Self-Adaptive Mobile Agent Population Control in Dynamic Networks Based on the Single Species Population Model

Tomoko SUZUKI\(^{(a)}\), Student Member, Taisuke IZUMI\(^{(†)}\), Fukuhito OOSHITA\(^{(†)}\), and Toshimitsu MASUZAWA\(^{(†)}\), Members

SUMMARY Mobile-agent-based distributed computing is one of the most promising paradigms to support autonomic computing in a large-scale of distributed system with dynamics and diversity: mobile agents traverse the distributed system and carry out a sophisticated task at each node adaptively. In mobile-agent-based systems, a larger number of agents generally require shorter time to complete the whole task but consume more resources (e.g., processing power and network bandwidth). Therefore, it is indispensable to keep an appropriate number of agents for the application on the mobile-agent-based system. This paper considers the mobile agent population control problem in dynamic networks: it requires adjusting the number of agents to a constant fraction of the current network size. This paper proposes algorithms inspired by the single species population model, which is a well-known population ecology model. These two algorithms are different in knowledge of networks each node requires. The first algorithm requires global information at each node, while the second algorithm requires only the local information. This paper shows by simulations that the both algorithms realize self-adaptation of mobile agent population in dynamic networks, but the second algorithm attains slightly lower accuracy than the first one.

key words: mobile agent, mobile agent population control, dynamic network, self-adaptation, single species population model

1. Introduction

Continuous and significant increase in scale, dynamics and diversity of network environments requires highly-adaptive distributed systems. Thus, autonomic distributed systems with self-* properties, such as self-adaptation, self-configuration, self-optimization and self-healing, are attracting widespread attention from researchers and engineers in the field of distributed systems.

Mobile-agent-based distributed computing is one of the most promising paradigms to support autonomic computing in a large-scale of distributed system with dynamics and diversity [1], [2]. Mobile agents are autonomous programs that can migrate from one node to another on the network, and traverse the distributed system to carry out a sophisticated task at each node. Since the adaptability and flexibility of mobile-agents simplifies the design of highly-adaptive distributed systems, several mobile-agent-based distributed systems have been proposed and developed. Network management systems [3]–[5] are successful examples of mobile-agent-based distributed systems: agents traverse the network to collect load information of nodes and links, and adjust their load adequately.

In such mobile-agent-based distributed systems, generally, a larger number of agents require shorter time to complete the whole task, but consume more resources (e.g., processing power and network bandwidth). Therefore, it is indispensable to keep an appropriate number of agents for the application on the mobile-agent-based system. For example, it is claimed, for network management system MARP, the appropriate number of agents is a half of the number of nodes in the network [3].

However, in dynamic networks, adjustment of agent population (i.e., the number of agents) for dynamical change of network size (i.e., the number of nodes) is not an easy task and requires investigation. Since most mobile-agent-based distributed systems (including MARP) have no facility for adjusting the agent population, developing a general mechanism for adjusting the agent population is invaluable.

Biological systems inherently have self-* properties to realize environmental adaptation. Thus, several biologically inspired approaches have succeeded in realizing highly adaptive distributed systems. Successful projects include Bio-Networking project [6] and AntNet project [7]. These projects adopt biologically inspired approaches to provide highly adaptive platform for mobile-agent-based computing [8], [9]. Concerning control of agent population, Kaizar et al. [10] proposed a biologically inspired method in agent-based distance vector routing system: each agent deposits small data (called pheromone) on the node it visits. The agent population is controlled depending on concentration of the pheromone. However the paper does not clarify the influence of parameters on the resultant population, and thus, it is not expectable how many agents exist in the network after execution with some parameter setting.

Contribution of this paper. In this paper, we first formulate the mobile agent population control problem in dynamic networks, and present biologically inspired solutions for the problem. The mobile agent population control problem requires adapting the number of agents (called agent population) to a given constant fraction of the current number of nodes (called network size).

We propose two distributed solutions for the problem. To realize self-adaptation of the agent population to the net-
work size, we borrow an idea from the single species population model, which is a well-known population ecology model. This model considers population of a single species in an environment such that individuals of the species can survive by consuming food supplied by the environment. The model is formulated by the logistic equation and shows that the population automatically converges to and stabilizes at some number depending on the amount of supplied food.

In the proposed algorithms, agents are regarded as individuals of a single species, and nodes supply food for agents. The algorithms try to adjust the agent population to a constant fraction of the network size by controlling the amount of food supplied at each node. Thus, the crucial point of the proposed algorithms is the method to determine the amount of food supplied at each node.

The amount of food supplied at each node should be determined from a frequency of agents’ visits at the node. To validate the claim, under the assumption that every agent traverses the network by random walk, the proposed algorithms determine the amount of supplied food from the stationary probability of random walk. The simulation results of the first algorithm show that the proposed strategy can adequately adjust the agent population. In addition, we also investigate the case that multiple kinds of agents coexist in the system. In this case, by applying the proposed strategy to each kind of agents, we can successfully control each agent population.

A debatable point of the first algorithm is that each node uses the link density of the network to determine the amount of supplied food since the stationary probability of random walk depends on the link density. Distributed solutions generally prefer to avoid using such global information as the link density. Thus, we propose the second solution to tackle this problem: in the second solution, each node uses local estimation of the link density that is calculated from information of nodes in its neighborhood. The simulation results show that the second solution can sufficiently adjust the agent population using only the local information, but with slightly lower accuracy than the first solution.

The rest of this paper is organized as follows. In Sect. 2, we present the model of distributed systems, and define the mobile agent population control problem. In Sects. 3 and 4, we propose the first distributed solution for the problem and show its simulation results. The modified algorithm is presented and simulated in Sect. 5. Section 6 concludes the paper.

2. Preliminaries

2.1 System Models

Dynamic networks. In this paper, we consider dynamic networks such that its node set and its link set vary with time. To define dynamic networks, we introduce discrete time and assume that each time is denoted by a non-negative integer in a natural way: time 0 denotes the initial time, time 1 denotes the time immediately following time 0 and so on.

Formally, a dynamic network at time $t$ is denoted by $N(t) = (V(t), E(t))$ where $V(t)$ and $E(t)$ are respectively the node set and the link set at time $t$. A link in $E(t)$ connects two distinct nodes in $V(t)$ and a link between nodes $u$ and $v$ is denoted by $e_{uv}$ or $e_{vu}$. We also use the following notations to represent the numbers of nodes and edges at time $t$: $n(t) = |V(t)|$ and $e(t) = |E(t)|$.

Mobile agent systems. A mobile agent is an autonomous program that can migrate from one node to another on the network. In dynamic networks, agents on node $u \in V(t)$ at time $t$ can start migrating to node $v \in V(t)$ only when link $e_{uv}$ is contained in $E(t)$. The agent reaches $v$ at time $t + \delta$ only when the link $e_{uv}$ remains existing during the period from $t$ to $t + \delta$, where $\delta$ is an integer representing migration delay between the nodes. The agent migrating from $u$ to $v$ is removed from the network when the link $e_{uv}$ disappears during the period from $t$ to $t + \delta$.

Each of nodes and agents has a local clock that runs at the same rate as the global time. However, we make no assumption on the local clock values: the difference between the local clock values in the system is unbounded.

An agent and a node can interact with each other by executing operations: agent $p$ on node $u$ can change its state and the state of $u$ depending on the current states of $p$ and $u$, and node $u$ can change its state and the states of the agents residing on $u$ depending on the current states of $u$ and the agents. Besides the above operations, each agent can execute operations to create new agents and to kill itself and each node can also execute operations to create new agents.

When agents reside on a node, the agents and the node may have operations they can execute. For execution semantics, we assume that the agents and the node execute their operations sequentially in an arbitrary order. We also assume that the time required to execute the operations can be ignored, that is, we consider all the operations are executed sequentially but at an instant time.

Concerning the migration patterns of agents, we assume that each agent makes a random walk independently: an agent migrates from one node to one of its neighboring nodes with equal probability. In real agent systems, migration patterns of agents differ from application to application. In this paper, however, we focus on a certain control of agents (population control defined in the next subsection) rather than the actual applications invoked by the agents. Thus, we adopt random walks as the migration patterns since we do not want to consider a particular application. It is worthwhile to mention that a random walk is one of the most typical migration patterns, and that some real agent systems adopt random walk as their migration patterns [3].

2.2 Mobile Agent Population Control

In this paper, we consider the mobile agent population control problem. The goal of the problem is to control the number of agents so that the ratio between the number of
agents (called \textit{agent population} hereinafter) and the number of nodes (called \textit{network size} hereinafter) is kept to be a given constant. The problem is defined as follows. In the definition, let \(a(t)\) be the agent population on the network \(N(t)\) at time \(t\).

\textbf{Definition 2.1:}  
The goal of the mobile agent population control problem is to adjust the agent population \(a(t)\) at time \(t\) to satisfy the following equality for a given constant \(\gamma\) (0 < \(\gamma\) < 1).

\[ a(t) = \gamma \cdot n(t) \]

In this paper, we propose distributed solutions for the agent population control problem. In the distributed solutions, we assume that the constant \(\gamma\) is initially given to every node.

We consider distributed systems such that mobile agents are distributed over the networks and nodes can leave from or join in the networks. In such environment, it is obviously impossible to keep satisfying the above equation all the time. Thus, our goal is to propose distributed solutions that realize quick convergence to and stability at the target population.

3. Agent Population Control Algorithm

In this section, we present a distributed solution for the mobile agent population control problem. This algorithm is inspired by the single species population model (the logistic model), which is well-known in the field of the population ecology.

3.1 Single Species Population Model

In this subsection, we introduce the \textit{single species population model} in the population ecology as the basis of our algorithm. This model considers an environment with a single species such that individuals of the species can survive by consuming food supplied by the environment. The model formulates the population growth of the species in the environment, and shows that the population (i.e., the number of individuals) in the environment automatically converges to and stabilizes at some number depending on the amount of food supplied by the environment.

We present more details of the single species population model. Each individual of the species periodically needs to take a specific amount of food to survive. That is, if an individual can take the specific amount of food then it can survive. Conversely, if an individual cannot take the specific amount of food then it dies. Moreover, in the case that an individual can take a sufficient amount of extra food, then it generates progeny. Consequently, the followings hold: The shortage of supplied food results in decrease in the population. Conversely, the excessive amount of food results in increase in the population.

The single species population model formulates the above phenomena as follows: Let \(p(t)\) be the population at time \(t\). The single species population model indicates that the population growth rate at time \(t\) is represented by the following nonlinear first-order differential equation known as the logistic equation [11]:

\[ \frac{\Delta p(t)}{\Delta t} = p(t) \cdot r(t) = p(t)(k \cdot f_a(t) - f \cdot p(t)), \]

where \(f_a(t)\) is the amount of food supplied by the environment at time \(t\), \(f\) is the amount of food consumed by one individual to survive, and \(k\) is a positive real constant.

The \textit{per capita growth rate} \(r(t)\) at time \(t\) is represented by

\[ r(t) = k(f_a(t) - f \cdot p(t)). \]

The expression \(f_a(t) - f \cdot p(t)\) represents the difference between the amounts of supplied food and consumed food. When the supplied food exceeds the consumed food, \(r(t)\) takes a positive value proportional to the difference, that is, the positive per capita growth rate \(r(t)\) proportional to the amount of the surplus food. A scarcity of the supplied food causes a negative value of \(r(t)\) proportional to the difference, that is, the negative per capita growth rate \(r(t)\) proportional to the shortage of the supplied food.

The logistic equation has two equilibrium points of the population size: \(p(t) = 0\) and \(p(t) = f_a(t)/f\). That is, the population remains unchanged, when the population size is at the equilibrium points. The equilibrium point \(p(t) = f_a(t)/f\) represents the maximum population that the environment can keep, and is called the \textit{carrying capacity} of the environment.

If the population is larger (resp. smaller) than the carrying capacity then the population decreases (resp. increases). Once the population reaches the carrying capacity, then it remains unchanged (see Fig. 1). Consequently, the single species population model implies that the population eventually converges to and stabilizes at the carrying capacity.

Notice that the carrying capacity depends on the amount of food supplied by the environment.

3.2 Algorithm for Mobile Agent Population Control

In this subsection, we present an algorithm for the mobile agent population control problem. The algorithm is inspired by the single species population model: agents are regarded
as individuals of a single species, and a network is regarded as an environment. That is, agents need to consume food to survive and the food is supplied by nodes of the network.

The goal of the mobile agent population control problem is to adjust the agent population \( a(t) \) to \( \gamma \cdot n(t) \). Remind that the single species population model shows that the number of individuals converges to and stabilizes at the carrying capacity. Thus, the algorithm tries to adjust \( a(t) \) to \( \gamma \cdot n(t) \) by adjusting the carrying capacity to \( \gamma \cdot n(t) \).

In the algorithm, we introduce time interval of some constant length denoted by \( CYCLE \). Behavior of each node and each agent can be divided into a series of the time intervals. It should be noticed that the start time of the intervals at different nodes or agents are not synchronized: a node or an agent may start a new time interval while another is in the middle of its time interval.

The primary behavior of nodes and agents is simple: each node supplies food every \( CYCLE \) time units (i.e., at the beginning of each time interval). Each agent traverses the network by a random walk and consumes the visited nodes have. The agent can survive into the next time interval if it can take a specific amount of food, denoted by \( AGT\_FOOD \), during the current time interval. The agent kills itself (i.e., removes itself from the network) if it cannot get food of amount \( AGT\_FOOD \) during the time interval.

In addition, each agent creates new agents by consuming surplus food of amount \( AGT\_FOOD \). This idea derives from the fact that the positive per capita growth rate \( r(t) \) in the single species population model is proportional to the amount of surplus food.

Figure 2 shows the detailed behavior of nodes and agents in the mobile agent population control algorithm.

In what follows, we explain some technical parts of the node’s behavior. Each node supplies an appropriate amount of food every \( CYCLE \) time units. Now, we consider the amount of food \( f(v, t) \) that node \( v \) should supply at time \( t \). As stated in the above, the algorithm tries to adjust the carrying capacity to \( \gamma \cdot n(t) \). The single species population model shows that the carrying capacity is represented by \( f_0(t) \). Since \( f_0(t) \) corresponds to the total amount of supplied food on the whole network \( \sum_{v \in V(t)} f(v, t) \) and \( f \) corresponds to the amount of food \( AGT\_FOOD \) consumed by an agent to survive, the following equation should be satisfied:

\[
\sum_{v \in V(t)} f(v, t) = \gamma \cdot n(t) \cdot AGT\_FOOD.
\]

If each node supplies the same amount of food, \( \gamma \cdot AGT\_FOOD \), the above equation can be easily satisfied. However, since each agent makes a random walk, frequencies of agents’ visits vary from node to node and, thus, such uniform allocation of food over the network cannot adjust the agent population appropriately: nodes with greater frequencies face the food shortage and nodes with less frequencies have surplus food that is never consumed. To resolve this problem, the amount of food each node supplies should be proportional to the frequencies of agents’ visits. Theory of Markov chain shows that the stationary probability \( q(v) \) of an agent at node \( v \) is represented by \( q(v) = deg_v / (2 \cdot e(t)) \), where \( deg_v \) is the number of links connecting to \( v \) (called degree of \( v \) hereinafter). Therefore, node \( v \) should supply food of amount

\[
q(v) \cdot \gamma \cdot n(t) \cdot AGT\_FOOD
\]

at every \( CYCLE \) time units.

When a node supplies new food and goes to the next time interval, the node stores the surplus food of the current time interval into variable \( surplus\_food \). The surplus food is consumed by agents to create new agents.

Another problem we should consider is the total extinction of agents. We consider dynamic networks. When a node \( v \) leaves from the network at time \( t \), agents on node \( v \) or on links connecting to \( v \) at time \( t \) are also removed from the network. This may cause the total extinction of agents. To resolve the problem, each node \( v \) creates a new agent if \( v \) is not visited by any agent during a sufficiently long time period. The length of the period should be also determined from the stationary probability \( q(v) \) of an agent. In the algorithm, each node \( v \) creates a new agent when no agent visits \( v \) during \( C/(\gamma \cdot n(t) \cdot q(v)) \) time units, where \( C \) is a constant.

4. Simulation Results

In this section, we present simulation results to show that the proposed algorithm can effectively adjust the agent population.

In the simulation, we assume that each agent repeatedly executes the following actions: each agent stays at a node for one time unit, and then migrates to one of its neighboring nodes by a random walk. We also assume that the migration delay between any pair of neighboring nodes is two time units. The following values are initialized randomly:

- the initial locations of agents
- the initial values of the local clocks (i.e., \( time_v \), \( time_p \))
- the initial amounts of food that nodes have (i.e., \( food_v \))
- the initial amounts of food that agents have consumed in the current time interval (i.e., \( eat\_food_p \))

The initial amounts of surplus food that nodes have (i.e., \( surplus\_food_v \)) and the initial amounts of surplus food that agents have consumed in the current time interval (i.e., \( eat\_surplus\_food_p \)) are set to 0. To resolve the total extinction of agents, each node \( v \) creates a new agent when no agent visits \( v \) during \( 50/(\gamma \cdot n(t) \cdot q(v)) \) time units.

We present the simulation results for random networks and scale-free networks. Scale-free networks are a specific kind of networks such that some nodes have a tremendous number of connections to other nodes, whereas most nodes have just a handful. The degree distribution follows a power law of \( k \): the number of nodes with degree \( k \) is proportional to \( k^{-r} \), where \( r \) is a positive constant. A scale-free network is said to be a realistic model of actual network structures [12], [13].

Simulation results for static networks. Figure 3 shows
Behavior of node $v$

- $\text{food}_v$: the amount of food
- $\text{surplus}_v\text{food}_v$: the amount of surplus food
- $\text{time}_v$: local clock time
  
  */ its value automatically increases at the same rate as the global time */

- at the beginning of each time interval (i.e., when $\text{time}_v \mod \text{CYCLE} = 0$ holds)
  
  $\text{surplus}_v\text{food}_v := \text{food}_v$
  $\text{food}_v := \frac{\text{mod}}{\text{q}} \cdot y \cdot n(t) \cdot \text{AGT}\text{FOOD}$

- on suspicion of the total extinction (i.e., when no agent visits $v$ during the last $\frac{\text{q}}{\text{y} \cdot \text{mod}(v)}$ time units)
  
  create one agent

Behavior of agent $p$

- $\text{eat}_v\text{food}_p$: the amount of food that $p$ consumes from $\text{food}_v$ of nodes
- $\text{eat}_v\text{surplus}_v\text{food}_p$: the amount of food that $p$ consumes from $\text{surplus}_v\text{food}_v$ of nodes
- $\text{time}_p$: local clock time
  
  */ its value automatically increases at the same rate as the global time */

- when $p$ is created

  $\text{eat}_v\text{food}_p := 0.0$
  $\text{eat}_v\text{surplus}_v\text{food}_p := 0.0$
  $\text{time}_p := 0$

- on arrival at node $v$
  
  */ if there are two or more agents on the same node $v$, the agents run the following processes independently in an arbitrary order */

  if ($\text{eat}_v\text{food}_p < \text{AGT}\text{FOOD}$) then
    
    $y := \min[\text{AGT}\text{FOOD} - \text{eat}_v\text{food}_p, \text{food}_v]$
    $\text{eat}_v\text{food}_p := \text{eat}_v\text{food}_p + y$
    $\text{food}_v := \text{food}_v - y$

  if ($\text{surplus}_v\text{food}_v > 0$) then
    
    $y' := \min[\text{AGT}\text{FOOD} - \text{eat}_v\text{surplus}_v\text{food}_p, \text{surplus}_v\text{food}_v]$
    $\text{eat}_v\text{surplus}_v\text{food}_p := \text{eat}_v\text{surplus}_v\text{food}_p + y'$
    $\text{surplus}_v\text{food}_v := \text{surplus}_v\text{food}_v - y'$

  if ($\text{eat}_v\text{surplus}_v\text{food}_p = \text{AGT}\text{FOOD}$) then
    
    create one agent
    $\text{eat}_v\text{surplus}_v\text{food}_p := 0$

- on departure from node $v$ */ when $p$ completes the application task at $v$ */

  migrate to one of the neighboring nodes

- at the end of each time interval (i.e., when $\text{time}_p \mod \text{CYCLE} = 0$ holds)

  if ($\text{eat}_v\text{food}_p < \text{AGT}\text{FOOD}$) then kill itself
  else $\text{eat}_v\text{food}_p := 0$ /* $p$ survives into the next time interval */

Fig. 2 Behavior of node $v$ and agent $p$.

the experimental results for “static” random networks and “static” scale-free networks where nodes and links of the networks remain unchanged. In the simulation, the length $\text{CYCLE}$ of the time interval is set to 200, and the number $n(t)$ of nodes is fixed at 500 during the simulation. Random graphs with $n$ nodes are generated as follows: each pair of nodes is connected with probability of $5.0/\left(n - 1\right)$. Scale-free networks are generated using the incremental method proposed by Balabasi and Albert [12]. More precisely, starting with 3 nodes, we add new nodes one by one. When a new node is added, three links are also added to connect the new node to three other nodes, which are randomly selected
with probability proportional to their degrees.

Figure 3 shows transition of the agent population $a(t)$ with time $t$. It shows the simulation results for six combinations of three values of $\gamma$ (0.8, 0.5, and 0.2), and two initial agent populations $a(0)$ (500 and 0). These simulation results show that the agent population quickly converges to the equilibrium point, and has small perturbation after the convergence.

Simulation results for dynamic networks. Figure 4 and Fig. 5 show the experimental results for “dynamic” random networks and “dynamic” scale-free networks where nodes and links of networks vary with time. When a new node joins in the network, the new node is connected to other nodes with probability $5.0 / n(t)$ for each other node on random networks, and the node is connected to three other nodes randomly selected with probability proportional to their degrees on scale-free networks. When a node $v$ leaves
from the network, the links connecting to \( v \) are also removed from the network, and agents on node \( v \) or these links are also removed from the network. So, some network partitions may be caused by leave of nodes, but it is not matter for the proposed algorithm. The algorithm can control agent population in a subnetwork; the agents can traverse and eat food on nodes only in the subnetwork. In the simulation, each node leaves from the network with a constant probability for random networks. For scale-free network, the leave of a hub node having tremendous number of connections is undesirable for us since a lot of partitions will be caused. Therefore, for scale-free network, each node leaves with probability inversely proportional to its degree, that is, node \( v \) leaves from the network with probability \( \text{avg}_\deg \cdot p_t/\deg_v, (0 < p_t < 1) \), where \( \text{avg}_\deg \) is the average degree of the network. So, there is little chance that some hub nodes leave from the network. To show the adaptiveness of the proposed algorithm, Fig. 4 and Fig. 5 also show the difference ratio of the agent population: the ratio is defined by \( |\gamma \cdot n(t) - a(t)|/(\gamma \cdot n(t)) \) and represents the ratio of difference between the adjusted and the target numbers of agents to the target number. (In static networks, the average of the difference ratio is about 0.01.)

Figure 4 shows simulation results for dynamic networks with continuous and gradual changes: some nodes join in the network and some nodes leave from the network constantly. In this simulation, the initial network size \( n(0) \) is 500, and the following dynamical changes occur every 500 time units. In the first half (from time 0 to time 10,000) of the simulation, two new nodes join in the network with probability 0.05 and each node \( v \) leaves from the network with probability 0.01 for random networks and with probability \( \text{avg}_\deg \cdot 0.01/\deg_v \) for scale-free networks. In the second half (from time 10,000 to time 20,000), two new nodes join in the network with probability 1.0 and each node \( v \) leaves from the network with probability 0.0001 for random networks and with probability \( \text{avg}_\deg \cdot 0.0001/\deg_v \) for scale-free networks.

In the simulation results of Fig. 4, the length \( CYCLE \) is set to 200, the value of \( \gamma \) is set to 0.5, and the initial agent population \( a(0) \) is set to 250. Since the difference ratio is kept to be less than 0.07 and does not widely diverge from 0, the simulation results show that the agent population is adaptively adjusted in response to changes in the network size.

Figure 5 shows the simulation results for dynamic networks with drastic changes: in a short term, a large number of nodes leave from the network or join in the network. In this simulation, the initial network size \( n(0) \) is 500, and 200 nodes leave from the network at time 7,000 of the simulation, and 400 nodes join in the network at time 14,000 of the simulation. The leaving nodes are chosen randomly.

In the simulation results of Fig. 5, the length \( CYCLE \) is set to 200, the value of \( \gamma \) is set to 0.5, and the initial agent population \( a(0) \) is set to 250. While the difference ratio widely diverges from 0 immediately after the drastic changes of the network, it quickly converges to the target number. The difference ratio is kept to be less than 0.04 after the convergence.

**Simulation results on lifetime and utilization rate of agents.** The goal of the mobile agent population control problem is to adjust the agent population to a given ratio of the network size. However, from the point of application view, lifetime and utilization rate of agents are also very important. In real applications, agents traverse the network to complete some tasks such as collection and/or distribution of information. Each task arises on each node and should be processed by agents that visit the node. Thus, such applications require that lifetime of agents should be sufficiently long to complete the task, and that each agent should process many tasks efficiently.

Lifetime \( l_{tp} \) of agent \( p \) is defined to be the time length from its creation to its elimination, i.e., \( l_{tp} = t_{dp} - t_{bp} \), where \( t_{dp} \) is the time when \( p \) is removed and \( t_{bp} \) is the time when \( p \) is created. Table 1 shows the average lifetime of agents of ten trials. To focus on the lifetime of agents after convergence of the agent population to the target value, the initial agent population \( a(0) \) is set to the equilibrium point. The simulation results show that lifetime quickly becomes longer when the length of the time interval \( CYCLE \) becomes longer. Therefore, by setting an appropriate value to \( CYCLE \), it is strongly expected that lifetime of each agent becomes sufficiently long.

Next, we show simulation results of utilization rate of agents. Utilization rate of an agent is defined to be the ratio of the actual number of tasks the agent processes to the possible maximum number of tasks that the agent can process. We assume that each agent processes one task on each visit to a node, if the node has tasks. Thus, the maximum number of tasks an agent can process in the interval of \( CYCLE \) time units is \( CYCLE/3 \) since two thirds of the interval is absorbed by migration. The tasks arise on each node at every \( CYCLE \) time units. Since the number of agents should be kept to be \( \gamma \cdot n(t) \), the number of tasks that can be processed by the whole agents during the time interval of \( CYCLE \) time units is \( (\gamma \cdot n(t) \cdot CYCLE)/3 \). Therefore, the number of tasks that arise on node \( v \) at every \( CYCLE \) time units is determined to

<table>
<thead>
<tr>
<th>CYCLE</th>
<th>100</th>
<th>200</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>4767</td>
<td>30102</td>
<td>129273</td>
</tr>
<tr>
<td>500</td>
<td>4906</td>
<td>32996</td>
<td>148427</td>
</tr>
<tr>
<td>1000</td>
<td>4954</td>
<td>35446</td>
<td>173367</td>
</tr>
</tbody>
</table>

Table 1 Lifetime of agents.

b. scale-free networks

<table>
<thead>
<tr>
<th>CYCLE</th>
<th>100</th>
<th>200</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>4703</td>
<td>40152</td>
<td>277297</td>
</tr>
<tr>
<td>500</td>
<td>4772</td>
<td>41421</td>
<td>566194</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( CYCLE )</th>
<th>100</th>
<th>200</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>4703</td>
<td>40152</td>
<td>277297</td>
</tr>
<tr>
<td>500</td>
<td>4772</td>
<td>41421</td>
<td>566194</td>
</tr>
</tbody>
</table>
be \( q(v) \cdot (\gamma \cdot n(t) \cdot CYCLE)/3 \) by a similar reason to that for the amount of food. Each agent traverses the network and process one task the visited node has. Utilization rate \( u_p \) of agent \( p \) for a time interval is defined by \( w_p/(CYCLE/3) \), where \( w_p \) is the number of tasks processed by agent \( p \) during the time interval.

Table 2 shows the average utilization rate of agents of ten trials. The initial agent population \( a(0) \) is set to the equilibrium point. The simulation results show that utilization rate are sufficiently high. When the value of \( \gamma \) is set to more than 0.5, almost all average utilization rate of agent becomes more than 90%.

Besides the simulations on random networks and scale-free networks presented in this section, we did simulations on several other networks such as complete networks, lollipop networks and star networks, and obtained similar results on these networks.

### 5. Modified Algorithm with Estimated Density

In the algorithm presented in Sect. 3.2, each node \( v \) supplies the amount \( q(v) \cdot \gamma \cdot n(t) \cdot AGT.FOOD \) of food every CYCLE time units, where \( q(v) \) is the stationary probability of an agent at node \( v \) and is represented by \( q(v) = deg_v/(2 \cdot e(t)) \). The algorithm assumes that the amount of food supplied at each node \( v \) can be locally computed by \( v \). It seems natural to assume that each node \( v \) initially knows the input parameter \( \gamma \), the constant AGT.FOOD, and its own degree \( deg_v \). However, the numbers of nodes \( n(t) \) and links \( e(t) \) are global information of the network, and thus, it is unrealistic, especially for dynamic networks, to assume that each node initially knows these values. In static networks, \( n(t) \) and \( e(t) \) can be computed by a simple distributed algorithm (e.g., the wave algorithm in [14]). However, computation of \( n(t) \) and \( e(t) \) essentially requires the number of messages proportional to \( n(t) \) because \( n(t) \) and \( e(t) \) are global functions over the whole network. In dynamic networks, the computation requires much greater cost since computation of \( n(t) \) and \( e(t) \) should be repeatedly executed to follow dynamic changes of the network.

In this section, to save the cost (e.g., the number of messages) for computing the amount of food supplied by each node, we propose a method for locally estimating the amount of supplied food, and evaluate its accuracy by simulation. We also show simulation results of the mobile agent population control using the estimated values.

We can see that \( n(t) \) and \( e(t) \) appear in the form of \( n(t)/e(t) \) in the formula that determines the amount of food supplied at node \( v \):

\[
\text{Utilization rate of agents.}
\]

<table>
<thead>
<tr>
<th>( n )</th>
<th>200</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma )</td>
<td>0.2</td>
<td>0.5</td>
<td>0.8</td>
</tr>
<tr>
<td>0.849</td>
<td>0.898</td>
<td>0.898</td>
<td></td>
</tr>
<tr>
<td>0.900</td>
<td>0.915</td>
<td>0.915</td>
<td></td>
</tr>
<tr>
<td>0.916</td>
<td>0.915</td>
<td>0.915</td>
<td></td>
</tr>
</tbody>
</table>

\( a \). random networks

\( b \). scale-free networks

\[
\frac{deg_v/(2e(t)) \cdot \gamma \cdot n(t) \cdot AGT.FOOD}{\text{Utilization rate of agents.}}
\]

The form \( n(t)/e(t) \) is the inverse of the link density (simply called density hereinafter) of the network. From the observation, we can expect that the amount of food supplied by each node can be locally computed, because the density \( e(t)/n(t) \) is expected to be estimated from local information if the network has some uniformity. In the followings, we propose a method for locally estimating the density of the network.

#### Local estimation of the density

To attain locality of the estimation, we concentrate our attention to the methods such that each node \( v \) estimates the density only from information on its \( k \)-neighbors for some small constant \( k \): the \( k \)-neighbors of \( v \) are the nodes at most \( k \) away from the node \( v \). Let \( Nb_k(v, t) \) be the set of \( k \)-neighbors of node \( v \) at time \( t \).

\[
\text{The form } n(t)/e(t) \text{ is the inverse of the link density (simply called density hereinafter) of the network. From the observation, we can expect that the amount of food supplied by each node can be locally computed, because the density } e(t)/n(t) \text{ is expected to be estimated from local information if the network has some uniformity. In the followings, we propose a method for locally estimating the density of the network.}
\]

#### Local estimation of the density

To attain locality of the estimation, we concentrate our attention to the methods such that each node \( v \) estimates the density only from information on its \( k \)-neighbors for some small constant \( k \): the \( k \)-neighbors of \( v \) are the nodes at most \( k \) away from the node \( v \). Let \( Nb_k(v, t) \) be the set of \( k \)-neighbors of node \( v \) at time \( t \).

\[
\text{(link) density of the network } e(t)/n(t) \text{ is represented by } e(t)/n(t) = \frac{\sum_{u \in V(t)} deg_u/(2n(t)) = (1/2) \cdot \sum_{u \in V(t)} deg_u/n(t)}{\text{Utilization rate of agents.}}
\]

The (link) density of the network \( e(t)/n(t) \) is represented by \( e(t)/n(t) = \frac{\sum_{u \in V(t)} deg_u/(2n(t)) = (1/2) \cdot \sum_{u \in V(t)} deg_u/n(t)}{\text{Utilization rate of agents.}} \).

Since \( \sum_{u \in V(t)} deg_u/n(t) \) is the average of degrees over all nodes, the equation leads us to the following straightforward local estimation of the density by node \( v \) at time \( t \):

\[
\frac{1}{2} \cdot \sum_{u \in Nb_k(v, t)} \frac{deg_u}{|Nb_k(v, t)|}.
\]

However, when each node determines the amount of supplied food from this estimation, the total amount of the supplied food over the whole network is smaller than the target value. The shortage of supplied food is caused by the tendency that the estimation of the density tends to be higher than the actual density in the whole network. The reason why the estimation demonstrates such tendency can be intuitively explained as follows: the degree \( deg_v \), is counted in all the density values estimated by the \( k \)-neighbors of \( v \). Since a node with a higher degree tends to have a larger number of \( k \)-neighbors, the above estimation tends to be higher than the actual density. On the assumption that the number of \( k \)-neighbors of each node \( v \) is proportional to its degree \( deg_v \), we make correction by multiplying a weight coefficient \( 1/deg_v \), to each node \( v \). The correction gives us the following estimation of the density by node \( v \) at time \( t \):

\[
\frac{1}{2} \cdot \sum_{u \in Nb_k(v, t)} \frac{deg_u/deg_v}{|Nb_k(v, t)|} \cdot \sum_{u \in Nb_k(v, t)} \frac{1}{deg_u} = (1/2) \cdot \frac{1}{|Nb_k(v, t)|} \cdot \sum_{u \in Nb_k(v, t)} \frac{1}{deg_u}
\]

We expect that the above formula gives us better estimation especially in large-scale of random networks with small degrees.
Based on the above estimation, the algorithm in Sect. 3.2 is modified so that each node $v$ should supply the following amount of food every CYCLE time units.

$$\sum_{u \in N_b_k(v,t)} (1/\text{deg}_u) \cdot \text{deg}_v \cdot \gamma \cdot \text{AGT\_FOOD}$$

$|N_b_k(v,t)|$

**Simulation results using the estimated density.** Now, we present simulation results to show accuracy of the estimation. Table 3 shows the value $(s'/s) \cdot 100$ where $s$ (resp. $s'$) is the total amount of supplied food over the whole network by the algorithm in Sect. 3.2 (resp. the modified algorithm). The table shows the simulation results for nine combinations of three values of $k$ (1, 2 and 3), and three network sizes $n(t)$ (200, 500 and 1000). Experimental results in the table are average values of ten different networks, and these networks are created by a similar way to that in Sect. 4.

For random networks, the simulation results show that the modified algorithm supplies almost the same amount of food as the algorithm in Sect. 3.2. Larger values of $k$ give closer approximation of the total amount of supplied food, but require larger cost to gather information from $k$-neighbors. We can conclude from the simulation results that setting $k = 1$ or $k = 2$ gives sufficiently close approximation.

From the simulation results for scale-free networks, we can see that the total amount of supplied food determined from 1-neighbors shows the closest approximation, and that larger value of $k$ cannot improve the approximation. This is because scale-free networks violates the assumption that the number of $k$-neighbors of node $v$ is proportional to the degree $\text{deg}_v$ for $k$ of two or more. We can conclude from the simulation results that setting $k = 1$ is appropriate for scale-free networks.

Now, we show simulation results of the modified algorithm. In the simulation, the behavior and the initial locations of agents are determined in the same way as that in Sect. 4. Figure 6 and Fig. 7 show the experimental results on the agent population $a(t)$. Figure 6 is for “static” networks
and Fig. 7 is for “dynamic” networks. The networks are created by the way presented in Sect. 4. In the simulation, the length CYCLE of the time interval is set to 200. The network size \( n(t) \) is fixed at 500 during the simulation for “static” networks. For “dynamic” networks, networks change in the same way as the simulation in Sect. 4. From the above observation on accuracy of the estimated density, each node uses the density estimated from 2-neighbors for random networks, and the density estimated from 1-neighbors for scale-free networks.

The simulation results in Fig. 6 show that the agent population converges to and stabilizes at slightly higher population than the target. The difference is caused by excess of supplied food presented in Table 3, and the difference ratio of population is less than 0.02 in random networks and is less than 0.07 in scale-free networks. The simulation results in Fig. 7 also show that the agent population is adaptively adjusted in response to changes in the network size. The agent population is slightly higher than the target, but the difference ratio of population is still less than 0.1. Consequently, we can conclude that the modified algorithm can also adjust the agent population using only the local information, but its accuracy is slightly lower than the algorithm using the global information \( n(t) \) and \( e(t) \).

6. Conclusions

In this paper, we have proposed two distributed algorithms for the mobile agent population control problem that requires adapting the number of agents to a given constant fraction of the current number of nodes in a dynamic network. These algorithms are inspired by the single species population model, which is well-known in the field of the population ecology. The simulation results show that the proposed algorithms can adequately adjust the number of agents in dynamic networks. In addition, from the simulation results, the lifetime of each agent becomes sufficiently long by setting an appropriate value to algorithm parameter CYCLE and the utilization rate of agent is also satisfactory.

These two algorithms are different in the information each node requires. That is, the first algorithm requires each node to know the numbers of nodes and links. On the other hand, in the second algorithm, each node needs no global information on networks, but locally estimates the link density. The simulation results have shown that the second algorithm can also adjust the number of agents but with slightly lower accuracy than the first algorithm.

The mobile agent population control problem is a fundamental problem common to the wide range of mobile-agent-based distributed systems. In the proposed algorithms, on the assumption that each agent makes random walk, each node utilizes the stationary probability of an agent to determine the amount of food it supplies. Our future work is to consider other migration patterns than random walks. The key idea of our algorithms is to adjust agent population by controlling the total amount of food supplied at each node. In the first algorithm, the amount of food supplied at each node is determined from the frequency of agents’ visits at the node. Therefore, the first algorithm is expected to control agent population appropriately under other migration patterns, as long as each node can know the frequency of agents’ visits at the node. On the other hand, the second algorithm require the assumption that frequencies of agents’ visits at each node is inversely proportional to the number of edges \( e(t) \), which does not generally hold for any migration patterns. Thus, to apply the second algorithm, we have to modify it not to use such assumption.

Another future work is to realize mobile agent population control only with use of agent-side processes. One of advantages of mobile agent systems is to be able to construct systems only using remote executable programs (or agents). In the proposed algorithm, however, we need actions of environments (i.e., food feeding by nodes) in addition to actions of agents. Therefore, we should consider control algorithms that require no action of network nodes.

Acknowledgment

This work is supported in part by a Grant-in-Aid for Scientific Research((B)(2)15300017) of JSPS, Grand-in-Aid for Young Scientists ((B)18700059) Grant-in-Aid for Scientific Research on Priority Areas(16092215) of MEXT, Grant-in-Aid for JSPS Fellows (2005, 50673), and “The 21st Century Center of Excellence Program” of MEXT, and Strategic Information and Communications R&D Promotion Programme (SCOPE) of MIC.

References


Tomoko Suzuki received the B.E and M.E. degrees in computer science from Osaka University in 2003 and 2005. She is now a student of Graduate School of Information Science and Technology, Osaka University. Her research interests include distributed algorithms. She is a student member of IEEE.

Taisuke Izumi received the M.E. and D.I degrees in computer science from Osaka University in 2003 and 2006. He is now an Assistant Professor of Graduate School of Engineering, Nagoya Institute of Technology. His research interests include distributed algorithms. He is a member of ACM and IEEE.

Fukuhito Ooshita received the M.E. degree in computer science from Osaka University in 2002. Since 2003, he has been an Assistant Professor in the Graduate School of Information Science and Technology at Osaka University. His research interests include parallel algorithms and distributed algorithms.

Toshimitsu Masuzawa received the B.E., M.E. and D.E. degrees in computer science from Osaka University in 1982, 1984 and 1987. He had worked at Osaka University during 1987–1994, and was an associate professor of Graduate School of Information Science, Nara Institute of Science and Technology (NAIST) during 1994–2000. He is now a professor of Graduate School of Information Science and Technology, Osaka University. He was also a visiting associate professor of Department of Computer Science, Cornell University between 1993–1994. His research interests include distributed algorithms, parallel algorithms and graph theory. He is a member of ACM, IEEE, EATCS and the Information Processing Society of Japan.