On Stigmergically Controlling a Population of Heterogeneous Mobile Agents Using Cloning Resource

W. Wilfred Godfrey[1], Shashi Shekhar Jha[2], and Shivashankar B. Nair[2]

[1] Department of Information and Communication Technology
ABV-Indian Institute of Information Technology and Management Gwalior
Gwalior, M.P., India
godfrey@iiitm.ac.in
[2] Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati, Assam, India
Tel.: +91-361-2582353
j.shashi@iitg.ernet.in, sbnair@iitg.ernet.in

Abstract. Cloning can greatly enhance the performance of networked systems that make use of mobile agents to patrol or service the nodes within. Uncontrolled cloning can however lead to generation of a large number of such agents which may affect the network performance adversely. Several attempts to control a population of homogeneous agents and their clones have been made. This paper describes an on-demand population control mechanism for a heterogeneous set of mobile agents along with an underlying application for their deployment as service providers in a networked robotic system. The mobile agents stigmergically sense and estimate the network conditions from within a node and control their own cloning rates. These agents also use a novel concept called the Cloning Resource which controls their cloning behaviour. The results, obtained from both simulation and emulation presented herein, portray the effectiveness of deploying this mechanism in both static and dynamic networks.

Keywords: Mobile Agents, Cloning, Population Control, Cloning Resource, Typhon

1 Introduction

Mobile agents are programs that can act autonomously and also carry their execution state and data as they migrate. This makes them ideal candidates for transferring and also embedding intelligence in distributed systems. These agents have been used in network management, monitoring, routing and load balancing [1–6] and also in variety of other applications including information retrieval, robotics, etc. [7–10].

Mobile agents have been used for patrolling [11, 12] the nodes of a network. These agents attempt to visit the nodes at some regular intervals and provide the service they carry. With all agents carrying the same service (homogeneous), the kind of patrolling addressed by most researchers is merely a mechanism to ensure that any one of the agents visits each of the nodes at uniform intervals. The mechanisms cited by Chu *et al.*

[11] assume that a node once visited will never require nor regenerate a request for the service of an agent for a specific period of time till its turn to be visited again. Nodes get serviced by the agents in a round-robin like manner. Thus if a node which has just been visited, generates a request for a service once again, it will have to wait for its turn which will occur only when all other nodes have been serviced. The patrolling mechanisms suggested in [11] and by Sempe *et al.* [12] do not seem to take care of on-demand servicing. Active patrolling would mean satisfying requests generated by nodes within a network as quickly as possible, irrespective of whether they were serviced in the recent past. Patrolling in the real-world means moving around in a manner so as to provide attention to both, recently visited and not-recently visited locations within an area alike based on the urgency of the situation at that location. Godfrey and Nair [13] have proposed an architecture for a multi-robot networked system which uses mobile agents to provide services to robots in a round robin manner. The robotic nodes communicate with each other over a wireless communication link. Each robotic node provides a software framework for hosting and enabling mobile agents to arrive from or migrate to another such node. The mobile agents carry a service as their payload and provide them to a robotic node whenever they reach a node that has requested for the same. The term service is used to generically denote what the agent carries as payload. This could be for instance the source code for a robotic task, rules or information.

In order to support active patrolling, Godfrey and Nair [14] have described a mechanism termed *PherCon*, which combines both the Pheromone and Conscientious strategies to achieve agent migration towards the requesting nodes. All robotic nodes forming the network are capable of diffusing virtual pheromones [15] as and when a service is required. These pheromones tend to attract the relevant mobile agents within the network towards the *Robotic node Requesting a Service* (RRS). The RRS pro-actively diffuses pheromones to its neighbouring nodes which in their turn diffuse them at lower concentrations to their neighbours thus generating a pheromone concentration gradient network across the RRS [14]. The mobile agents each carrying different services (or code for the tasks) patrol the network in a conscientious [16] manner avoiding visits to recently visited nodes. When they hit upon a pheromone trail at a node, they switch to pheromone tracking and follow the shortest path along the pheromone concentration gradient to eventually reach and service the RRS. Such an active form of patrolling serves to avoid longer waiting times on part of those nodes which have been visited recently but have generated their request immediately after a visit by the relevant mobile agent. The model is well suited for applications that make use of heterogeneous mobile agents wherein every agent carries a different set of services as its payload. Figure 1 depicts the structure of the pheromone. The first field contains the RRS ID that initially generates the pheromone. The second field contains the requested service type. The concentration and life-time occupy the third and fourth fields while the last one points to the previous node. Godfrey and Nair [17] have also shown how localized cloning within the pheromone gradient can decrease RRS service times. Localized cloning however yields only marginal effects. In order to effectively patrol and provide on-demand services in a networked scenario using heterogeneous mobile agents, cloning needs to be performed by every agent based on its demand. Agents whose services (payloads) are more often requested should clone proportionately so as to provide a quicker service. With the

build-up of more clones within a network, the service times are bound to fall but not for long. A large number of clones can clutter the network increasing migration times and thus deteriorating the performance of the system.

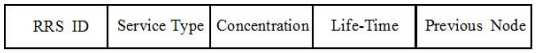


Fig. 1. Structure of the virtual pheromone [14]

In scenarios as in [14] where the nodes (RRSs) asynchronously generate requests for a service, the waiting times can be further decreased if the mobile agents populating the network are made to clone proportionate to their demand within the network. Cloning a mobile agent carrying a certain task as its payload, in large numbers based on its utility may theoretically decrease waiting times of those nodes that request this specific task. However in practice, this may not be a viable alternative since the network may have several nodes requesting different tasks. If all the pertinent agents were to clone in large numbers at the same time, they would quickly clutter and choke up the system. Cloning thus needs to be carried out judiciously based on the network conditions and also demand (number of nodes requesting a service or RRSs). Mobile agents need to also back-off and die in case of choke-ups. Decisions on whether to or not to clone or to back-off have to be made autonomously by the agent and not in consultation with others. Mobile agents in real-world application scenarios such as the robotic network described in [13] will need to make a decision on whether to or not to clone by sensing the active medium they migrate through, in a stigmergic manner [18]. In this paper we describe a method by which the mobile agents, populating and migrating within the robotic network as in [14, 17], unilaterally approximate the status of the network and judiciously decide the extent to which they need to clone. This paper extends our previous work [19] and describes an attempt to design an adaptive, on-demand cloning controller for controlling a population of a heterogeneous set of mobile agents in both static and dynamic networks.

2 Motivation and Related Work

Mobile agents share all characteristics of their static counterparts but stand apart in their capability to migrate and clone autonomously. The concept of cloning featured in mobile agents helps increase their population and indirectly allows for parallel operation and information transfer across a network. It can also aid in the upward scalability of a distributed system. Cloning can thus make a large impact in the performance of mobile agent based applications. Glitho *et al.* [20] discuss a mobile agent based approach that outperforms traditional client-server based architectures in a multi-party event scheduling problem. The authors argue that if the number of agents is increased it would reduce the response time and enhance performance. Such an increase in number of agents could be made inherent to the system by using the concept of cloning. In grid computing scenarios too, mobile agents could increase their numbers by cloning

to find the requested resource at a faster pace. Jin *et al.* [21] have proposed a routing model for grid computing using mobile agents. They have used two types of agents to collect and update the dynamic changes in the routing information coupled with a naive form of cloning to boost performance. Hamza *et al.* [22] describe the use of mobile agents in the domain of cloud computing for discovery of web services. Cloning, if used in their model, could expand the search area leading to faster discovery of services. Various other application domains such as agent based data-mining [23], distributed information management [2], Internet based e-commerce [6], monitoring and routing the networks [3] can greatly benefit if cloning of mobile agents is used.

Having multiple copies of an agent can not only enable parallel execution but can also provide fault-tolerance, thus increasing both robustness and efficiency of the system. The down side is that an increased population of mobile agents may result in higher network resource utilization which in turn may deteriorate the performance of the system. While on one side increasing the population of agents using cloning increases its performance, uncontrolled cloning can flip the same and cause the network to become overcrowded. To substantiate such a claim, we performed experiments using a 200-node connected network. The network capacity or the maximum number of agents that the network could accommodate was varied from 0 to 1200. As a performance measure, we recorded the number of Step-Counts required to service 100 RRSs. The agents were allowed to clone to maximize their performance.

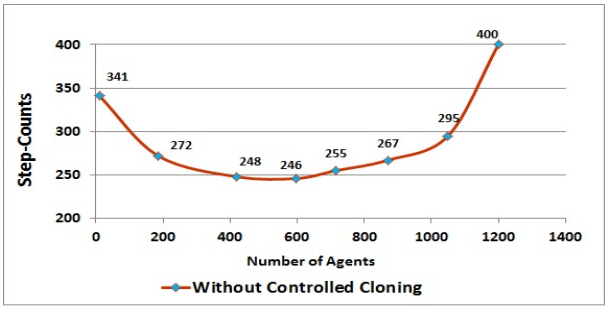


Fig. 2. Graph depicting the Step-Counts required to service 100 RRSs without any control on cloning

Figure 2 shows the graph depicting the nature by which the number of Step-Counts (time) required to service 100 RRSs decreases initially with increasing number of agents (including their clones) and then increases due to cluttering within the network. It can be observed that when the dots, that indicate the Step-Counts, are connected the resulting curve seems to be parabolic in nature. Ideally, the value of the number of agents in the network should coincide with that at the vertex of the parabola for the fastest possible service. Thus it can be inferred that in practice one should ensure that the number of agents (including clones) should always lie to the left of the axis of this parabola. The graph therefore endorses the imperative need for controlled cloning so as to ensure that the total number of agents including clones populating the network remains high

while also providing minimal service times. In scenarios where heterogeneous mobile agents populate the network, it is essential that the total number of such agents remain optimal so as to provide low RRSs service times and also that the agents clone proportionate to their demand. The mechanism should therefore facilitate the depletion of agents that are no more in demand thereby giving way to other in-demand agents while also maintaining the net population at an optimal value that does not cause cluttering or severe network contention. Hence, a controlled, adaptive and demand based cloning can ensure that the performance remains well near the achievable optimum. The work described in this paper augments the *PherCon* [14] migration strategy to achieve lower waiting times to service a node in a network populated by heterogeneous agents.

Suzuki *et al.* [24, 25] have addressed the control of a population of mobile agents in dynamic networks. Their mechanism is based on the popular ecological model which assumes that the population of a single species converges to a number in proportion to the amount of *food* available in its environment. Every node in their network generates *food* at regular intervals in proportion to the number of its links to other nodes. An agent migrates to a node, consumes the *food* within the node and clones in proportion to the excess *food* it cannot consume. An agent which does not get *food* starves and thus is eliminated. Two algorithms have been cited - one in which an agent is provided with the actual number of links in a vertex and the other wherein the agent makes an estimate of the link density. The latter algorithm proves to be a useful aid in calculating the *food* to be generated in case of dynamic networks where the nodes are mobile causing a change in topology and hence the links. Ma *et al.* [26] and Golebiewski *et al.* [27] describe algorithms that use a mobile agent population control protocol wherein each node keeps at most one copy of an agent. If there is a single agent in a node then a new agent is born with a certain probability p, computed based on the fraction of target nodes. In this case, the agents make use of a random migration policy. Another mechanism for population control of agents within the Internet described in [28] uses three types of entities - a node, an executor agent and a controller agent. Each of these agents has some energy which gets consumed as it performs its respective tasks. When the energy level falls below a certain threshold, the agent requests for more energy from its controller. If the latter does not respond immediately, the agent becomes an *orphan* and is thus removed from the network. This protocol is complex in terms of inter-entity communications and the control is not fully decentralized. There is a heavy dependency on the controller agent whose failure would mean the same for the entire system. A biologically inspired mechanism for a homogeneous mobile agent patrolling system has been suggested by Amin and Mikler [29] using pheromones. An agent that visits a node lays pheromone which is volatile in nature. Based on the amount of pheromone an inter-arrival time of an agent at a node is calculated by the agent that has just reached this node. This inter-arrival time is used to estimate the frequency of visits made by an agent to this node. If this time is higher than a certain threshold, the agent assumes that there are lesser number of agents in the network and thus clones to cope up with the situation at hand. On the contrary if it finds this value below another threshold the agent kills itself assuming it to be redundant. If the number of visits is in between these two thresholds the agent merely migrates to a neighbouring node. Bakhouya and Gaber [30] focus on the dynamic regulation of the mobile agent population in a distributed system inspired

by concepts from the biological immune system. They embed three basic behaviors onto the mobile agents viz. cloning, moving and killing and attribute them to three different antibodies, each suppressing or stimulating the other. The behaviour to be chosen depends on the inter-arrival time of the agents at a particular node, similar to what has been suggested by Amin and Mikler [29], which in turn controls the agent population.

All the mechanisms cited so far support population control of a homogeneous set of mobile agents and are inherently suited for a regular patrolling problem where the inter-arrival times between visits made by the same type of agent are to be kept a constant or a minimum. Further in all these mechanisms the maximum number of agents is a factor of the total number of nodes in the network which needs to be known *a priori*. These mechanisms are thus neither scalable for different types of agents nor adaptive in terms of their manner of controlling the agent population. Instead they merely try to ensure that the existence of a certain number of agents of the same type (homogeneous) will not clutter the network and thus avoid network contention. For instance, if each agent were to take an opinion (such as stimulations or suppressions as mentioned in [30]) from other agent peers, this additional communication would cause further overheads on the network. With a large number of such agents communicating and transacting stimulations and suppressions to one another, as suggested by Farmer *et al.* [31], progress of their movement towards the respective RRSs would be greatly retarded.

In the next section, we discuss the architecture of the proposed cloning controller and the inherent mechanism to regulate a heterogenous population of mobile agents and their clones.

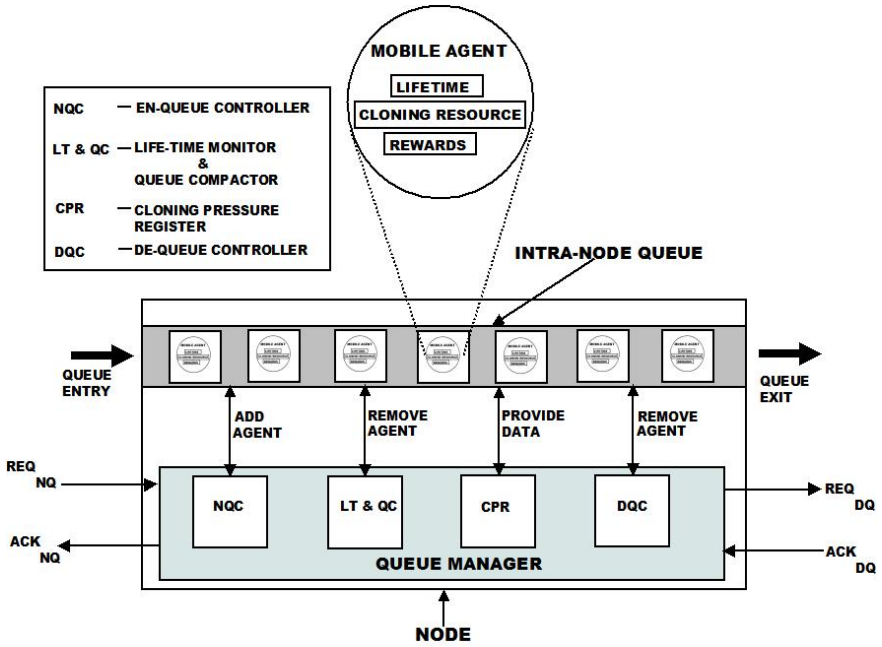
3 The Proposed Cloning Controller

We describe herein, a cloning control mechanism suited for active patrolling which is both scalable and adaptive in nature. Cloning is controlled egocentrically by each mobile agent without imposing any overheads for communicating with other mobile agents across the network, thus making the mechanism well suited for distributed systems. There is no centralized mechanism to serve information of any kind to these agents.

3.1 Architecture of Cloning Controller

Figure 3(a) depicts the composition of the cloning controller residing within each node. Each node in the network has a queue into which all the mobile agents hosted by it are lined up before migrating to another node. Thus mobile agents migrate from the intra-node queue of one node to that of a neighbour. If the node is an RRS, and the incoming agent has the requested service then the same is downloaded onto the node before the agent enters the queue for onward migration to a neighbouring node. Apart from the service or code for a task, the mobile agents carry within themselves a record of their current life-time, cloning resource and the number of rewards received. These concepts have been dealt within subsequent sections.

A *Queue-Manager* controls the intra-node queue. The *De-queue* (DQ) Controller within this manager performs the job of handshaking with a node in its immediate



(a)

The following operations occur in every step at each node:

AtNodeQueue

1. If an Agent is at the top of the Intra-node Queue
   a. CloneifNecessary()
   b. Compute "NextNode" using Pheromone-Conscientious Algorithm
   c. Check if movement to the next node is possible. If true goto (d) else goto(3).
   d. Execute OnDeparture() Method
2. If an Agent is permitted into the Intra-NodeQueue
   a. Execute OnArrival() Method.
3. Decrement the Life-time of every agent in the queue

OnArrival()

1. Execute the service if this is the RRS that requested for its service. If true goto (2) else goto (3)
2. Charge the Cloning Resource and the Life-time based on rewards.
3. Enter into the Intra-NodeQueue of the entering Node

OnDeparture()

1. Remove from the Intra-NodeQueue of the exiting Node

CloneifNecessary()

1. Find Resource needed for Cloning.
2. Find the Number of Clones
3. Decrement the Resource based on the Number of clones
4. Create clones
5. Recharge the Cloning Resource

(b)

Fig. 3. (a) Architecture of the Cloning Controller (b) The Cloning Control Mechanism

neighbourhood by sending a request for the migration of an agent within its associated intra-node queue to the next node via REQ_{DQ}. It receives the acknowledgement from its peer in the other node via ACK_{DQ}. The *En-queue* (NQ) Controller performs the task complementary to the *De-queue* Controller and caters to requests for migration of an agent residing in another node into its intra-node queue using REQ_{DQ} and ACK_{DQ} in a similar fashion. Migrations are performed only if the queue in the next node has a vacant slot. The *Life-time Monitor cum Queue-Compactor* unit ensures that the life-times of each of the agents within the queue are decremented in each step and the queue is compacted as and when an agent dies within the queue. The agent reads the *Cloning Pressure*, ρ, from the *Cloning Pressure Register* resident within the *Queue-Manager*. The cloning pressure is calculated based on the number of agents populating the intra-node queue and has been dealt with in a subsequent section on the dynamics of the controller.

3.2 Stigmergic Sensing

Stigmergy is a form of indirect communication wherein the entities of a system communicate amongst each other through their environment [18]. This type of communication is prevalent in social insects such as ants, wasps, honey bees, etc [32]. In this work, the mobile agents form the entities while the nodes in the network makes up the environment. Whenever an agent lands on a node, it provides its services to the node if the latter has generated a request for the same. The agent is then queued in the intra-node queue present at this node and made to wait for its turn to migrate to the next robotic node. A higher number of agents in this queue essentially means more waiting times within it and also that node to node migrations have been taking more time. Such increased migration times convey a cluttered network condition. Likewise, a lesser number of agents within the queue means less number of the agents populating the network. The intra-node queue form part of the environment of the agent and indirectly conveys the current status of the network. By examining this queue, a mobile agent thus stigmergically senses the current network condition based on the number of agents, and clones proportionately.

Figure 3(b) shows the mechanism behind the stigmergic cloning control. A queue threshold (Q_{Th}) determines the maximum number of agents allowed in the intra-node queue. The basic objective is to ensure that this queue is moderately filled so as to achieve quicker migrations and hence faster service for all the RRSs. Agent migrations can be realized only if there is a vacancy in the intra-node queue of the destination node; else the agent needs to wait within the queue of the current node while also expending its life-time. If we assume that cloning by agents causes their numbers to increase to a level that there are no empty slots within any of the intra-node queues in the network, then no agent will be in a position to migrate. This will mean that the network is choked-up. Only when some of the mobile agents within the intra-node queues die out due to their decreasing life-times, will other agents with higher-life times be able to migrate to vacancies created by their dead counterparts in other intra-node queues. Thus, the agents need to clone in such a contained manner as to increase their numbers to result in a faster service of the RRSs but at the same time ensure that such a choke-up of the network is never reached.

3.3 Cloning Resource - The Underlying Rationale and Functions

The concept of a resource [33, 34] can greatly alter an otherwise linear cloning mechanism. We argue that biological glands [35] cannot secrete in large amounts continuously and are limited by an inherent *resource*. This is much like squeezing a completely wet piece of sponge to extract a certain amount of water. The squeezing force required to extract a certain quantity of water initially is far less than the force required to extract the same amount of water a second time. Water, in this case, is the *resource* being extracted. In biological systems, the resource is recharged by various factors which include the nutrients supplied to the gland over periods of time.

We embed a similar mechanism to emulate a *cloning resource* which is replenished by rewards that a mobile agent gains after servicing an RRS. With the cloning resource charged using such rewards, its chances of generating more clones also increases. Apart from rewards, the cloning resource of a mobile agent is also boosted periodically based on its current value and certain ambient conditions. The dynamics of cloning and the manner of resource charging have been discussed in later sections.

The cloning controller works based on a reactive mechanism to maintain the population of mobile agents within the networked system. This mechanism, embedded within each agent, senses the number of existing mobile agents in the intra-node queue waiting for their turn to migrate to the next node. The number of agents within this queue potentially gives a feeling of the network conditions based on which the agent decides whether or not to clone. The extent of cloning depends on the following parameters:

1. The *Cloning Resource* available within the agent,
2. The *Rewards* it has gained by servicing the RRSs and
3. The *Cloning Pressure* which is proportional to the number of vacant slots in the queue.

Cloning resource is charged partly by rewards and partly by an inherent charging mechanism embedded within the agent. Apart from a cloning resource, agents also have a life-time stamped on them which increases with the rewards they acquire as they service the RRSs.

3.4 Dynamics of Mobile Agent Cloning

Initially at time $t = 0$, the system consists of only parent agents –at least one per type of agent which always exists and never dies. These agents continue to populate the network all through ensuring that at least one copy of their service (payload) exists within the network. In subsequent discussions we will use the term *agent* to refer to either the parent agent or the clone unless otherwise specified. An agent makes the decision to clone based on several factors and the total number of clones generated affects the overall performance of the system in terms of both the utilization of the network resources and also RRS servicing times. The decision as to whether or not an agent, resident in node N at time t, should clone is made based on the *Cloning Pressure*, $\rho_c(t)$ given by equation (1).

$$\rho_c^N(t) = \begin{cases} Q_{Th} - Q_N(t) & \text{for } \rho_c(t) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where Q_{Th} is the *Queue Threshold* and $Q_N(t)$ is the number of mobile agents populating the queue at time t in node N. A mobile agent M residing at node N, generates C number of clones at time t based on equation (2).

$$C^M(t) = \rho_c^N(t)\{R_{av}^M(t)/R_{max}\} \tag{2}$$

where R_{av}^M is the available cloning resource within the agent M and R_{max} is the maximum cloning resource an agent can possess. C is rounded off to the next lowest integer. Initially all parent agents have the maximum cloning resource to their credit i.e. $R_{av} = R_{max}$. The cloning resource is depleted from an agent M for every cloning session based on the equation (3).

$$R_{av}^M(t+1) = R_{av}^M(t) - C^M(t)R_{min} \tag{3}$$

where R_{min} is the minimum cloning resource required to generate a clone. Each of the clones needs to be conferred this minimum value of resource. This value is extracted from the personal cloning resource of the agent that created the clones. If this agent cannot afford to confer this minimum amount of resource then its cloning is inhibited. The cloning resource is charged based on the equation (4).

$$R_{av}^M(t+1) = \begin{cases} R_{av}^M(t) + \tau_c e^{-1/R_{av}(t)} + \tau_r Rew(t) & \text{for } R_{av}^M(t) \geq 1, \rho_c^N(t) \leq 1 \\ R_{av}^M(t) + \tau_c + \tau_r Rew(t) & \text{for } R_{av}^M(t) < 1, \rho_c^N(t) < 1 \\ R_{av}^M(t) + \tau_c e^{(1-1/x)} + \tau_r Rew(t) & \text{for } \rho_c^N(t) > 1 \text{ where } x = \rho_c^N(t) \\ R_{max} & \text{if } R_{av}^M(t+1) > R_{max} \end{cases} \tag{4}$$

where $Rew(t)$ is taken to be 1 if the agent is able to service an RRS; else it is zero at time t. τ_c and τ_r are non-zero positive constants. Care is taken to ensure that the cloning resource, R_{av}, never exceeds its maximum value R_{max} and if so it is brought down to the latter. Every mobile agent or clone has, in addition to the cloning resource, a life-time conferred on it at the time it is created. The life-time $L(t)$ is decremented in every time step according to equation (5).

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

Life-times of the agents are increased by a factor as and when these agents earn rewards as given by equation (6).

$$L(t+1) = L(t) + \sigma Rew(t) \tag{6}$$

where L(t) is the life-time of an agent at time t and σ is a non-zero positive integer constant.

4 Results and Discussions

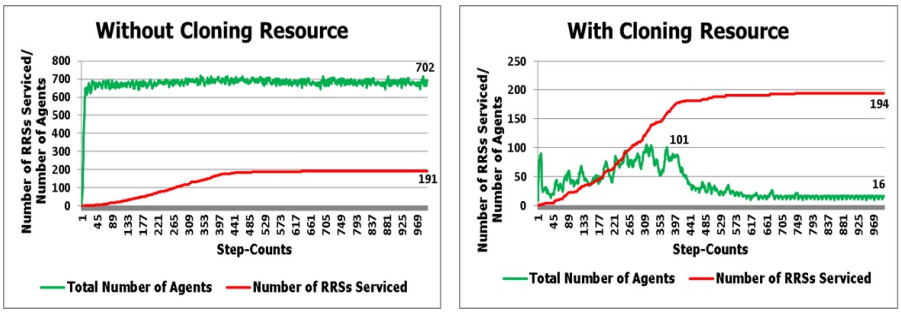
The mechanism of cloning controller was simulated together with the mobile robotic nodes hosting mobile agents which migrate using the Pheromone-Conscientious (*Pher-Con*) [14] mechanism. A stand-alone simulator for the above setup was coded in Java™ as a discrete event system simulator together with a rendering engine.

Two pertinent terms with respect to the simulator are: (i) *Run*: It signifies one complete simulation cycle. (ii) *Step*: A run comprises several discrete steps. Many operations may be executed during each such step by both the robotic nodes and the mobile agents comprising the network. The count of the steps is referred to as the *Step-count*.

Simultaneous multiple mobile agent migrations do not occur in the same step-count. This ensures that the simulation closely matches the real world wherein when two entities transact with one another, a third entity desirous of a transaction with any of the former two, needs to wait for the transaction to end. Hence at any given step-count while an agent is migrating from the queue in node A to that in node B, no other agents are allowed to migrate to either of these nodes. Further, in such a state no other agents from A or B can migrate to other nodes.

The simulation was carried out using 200 nodes with an initial population of 8 parent agents designated *Agent-0* through *Agent-7* where *Agent-0* carries code for *Service-0* and *Agent-1* carries code for *Service-1* and so on. One run of the simulation was carried out for 1000 step-counts. The values of the parameters used in the simulations are $R_{max} = 100$, $R_{min} = 10$, $R_{av} = R_{max}$ for the initial population of agents and $R_{av} = R_{min}$ for the clones when they are created, $L = 30$, $Q_{Th} = 4$, $\tau_c = 0.1$, $\tau_r = 100$ and $\sigma = 7$. A set of 194 RRSs requesting for *Service-0* through *Service-7*, were generated at discrete simulation steps in a random manner. The RRSs then diffuse pheromones for the requested service across their neighbours. One simulation run, each comprising 1000 step-counts was carried out separately in the absence and presence of the cloning resource. Results were obtained by observing the maximum number of mobile agents generated and the total number of step-counts required to satisfy 194 RRSs.

The graph in Figure 4(a) shows a plot of the variation of the total number of agents and the number of RRSs serviced with the simulation step-counts when the cloning resource feature was disabled. This was done by assuming that all agents and their clones are free to always clone repeatedly irrespective of their cloning resource value. Disabling the feature was achieved by considering the factor (R_{av}/R_{max}) in equation (2) to be unity always. It can be observed from this graph that only 191 out of 194 RRSs were satisfied in 1000 step-counts using a total agent population of around 700 in the



(a) (b)

Fig. 4. Graph depicting the variation of the total number of agents and the RRS service times for (a) Without Cloning Resource and (b) With Cloning Resource

network. The population of mobile agents seems to continue to hover around 700 even though no more RRSs were generated and a few were still waiting to be serviced, thus needlessly consuming precious bandwidth and resources.

The graph in Figure 4(b) depicts the performance of the cloning controller using the concept of the cloning resource. Using just 101 agents, all 194 RRSs were satisfied in 743 step-counts. This graph also shows a sharp increase in the number agents proportional to the RRSs generated within the network followed by its decrease to a very low value after most of the RRSs are serviced. The increase and decrease conforms to the rate at which the RRSs are generated and serviced thus exhibiting an adaptive behaviour on part of the cloning controller. Further the number of agents populating the network when almost no RRSs exist, is just about 15 in contrast to 700 in the previous case saving the precious computational resource which could be utilized for other purposes or entities within the network. The heterogeneous mobile agents thus clone based on their own decision but also ensure that they do not grow too large in number so as to curb the migrations of the other types of agents.

Simulations were also conducted with a total of 292 RRSs generated over 1000 step-counts. On an average around 30 RRSs requested for the same type of service. RRSs requesting a certain type of service were generated in an interleaved fashion every 100 step-counts. In addition 27 RRSs once again requesting *Service-0* carried by *Agent-0* were generated starting from step-count 250 onwards. Figure 5 shows the nature by which the number of agents of each type increase to a maximum and then go down to a minimum after the corresponding RRSs are satisfied. It can also be seen that the rate of generation of the clones of *Agent-0* in the second phase after the 250^{th} step-count is much faster than that in the initial stage. As an agent services more RRSs it gains both cloning resource and life-time due to the rewards it accrues in doing so.

Increased life-times make some of these agents to live longer. When the second burst of RRSs occurred, the population of *Agent-0* and its clones had not died down to the minimum. This facilitated the generation of a larger number of clones in a short period of time causing the steep peak. The graph also shows how the overall population of heterogeneous agents varies. It can be seen that this value is contained below 120 and goes down rapidly when no RRSs populate the network indicating clearly that the

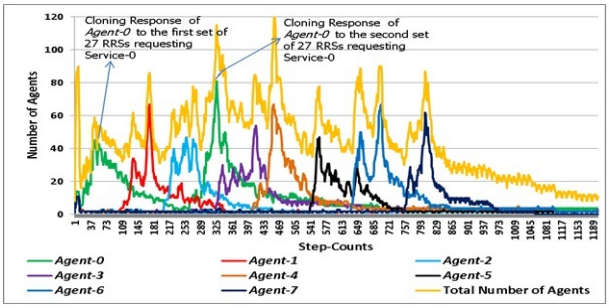


Fig. 5. Graph showing the increase and decrease in the number of each type of agent— *Agent-0* through *Agent-7*

heterogeneous mobile agent population controls itself selectively and on-demand. All unnecessary clones are automatically flushed once all the RRSs have been serviced. The existing parent agents (*Agent-0* through *Agent-7*) then continue to patrol the network in a conscientious manner thereby allowing other network related activity.

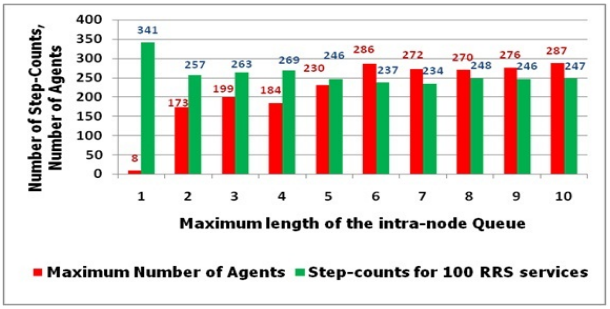


Fig. 6. Variation in the number of Mobile Agents and Clones and Step-counts to service 100 RRSs for different values of Q_{Th}

Figure 6 shows the effect of the variation of the intra-node queue length, Q_{Th}, on the number of agents and clones generated and the step-counts taken to service a constant number of RRSs (100 in this case) using the cloning resource. It can be observed from the graph that at low Q_{Th} values the total number of agents and clones generated is also low thereby taking more steps to service all the 100 RRSs. As the Q_{Th} value increases the performance increases due to an increase in number of agents and clones generated. However further increase in Q_{Th} beyond 4, does not portray any drastic change in performance. Both the number of agents and clones and the number of step-counts consumed vary within a narrow band between 240 and 250. Thus even though with higher values of Q_{Th} the agents or their clones find more empty spaces in the queue they refrain from needless over-cloning. Such a judicious cloning coupled with minimal service times of the RRSs, can provide more space to other entities that populate the network. In real networks, inter-node communications and message passing will also use network resources such as the bandwidth. Since the cloning controller consumes only a part of these resources and that too only on-demand it ensures that a fair amount of networking resources is always available for the other communications. Thus choke-ups will rarely occur in such networks.

4.1 Performance in Dynamic Networks

Since the underlying application was aimed towards controlling mobile agent cloning in multi mobile robot networked systems as cited in [14], it is important to evaluate the performance of the same when the robotic nodes hosting mobile agents move relative to one another. In order to compare the performance of the cloning controller in both static and dynamic networks, simulations similar to that described in Section 2 were

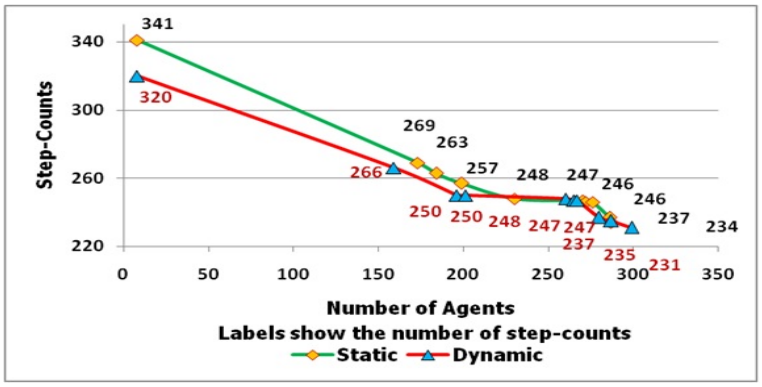


Fig. 7. A comparison of the performance of the Cloning Controller when used in Static and Dynamic Networks

carried out using a 200 node network. The agents however used the proposed cloning control mechanism.

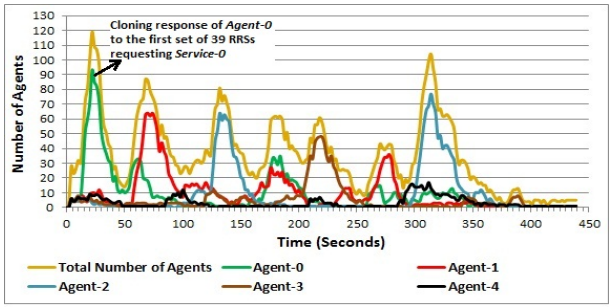
Figure 7 shows the variation in the performances for servicing 100 RRSs for both the static and dynamic cases. With cloning controller mechanism active within the agents, the maximum number agents to satisfy 100 RRSs oscillate in between 150 to 300 for both the static and dynamic networks. For lower number of agents and clones, the Step-Counts required for the service to be completed is high. As the number of these agents and clones increases the Step-Counts decrease and hover around 230-260, for both the static and dynamic cases. The graphs clearly indicate that the cloning controller works almost the same way in both the scenarios and contains the mobile agent population to the left of the axis of the parabola as discussed in Section 2.

4.2 Emulation on Real Networks

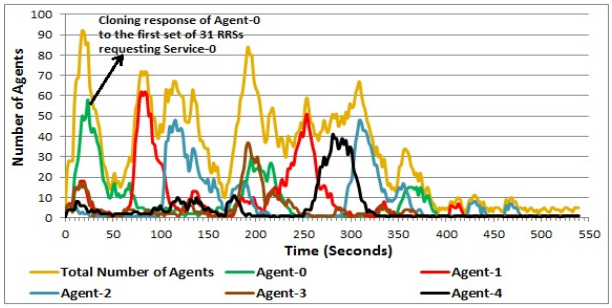
To validate the efficacy of the proposed cloning control mechanism on real networks, the same was emulated over a LAN using the Typhon [36] mobile agent framework. Typhon facilitates various mobile agent functionalities such as mobility, code-carrying ability, cloning of mobile agents, etc. Each instantiation of Typhon within a computer emulates a robotic node. The Typhon framework allows several such instantiations or nodes to co-exist in a single computer (localhost).

50 Typhon based nodes (instantiations) were distributed amongst seven PCs connected within a LAN to emulate a robotic network. Each node was capable of running its own cloning controller. Experiments were performed on both static and dynamic networks. In the dynamic case, nodes were made to freely connect to any other node in the network and also drop connections to any of their neighbours thus emulating mobility of robotic nodes. The dynamic network was generated using the Erdős-Rényi $G(n,p)$ model [37] where n is the nodes in the network and p is the probability of making or breaking a connection. We have chosen $p = 0.5$. The emulation was performed with an initial population of 5 different types of parent agents designated as *Agent-0* through

Agent-4. The Pheromone-Conscientious (*PherCon*) [14] mechanism was embedded in the agents for migration within the network. Except for the value of L which was taken to be 10 seconds, all other parameter values used were same as those used in the simulation.



(a) Static network



(b) Dynamic network

Fig. 8. Graph showing the change in population of agents viz. *Agent-0* through *Agent-4* in the emulation - (a) In a Static Network and (b) In a Dynamic network

Figures 8(a) and 8(b) depict the variations in the populations of the different types of agents (including clones) in the static and dynamic networks respectively. The RRSs requesting different types of services (*Service-0* through *Service-4*) were randomly generated in both the networks. In order to test the efficacy of the cloning controller, a relatively high demand for a particular service was created within the network by increasing the probability of generating RRSs that requested this service. It was found from the log that in the static case 237 RRSs were generated while that in the dynamic was 252. It can be noted that the results obtained from the emulation conform to those obtained from simulation (see Figure 5). In Figure 8(a) since a high demand for *Service-0* was generated in the first 50 seconds (37 RRSs), the population of *Agent-0* increased drastically while those of the others were found to be low. After the 50^{th} second the demand for *Service-1* increased causing a rapid increase in the population of *Agent-1*. Subsequently, the graph shows similar responses for agents viz. *Agent-2*, *Agent-0*, *Agent-3*,

Agent-1 followed by *Agent-2*, indicating their greater demands. A similar trend can be observed for the dynamic network, response of which is shown in Figure 8(b). Further, it can be observed that the populations of all agents (or clones) die rapidly as soon as the respective RRSs are serviced.

In order to find the approximate upper bound of the number of agents that could actually populate the 50-node network we forced all nodes to become RRSs repeatedly. Figure 9 shows the variation in the number of agents when all nodes were made to request *Service-0* repeatedly. As can be seen the maximum number of clones (including the parent agent) that populate the network during this run is 160 which is less than 200 (queue length at each node $\times$ total number of nodes in the network). If 200 agents were to populate this 50-node network there would be no space for their movement to other nodes since the intra-node queues within all the nodes are full. This would thus result in a cluttering and eventually a choke-up. The graph indicates clearly that even if we load the network by generating the largest number of RRSs (in this case 50) repeatedly, the total number of agents remains well below 200 thus providing vacancies in the intra-node queues and facilitating proper mobility of agents within the network.

In Figures 8(a) and 8(b), the total population of the agents (including clones) within the network always remains below 120 in case of static and 92 in case of the dynamic network indicating clearly that mobility of agents within the network is facilitated all through. The total population is well below the empirically found upper limit of 160. Thus the cloning control mechanism is able to stigmergically curtail the agent population well within limits thus avoiding choke-ups and also ensuring effective service to all RRSs.

5 Comparison with other approaches

Earlier approaches [24, 25, 27, 28, 30] to mobile agent population control endeavour to ensure that their total number never exceeds a fraction of the number of nodes of the network they populate. The agents are assumed to be homogeneous. They do not mention how a heterogeneous set of agents can be handled or whether demand-based selective rise and fall of different agent populations can be controlled. For instance

Fig. 9. Variation in the population of *Agent-0* when all nodes were forced to become RRSs requesting *Service-0* repeatedly

Golebiewski *et al.* [27] and Suzuki *et al.* [24, 25] describe mechanisms to maintain a population of homogeneous agents both in static and dynamic networks. The total agent population is maintained to a fraction of the number of nodes in the network. Their target application domain is different from the one described in this paper, though.

The mechanism proposed by Golebiewski *et al.* [27] preserves a parameter p (referred to as the reproduction probability) at each node and maintains at least one copy of an agent at each node. Though their algorithm may seem intuitive, the calculation of this parameter p is non-trivial. They assume the network to be fully connected and node and agents to be operating synchronously. Their results were obtained through simulations which allow such assumptions. In real networks these assumptions do not hold, making such workable implementations impossible unless some global parameters such as the total number of nodes and the desired number of agents are known a priori. Heterogeneity of agents and the selective control, of their respective populations, is never considered.

Marina Flores-Badillo *et al.* [28] have discussed an algorithm to manage the agent population in work flow automation. Their system uses three types of agents viz. controller agents, executer agents and blackboard agents. The protocol uses the concept of energy as the time-to-live (ttl) for the controller and executor agents for their functioning in the network. These ttls are controlled by the application. This protocol is complex and has tightly coupled entities as there is a large amount of communication and dependency among all the three types of agents. Further, the control is not distributed and involves a large amount of book-keeping to keep track of the locations of the agents and their respective energies. Though their protocol may be suitable for the domain of work-flow automation, communication complexities and non-localized decisions render it to be unsuitable for use in real dynamic and distributed environments.

Suzuki *et al.* [24, 25] proposed an agent population mechanism in dynamic networks using the single species (homogeneous) population model. They estimate the link densities of each node in a dynamic network and then generate food required to be consumed by agents so as to live and clone. Though, their simulation results portray that it is possible to maintain the agent population up to a given fraction of the nodes in a network (within 20% of error), they do not account for the overhead generated due to the communication complexity ($O(|Nbk| * N)$, $Nbk =$ Set of nodes at k distance; $N =$ Number of nodes in the network) of their approximation. If we extend their mechanism to cater to the control of the populations of a heterogeneous set of agents, this complexity further increases per node and the food stored and maintained within the node would also be heterogeneous in nature. Further an *a priori* knowledge of the actual number of different types of (heterogeneous) agents in the system needs to be known. An approximation on the frequency of visits made by these agents also needs to be reformulated.

Bakhouya and Gaber [30] have proposed an algorithm inspired by the immune network theory [31]. Their algorithm caters to the need of a dynamic distributed system wherein they have used the inter-arrival-times of different agents as cues to tune the agent population. They have modeled three behaviours for the mobile agents viz. Migration, Cloning and Termination and just like B-Cells in the immune system, their relative concentration decides which of these is triggered. Their method does not re-

quire the number of agents and nodes in the network to be known *a priori* and uses agent's inter-arrival times to sense the agent population.

However, heterogeneity of the agents is not taken into account. Since the number of agents to be populated in the network (N) is optimized independent of number of different types of agents (α) available in the system, their algorithm could choke the system by generating αN agents (N number of agents for each type). In real networks populated with a heterogeneous set of agents, delays could occur due to large queues at the bottle-neck links. If these network delays are to be taken into account, they may inadvertently cause the respective nodes to process based on larger inter-arrival times to eventually destabilize the agent population. If one were to extend their model to support a heterogeneous set of mobile agents it would mean that each node maintain the inter-arrival times of every type of agent. Every time a new type of agent is introduced into the system the nodes may require be either reprogramming or embedding with a mechanism to identify such new agents. The knowledge of the number of the types of agents (α) would need to be known *a priori* thus affecting scalability of the system. All this naturally would increase the computational and space overheads within each node in the network.

The population control mechanism proposed in this paper remedies the problems with these earlier approaches by allowing a heterogeneous set of agents to co-exist in a dynamic distributed network without compromising the upper bound on the total agent population. Since the per node computations do not require any specific information on the type of an agent, new agents may be introduced into the network on-the-fly without hindering performance. The control mechanism described herein focuses on a demand based system that is capable of selectively controlling the rise and fall of different types of mobile agents based on their requirement at the nodes in the network. If the need for certain types of agents is high, their population grows proportionately based on their demand in the network while those of the others wane, ensuring that the network does not get choked up. One may intuitively infer that the inherent demand based policy frees up network resources by lowering the number of agents and allowing for quicker movement of high demand agents within the network. Further, the proposed mechanism uses minimal node-to-node communications (for migrating an agent from one queue to another) which greatly reduces communication overheads within the network and hence saves on bandwidth.

6 Conclusions

Most of the state of the art population control mechanisms are directed towards controlling the cloning of a homogenous set of agents. Controlling a clone population in heterogeneous mobile agent based scenarios is non-trivial. We described a novel demand based adaptive mechanism for controlling the population of a heterogeneous set of mobile agents using stigmergic sensing of the condition of the network . Stigmergic sensing becomes essential in the real world since it is not possible for each mobile agent to communicate with all the others to compute the true global information on its population. This sensing, based on the number of mobile agents queued up for migration, coupled with a cloning resource charged by rewards accrued by servicing RRSs, facil-

itates a more effective utilization of the available network resources. Clonal resource and life-time also facilitate a faster response when the same type of service is requested very frequently by the nodes. The mechanism provides for a faster service of numerous RRSs and at the same time judiciously consumes network resources allowing other entities that may populate the network to use the same. The mechanism described herein, uses a stigmergic technique to allow both mobile agents and other information to flow efficiently within the network. The results obtained from the emulation reveals the practical viability of the mechanism of cloning control. The paper also explains the adaptive nature of the mechanism and its robustness even in dynamic scenarios.

The mechanism can be further enhanced using concepts from the Natural Immune Systems so that a highly rewarded agent could become an Immune memory with life-time comparable with that of its original parent. This will ensure that further responses to the generation of RRSs will be much faster. In future we envisage using mobile agents coupled with such a clonal controller in the domain of wireless mobile networks so as to improve their quality of service.

Acknowledgements

The authors wish to thank the Department of Science and Technology, Government of India, for the funding provided under the FIST scheme to set up the Robotics Lab.[3] at the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, where the entire reported work was carried out.
The second author would like to acknowledge Tata Consultancy Services for their support under TCS-RSP.

References

1. Chess, D., Harrison, C., Kershenbaum, A.: Mobile agents: Are they a good idea? In: Mobile Object Systems Towards the Programmable Internet. Springer (1997) 25–45
2. Dale, J.: A Mobile Agent Architecture to Support Distributed Resource Information Management. Master's thesis (January 1996)
3. Manvi, S.S., Venkataram, P.: Mobile agent based approach for QoS routing. IET communications **1**(3) (2007) 430–439
4. Van Thanh, D.: Using mobile agents in telecommunications. In: Proceedings of 12th International Workshop on Database and Expert Systems Applications, 2001., IEEE (2001) 685–688
5. Wei, C., Yi, Z.: A multi-constrained routing algorithm based on mobile agent for MANET networks. In: Proceedings of International Joint Conference on Artificial Intelligence, 2009. JCAI'09., IEEE (2009) 16–19
6. Pathak, H., Nipur, Garg, K.: A Fault Tolerant Comparison Internet Shopping System: Best Deal by Using Mobile Agent. In: Proceedings of International Conference on Information Management and Engineering, 2009. ICIME '09. (2009) 541–545
7. Yamaya, T., Shintani, T., Ozono, T., Hiraoka, Y., Hattori, H., Ito, T., Fukuta, N., Umemura, K.: MiNet: Building ad-hoc peer-to-peer networks for information sharing based on mobile agents. In: Practical Aspects of Knowledge Management. Springer (2004) 59–70

[3] www.iitg.ernet.in/cse/robotics

8. Cragg, L., Hu, H.: Application of mobile agents to robust teleoperation of internet robots in nuclear decommissioning. In: Proceedings of IEEE International Conference on Industrial Technology, 2003. Volume 2., IEEE (2003) 1214–1219
9. Cragg, L., Hu, H.: A Multi-Agent System for Distributed Control of Networked Mobile Robots. Measurement and Control **38**(10) (2005) 314–319
10. Cragg, L., Hu, H.: Mobile agent approach to networked robots. The International Journal of Advanced Manufacturing Technology **30**(9-10) (2006) 979–987
11. Chu, H.N., Glad, A., Simonin, O., Sempe, F., Drogoul, A., Charpillet, F.: Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation. In: Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence, 2007. ICTAI 2007. Volume 1., IEEE (2007) 442–449
12. Sempé, F., Drogoul, A.: Adaptive patrol for a group of robots. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.(IROS 2003). Volume 3., IEEE (2003) 2865–2869
13. Godfrey, W., Nair, S.B.: An Immune System Based Multi-robot Mobile Agent Network. In Bentley, P., Lee, D., Jung, S., eds.: Artificial Immune Systems. Volume 5132 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2008) 424–433
14. Godfrey, W.W., Nair, S.B.: A Pheromone Based Mobile Agent Migration Strategy for Servicing Networked Robots. In: Bio-Inspired Models of Network, Information, and Computing Systems. Springer (2012) 533–541
15. De Castro, L.N.: Fundamentals of natural computing: basic concepts, algorithms, and applications. Volume 11. CRC Press (2006)
16. Minar, N., Kramer, K.H., Maes, P.: Cooperating mobile agents for dynamic network routing. In: Software agents for future communication systems. Springer (1999) 287–304
17. Godfrey, W.W., Nair, S.B.: Mobile agent cloning for servicing networked robots. In: Principles and Practice of Multi-Agent Systems. Springer (2012) 336–339
18. Bonabeau, E.: Editor's introduction: stigmergy. Artificial Life **5**(2) (1999) 95–96
19. Godfrey, W.W., Nair, S.B.: A Mobile Agent Cloning Controller for Servicing Networked Robots. In: Proceedings of 2011 International Conference on Future Information Technology, IPCSIT 2011., IACSIT Press (2011) 81–85
20. Glitho, R., Olougouna, E., Pierre, S.: Mobile agents and their use for information retrieval: a brief overview and an elaborate case study. IEEE Network **16**(1) (2002) 34–41
21. Jin, Y., Qu, W., Zhang, Y., Wang, Y.: A mobile agent-based routing model for grid computing. The Journal of Supercomputing **63**(2) (2013) 431–442
22. Hamza, S., Okba, K., Aïcha-Nabila, B., Youssef, A.: A Cloud computing approach based on mobile agents for Web services discovery. In: 2012 Second International Conference on Innovative Computing Technology (INTECH). (2012) 297–304
23. Stahl, F., Gaber, M., Bramer, M., Yu, P.: Pocket Data Mining: Towards Collaborative Data Mining in Mobile Computing Environments. In: 2010 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI). Volume 2. (2010) 323–330
24. Suzuki, T., Izumi, T., Ooshita, F., Masuzawa, T.: Biologically inspired self-adaptation of mobile agent population. In: Proceedings of Sixteenth International Workshop on Database and Expert Systems Applications, 2005., IEEE (2005) 170–174
25. Suzuki, T., Izumi, T., Ooshita, F., Masuzawa, T.: Self-adaptive mobile agent population control in dynamic networks based on the single species population model. IEICE transactions on information and systems **90**(1) (2007) 314–324
26. Ma, J., Voelker, G.M., Savage, S.: Self-stopping worms. In: Proceedings of the 2005 ACM workshop on Rapid malcode, ACM (2005) 12–21
27. Golebiewski, Z., Kutylowski, M., Luczak, T., Zagórski, F.: Self-stabilizing population of mobile agents. In: Proceedings of IEEE International Symposium on Parallel and Distributed Processing, 2008. IPDPS 2008., IEEE (2008) 1–8

28. Flores-Badillo, M., Padilla-Duarte, A., López-Mellado, E.: A population control protocol for mobile agent based workflow automation. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 2009. SMC 2009., IEEE (2009) 4059–4064
29. Amin, K.A., Mikler, A.R., Prasanna, V.I.: Dynamic agent population in agent-based distance vector routing. Neural, Parallel & Scientific Computations **11**(1 & 2) (2003) 127–142
30. Bakhouya, M., Gaber, J.: Adaptive approach for the regulation of a mobile agent population in a distributed network. In: Proceedings of The Fifth International Symposium on Parallel and Distributed Computing, 2006. ISPDC'06., IEEE (2006) 360–366
31. Farmer, J.D., Packard, N.H., Perelson, A.S.: The immune system, adaptation, and machine learning. Physica D: Nonlinear Phenomena **22**(1) (1986) 187–204
32. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems. Number 1. OUP USA (1999)
33. Nair, S.B., Godfrey, W.W., Kim, D.H.: On Realizing a Multi-Agent Emotion Engine. International Journal of Synthetic Emotions (IJSE) **2**(2) (2011) 1–27
34. Godfrey, W.W., Nair, S.B., Kim, D.H.: Towards a dynamic emotional model. In: Proceedings of IEEE International Symposium on Industrial Electronics, 2009. ISIE 2009., IEEE (2009) 1932–1936
35. Mishra, R., Srivastava, A., Bhaumik, K., Chaudhary, S.: Acth and regulation of adrenocortical secretion: A mathematical model. Indian Journal of Pure Applied Mathematics **13**(12) (1982) 1503–1512
36. Matani, J., Nair, S.B.: Typhon-A Mobile Agents Framework for Real World Emulation in Prolog. In: Multi-disciplinary Trends in Artificial Intelligence. Springer (2011) 261–273
37. Erdős, P., Rényi, A.: On the evolution of random graphs. Publ. Math. Inst. Hungar. Acad. Sci **5** (1960) 17–61