An Adaptive Regulation Approach of Mobile Agent Population Size in Distributed Systems

M. Bakhouya,[1,*] M. Nemiche,[2,†] J. Gaber[3,‡]
[1]*International University of Rabat, Faculte FIL, Parc Technopolis, 11 100 Sala el Jadida, Morocco*
[2]*Faculty of Sciences, Ibn Zohr University, Morocco*
[3]*Universite de Technologie de Belfort-Montbeliard, Rue Thierry Mieg, 90010 Belfort, France*

The development of ubiquitous and pervasive computing systems requires new approaches and paradigms. Mobile agent based approaches have received a great attention for developing distributed applications. Agents are programs that can migrate from a machine to another in a network and perform tasks on distant machines. However, it is difficult to estimate a priori the appropriate number of agents allowed to be spawned in the network without any global information or controller. Indeed, increasing agent population size, with cloning operation, will increase resource demands in the network, which would indirectly affect the network performance. This paper focuses on the problem of dynamic regulation of mobile agent population size in a distributed system and proposes an approach that takes inspiration from the immune system concepts. Simulations have been conducted and results are reported to show the effectiveness of the proposed approach.

1. INTRODUCTION

Ubiquitous and pervasive computing is an emerging paradigm that aims to provide users with access to services all the time, everywhere and in a transparent way, by means of devices embedded in the surrounding physical environment and/or carried by the user.[1] The goal is to develop systems where highly heterogeneous hardware and software resources can seamlessly and spontaneously interoperate to provide a variety of services to users regardless of the specific characteristics of the environment and user devices. As stated in Ref. 2 the aim of ubiquitous computing is to provide any mobile device an access to available services in an existing network all the time and everywhere while the main objective of pervasive computing is to provide spontaneous services created on the fly by mobiles that interact by ad hoc connections. In other words, pervasive computing concerns the use of mobile

*Author to whom all correspondence should be addressed; e-mail: mohamed .bakhouya@uir.ac.ma.
†e-mail: nemiche@uv.es.
‡e-mail: gaber@utbm.fr.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 31, 173–188 (2016)

View this article online at wileyonlinelibrary.com. • DOI 10.1002/int.21750

computing technology and recently data coming from social networks to enhance people's interactions during unexpected contexts. Interactions between users are based on geographic proximity and context awareness and can occur in a shared local such as in the office, in the elevator, at the grocery store, in the class, in the library, in the pool area, and in the gym. Users live close enough so they can interact and eventually see and meet each other in real life if they wish. Usually, interactions can occur when users find that they share common interests, such as hobbies and jobs.

Author in Ref. 2 also point out that most of research works to date in resource management are based on the traditional client–server paradigm (CSP). Indeed, with the CSP, it is the user who should initiate a request, should know a priori that the required service exists, and should be able to provide the location of a server holding that service. However, this paradigm is impracticable in ubiquitous and pervasive environments and cannot meet simultaneously their related needs and requirements that are scalability and adaptability to dynamic environments. Many efforts have been dedicated to develop decentralized architectures for resource management. Thus, network resources must be able to scale, adapt to dynamic conditions in the network, be highly available, and should require minimal human configuration and management.

According to Gaber's classification, two alternative paradigms to CSP have been introduced to design and implement ubiquitous and pervasive applications: the adaptive services-to-client paradigm (SCP) and the spontaneous service emergence paradigm (SEP).[3] With the alternative SCP paradigm, it is the service that comes to the user. In other words, in this paradigm, a decentralized and self-organizing middleware should be able to provide services to users according to their availability and the network status. Such a middleware can be inspired, for example, from a biological system such as the natural immune system. The second alternative SEP paradigm is more adequate for pervasive applications. It involves the concept of spontaneous emergence service composition that suits pervasive environments. More precisely, spontaneous services can be created on the fly and be provided by mobiles that interact by ad hoc connections. The SEP could be also implemented by a natural systems that involve self-organizing and emergence behaviors.[2] It is worth noting that research works in artificial intelligence, agent-based systems, mobile and autonomous robots, distributed systems, and autonomic systems have focused on the development of adaptive approaches and systems that modify their own behavior at run-time to address constantly changing environments.[4] Most approaches are inspired by features and capabilities seen in natural and biological systems, for example, human brain, immune systems, ant colony, and flocks of birds.[5,6] The main goal is to develop run-time mechanisms so that the system autonomously adapts its structure and its behavior during the course of operation.

In the past few years, several approaches, such as in Ref. 7, have been proposed to implement ubiquitous and pervasive applications. For example, an agent-based approach, with self-adapting and self-organizing capabilities, has been proposed in Ref. 8 to implement the SCP paradigm. This approach is based on the mobile agent paradigm and inspired by the immune system to organize resources into communities by creating dynamic affinity relationships to represent services in the network. In this approach, mobile agents are used as an alternative to CSP. They

have the ability to move from location to location to meet other agents or to access resources provided at each location.[9] Furthermore, agents can clone themselves in order to increase system robustness and performances. However, it should be noted that increasing agent population, with cloning operation, will increase resource demands in the network, which would indirectly affect network performance. Since mobile agents operate in a dynamic and distributed environment, it is difficult or even impossible to estimate a priori an appropriate number of agents in the network.[10] Also, changing the population dynamically in response to its environment is a complex issue in the absence of central controller.

In this paper, a distributed approach for the regulation of mobile agent population and inspired by the human immune idiotypic network is presented. In this approach, an agent retrieves local information from its environment and makes appropriate decision to either *kill*, *move without cloning* or *clone itself and move* to another node of the network. We demonstrated through extensive simulations that the number of agents in the network stabilize without having a global knowledge of the network. The approach is also compared to the ant-based approach[10] and results show the effectiveness of the immune-based approach. The rest of the paper is structured as follows. In Section 2, the related work of dynamic regulation of agent population size is presented. Section 3 presents an approach of autonomous adaptive mobile agent by getting inspired from immune system concepts. Simulation results are presented in Section 4. Conclusions and future work are given in Section 5.

2. RELATED WORK

Changing the population dynamically in response to its environment requires a high degree of coordination among agents to analyze the global environment from local information. In Ref. 10, Amin and Mikler have proposed an approach inspired by stigmergetic propriety of *ant colony* to facilitate coordination between agents. More precisely, mobile agents with minimum cognitive capabilities communicate with each other using pheromones that assist them to select an appropriate action. In other words, the stigmergetic propriety may be defined as the indirect communication mediated by modifications of environmental states that are locally accessible by the communicating agents. The intensity of pheromones disposed by agents at each node visited is determined by the equation $e^{-\lambda \Delta t}$, where Δt is the time since the deposition of pheromone and λ is a constant value fixed between 0 and 1.

In this approach, the action selection algorithm for each agent is defined as follows. An agent visiting a node at time t_b extracts the value of the pheromone that was disposed at time t_a ($t_a < t_b$) using the equation $e^{-\lambda \Delta t}$. If this value is above a certain termination threshold Max, the agent kills itself. On the other hand, if the pheromone value reduces below a cloning threshold Min, the agent clones itself. But, if the pheromone is comprised between the termination and cloning thresholds, the agent neither clones nor kills itself. In this case, the agent migrates to another node. This approach was proposed to implement an algorithm for agent-based distance vector routing (ADVR).[10] In ADVR, agents migrate among nodes, thereby establishing routes for every pair of nodes in the network in a distributed way. However, agents are homogeneous, when one or more agents are killed there

are other agents that can achieve the objective. More precisely, in ADVR algorithm an ideal number of agents in the network to enhance the path-cost convergence of shortest-path routes is not required.[10] Therefore, this approach is not applicable when the agents are heterogeneous. In fact, when an agent is eliminated just after creation and that its objective is not then accomplished, the node dispatches other agents to accomplish it.

However, in this approach, values of λ, Min, and Max are constant and fixed a priori for each node. Hence, the experiments conducted for this approach demonstrate that the agent population size depends only on these global values.[8,10] However, the resource utilization will change dynamically, and therefore, changing the values of λ, Min, and Max is required. Dynamically adjusting these thresholds in a distributed fashion is a nontrivial issue problem. Authors consider the existence of a mechanism that map these values to the global state of the network based on local information but this issue is not addressed in their work.[10] Hence, an appropriate mapping of these values to the resource availability must reflect the global state of the network. To change values of λ, Min, and Max a reinforcement learning mechanism is needed. More precisely, the reinforcement learning permits for nodes to adapt the threshold values according to the value of Δt. For example, if a node perceives that, during an interval of time, the value of Δt decreases, then that could be interpreted by the fact that the number of agents present in the network is increased. In this case, the node should adjust the parameter Min to reinforce the elimination of the received agents and thus to cause the reduction in the size of the population in the network.[8]

To build intelligent agents capable of autonomously achieving goals in a complex environment, several research work have been proposed.[11] More precisely, researchers address the problem of how the agent should act and adapt to environment changes.[12] For example, reinforcement learning could provide a framework for an agent to act and adapt to its environment.[12–17] In parallel, algorithms designers have proposed several techniques that could be used to develop algorithms with increasing level of adaptiveness[18] (see Figure 1). The first technique uses *if/switch statements* to evaluate the local function or expression to select a suitable action. Another technique named online parameterized technique is used to select an action based on inputs and parameters that can evolve over time. The algorithm selection technique chooses the most effective algorithm among a fixed set of available algorithms based on given properties, for a specific task or environment state. The last technique, which recently attract researchers' attention, is the use of techniques and principles from natural and biological systems to select suitable actions and generate new actions according to the environment.

It is worth noting that biological and natural systems, such as immune systems, honey bee, and ant colonies, have several features and organizing principles that can be exploited in designing and developing adaptive systems. These superorganisms often use *self-organizing* behaviors and *feedback loops* that allow the system achieving reliable and robust solutions using information gathered from entities.[19] As also stated by Kholodenko,[20] positive and negative feedback loops are key elements of information processing in all biological systems. These feedback loops allow improving information flow and decision making at multiple levels, without a centralized control.[4] For example, the biological immune system is composed of a

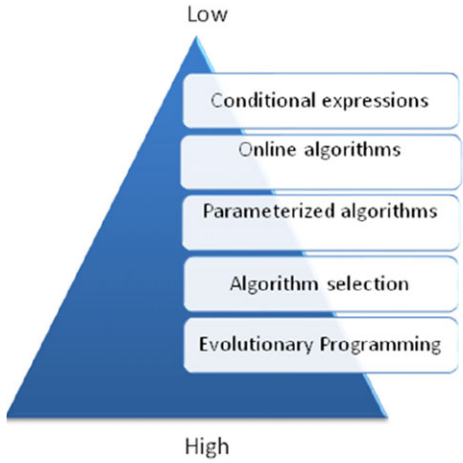


Figure 1. Adaptation rules and techniques: from low to high adaptiveness.

diverse set of cells that are distributed throughout the body and are communicating without a central controller. The immune system *evolves to adapt* and improve the overall system performance (e.g., organizational memory).[21–24]

These natural and biological features have also inspired researchers from the software engineering, autonomic computing, and networking communities for building self-adaptive systems.[25–27] They have also highlighted that *feedback loops* are core design elements and should be made explicit in modeling, design, and implementation of adaptive systems. In this paper, the proposed approach uses a mechanism inspired by the immune system similar to the one used by Watanabe et al.[24] and Ishiguro et al.[28] for intelligent selection of actions by a mobile robot. This mechanism was adapted from a model proposed by Farmer et al.,[22] in which the authors describe a nonlinear dynamical model using differential equations for the immune system based on the immune idiotypic network hypothesis proposed by Jerne.[21] The use of linear equations formulation and iterative methods, which is preferred to a nonlinear system or coupled differential equations that can have multiple attractors, to model adaptive behaviors was proposed in Ref. 29.

3. BIOINSPIRED AUTONOMOUS AGENTS

3.1. General Architecture

A general architecture of autonomous agents was proposed by Ranjan et al. in Ref. 30. It consist of two modules, the interface of the agent to its environment and the controller. The interface contains sensors that sense the environment, analyses them, and creates a view of the environment called percept. The percept is passed on to the controller that decides whether an action (i.e., policy) is appropriate to the current situation. Once the action is selected, it is passed to the interface to be carried out by effectors. As also stated in Ref. 29 to develop the actions selection mechanism, we consider that each autonomous agent can keep track of environment

changes, perform interactions, and communicate with others surrounding agents and the environment. The agent's behavior is governed by the system's dynamics and the peering relationships between the different action rules.

By analogy with the immune system, the current situation of the environment (i.e., percept) detected by the interface (i.e., sensors) works as antigen, and prepared behavior or action is regarded as a B-cell or antibody, while the interaction between actions is considered as stimulation/suppression chain (or a feedback loop) between B cells.[31] The antibody population is represented by the concept of concentration. In this architecture, the description of all variables is integrated and consequently, it is not necessary to have an internal state to describe the dynamics of the system such as in the resolution approach proposed in Ref. 12. More precisely, the agent cannot make the difference between the stimuli and the hidden information (i.e., internal state) to select an appropriate action. Indeed, the differential equations,[22] which describe the evolution of agent behaviors, integrate variables characteristic of actions, the internal state, and variables describing the stimuli coming from the environment of the agent.

It is worth noting that the pioneering work in this area was introduced by Holland,[32] which proposed the classifier system, one of the most popular approaches to machine learning and artificial intelligence. Farmer et al.[22] have explored the similarities between the immune and classifier systems and described a nonlinear dynamical model based on the network hypothesis of Jerne.[21] Jerne has pointed out that the immune system is a sophisticated biological system interpreted as a network of billions of highly specialized cells that specifically stimulate and inhibit each other, and thus exhibits powerful capability for learning, memory, and pattern recognition. The dynamical model using differential equations formulation described in Ref. 22 has been used in several artificial intelligence approaches such as proposed in Refs. 24,28, and 30.

Similarly to the work in Refs. 24,28, and 33 to regulate the agent population size, we incorporate in each agent a controller which is equivalent to the immune idiotypic network. More precisely, each behavior (i.e., action) of the agent is a B cell and the environment is the antigen. As described in Ref. 29, starting from the ideas underlying Holland's classifier system and Farmer et al. nonlinear dynamical system, the action–selection problem in an environment is mathematically identical to solving simple linear systems when potential actions are arranged on an affinity network. The agent behavior can be described by a well-defined finite set of internal actions. Each action has a numerical weighting. Actions are connected to each other by links to encode stimulatory and/or inhibitory interactions. For each environment change, all the actions whose condition matches those changes compete to be acted upon. The action numerical weights are calculated on the basis of the action strength, that is, condition match, and changes according to the peering interactions with the other actions, until they settle down to stable steady values. Then, the algorithm applies the operations that correspond to the actions that best fit the environment, for example, with the highest values. The peering relationships between the different actions can be compactly described by means of an affinity network, that is a directed weighted graph.

For regulating agent population size, mobile agent behaviors are death or *kill*, *move,* and *clone*. As depicted in Figure 2, these behaviors are linked with a

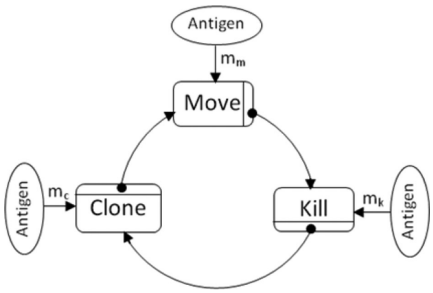


Figure 2. Stimulation/suppression chain between the three behaviors of the mobile agent.

stimulation/suppression chain or feedback loop. Formally, let consider A_c, A_m, and A_k be the concentrations associated, respectively, with the *clone*, *move,* and *kill* behaviors (i.e., B cells). Their variations can be expressed as follows:

$$A_c(t+1) = A_c(t) + (a_k(t) - a_m(t) + m_c - k_c)a_c(t)$$
$$A_m(t+1) = A_m(t) + (a_c(t) - a_k(t) + m_m - k_m)a_m(t)$$
$$A_k(t+1) = A_k(t) + (a_m(t) - a_c(t) + m_k - k_k)a_k(t)$$

where the values k_c, k_m, and k_k are constants and denote the dissipation factor representing the antibody's natural death corresponding to the behaviors *clone*, *move,* and *kill,* respectively. Variables m_c, m_m, and m_k correspond to the affinity of the antigen with the three respective behaviors (i.e., B cells). The values m_c, m_m, and m_k are calculated with the following logistic equation to squash the values A_c, A_m, and $A_{k,}$ respectively, between 0 and 1, as in Refs. 33 and 34: $a_i(t) = \frac{1}{1+e^{(0,5-A_i(t))}}$.

This approach controls the agent population based on the interarrival time. If the interarrival time is small, then this could be interpreted by the fact that an excessive number of agents are present in the system. In this case, the *kill* and *move* behaviors should be stimulated with a higher affinity. On the other hand, if the interarrival time is large, it implies that there are a suboptimal number of agents in the system. In this case, the *clone* and *move* behaviors should be stimulated with a higher affinity. More precisely, an agent visiting a node at time t_b extracts the value of the time t_a of last visit to this node and calculates the value separating the two last visits Δt $(t_b - t_a)$. Using this value, which corresponds to the antigen, the kill action is stimulated with the affinity value m_k equal to $\frac{1}{2}x$, the clone action with the affinity equal m_c to $1 - x$, and the move action with the affinity m_m value equal to $\frac{1}{2}x$, where x is equal $1/(1 + e^{-\Delta t})$. Each behavior concentration is calculated repeatedly (e.g., during 10 calculation steps) and the behavior with the highest concentration is selected. The initial concentration value for each behavior (i.e., $a_i(0)$) in our experiments is considered 0.01. The values of constants parameters K_c, K_m, and K_k are fixed to 0.001. It should be noted that, unlike the approach of Amin and Mikler,[10] the selection of appropriate action is not deterministic. Indeed, behaviors of the agent are stimulated in parallel and the most appropriate behavior to the environment state will emerge and be selected. More precisely, each mobile

agent selects an appropriate behavior to its environment without using any global or static thresholds parameters (i.e., Min and Max).

3.2. Specification of Agents

We have two types of agents: mobile agents (Magents) and agents that represent the nodes of networks (Nagents). In the rest of this section, we present the specification of these agents and their behaviors using input/output automata formalism.[35] This formalism was proposed to model discrete event systems consisting of concurrent and distributed components that receive inputs from and react to their environment. Each component can be modeled as an I/O automata with an internal state and three actions, *input*, *output*, and *internal*. For example, using the I/O automata formalism, the state of an agent *Nagent* is described as follows:

- The address n belonging to $Nodes$, the set of all nodes of the network.
- The neighboring set V_n.
- The value t_a that represents the time of last visit to that node by another mobile agent.

In the I/O automaton model, *Nagent* receives requests from mobiles agents, via the request input action $request_n()$, where n is the unique identifier assigned to each node of the network. Upon receiving the request, *Nagent* communicates its neighbor V_n and the interarrival time Δt to this mobile agent via the message $inform_n(\Delta t, V_n)$. It also adds the time t_a of last visit of this agent. The specification of *Nagent* is as follows:

Node Agent : Nagent

Signature

Input:

$request_n()$, the node agent receives the request from the mobile agent

initiallement : $\{request_n()\}$, initially, the node agent waits the mobile agent requests

Output:

$inform_n(\Delta t, V_n)$, the agent communicates its neighbor set V_n and the interarrival time Δt

States

$n \in Noeuds$, the node address

$V_n \subseteq Noeuds$, the neighbors of the node

$t_a \in \mathbb{R}^+$, the time of the last visit to this node

Actions

Input $request_n()$

Eff: $t_b \leftarrow date()$,
$out \leftarrow \{inform_n(\Delta t, V_n) : \Delta t = (t_b - t_a)\}$;

Output $inform_n(\Delta t, V_n)$

Eff: $t_b \leftarrow t_a$

Unlike *Nagent*, which are stationary agents, *Magent* are mobile agents with the following state:

- The current agent address $location$, initially, it corresponds to the address of the initiator node.

- A Boolean $initiator$ initialized at $TRUE$ to indicate that the agent is belonging to the initial population.
- The real value A_c corresponds to $clone$ action.
- The real value A_m corresponds to $move$ action.
- The real value A_k corresponds to $kill$ action.
- The iteration number $IterMax$.
- The dissipation factor k_c, k_m, and k_k representing the antibody's natural death of the behavior $clone$, $move$, and $kill$, respectively.

The specification of *Magent* is as follows:

Mobile Agent: Magent

Signature

Input:

$inform_n(\Delta t, V_n)$, the agent receives the response to its request $request_n()$

$move(n, n')$, the agent moves from the node n to the node n' randomly selected from V_n

initialement:$\{move(n, n')\}$, initially, upon its creation the agent should execute this action

Output:

$request_n()$, the mobile agent sends its request to the node agent n

Internal:

clone(Magent), the operation that allows the agent to clone itself with *initiator* variable is set to *false*

States:

$location \in Noeuds$, localization of *Magent*, initially the initiator node c

$A_c \in \mathbf{R}$, the concentration value of *clone* action, initially 0

$A_m \in \mathbf{R}$, the concentration value of *move* action, initially 0

$A_s \in \mathbf{R}$, the concentration value of *kill* action, initially 0

$initiator \in Bool$, boolean, initially *False* if the agent is a clone, *True* overwise

$IterMax \in \mathbf{N}$, the maximum number of iterations

Actions:

Input $inform_n(\Delta t, V_n)$
Eff:$loop \leftarrow 1$
  repeat
  update A_c, A_m, A_s
  $loop \leftarrow loop + 1$
until $loop = IterMax$
switch $(Max(A_c, A_m, A_s))$
case A_c:
  $int \leftarrow \{\text{clone(Magent)}\}$; $int \leftarrow \{move(n, n')\}$;
case A_m:
  $int \leftarrow \{move(n, n')\}$;
case A_s:
  $int \leftarrow \emptyset$; $in \leftarrow \emptyset$; $out \leftarrow \emptyset$;
end switch

Input $move(n, n')$
  Eff: $location \leftarrow n'$
    $out \leftarrow request_{n'}()$;
  $in \leftarrow \{inform_{n'}(\Delta t, V_{n'})\}$;

Output $request_n()$
  Pre: $location = n$
  Eff: none

Each mobile agent *Magent* move from one node to another node in the network, querying each node n using $request_n()$ action to obtain its neighbor V_n and the inter-arrival time Δt. Upon receiving these information via $inform_n(\Delta t, V_n)$ action, the behaviors $clone$, $move$, and $kill$ are stimulated with the affinity value m_c, m_m, and m_k, respectively. Each behavior concentration is calculated repeatedly and the behavior with the highest concentration is selected. The $move(n, n')$ action corresponds to the agent migration, where n' is the neighbor of the node n randomly selected from the set V_n. The $clone(Magent)$ action corresponds to cloning operation. When an agent *Magent* is cloned, its clone agent moves to a node randomly selected.

4. SIMULATION RESULTS

In this section, the effectiveness of the proposed approach is demonstrated using simulations, which were conducted using the ns2 simulator.[36,37] Two network topologies composed of 100 nodes were generated using Brite tool.[38] The first network topology was generated according to the Waxman model. This model refers to a generation model for a random topology using Waxman's probability model for interconnecting the nodes of the topology, which is given by $P(u, v) = \alpha e^{-d/(\beta L)}$, where $0 \leq \alpha, \beta \leq 1$, d is the Euclidean distance from node u to node v, and L is the maximum distance between any two nodes (α=0.15, β=0.2 in the generated topology). In the generated topology, the number of bidirectional links is 201, the average link bandwidth is 19.97 Mbps and the average node degree is 4.

The second network topology composed of 100 nodes was generated according to BA model. This model is based on the gradual increase in the network size by continual addition of new nodes. More precisely, at generation time, when a node i is added to the network, the probability to connect it to a node j already belonging to the network is $P(i, j) = d_j / \sum_{k \in V} d_k$, where d_j is the degree of the target node, V is the set of nodes that have joined the network, and $\sum_{k \in V} d_k$ is the sum of outdegrees of all nodes that have previously joined the network. In this topology, the number of bidirectional links is 198, the average bandwidth is 30.14 Mbps, and the node degree is 4.

In these two topologies, links' bandwidth can be assigned to links using the following distributions: *uniform*, *exponential,* and *heavy*. In the first one, a value uniformly distributed between $BW\text{min}$ and $BW\text{max}$ is selected. In the second distribution, a value exponentially distributed with mean $BW\text{min}$ is selected. In the third distribution, a value heavy-tailed distributed (Pareto with shape 1.2) with minimum value $BW\text{min}$ and maximum value $BW\text{max}$ is selected. In the simulations, we fixed the value of $BW\text{min}$ to be 10 Mbps and the value of $BW\text{max}$ to 1024 Mbs. Furthermore, simulation results are produced by considering these three distributions to generate the links' bandwidth. These two topologies have the same number of nodes and slightly number of bidirectional links, but the degree distribution (nodes connectivity) among nodes is different.

Figure 3 shows the evolution of dynamic agent population size during the simulation using the ant-based approach when uniform, heavy, and exponential distributions are applied and the topology is generated using the Waxman model.

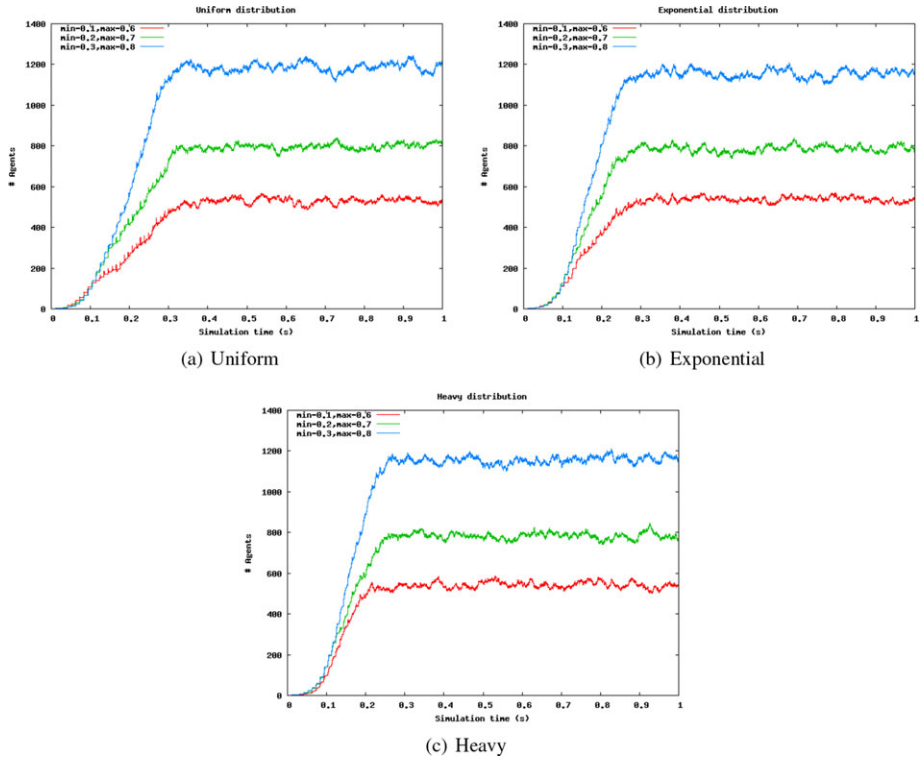


Figure 3. The evolution of the population of agents for uniform, exponential, and heavy distributions using the ant-based approach.

As shown in this figure, by changing the values of Min and Max, the population size changes and mainly depends on these global values that are fixed a priori for each node. We can also see that bandwidth assigned to links using the uniform, exponential, and heavy distributions has no influence on the total population size.

Figure 4 shows the evolution of dynamic agent population size during the simulation using the immune-based approach when the uniform, heavy, and exponential distributions are applied and the topology is generated using the Waxman model. We have also considered three scenarios, similar, fixed, and random. In the first scenario, when an agent clone itself the clone takes the same values of the parameters A_c, A_m, andA_k. In the second scenario, a fixed value for these parameters was given, 0.5 in this simulation. In the third scenario, random values are generated between 0.5 and 1.0. We can see that the population size is still similar for the three scenarios and the three models used for bandwidth generation.

As shown in Figures 3 and 4, unlike the ant-based approach, in the immune-based approach, the population size converges without using any global values. Agents learns while moving in the network and select the suitable action based on the interarrival time between agents.

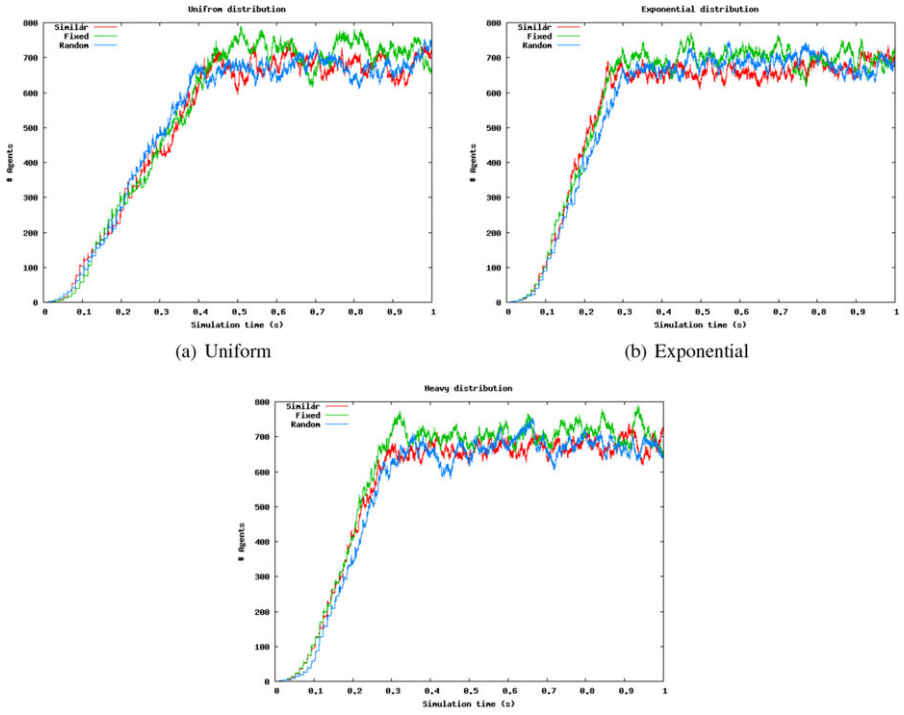
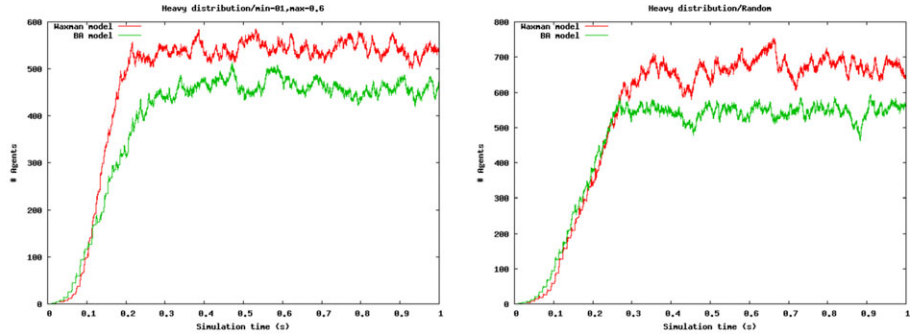


Figure 4. The evolution of the population of agents for uniform, exponential, and heavy distributions using the immune-based approach.

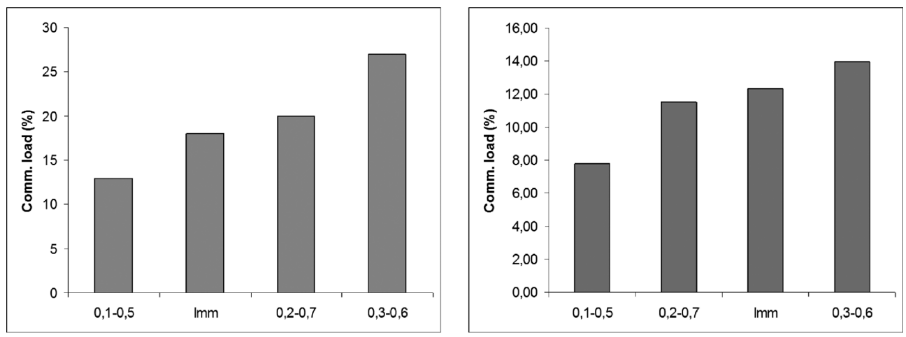
To show the influence of the topology on the agent population size, another topology composed of 100 nodes was generated according to BA model. Figure 5 shows the evolution of the population size when using two topologies that are generated using Waxman and BA models, respectively. As depicted in the figure, the size of the mobile agent population size when the Waxman model is used is greater than the size of mobile agent population when using the BA model. Notice that similar simulations have been conducted using uniform and exponential distributions and other values of min and max but have not presented because they show similar results like the heavy distribution. To figure out the reason why this is happened, we calculate the nodes' degree for each topology and the standard deviation, which are 3.59 and 2.68 for the topologies generated by BA and Waxman models, respectively. We concluded that BA model generates a network topology that contains relatively more nodes with high connectivity. Let us, for instance, consider the network node with the highest degree (i.e., the largest number of neighbors). This node, by its particular situation within the network topology, will be visited more frequently than other nodes of the network. Therefore, the interval δt between two successive visits could be very small, which leads these nodes to make decisions to kill mobile agents while the network may not be loaded.



(a) Ant-based approach: min and max are fixed to 0.1 and 0.6

(b) Immune-based approach: random strategy

Figure 5. The evolution of the population of agents when the heavy distribution is applied. The topology was generated using BA and Waxman models.

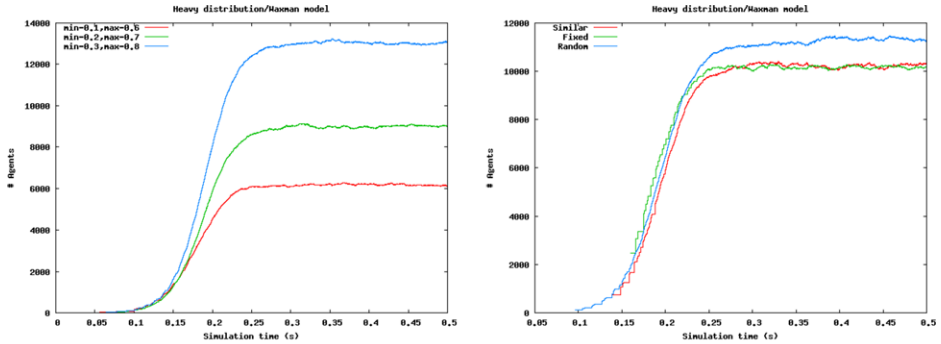


(a) Waxman model

(b) BA model

Figure 6. The communication load for ants and immune approaches. The topology was generated using BA and Waxman models.

We have also computed the communication load as depicted in Figure 6. The communication load is a relative value of arrival rate versus departure rate on all links and obtained as follows. Let us consider $\mathcal{D}_r$ as the maximum number of agents that can possibly, under ideal circumstances, be transmitted over all links during the simulation time $(t_e - t_s)$. $\mathcal{A}_r$ is the actual number of agents that have arrived over all links during this period. The communication load $\mathcal{L}$ can be defined as the ratio between the departure rate $\mathcal{D}_r$ and the arrival rate $\mathcal{A}_r$ as follows: $\mathcal{L} = \frac{S_a}{t_e - t_s} \sum_{i=1}^{N_\ell} \frac{A_{\ell_i}}{R_{\ell_i}}$, where A_{ℓ_i} is the number of agents arrived to the link ℓ_i, R_{ℓ_i} is the bandwidth of each link ℓ_i, N_ℓ is the number of unidirectional links, and S_a is the size of the each agent. The actual number of agents arrived to each link was obtained by monitoring links during the simulation, but the number of agents that can be transmitted was calculated based on the bandwidth value of each link (400 unidirectional links in the topology generated) and the size of agents (fixed to 1028 bytes).



(a) Ant-based approach (b) Ant-based approach

Figure 7. The evolution of the population of agents. The network with 1000 nodes generated using the Waxman model and the heavy distribution.

Figure 6 shows that the communication load changes as we change Min and Max values because of the number of agents in the network. Furthermore, the communication load in the network topology generated by the BA model is slightly smaller than the Waxman model because of the decrease in the number of agents in the network and the greater values of links bandwidth. In the immune approach, this decrease is due only on the links bandwidth values, which is greater for BA model (31 Mbps on average). According to these results, we conclude that the population size depends crucially on the topology structure given in a network.

To show the influence of the network size, we computed as shown in Figure 7 the evolution of dynamic agent population size during the simulation for both approaches in a network of 1000 nodes. The heavy distribution was applied to generated bandwidth values related to links. As illustrated above, these results show that the population size depends on these global values that are fixed a priori for each node. Furthermore, for both approaches the total population size increased significantly and adapts to the nodes increases.

5. CONCLUSIONS AND FUTURE WORK

Regulating a mobile agent population size in a dynamically changing environment is particularly challenging. This paper describes an adaptive immune-inspired approach for dynamic regulation of mobile agent population size in a distributed network. In this approach, an agent retrieves local information from its environment and makes appropriate decision to either kill, move without cloning or clone itself, and move to another node of the network. With this approach, the number of agents in the network stabilizes without having a global knowledge of the network. Other approaches are under investigation, and more simulations will be conducted. Analytical performance analysis to study the behavior of the composed automata of *Nagent* and *Magent* to prove safety and liveness properties is our ongoing work.

References

1. Weiser M. Some computer science issues in ubiquitous computing. Commun ACMs 1993;36:74–84.
2. Gaber J. Spontaneous emergence model for pervasive environments. Globecom Workshops, Washington, DC; 2007. pp 1–6.
3. Gaber J–. New paradigms for ubiquitous and pervasive applications. In First Workshop on Software Engineering Challenges for Ubiquitous Computing, Lancaster, UK; July 2006.
4. Bakhouya M, Gaber J. Bio-inspired approaches for engineering adaptive systems. 5th Int Conf on Ambient Systems, Networks and Technologies (ANT-2014). Procedia Computer Science 2014;32:862–869.
5. Caro GD, Dorigo M. Antnet: distributed stigmergetic control for communications networks. J Artif Intell Res 1998;9:317–365.
6. Cohen IR. Real and artificial immune systems: computing the state of the body. Nat Rev Immunol 2007;7:569–574.
7. Moore M, Suda T. A decentralized and self-organizing discovery mechanism. In: Proc First Annual Symposium on Autonomous Intelligent Networks and Systems, University of California, Los Angeles, CA; May 2002.
8. Bakhouya M. Self-adaptive approach based on mobile agent and inspired by human immune system for service discovery in large scale networks, PhD Thesis, Universite de Technologie de Belfort-Montbeliard; December 2005.
9. Baala H, Flauzac O, Gaber J, Buid M, El-Ghazawi T. A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks. J Parallel Distrib Comput 2003;63(1):97–104.
10. Amin KA, Mikler A. Dynamic agent population in agent-based distance vector routing. In: Second Int Workshop on Intelligent Systems Design and Applications, Atlanta, GA; August 2002.
11. Maes P. Modelling adaptive autonomous agents. Artif Life 1994;1(1):135–162.
12. Sigaud O. Automatisme et subjectivite: l'anticipation au coeur de l'experience, PhD Thesis, Universite Paris I, spécialite Philosophie; 2002. Available at http://www.risc.cnrs.fr/detail_memt.php?ID=555, accessed June 2012.
13. Bakker B, van der Voort G. Trading of perception with internal state: Reinforcement learning and analysis of q-elman networks in a markovian task. In: Proc Int Joint Conf on Neural Networks, Como, Italy; 2000. Vol III, pp 213–218.
14. Buffet B. Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs. PhD Thesis, Universite Henri Poincare Nancy 1, 2003. 238 p.
15. Hasinoff SW. Reinforcement learning for problems with hidden state. Research Report, Department of Computer Science, University of Toronto; 2002. Available at http://www.cs.toronto.edu/hasinoff/.
16. Zhang W. Algorithms for partially observable markov decision processes. PhD Thesis; 2001. Available at https://circle.ubc.ca/handle/2429/29073, accessed June 2014.
17. Aberdeen D. A (revised) survey of approximate methods for solving partially observable markov decision processess. Research Report, National ICT Australia; 2003. Available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.64.3376&rep=rep1&type=pdf, accessed June 2014.
18. Heinzemann C. Modeling behavior of self-adaptive systems; 2012. Available at www2.cs.uni-paderborn.de/, accessed December 2012.
19. Nieh JC. A negative feedback signal that is triggered by peril curbs honey bee recruitment. Curr Biol 2010;20:310–315.
20. Kholodenko BN. Cell-signalling dynamics in time and space. Rev Mol Cell Biol 2006;7:165–176.
21. Jerne NK. Towards a network theory of the immune system. Ann Immunol 1974;125C:135–162.
22. Farmer JD, Packard NH, Perelson A. The immune system, adaptation and machine learning. 1986;22D: 72–81.

23. Detrain C, Deneubourg J-L. Self-organized structures in a superorganism: do ants behave like molecules. Phys Life Rev 2006;3(3):162–187.
24. Watanabe Y, Ishiguro A, Shirai Y, Uchikawa Y. Emergent construction of behavior arbitration mechanism based on the immune system. Adv Robotics 1998;12(3):227–242.
25. Cheng BHC, de Lemos R, Giese H, Inverardi P, Magee J. Software engineering for self-adaptive systems: a research roadmap. LNCS 2009;5525:1–26.
26. Brun Y, Serugendo GDM, Gacek C, Giese H, Kienle MLH, Muller H, Pezze M, Shaw M. Engineering self-adaptive systems through feedback loops. Software Engineering for Self-Adaptive Systems. Lect Notes Comput Sci 2009;5525:48–70.
27. Kephart JO, Chess DM. The vision of autonomic computing. Computer 2003;36(1):41–50.
28. Ishiguro A, Watanabe Y, Kondo T, Uchikawa Y. Proposal of decentralized consensus-making mechanisms based on immune system-application to a behavior arbitration of an autonomous mobile robot. Proc AROB, Oita, Japan; 1996. pp 122–127.
29. Gaber J. Action selection algorithms for autonomous system in pervasive environment: a computational approach. TAAS 2011;6(1):10.
30. Ranjan S, Gupta A, Basu A, Meka A, Chaturvedi A. Adaptive mobile agents: modeling and a case study. 2nd Workshop on Distributed Computing, Calcutta, India; 2000; 238 p. Available at http://www-ece.rice.edu/sranjan/publications/WDC.doc.
31. Cabri G, Leonardi L, Zambonelli F. Mobile agent technology: current trends and perspectives. Congresso annuale AICA, Napoli, Italy; 1998. Available at http://www.agentgroup.unimo.it/MOON/papers/pdf/aica98.pdf.
32. Holland J. Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In: Luger GF, editor. Computation and intelligence: collected readings. Menlo Park, CA: American Association for Artificial Intelligence; 1995. pp. 275–304.
33. Suzuki J, Yamamoto Y. Building an artificial immune network for decentralized policy negotiation in a communication endsystem: openwebserver/inexus study. In: Proc 4th World Multiconference on Systemics, Cybernetics and Informatics, Orlando, FL; 2000.
34. Watanabe Y, Ishiguro A, Uchkawa Y. Decentralized behavior arbitration mechanism for autonomous mobile robot using immune system. Artificial immune systems and their applications. Springer-Verlag, Berlin; 1999. pp 186–208.
35. Lynch NA, Tuttle MR. Hierarchical correctness proofs for distributed algorithms. In: PODC '87: Proc Sixth Annual ACM Symposium on Principles of distributed computing. ACM: New York, NY; 1987. pp 137–151.
36. NS2. The network simulator. Available at http://www.isi.edu/nsnam/ns/.
37. Wittner O. Ant-like mobile agents: ns2 patch. Available at http://www.matlab.nitech.ac.jp/khpoo/research/ant.htm.
38. Medina A, Lakhina A, Matta I, Byers J. Brite: universal topology generation from a user's perspective. Research report N BUCS-TR-2001-00, 2001.