *de-queue* operation to enable its exit from Qn_i resulting in its migration to another node n_k from n_i.
(c) Execution of L: A mobile agent m_j at a node n_i, can execute the user-defined learning algorithm L using the computing environment provided by the static agent a_i resident at the node n_i.
(d) Execution of $\eta(.)$: A mobile agent m_j can request the static agent a_i on the node n_i to execute a set of actions and provide feedback. As $\eta(.)$ leads to the goal G (Sect. 3.1) which is dependent on the application under consideration, the executions mentioned here could imply either a set of movements for a navigating mobile robot, a set of rules for mining a large database, a set of input parameters for a control algorithm, etc.

Mobile-to-Mobile Agent Interaction. A mobile agent m_i *interacts-with* ($\otimes$) another mobile agent m_j resulting in sharing of intelligence if both the agents are present within the queue of a node n_k i.e.

$$m_i \otimes m_j \quad \textbf{iff} \quad m_i, m_j \in Qn_k, i \neq j$$

Hence, all the interactions among the mobile agents are always local and take place only *within the queue* of a node where the agents reside after migrating to a node.

3.3 Mobile Agent Migration Strategy

The mobile agents migrate within the network W using the ϵ-*Conscientious* migration strategy which is a combination of the *Random* and *Conscientious* migration strategies [33]. In the *Random* migration, a mobile agent chooses one of the neighbours of the current node at random and migrates to that node. In *Conscientious* migration strategy the mobile agent maintains a list of previously visited nodes (say V) and migrates to one that it has not visited so far. If it has visited all, it moves to the node which it has visited least recently. It can be noted that due to this migration strategy, the mobile agents tend to evenly distribute their frequency of visits at each node within the network. The *Conscientious* migration may intuitively seem better from the perspective of a single mobile agent. However, in a multiple mobile agent scenario, this strategy may lead the mobile agents to follow one another along a fixed path within the network.

In ϵ-*Conscientious* migration, a mobile agent employs the *Random* migration with a probability ϵ while it follows the *Conscientious* migration with probability $(1 - \epsilon)$. Thus, with an ϵ-*Conscientious* migration strategy, agents always try to reach out to non-visited or least visited nodes with a higher probability while reducing the drawback of a purely conscientious migration strategy.

3.4 Distributed Asynchronous Intelligence-Sharing and Learning

Let S^{m_i} be a set of shareable intelligence units (s_i) that a mobile agent $m_i \in M$ receives from the static agent $a_j \in A$ at a node $n_j \in W$. The set of shareable intelligence S^{m_i} is the learned information gathered as a result of the feedbacks obtained by m_i via the static agent a_j when it executes $\eta(.)$. Hence, the structure of individual elements $s_i \in S^{m_i}$ depends on the learning problem P of the application under consideration.

Each mobile agent m_i carries a *Bag*, B^{m_i}, (similar to the casebase of an agent as mentioned in [15]) which is a set of s_is obtained from the sets of shareable intelligence S^{m_i}, of other mobile agents as a result of sharing between m_i and the other mobile agent. Hence, B^{m_i}, which forms a part of the mobile agent's payload, can be defined as:

$$B^{m_i} = \{b_i | \exists m_j \text{ such that } b_i = s_k, s_k \in S^{m_j}, b_i \notin S^{m_i} \quad i, j, k \in I\}$$

Below we enumerate the definitions of sharing and learning within the scope of the proposed framework.

Definition 1. *A mobile agent m_i is said to have shared its intelligence with another mobile agent m_j if*

$$m_i \otimes m_j, s_i \in S^{m_i}, s_i \notin S^{m_j}, s_i \Rightarrow m_j, \quad m_i, m_j \in M, i \neq j$$

where '$\Rightarrow$' denotes that s_i is assigned to the mobile agent m_j resulting in $s_i \in B^{m_j}$.

The sharing of information amongst the mobile agents is completely asynchronous in nature. This essentially means that there is no global clock to synchronize the sharing events among the multiple mobile agents at various nodes within the framework. Thus sharing between the several mobile agents populating the network could take place concurrently at different nodes.

Definition 2. *As mentioned in Sect. 3.1, $\eta(.)$ is a function (set of actions) provided by a mobile agent m_i to a static agent a_j at a node n_j. The execution of $\eta(.)$ which is facilitated by a_j at n_j returns a new S^{m_i} which is passed on to m_i by a_j as the feedbacks. Thus $S^{m_i}_{new} = \eta(.)$. The mobile agent m_i is said to have learned new information if*

$$|S^{m_i}_{new}| > |S^{m_i}_{old}|$$

where, $S^{m_i}_{old}$ is the shareable intelligence possessed by the mobile agent m_i before the execution of $\eta(.)$ and $S^{m_i}_{new}$ is the same that the mobile agent m_i receives after this execution by the static agent a_j at node n_j.

Definition 3. *The problem P is said to have solved if a solution to goal G is found by all the mobile agents. This convergence is said to have achieved **iff***

$$\forall i, m_i \rightarrow G, m_i \in M$$

where, $\rightarrow$ denotes convergence. Hence, the main objective of the proposed framework is to ensure the convergence of all the mobile agents within the network to the common goal G using local sharing and consequent learning.

3.5 Inherent Mechanisms Within the Framework

The proposed framework provides a distributed model for sharing and learning amongst a set of location-unaware nodes in a network. In this framework we use asynchronous local sharing as a basis of information exchange by providing mobility to the learned information. Figure 2 depicts the cycle of learning that a mobile agent goes through within the proposed framework.

Each mobile agent (m_i) starts its operations initially from an *Executor* state. In this state, the mobile agent m_i residing in the queue, Q_{n_j}, of node n_j interacts with the static agent a_j also resident at the node n_j. Each mobile agent m_i is conferred with an *Execution Potential* (ξ_{m_i}) which is consumed gradually (discussed later) with every execution that the static agent a_j performs based on a request from m_i at the node n_j. The *Execution Potential* (ξ_{m_i}) restricts the mobile agent to reside at a node indefinitely. In the *Executor* state, the mobile agent m_i provides a sequence of actions derived from $\eta(.)$ to the static agent a_j as an attempt to achieve the goal G at the node n_j. This reduces the value of ξ_{m_i} based on the length of the sequence. The static agent a_j executes these sequence of actions at the node n_j. As a result of this execution, the mobile agent m_i receives feedbacks on the derived sequence of actions from the static agent a_j. This constitutes the part of the shareable-intelligence S^{m_i} gained by the mobile agent m_i at node n_j. These feedbacks also replenish ξ_{m_i} based on

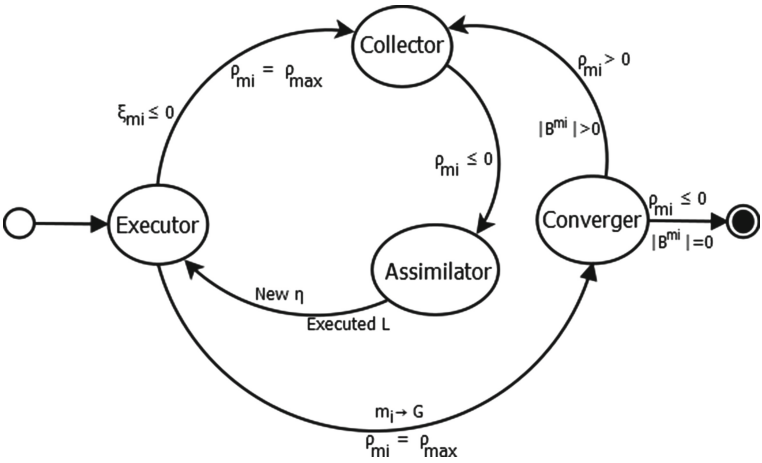


Fig. 2. The learning cycle of a mobile agent in the proposed framework

criteria discussed later. A mobile agent continues to remain in the *Executor* state until its ξ_{m_i} is exhausted. Once, $\xi_{m_i} \leq 0$, the mobile agent m_i assimilates the shareable-intelligence S^{m_i} it received and transits to the *Collector* state. In this state, the mobile agent traverses the network W, to share its shareable-intelligence with other mobile agents as well as to get shareable-intelligence from them as and when they co-exist within the same queue at a node. The mobile agent communicates locally with other mobile agents in the *Collector* state to share information. As a result of these local communications within a queue, the mobile agent m_i may receive a new set of information from the shareable-intelligence of other mobile agents. This new set of information is deposited into the *Bag*, B^{m_i} which is unique to the agent.

Every mobile agent is also empowered with a quantity termed as *Migration Resource* (ρ_{m_i}) [24] *a priori* which is also carried as its payload. The parameter ρ_{m_i} governs and regulates the duration for which the mobile agent migrates around in the network W, so as to meet and share information, in the *Collector* state. A mobile agent m_i enters the Collector state with $\rho_{m_i} = \rho_{max}$, ρ_{max} being the maximum possible value of ρ_{m_i} as defined by the user. The mobile agent m_i continues to traverse the network W, till its *Migration Resource* (ρ_{m_i}) is exhausted i.e. $\rho_{m_i} \leq 0$. This resource provides the impetus to the mobile agent to migrate within the network in search of other mobile agents having better information. The dynamics governing ρ_{m_i} have been discussed later. The structure of a mobile agent with all its components (payloads) can be seen in Fig. 3.

The mobile agent m_i transits to the *Assimilator* state from the *Collector* state as and when ρ_{m_i} degrades to a value less than or equal to zero. This is the state where the learning takes place. In this state, the mobile agent m_i uses the learning algorithm (L) which it carries as payload. It combines the information

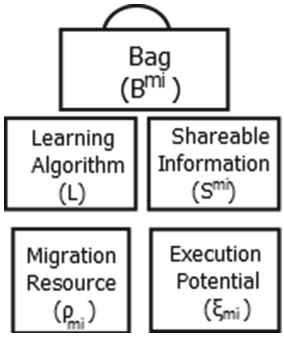


Fig. 3. Contents within the payload of a mobile agent in the proposed framework.

in S^{m_i} and B^{m_i} as the input to L and churns out a new execution plan (i.e. a new sequence of actions from $\eta(.)$) to reach the goal G at a node n_j. The generation of the new execution plan triggers the mobile agent m_i to transit to the *Executor* state.

The static agent a_j in turn executes this new plan within node n_j and provides the related feedback to the mobile agent m_i. This makes the mobile agent m_i transit back to the *Collector* state (unless of course the goal G is achieved) and the cycle in Fig. 2 continues. The value of ρ_{m_i} is again changed to ρ_{max} before it enters this cycle.

If the static agent reports that the goal G has been achieved, the mobile agent then transits to the *Converger* state with $\rho_{m_i} = \rho_{max}$. Mobile agents in this state tend to verify whether the converged goal G is an optimum or not. A mobile agent m_i in the *Converger* state migrates in the network W to find other mobile agents having a better solution than the one it has found. If the mobile agent m_i finds another mobile agent m_j having a better solution (as per the criteria defined by the user) then m_i receives the shareable-intelligence from m_j into its *Bag* B^{m_i}. This makes the mobile agent m_i to transit back to the *Collector* state and once again join the cycle. However, if ρ_{m_i} becomes less than or equal to 0 for the mobile agent m_i in the *Converger* state and B^{m_i} is still empty, it triggers the mobile agent m_i to exit the learning cycle. Hence a mobile agent m_i exits the learning cycle **iff**

$$(m_i \rightarrow G) \wedge (\rho_{m_i} \leq 0) \wedge (|B^{m_i}| = 0)$$

The overall process halts when all the mobile agents exit the learning cycle through the *Converger* state. The Algorithm 1 depicts all the functions of a mobile agent in the proposed framework as described above.

3.6 Dynamics Within the Framework

As mentioned, both ρ_{m_i} and ξ_{m_i} act as fuel for migration and on-node execution on part of the mobile agents, respectively. The dynamics that regulates the values ρ_{m_i} and ξ_{m_i} are discussed below.

Algorithm 1. Algorithm embedded within Mobile Agents

```
Input          : η, L, G, P, ρ_max
Output         : Convergence path to G
Initialization: B^{m_i} = {}, S^{m_i} = {}
while Not converged to G do
    enter_node(n_j);                                              // Executor State //
    while ξ_{m_i} > 0 do
        Exec_Plan = action_sequence(η);
        initialize(ξ_{m_i}, Exec_Plan);
        to_StaticAgent(a_j, Exec_Plan) ;        // Mobile to Static Agent Interaction
        receive_feedback();
        retrieve_intelligence(S^{m_i});
        update_potential(ξ_{m_i});
    end
    if G is not reached then
        ρ_{m_i} = ρ_max ;                                         // Collector State //
        while ρ_{m_i} > 0 do
            migrate_next_node();
            en_queue(Q_{n_i});
            ρ_{m_i} = migration_penalty(ρ_{m_i}, B^{m_i});
            if information request received then
                share_intelligence(S^{m_i});        // Mobile to Mobile Agent Interaction
            end
            if end_of_queue then
                find_shareable_agent();
                if agent is available then
                    W_old = Π_{m_i}(B^{m_i});
                    b_i = get_shareable_intelligence();       // Mobile to Mobile Agent
                                                              Interaction
                    B^{m_i} = B^{m_i} ∪ b_i;
                    W_new = Π_{m_i}(B^{m_i});
                    ρ_{m_i} = migration_reward(ρ_{m_i}, W_old, W_new);
                end
            end
        end
    end
    else if G is reached then
        ρ_{m_i} = ρ_max;                                          // Converger State //
        while ρ_{m_i} > 0 do
            migrate_next_node();
            en_queue(Q_{n_i});
            ρ_{m_i} = migration_penalty(ρ_{m_i}, B^{m_i});
            validate_convergence(G);
            if G is non-optimal then
                Jump to Collector State ;
            end
        end
    end
    if G ∧ (ρ_{m_i} ≤ 0) ∧ (|B^{m_i}| = 0) then
        exit() ;                                                  // Convergence
    end
    // Assimilator State //
    run_learning_algorithm(L, B^{m_i}, S^{m_i});
    get_new_sequence(η);
end
```

Dynamics of ρ_{m_i}. We assume that each piece-wise intelligence $b_i \in B^{m_i}$ gained in the learning exercise by the mobile agent m_i in the *Collector* state, to achieve the goal G, has a value or weight associated to it. This weight is synonymous to the profit gained or loss incurred, as the case may be, in using this piece of intelligence to advance towards the goal G. Let $\phi(.)$ be the function which returns this weight for each piece-wise intelligence $b_i \in B^{m_i}$. Hence, the net weight (Π_{m_i}) of the *Bag*, B^{m_i}, is calculated as:

$$\Pi_{m_i} = \sum_i \phi(b_i), \forall b_i \in B^{m_i} \tag{1}$$

The weight function $\phi(.)$ depends on the underlying application and can be designed based on a knowledge-sharing model as proposed in [13].

A mobile agent always enters the *Collector* state with $\rho_{m_i} = \rho_{max}$, where ρ_{max} is the maximum possible value of ρ_{m_i} conferred on it *a priori*. A migration penalty is incurred on ρ_{m_i} whenever a mobile agent m_i in the *Collector* state moves to a new node.

The value of ρ_{m_i} at the new node is computed as:

$$\rho_{m_i}(x_{n+1}) = \begin{cases} \rho_{m_i}(x_n)e^{-\Pi_{m_i}} & \text{, if } \Pi_{m_i} > 0 \\ \rho_{m_i}(x_n)(1 - \frac{1}{\rho_{max}}) & \text{, otherwise} \end{cases} \tag{2}$$

where, x_n denotes the n^{th} instance.

As can be observed in the above equation, the value of ρ_{m_i} degrades exponentially with increase in the weight of the *Bag* B^{m_i}. As more nuggets of information populate the *Bag* B^{m_i} of a mobile agent m_i due to local communication with other mobile agents, the payload of the mobile agent m_i increases and makes its movement across the network W sluggish. The migration resource ρ_{m_i}, however, decreases accordingly and inhibits the mobile agent m_i from further migration when $\rho_{m_i} \leq 0$. This triggers the mobile agent to enter *Assimilator* state wherein it generates a new plan using the algorithm L. On the contrary, if the *Bag* B^{m_i} is empty, the mobile agent's payload is lighter and ρ_{m_i} is high, forcing it to explore for fresh information across the network W by migrating to other nodes in search of other mobile agents that can provide the same.

While the above equation tends to reduce ρ_{m_i} of an agent due to migrations, it is also *recharged* whenever a mobile agent in the *Collector* state receives information from another mobile agent as a result of local sharing. This also means that whenever there is an increase in weight Π_{m_i} of the *Bag* within a mobile agent m_i, the value of ρ_{m_i} increases empowering it to travel further into the network W in spite of its sluggishness caused by the heavy *Bag* B^{m_i}. The value of ρ_{m_i} when a mobile agent m_i receives and accumulates new information into its *Bag* B^{m_i} is calculated as:

$$\rho_{m_i}(x_{n+1}) = \rho_{m_i}(x_n) + c\frac{\delta}{\Pi_{m_i}}\rho_{max} \tag{3}$$

where,

$$\delta = \Pi_{m_i}^{after\_sharing} - \Pi_{m_i}^{before\_sharing} \tag{4}$$

c is a constant and $c > 0$.

Dynamics of ξ_{m_i}. The *Execution Potential* (ξ_{m_i}) of a mobile agent decreases linearly with the execution of every action within the action-sequence when the mobile agent m_i is in the *Executor* state. As mentioned earlier, the action-sequence is derived from $\eta(.)$ which is performed by the static agent.

$$\xi_{m_i}(x_{n+1}) = \xi_{m_i}(x_n) - k_1,\ k_1 > 0 \tag{5}$$

However, when the mobile agent receives a positive feedback (defined by the user for the specific problem whose solutions are being learnt using the algorithm L) from the static agent as a result of executing an action, the *Execution Potential* (ξ_{m_i}) is topped up as:

$$\xi_{m_i}(x_{n+1}) = \xi_{m_i}(x_n) + k_2,\ k_2 \geq 0 \tag{6}$$

k_1 and k_2 are constants.

The increase in ξ_{m_i} allows for further exploration into the search space of the problem that has not been achieved as a result of sharing with other mobile agents. It thus aids the generation of new piecewise information and subsequent enhancement of the set S^{m_i}.

4 Implementation

Since the work reported herein exploits the simultaneous or concurrent executions of multiple agents running and sharing in parallel at different locations (nodes) within a network, experiments if conducted on an inherently sequential simulator would grossly undermine the strength and ability of the proposed framework. Hence, we have implemented the whole framework using the *Typhon* [31] platform over a Local Area Network (LAN). *Typhon* provides for mobile agent programming on real networks and runs over the proprietary *Chimera* Static Agent Platform that is shipped along with LPA WIN-PROLOG (http://www.lpa.co.uk/). Hence, along with static agents, *Typhon* provides all the mobile agent related functionalities such as migration, execution, cloning, payload carrying capability, etc. One instantiation of a *Typhon* based platform over *Chimera* acts as a node in the network. Several such instantiations were used to realize various overlay networks of different sizes varying from 10 nodes to 50 nodes over the LAN. Further, to evaluate the robustness of the proposed framework, experiments were conducted using a dynamic network of 50 nodes emulating a mobile computing environment. As the experimental test-bed was completely implemented over a LAN, the results gathered also involve the real-time states (such as processing speed, network conditions, etc.) of different machines used in the experiments. One separate dedicated computer served as a log server to record events such as sharing, executions, etc., at the various nodes.

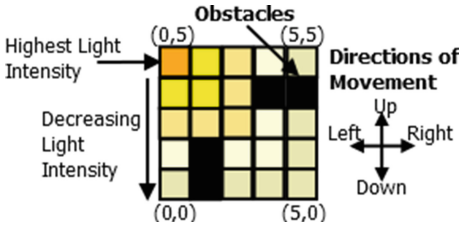


Fig. 4. The schematic of a 5×5 *Maze-World*. The top left corner in the *Maze-World* is the location of highest light intensity and formed the Destination location. The difference in colour variation of the cells show the change in light intensity as we move away from the destination. The black cells depict obstacles. The four directions of movement in the *Maze-World* are shown separately (Color figure online).

The events were time-stamped using the local time at the log server as and when the relevant pieces of information were received from the individual nodes in the network.

4.1 Terms Used in the Implementation

In this section, a glossary of various terms used in the implementation has been provided for better clarity. The meaning of terms used are as follows:

- Virtual robot: This is a simulated robot which has been tasked to learn a sequence of actions to reach a specified goal.
- *Maze-World*: This is an $n \times n$ grid structure with each cell of the grid having a set of sensor vectors.
- Sensor Vector (SV): This is a set of sensor values perceived by the Virtual Robot in a cell within the *Maze-World*.
- Static agents: These agents have the responsibility of executing operations on the virtual robots and to interact with the mobile agents.
- Mobile agents: They act as the carrier of learned information from one virtual robot to another.
- Node: The node comprises the *Typhon* Agent Platform, Queue for mobile agents, *Maze-World* and the virtual robot managed by a static agent.
- Network: This is the inter-connection of nodes which facilitates migration of mobile agents from one node to another.

4.2 The Distributed Learning Problem (P)

For evaluation of the proposed framework, a problem involving a set of virtual robots assigned with the task to learn a path from a fixed source (S_S - lowest light intensity) to a fixed destination (S_D - highest light intensity) in a *Maze-World* was used. The visualization of the *Maze-World* is shown in Fig. 4 and the corresponding networked-framework is depicted in Fig. 5. As can be seen,

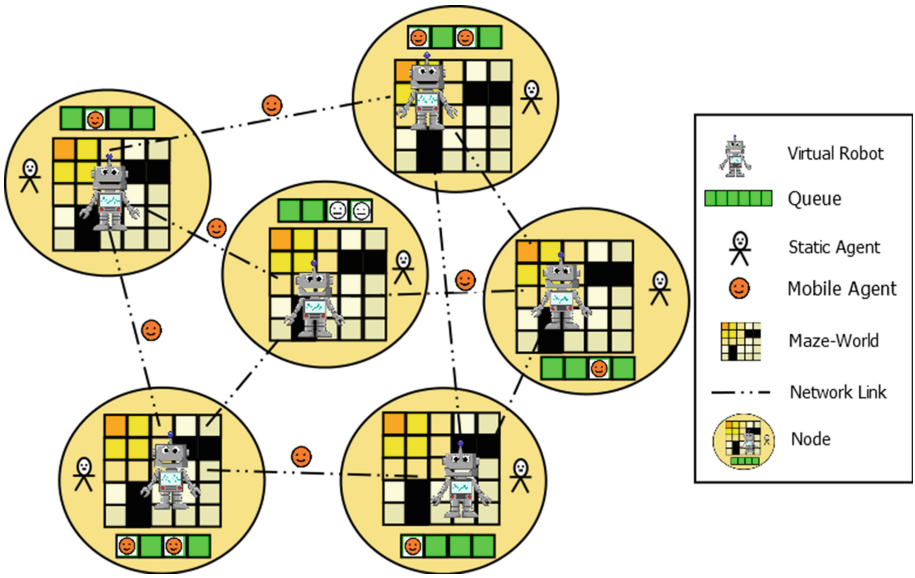


Fig. 5. An approximate visualization of the virtual robot network along with the *Maze-World*.

the *Maze-World* is available within each virtual robot locally. The set of virtual robots form the nodes of the distributed network. A static agent was also stationed at each node to manage all the functions of the virtual robot on the *Maze-World* available within the node. As can be observed, the *Maze-World* and virtual robot replace the problem component P and the node in Fig. 1 respectively. The network W is formed by the set of virtual robots situated within the nodes with the *Maze-World* and static agent. The mobile agents use this distributed network to disseminate the learned information received from the virtual robots via the static agents.

4.3 Specifications of the Virtual Robot

Each virtual robot is equipped with three virtual sensors viz. a *Light* sensor, an *Obstacle* sensor and a *Direction* sensor. The value of the *Light* sensor increases as the virtual robot moves towards the destination and is highest at the destination. Its value is inversely proportional to the distance between the destination and the current location of the virtual robot in the *Maze-World*. Obstacles may also populate the *Maze-World*. A virtual robot can sense these obstacles using its Obstacle sensor which returns only a binary value viz. *true* or *false*. The sensor returns a *true* if there is an obstacle in the direction of the virtual robot's heading or when it encounters the boundary of the *Maze-World*; else it returns a *false*. A virtual robot cannot cross any obstacle or the boundary of the *Maze-World*. Further, the value of the *Light* sensor becomes 0 if the value of *Obstacle* sensor is

true. The *Direction* sensor returns the direction of movement of the virtual robot in the *Maze-World*. There can be four possible values of the *Direction* sensor viz. *Up* (*positive Y-axis*), *Down* (*negative Y-axis*), *Right* (*positive X-axis*) and *Left* (*negative X-axis*). Hence, a virtual robot is not allowed to move diagonally from one location to another.

The virtual robot can perform five actions within the *Maze-World*. These actions are as follows:

- *Move_Forward:* Execution of this task makes the virtual robot to change its location in the direction of its heading one step at a time. For example, if the current location of the virtual robot is (2, 3) and its current heading is *Up*, then the new location of the virtual robot would be (2, 4) after the execution of the action *Move_forward*.
- *Move_backward:* Execution of this action changes the location of the virtual robot in the diametrically opposite direction of its heading without changing the current heading of the virtual robot. For example, if the current location of the virtual robot is (3, 3) and the current heading direction is *Left* then after executing the *Move_backward* action, the new location of the agent would become (4, 3) and the heading will remain *Left*.
- *Turn_Left:* This action changes the heading of the virtual robot 90° towards the clockwise direction of its current heading. The location of the virtual robot within the *Maze-World* is not affected by this action.
- *Turn_Right:* This action changes the heading of the virtual robot 90° towards the anticlockwise direction of its current heading. The location of the virtual robot within the *Maze-World* is not affected by this action.
- *Turn_Back:* This action changes the heading of the virtual robot to 180° of its current heading without changing the location of the virtual robot in the *Maze-World*.

4.4 Embedding the Problem *P* in the Framework

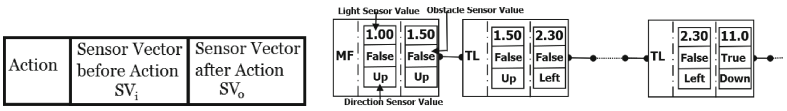
All the virtual robots have been embedded with the sensor vectors (SVs) of the source (S_S), and the destination (S_D) within the *Maze-World* using which they come to know about the source and the destination. The goal G for the virtual robot is to find a series of SV transitions using the available actions to reach the destination viz. S_D from the source S_S. The network of virtual robots uses the proposed framework to realize the mentioned objective.

$\phi(.)$ is defined as the difference of the norm between the SV before and after an action is executed i.e.

$$\phi(X) = ||SV_o|| - ||SV_i|| \tag{7}$$

where, X is an action, SV_i is the sensor vector before the execution of the action and SV_o is the sensor vector after the execution of the action.

The S^{m_i} in this case is the set of SVs with $\phi(X) > 0$. The structure of a piecewise unit of intelligence $s_i \in S^{m_i}$ together with an example s_is sequence are shown in Fig. 6(a) and (b).



(a) (b)

Fig. 6. (a) Structure of a piecewise intelligence $s_i \in S^{m_i}$ that a mobile agent m_i shares with other mobile agents (b) An example of s_is forming a sequence

4.5 The Learning Algorithm (L)

In our implementation, we have used a greedy approach to construct the sequence of actions towards the destination. When the mobile agents are in the *Assimilator* state of the learning cycle (Fig. 2), they perform an incremental search within their *Bag*s to find the associated transitions of SVs to reach the destination S_D. This search explores various combinations of SVs available within the *Bag*, B^{m_i} and S^{m_i}, of a mobile agent m_i and outputs the largest possible sequence of actions based on the SV transitions. Two piece-wise intelligence $b_i, b_j \in B^{m_i}$ such that $i \neq j$, can be connected to form a link if $SV_o \in b_i$ (i.e. $SV_o^{b_i}$) is equal to $SV_i \in b_j$ (i.e. $SV_i^{b_j}$) i.e.

$$SV_o^{b_i} = SV_i^{b_j}$$

Thus, SV transitions or b_is result in a tree with either the start vector S_S (the SV of the starting point in the *Maze-World*) or an SV closest to S_S (based on the *cosine* distance among the SVs) at the root of the tree. Other b_is within the *Bag*, B^{m_i}, form the node of the tree. The algorithm L uses *Depth-First-Search* (*DFS*) and outputs the branch having the highest length. In case, two branches have the same length, it outputs the branch with highest weight, Π_{m_i} (calculated using Eq. 1).

After assembling the sequence of actions towards the goal, G, the mobile agent transits to the *Executor* state. As mentioned earlier, in *Executor* state the static agent at a node executes the sequence of actions on the virtual robot thus evaluating the sequence of actions obtained using L and records the fresh SVs. If the execution of all the actions is over and the *Execution Potential*, ξ_{m_i}, is still greater than zero, then the static agent selects an action at random out of the set of actions ($\eta(.)$) and executes them. This allows for the self-discovery on the part of the agents which can be shared with others.

4.6 Complexity Analysis Within the *Maze-World*

Let us assume that the size of the *Maze-World* is $n \times n$. One may note that there could be more than one SV possible within a cell in the *Maze-World*. For example in the present *Maze-World* there are four different SVs, one for each direction within a cell. If S is the number of SVs possible within each cell of the *Maze-World*, the total possible SVs within the *Maze-World* would be Sn^2. Let

T be the number of actions that can be performed and l be the length of the sequence of actions to be executed to reach the destination (S_D). If we consider that the starting point for all virtual robots is fixed (S_S) within the *Maze-World*, then the total search space would reduce to $O(T^l)$.

If we relax our assumption that the length, l, of the sequence of actions is known, then for a single agent, the total time complexity to reach to the destination location (S_D) in the worst-case would be:

$$\beta := O\left(\sum_{l=1}^{Sn^2-1} T^l\right)$$

This gives us the upper bound on the time complexity of the search space using a single virtual robot positioned at a fixed location within the *Maze-World*.

Further, let us assume that α number of virtual robots at a fixed location within the *Maze-World* are trying to move towards the given destination. In the best case when no redundant executions are performed due to one-to-all sharing amongst virtual robots, the total time complexity of exploring the *Maze-World* by α virtual robots is of the order of β/α plus the overheads incurred in sharing.

Let the time taken for sharing intelligence between two virtual robots be τ. Apparently, the sharing of information involves two virtual robots (one who provides the information and the other one who receives it) at a time. Hence, the total number of sharing events required to disseminate the intelligence within each virtual robot among the remaining virtual robots would be ${}^{\alpha}C_2$. It may be noted that the virtual robots can share intelligence concurrently. If α is even, then the number of concurrent sharing events possible is $\alpha/2$. Thus the total time required to share the intelligence among the α virtual robots would be $\frac{(2\tau\ {}^{\alpha}C_2)}{\alpha}$.

In case if α is odd, this time complexity would become $\left(\frac{2\ {}^{\alpha-1}C_2}{\alpha-1} + \alpha - 1\right)\tau$.

Hence, the total time complexity to reach the destination in the best case for α virtual robot (α being even) would be:

$$\gamma := O\left(\frac{\beta + (2\tau\ {}^{\alpha}C_2)}{\alpha}\right)$$

This forms the lower bound on the time complexity of the overall search space. Let θ be the time complexity of solving the problem P using the proposed framework. Since, in our proposed framework $\alpha > 1$ along with localized sharing using the concept of mobility of learned information, intuitively θ will be bounded as: $\gamma < \theta << \beta$

It may be noted that these bounds are not so tight on θ, yet they give us an approximation of the reduction in the search space and time complexity when the proposed sharing framework is used.

5 Results

Experiments were carried out on various networks of virtual robots with different populations of mobile agents to learn a sequence of actions (i.e. the goal G in this case) to reach the destination. Each experiment was performed at least 10 times to counter any stochastic influence. The time required to complete each of the experiments varied between 300 to 3000 s. The average of 10 runs has been portrayed in the results. The values of various parameters used for the experiments are:
$\epsilon = 0.2$, $\rho_{max} = 10$, $n \times n$ (Size of maze) $= 50 \times 50$, Start Location (S_S) = (50, 0), Destination Location (S_D) = (0, 50)

The mobile agents were placed arbitrarily at different nodes within the network of virtual robots during the start of each of the experiments. All the nodes in the network of virtual robots were connected in the form of a mesh.

For experimentation, *Typhon* [31] based networks with sizes varying from 10 nodes to 50 nodes were created. The densities (D) of the mobile agents within a network, which can be defined as the ratio of number of agents to the number of nodes in the network, were varied from $D = 0.1$ to 0.5 on each of the networks and all the executions were logged. The experiments involved the problem of finding the sequence of actions required to traverse from the starting location S_S to the destination location S_D within the *Maze-World* with no obstacles. To vary the problem setting, we have also experimented on a 50-node network with the *Maze-World* having contiguous obstacles occupying co-ordinate locations (40, 0) to (40, 40).

Two important factors that are crucial to verify the effectiveness of the proposed framework are - the size of the network (i.e. number of nodes in the network) and the number of mobile agents involved. While the former tests the scalability of the framework, the latter can ensure a faster convergence. Hence, as a performance yardstick, we varied the density (D) of mobile agents in networks of different sizes and recorded the average number of executions that the virtual robots took to converge to the goal G. By number of executions of virtual robots, we mean the number of times each mobile agent entered the *Executor* state (Sect. 3.5). Thus, the average number of executions is the ratio of the total number of executions (until the convergence of all the mobile agents in the network) to the total number of mobile agents.

5.1 Converged Sequences of Actions

The graphs in Fig. 7(a) and (b) depict the final converged paths taken by the virtual robots with 5 mobile agents in the 50-node network. Figure 7(a) shows the converged paths of five virtual robots without sharing i.e. when the proposed sharing framework was not used. While, a total of 5 mobile agents ($D = 0.1$) in the 50-node network were used with the proposed framework for the graph in Fig. 7(b). As can be observed in Fig. 7(a), when the proposed sharing framework is not used by the virtual robots, all the virtual robots discover different sequences of actions (paths) to the destination S_D for the goal G. However, all

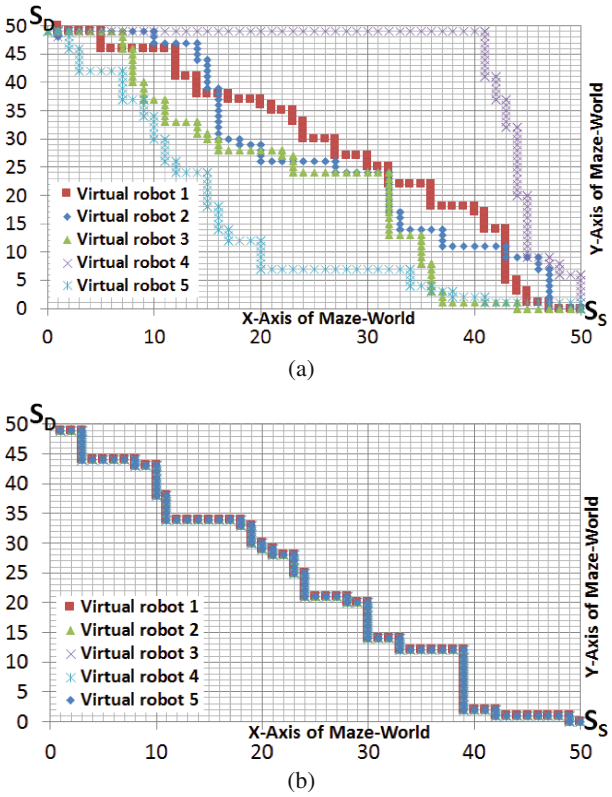


Fig. 7. Final converged sequences of five agents in the *Maze-World* from the Source at (50, 0) to the Destination at (0, 50) in a 50-node network - (a) Without Sharing (Paths are distinct) (b) With Sharing (Paths are highly overlapped).

the virtual robots converge to the same sequence of actions when they share information with one another. These graphs not only show the impact of sharing but also reveal the expected performance of the proposed framework. The goal G, assigned to the virtual robots, was to find a sequence of actions (as discussed in Sect. 3.5) that facilitate their movement from a source at (50,0) to a destination at (0,50). It can be seen clearly that sharing of information definitely helps these virtual robots achieve their goals in lesser time with fewer number of executions. From the logs, we have found that the average number of executions was 195 when the virtual robots did not share information whereas it was a mere 92 when sharing was embedded using the proposed framework. It should be noted that all the virtual robots are required to find their path individually. The results indicate that the proposed framework allows mobile agents to search in different directions (possibilities) and the best amongst them is taken up by all the virtual robots to beeline towards the goal. Similar trends were observed in all other cases considered for the experimentation.

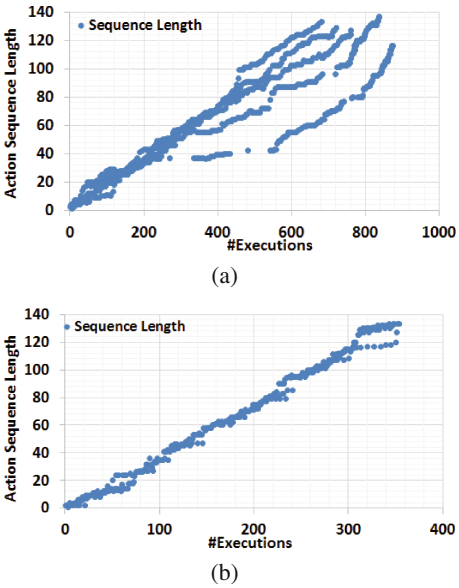


Fig. 8. Variations in the length of the sequences of actions of the virtual robots with executions in case of five mobile agents in a 50-node network - (a) Without Sharing (Mostly dissimilar sequences) (b) With Sharing (Converge on the similar sequences).

Further the converged sequence shown in Fig. 7(b) is achieved by five different mobile agents when they are completely oblivious of the information about other mobile agents within the network along with any knowledge pertaining to the network itself (i.e. the number of virtual robots). Hence, the proposed framework augments the algorithm L and facilitates the sharing of information amongst the virtual robots in a truly distributed sense and also yields better performance.

5.2 Length of the Sequences of Actions

The graphs in Fig. 8(a) and (b) depict the variations in the length of the sequence of actions with respect to the executions on part of the virtual robots with five mobile agents whose converged sequences are shown in the graphs in Fig. 7(a) and (b) respectively. The differences in the lengths of the sequences discovered by the five virtual robots against their executions can be observed in Fig. 8(a) when they are not sharing any intelligence amongst each-other. As can be observed in the case when the mobile agents shared intelligence (Fig. 8(b)), the lengths of the sequences of all the virtual robots are spread evenly and confined along a common line even before convergence when they are in the process of discovering the paths which is the period when execution increases. This illustrates that with no sharing of intelligence, the virtual robots search egocentrically and explore to find their individual solutions. Whereas the virtual robots are able to effectively constrict their search space towards the goal by sharing their piecewise knowledge

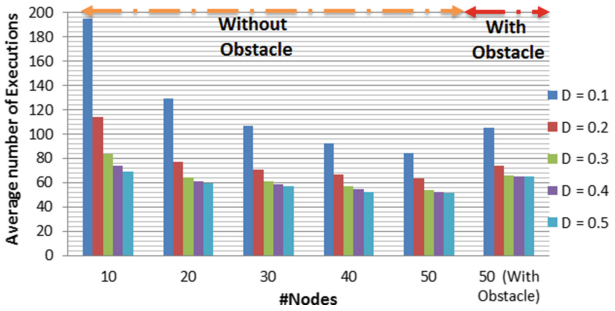


Fig. 9. Variations in the average number of executions to converge to a path from the source to the destination within a *Maze-World* of size 50×50 by all the mobile agents for different densities. The graph is plotted by taking the average of 10 runs in each case on *Typhon* based networks with no obstacles in the *Maze-World* for the first five sets. The last set is one with obstacles stretching from (40,0) to (40,40) in the *Maze-World* (Color figure online).

using the dynamics of the proposed framework. Each virtual robot thus tries to solve the sub-problems of a common agenda. It can also be observed that the total number of executions performed by all the five mobile agents taken together which is around 350 is less than half of the total number of executions taken together of all the non-sharing virtual robots (around 900). Hence, the results portrayed herein show that the proposed framework makes the virtual robots agree on a common solution and share the best available information amongst each other using the mobility of the learned information. The graphs in Fig. 8 show one instance of the results obtained. Similar trends were observed when the number of nodes and mobile agents were varied in all other cases considered for experimentation.

5.3 Average Number of Executions

Figure 9 depicts the average number of executions required to learn a path from source to the destination within the *Maze-World* of size 50×50 with varying number of nodes and mobile agents.

As can be observed from the graph in Fig. 9, the average number of executions (until all the mobile agents converge to a path to S_D) reduces monotonically with increase in the density of the mobile agents. Further, the trend remains the same as the size of the network (number of nodes in the network) grows. One can also observe that as the density becomes high (50 % of the total number of nodes), no significant difference in the average number of executions across different network sizes is observed. It may be noted that in the real-world, the execution of a series of actions is a costly affair both in terms of energy and time. Also, $D = 0.1$ in a 10-node network depicts a scenario when there is no sharing since there is only 1 mobile agent in the network. The average number of executions in this case is 195. While the average number of execution reduces to almost half (110 executions)

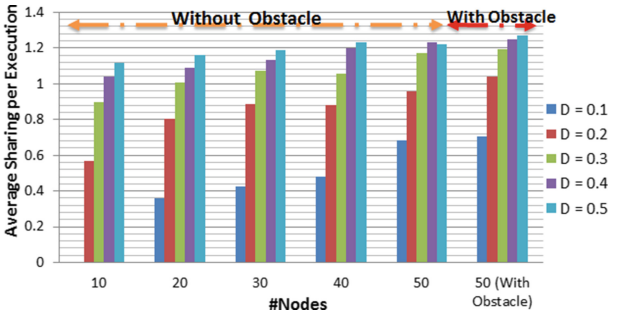


Fig. 10. Average number of sharing events per execution with different densities. The graph is plotted by taking the average of 10 runs in each case on *Typhon* based networks with no obstacles in the *Maze-World* for the first five sets. The last set is one with obstacles stretching from (40,0) to (40,40) in the *Maze-World* (Color figure online).

with $D = 0.2$ in case of 10 nodes. This depicts the huge gain in terms of average number of executions when the mobile agents are sharing information against the case when they are not doing so. Further, the average numbers of executions are higher in case of 50 nodes when there was obstacles within the *Maze-World* against the no-obstacle case. This shows that the complexity of the problem does affect the performance though the trend of executions remains the same. It may also be noted that the reduction in the average number of executions is high when the density of mobile agents varies from $D = 0.1$ to 0.2 in all the cases considered for experimentation. This reduction slows down from 0.2 onwards and almost becomes asymptotic. Moreover, increasing the density above 0.5 is not advisable as the population of mobile agents would clutter the network and entail more communication overheads [19]. Hence, the results clearly show that the proposed framework can effectively bring down the number of executions required to achieve the goal G in a fully distributed and asynchronous manner.

5.4 Average Number of Sharing Events per Execution

The graph in Fig. 10 depicts the average number of times the sharing was performed per execution with varying densities and sizes of the network. The average number of sharing events per execution is calculated as the ratio of the average of the total number of sharing events performed in 10 runs of each experiment to the average of the total number of executions in 10 runs. The cases considered are same as described in the previous section. As can be observed there is an increase in the average number of sharing events per execution with increase in the density of mobile agents in each of the networks of different sizes. Further, as observed in the previous case, the trend remains the same across different network sizes and problem setting. Apparently, the average number of sharing events per execution is 0 in case of $D = 0.1$ in the network with 10 nodes. The graph also reveals the fact that an initial increase in the density of mobile agents aids to expand the search in multiple directions which increases the amount of

sharing. However, a high density of mobile agents within the network causes redundancy in the search space and hence creates less amount of shareable intelligence amongst each other. Further, the graph also shows that a change in the problem setting (*Maze-World* with obstacles) does not alter the trend or the behaviour of the sharing among the mobile agents. It can also be observed that the amount of sharing increases when the problem becomes more complex (*Maze-World* with obstacles) because of the increase in size of the search space.

It may be noted that there is a marginal difference (< 0.05) in the average sharing per execution between $D = 0.4$ and $D = 0.5$ across all cases. These are possibly the best operating densities of the number of mobile agents in the proposed framework to effectively use distributed asynchronous sharing in the current problem setting of the *Maze-World*. Though a lower number of mobile agents could eventually find the solution (taking more number of executions and less sharing), an optimum number of mobile agents in the network could hasten the process while also effectively utilizing the network resources. A mechanism to dynamically vary the density of mobile agents based on the current size of the network as reported in [19] could aid the performance of the proposed framework.

5.5 Distinct Movements of Virtual Robots Within the *Maze-World*

The graphs in Fig. 11(a) and (b) depict the movement of all the virtual robots within the *Maze-World* of size 50×50 for $D = 0.1$ in a 50-node network without and with obstacles. The final converged sequences of actions (G) of all the virtual robots are also depicted in the graphs. These two cases of *Maze-World* exhibit different levels of complexities for the virtual robots to reach the destination location (50, 0) from the starting location (0, 50). When there is no obstacle within the *Maze-World*, the obvious way to traverse the *Maze-World* to reach the destination is to move diagonally while in the presence of an obstacle the best choice is to follow the path alongside the obstacle towards the destination. In case of an obstacle, the virtual robots need to exert more energy in terms of the number of executions required to explore the path alongside the obstacle as compared to a diagonal path in the absence of the same. Insertion of multiple obstacles along the path also gave similar results. As can be observed from the graphs, the final converged sequence of actions (G) in both without and with the obstacles, for all the virtual robots is the best possible intersection of the sequences available to reach the destination location at (50,0). Since the learning algorithm discussed in Sect. 4.5 does not ensure the optimality in the movement of virtual robots towards their destination, such an effect evolves as a result of the distributed asynchronous sharing of the piecewise intelligence available within each mobile agent which in turn circulates the knowledge of the global best among all the virtual robots. Sharing of information on part of the mobile agents seems to bring back the virtual robots that drift away from the optimal path resulting in a drastic reduction in the number of otherwise futile executions. It thus motivates the entire population of virtual robots to pursue a common and possibly more optimal path in an *orderly* and *unified* manner.

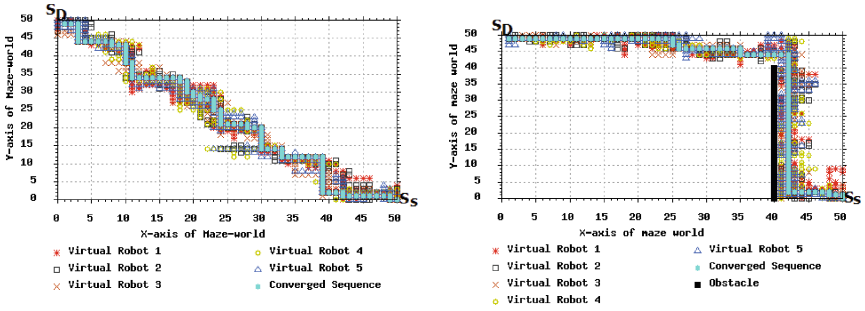


Fig. 11. Movements of all five virtual robots with sharing from the start of the experiment until convergence - (a) with $D = 0.1$ in a 50-node network with no obstacles in the *Maze-World* and (b) with $D = 0.1$ in a 50-node network with an obstacle stretching from location (40,0) to (40, 40) in the *Maze-World*.

5.6 Frequency of Sharing

Figure 12 shows the change in number of sharing events versus time along with the associated linear trendline for $D = 0.5$ in a 50-node network. It may be noted that the X-axis indicates the time-stamps recorded by the log server (as mentioned in Sect. 4). As can be seen, the linear trendline drawn based on the frequency of sharing events has a negative slope. Since the mobile agents try different combinations of actions on to the virtual robots initially they tend to discover diverse piecewise solutions causing the sharing events to be high. The number of sharing events decreases gradually as the virtual robots move towards the goal. This is because the initial high level of sharing causes the mobile agents to converge closer to a common path causing lesser diversity of information within their respective bags resulting in a gradual reduction in sharing. A similar

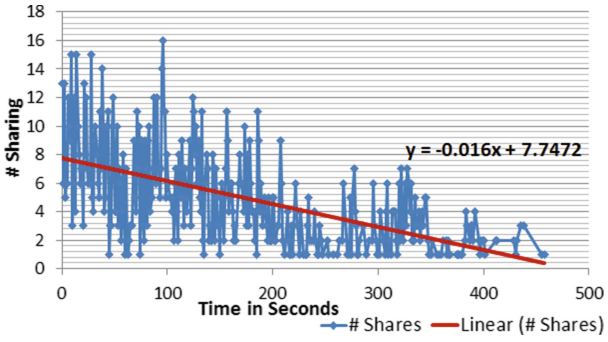


Fig. 12. Number of sharing events among 25 agents in a 50-node network ($D = 0.5$) against time in seconds.

trend was observed in all other cases of varying nodes and mobile agents used in the experimentation.

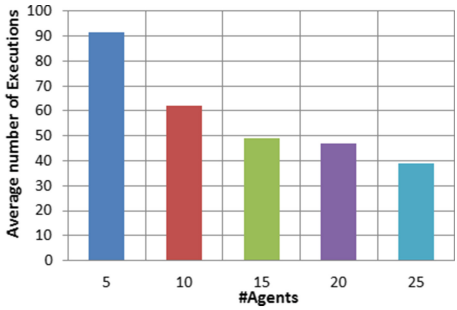
Sharing thus seems to constrain the mobile agents to search within a common and narrower search space. This could lead the virtual robots to follow a non-optimal path (local optima). One alternative to circumvent this could be to embed a different learning algorithm within a few mobile agents so as to force them explore the search space in a different manner ensuring diversity of the contents in the various *Bag*s for a longer time. This increase in diversity of the piecewise solutions carried within the individual *Bag*s of the mobile agents will consequently increase sharing and aid in an exit path from possible local optima.

5.7 Performance in a Dynamic Network

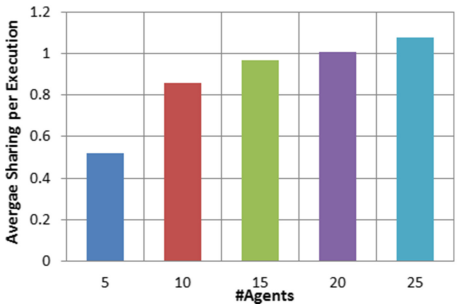
Figure 13 depicts the performance of the proposed distributed intelligence-sharing and learning framework in a 50-node dynamic network. The dynamic network was created using a variant of the Erdős-Rényi $G(n, p)$ model [11]. Initially a mesh network of 50 nodes was established over a LAN. Each node within the network was provisioned to break their connections with any of their current neighbours with a probability of 0.3. A node was allowed to make a new connection with any other node in the network with a probability of 0.5. Each node exercises the event of altering their connections after a time interval randomly chosen between 10 and 20 s in real time. Thus, the virtual robots within the dynamic network mimicked the movement or mobility of actual mobile robots just as in a mobile ad-hoc network by breaking and making new connections with other robots as they move physically. From the logs, the minimum degree of a node (d) was found to be 0 while the maximum degree was found to be 27. When a virtual robot becomes isolated ($d = 0$), then all the mobile agents within that node become *dormant* and wait till the virtual robot reconnects. Hence, the total number of nodes in the connected network keeps on varying with time.

As can be observed from Fig. 13(a), the average number of executions (until all the mobile agents converge to G) reduces with increase in the density of the mobile agents within the 50-node dynamic network. This follows the same trend as observed in case of the static networks discussed earlier. Further as shown in Fig. 13(b), the average number of sharing events per execution increases with increase in the number of mobile agents within the network. The trend of the graph in this case also remains similar to the previously cited cases of static networks. As can be seen from the graph, there is an initial boost in the average number of sharing events per execution in case of 5 mobile agents to 10 mobile agents while the same slows down as we move from 10 mobile agents to 25 mobile agents. As mentioned earlier, this is due to the fact that an initial increase in the density of agents aids to expand the search in multiple directions resulting in an increase in the overall sharing. As the density of mobile agents increases, redundancy in information discovered by the mobile agents increases thereby lowering the quantum of shareable intelligence.

It may thus be observed that the proposed framework is suited to networks wherein the *nodes are mobile*. This emphasizes the robustness and flexibility of



(a)



(b)

Fig. 13. Performance in the *Typhon* based 50-node dynamic network of virtual robots with varying densities from $D = 0.1$ (5 agents) to 0.5 (25 agents) (a) Average number of executions to converge to a path from the source to the destination within a *Maze-World* of size 50×50 by all the mobile agents (b) Average number of sharing events per execution.

the proposed framework for intelligence-sharing and learning in distributed environments. The results substantiate the viability of using the proposed framework in distributed mobile computing environments.

6 Discussions

The proposed framework opens numerous avenues for testing and verifying a gamut of learning algorithms over distributed, decentralized and asynchronous environments. Convergence is hastened due to faster access to newly weaned information gained from sharing on part of the mobile entities in the network. The multi-agent paradigm coupled with this mobility of learned information, facilitates collaborative learning among a set of location-unaware entities or nodes in a large distributed, decentralized and asynchronous system. The intrinsic flexibility of agent based technology used by the framework could also allow learning of multiple goals by the various entities in a network. Thus, in an N-node network, I nodes could be trying to find a solution to a problem P_1, J nodes

could be doing the same for another problem P_2, while the rest $(N - I - J)$ try to solve P_3. These problems may need to be solved concurrently or could even be made sequential using techniques cited in [24,25]. In such a case, the N-node network would be hosting heterogeneous sets of mobile agents which use the proposed framework to solve their respective problems. Heterogeneity in this case refers to the difference in the parameters and the learning algorithms used by the three sets of mobile agents attempting to solve the problems P_1, P_2 and P_3 respectively as mentioned above. Given a heterogeneous set of algorithms say L_1, L_2 and L_3 to be used to solve a particular problem P, such a set up of the framework could also be used to evolve the best algorithm. In this case, three sets of mobile agents could be primed to solve the problem P using an algorithm respectively within the proposed framework. Consequent to this, each set of mobile agents would evolve three solutions S_1, S_2 and S_3 based on the algorithms L_1, L_2 and L_3 respectively. In order to find the efficacy of the solutions S_i and find the best among them, an Idiotypic network [23] based mechanism such as the one proposed in [26] could be used. The mechanism proposed in [26] could allow the N-node network to automatically evolve the best solution out of S_1, S_2 and S_3.

In addition, if it so happens that an algorithm L_1 performs optimally during the initial phase of learning but unfortunately deteriorates in its performance at later stages while another algorithm L_2 behaves just the reverse way then running them together within the framework could be meaningful. While the set of mobile agents using L_1 will clearly dominate in performance initially, the other set of mobile agents using L_2 will use this learned information and enhance the convergence at a rapid pace. The heterogeneous set of mobile agents using different learning algorithms can thus co-operate and lead to a better solution using this framework. Since the performance of different algorithms could vary depending on the problem at hand such a hybridizing of algorithms could pave ways to newer and more efficient mechanisms.

The framework could also be enhanced by supplementing it with cloning of the best performing sets of mobile agents as described in [19]. Accordingly, the cloning of mobile agents would not only enhance concurrent processing by increasing the population of the best performing agents, but also restrict the lesser performing ones from consuming precious system resources such as bandwidth and computation times. Modifying and emulating variants of population based algorithms such as Evolutionary Computing and Genetic Algorithms (GA) including Island models [7] can also be performed using this framework.

As discussed earlier, the design methodology of the proposed framework is inspired by the interactions in social insect colonies and draws concepts from the domain of Evolutionary Computing and Genetic Algorithms (GA). Modifying and emulating variants of such population based algorithms can also be performed using this framework. If we consider each mobile agent carrying the learned information as a candidate solution, then this framework can be viewed as a real-time formulation of a distributed GA. The support for mobility of candidate solutions in this framework also allows the real implementation of Island

models of parallel GA [7]. Such models have been proposed as a distributed implementation but have hardly been used, possibly due to the non-existence of a convincing platform to realize them. If we look for the analogies between the proposed framework and GA, it may be observed that the Migration Resource, ρ_{mi}, mildly mimics the crossover operator used in a GA. It also makes candidate solutions to be nomadic and forces their movement from one island population to another. Further, the Execution Potential, ξ_{m_i}, partially mimics the mutation operator used in a GA by forcing the system to try out new solutions. Given a set of actions, it makes the system try out new permutations and combinations of these actions. The feedback provided by the static agent at the nodes participating in the learning exercise can be attributed to the fitness of the solutions in a GA. Similar analogies can also be derived for other population based bio-inspired algorithms such as Particle Swarm optimization (PSO) [27]. The proposed framework can thus be effectively utilized as a potential tool for intelligence-sharing and learning in large distributed systems in a variety of scenarios.

7 Conclusions

Distributed intelligence-sharing and learning in multi-agent systems, open up a wide variety of applications, that range from sensor beds to smart cities [5]. Mobile agent technology can play a crucial role in optimizing the use of local computing resources and distributed decision making in highly complex and scalable systems. This paper attempts to highlight the advantages of distributed intelligence-sharing and learning using a set of mobile agents. The *emulation* experiments presented in this paper provide valid proof-of-concept for the same. The model of intelligence-sharing and learning proposed herein could be used as a framework to realize distributed and continuously evolving systems which in turn could exhibit emergent behaviours. A variety of user-defined learning algorithms could also be used concurrently by different sets of agents within this framework. Further, since the number of mobile agents required is unknown, the initial discovery could commence with a moderate number of mobile agents, some of which may be allowed to clone over a period of time based on a performance measure so as to hasten the process of convergence. Excessive cloning can however lead to cluttering within the network which could be prevented using an appropriate controller such as the one described in [19].

It may be noted that the proposed framework makes use of information available locally (within a node) to achieve a global objective. This is akin to the functioning of densely populated insect colonies such as ants, honeybees, etc. [10]. The asynchronous sharing and consequent learning observed in the proposed framework could provide insights to the manner in which these swarms interact and converge towards a goal. It further opens up avenues to explore applications in the domain of asynchronous robotic swarms wherein every execution consumes precious energy stored within their batteries. Such a system could thus enhance the capabilities of the robotic swarm several manifolds. It is possible to

conjecture each robot in a swarm as a mobile agent and then embed the whole mechanism of local sharing and learning to constitute an intelligent mobile ad hoc network of robots. The framework described herein can also be used in many Internet of Things (IoT) and Cyber-Physical Systems (CPS) based applications. Some of the examples of such applications can be - energy management in smart buildings [44] wherein a set of mobile agents could be used to learn a schedule of energy consumption by various networked devices, intelligent water distribution systems [17] wherein such agents could share the information of a distributed network of various water reservoirs and optimize water distribution. Other applications could be the learning of the occupancy patterns of large buildings [34] for regulating HVAC systems and distributed intelligent traffic monitoring and management systems [4] wherein the agents can share the information of heavy traffic routes and learn traffic regulation rules to cater to different times of the day thus evolving optimized route schedules. In industry automation such agents can share the information of workloads on different remotely located machines and learn an optimized schedule for job allocations on-the-fly while in intelligent warehouse management systems these agents can facilitate sharing of information of the items placed in various smart racks and shelves to come up with the better and optimal strategies to improve logistics and placement of goods, etc. Use of a set of mobile agents, empowered with the sharing and learning mechanisms, in all these scenarios is bound to instil and enhance intelligence in such networked environments.

Acknowledgements. The first author would like to acknowledge Tata Consultancy Services (TCS) and Ministry of Human Resource Development, Govt. of India for the support rendered during the research reported in this paper.

References

1. Alonso, E.: Multi-agent learning. Auton. Agent. Multi-agent Syst. **15**(1), 3–4 (2007). http://dx.doi.org/10.1007/s10458-007-0019-1
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010). http://www.sciencedirect.com/science/article/pii/S1389128610001568
3. Berenji, H., Vengerov, D.: Advantages of cooperation between reinforcement learning agents in difficult stochastic problems. In: The Ninth IEEE International Conference on Fuzzy Systems, 2000, FUZZ IEEE 2000, vol. 2, pp. 871–876 (2000)
4. Bode, M., Jha, S.S., Nair, S.B.: A mobile agent based autonomous partial green corridor discovery and maintenance mechanism for emergency services amidst urban traffic. In: Proceedings of the First International Conference on IoT in Urban Space, URB-IOT 2014, pp. 13–18. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels (2014). http://dx.doi.org/10.4108/icst.urb-iot.2014.257297
5. Brenna, M., Falvo, M.C., Foiadelli, F., Martirano, L., Massaro, F., Poli, D., Vaccaro, A.: Challenges in energy systems for the smart-cities of the future. In: 2012 IEEE International on Energy Conference and Exhibition (ENERGYCON), pp. 755–762, September 2012

6. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **38**(2), 156–172 (2008)
7. Cantú-Paz, E.: A survey of parallel genetic algorithms. Calculateurs Paralleles, Reseaux et Systems Repartis **10**(2), 141–171 (1998)
8. Cao, J., Das, S.K.: Mobile Agents in Networking and Distributed Computing, vol. 3. Wiley, Hoboken (2012)
9. Cicirello, V., Smith, S.: Wasp-like agents for distributed factory coordination. Auton. Agent. Multi-agent Syst. **8**(3), 237–266 (2004). http://dx.doi.org/10.1023/B%3AAGNT.0000018807.12771.60
10. Dukas, R.: Insect social learning. In: Moore, M.D.B. (ed.) Encyclopedia of Animal Behavior, pp. 176–179. Academic Press, Oxford (2010). http://www.sciencedirect.com/science/article/pii/B9780080453378000589
11. Erdős, P., Rényi, A.: On the evolution of random graphs. Publ. Math. Inst. Hungar. Acad. Sci **5**, 17–61 (1960)
12. Ferber, J.: Multi-agent Systems: An Introduction to Distributed Artificial Intelligence, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
13. Fisch, D., Jnicke, M., Kalkowski, E., Sick, B.: Learning from others: exchange of classification rules in intelligent distributed systems. Artif. Intell. **187188**, 90–114 (2012). http://www.sciencedirect.com/science/article/pii/S0004370212000410
14. Franks, N.R., Richardson, T.: Teaching in tandem-running ants. Nature **439**(7073), 153–153 (2006)
15. Garland, A., Alterman, R.: Autonomous agents that learn to better coordinate. Auton. Agent. Multi-agent Syst. **8**(3), 267–301 (2004). http://dx.doi.org/10.1023/B%3AAGNT.0000018808.95119.9e
16. Ghavamzadeh, M., Mahadevan, S., Makar, R.: Hierarchical multi-agent reinforcement learning. Auton. Agent. Multi-agent Syst. **13**(2), 197–229 (2006). http://dx.doi.org/10.1007/s10458-006-7035-4
17. Giannetti, L., Maturana, F.P., Discenzo, F.M.: Agent-based control of a municipal water system. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 500–510. Springer, Heidelberg (2005). http://dx.doi.org/10.1007/11559221_50
18. Godfrey, W.W., Jha, S.S., Nair, S.B.: On a mobile agent framework for an internet of things. In: 2013 International Conference on Communication Systems and Network Technologies (CSNT), pp. 345–350, April 2013
19. Godfrey, W.W., Jha, S.S., Nair, S.B.: On stigmergically controlling a population of heterogeneous mobile agents using cloning resource. In: Nguyen, N.T. (ed.) TCCI XIV 2014. LNCS, vol. 8615, pp. 49–70. Springer, Heidelberg (2014). http://dx.doi.org/10.1007/978-3-662-44509-9_3
20. Harrison, C.G., Chess, D.M., Kershenbaum, A.: Mobile Agents: Are They a Good Idea?. IBM TJ Watson Research Center Yorktown Heights, New York (1995)
21. Holland, O.E.: Multiagent systems: lessons from social insects and collective robotics. In: The 1996 AAAI Spring Symposium on Adaptation, Coevolution and Learning in Multiagent Systems, pp. 57–62 (1996)
22. Ilarri, S., Mena, E., Illarramendi, A.: Using cooperative mobile agents to monitor distributed and dynamic environments. Inf. Sci. **178**(9), 2105–2127 (2008). http://www.sciencedirect.com/science/article/pii/S002002550700583X
23. Jerne, N.K.: Towards a network theory of the immune system. Annales d'immunologie **125**, 373–389 (1974)

24. Jha, S.S., Godfrey, W.W., Nair, S.B.: Stigmergy-based synchronization of a sequence of tasks in a network of asynchronous nodes. Cybern. Syst. **45**(5), 373–406 (2014). http://dx.doi.org/10.1080/01969722.2014.917235
25. Jha, S.S., Nair, S.B.: Orchestrating the sequential execution of tasks by a heterogeneous set of asynchronous mobile agents. In: Müller, J.P., Weyrich, M., Bazzan, A.L.C. (eds.) MATES 2014. LNCS, vol. 8732, pp. 103–120. Springer, Heidelberg (2014)
26. Jha, S.S., Shrivastava, K., Nair, S.B.: On emulating real-world distributed intelligence using mobile agent based localized idiotypic networks. In: Prasath, R., Kathirvalavakumar, T. (eds.) MIKE 2013. LNCS, vol. 8284, pp. 487–498. Springer, Heidelberg (2013)
27. Kennedy, J.: Particle swarm optimization. In: Gass, S.I., Fu, M.C. (eds.) Encyclopedia of Machine Learning, pp. 760–766. Springer, New York (2010)
28. Konstantinidis, A., Yang, K., Zhang, Q., Zeinalipour-Yazti, D.: A multi-objective evolutionary algorithm for the deployment and power assignment problem in wireless sensor networks. Comput. Netw. **54**(6), 960–976 (2010). http://www.sciencedirect.com/science/article/pii/S1389128609002679, new Network Paradigms
29. Korst, P., Velthuis, H.: The nature of trophallaxis in honeybees. Insectes Soc. **29**(2), 209–221 (1982). http://dx.doi.org/10.1007/BF02228753
30. Leadbeater, E., Chittka, L.: Social learning in insects from miniature brains to consensus building. Curr. Biol. **17**(16), R703–R713 (2007)
31. Matani, J., Nair, S.B.: *Typhon* - a mobile agents framework for real world emulation in prolog. In: Sombattheera, C., Agarwal, A., Udgata, S.K., Lavangnananda, K. (eds.) MIWAI 2011. LNCS, vol. 7080, pp. 261–273. Springer, Heidelberg (2011). http://dx.doi.org/10.1007/978-3-642-25725-4_23
32. Miller, K., Mansingh, G.: Towards a distributed mobile agent decision support system for optimal patient drug prescription. In: 2013 Third International Conference on Innovative Computing Technology (INTECH), pp. 233–238, August 2013
33. Minar, N., Kramer, K., Maes, P.: Cooperating mobile agents for dynamic network routing. In: Hayzelden, A., Bigham, J. (eds.) Software Agents for Future Communication Systems, pp. 287–304. Springer, Berlin Heidelberg (1999). http://dx.doi.org/10.1007/978-3-642-58418-3_12
34. Oldewurtel, F., Sturzenegger, D., Morari, M.: Importance of occupancy information for building climate control. Appl. Energy **101**, 521–532 (2013). http://www.sciencedirect.com/science/article/pii/S0306261912004564, sustainable Development of Energy, Water and Environment Systems
35. Outtagarts, A.: Mobile agent-based applications: a survey. Int. J. Comput. Sci. Netw. Secur. **9**(11), 331–339 (2009)
36. Panait, L., Luke, S.: Cooperative multi-agent learning: the state of the art. Auton. Agent. Multi-agent Syst. **11**(3), 387–434 (2005). http://dx.doi.org/10.1007/s10458-005-2631-2
37. Papaioannou, T., Edwards, J.: Building agile systems with mobile code. Auton. Agent. Multi-agent Syst. **4**(4), 293–310 (2001). http://dx.doi.org/10.1023/A%3A1012758908423
38. Queloz, P.A., Villazn, A.: Composition of services with mobile code. Auton. Agent. Multi-agent Syst. **4**(4), 311–337 (2001). http://dx.doi.org/10.1023/A%3A1012711025262

39. Rajkumar, R.R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: the next computing revolution. In: Proceedings of the 47th Design Automation Conference, DAC 2010, pp. 731–736. ACM, New York (2010). http://doi.acm.org/10.1145/1837274.1837461
40. Ren, W., Beard, R., Atkins, E.: A survey of consensus problems in multi-agent coordination. In: American Control Conference, 2005, Proceedings of the 2005, vol. 3, pp. 1859–1864, June 2005
41. Santos, A., Delbem, A., London, J.B.A., Bretas, N.: Node-depth encoding and multiobjective evolutionary algorithm applied to large-scale distribution system reconfiguration. IEEE Trans. Power Syst. **25**(3), 1254–1265 (2010)
42. Stone, P., Veloso, M.: Multiagent systems: a survey from a machine learning perspective. Auton. Robots **8**(3), 345–383 (2000). http://dx.doi.org/10.1023/A%3A1008942012299
43. Van Dyke Parunak, H.: "Go to the ant": engineering principles from natural multi-agent systems. Ann. Oper. Res. **75**, 69–101 (1997). http://dx.doi.org/10.1023/A%3A1018980001403
44. Zhao, P., Suryanarayanan, S., Simoes, M.: An energy management system for building structures using a multi-agent decision-making control methodology. IEEE Trans. Ind. Appl. **49**(1), 322–330 (2013)