The Journal of Supercomputing, 24, 327–340, 2003

Analysis on Extended Ant Routing Algorithms for Network Routing and Management

JOHN SUM cspfsum@comp.polyu.edu.hk

Department of Computing, Hong Kong Polytechnic University, Hung Hom, KLN, Hong Kong

HONG SHEN shen@jaist.ac.jp

Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Tatsunokuchi, Ishikawa 923-1292, Japan

G. YOUNG

Computer Science Department, Cal Poly Pomona, Pomona, CA 91768, USA

JIE WU jie@cse.fau.edu

Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, Florida, 334332, USA

CHI-SING LEUNG eeleungc@cityu.edu.hk

Department of Electronic Engineering, City University of Hong Kong, Kowloon Tong, KLN, Hong Kong

Abstract. Advances in mobile agent research have brought in a new method for network routing, ant routing. Recently, we have derived some preliminary results regarding the agent population growth property and the jumping behavior for an ant routing algorithm. The focus was on the expected number of agents in a node. In practice, the number of agents propagating on each network channel is also critical as the network channel bandwidth is limited. In this paper, we first propose two extended ant routing algorithms, and then provide an in-depth analysis on the population growth behavior of the propagating agents for these algorithms, both at nodes (hosts) and on edges (channels) of the network.

Keywords: Internet, mobile agents, routing algorithms

1. Introduction

To meet the requirement of rapid growth of the network-centric programming [6, 8, 12, 15, 22, 28], and applications due to the widespread availability of the Internet and popularity of the WWW [19, 20, 29], use of mobile agents appears to be an attractive technique evolved in the last few years. A mobile agent is a program that acts on behalf of a user to perform intelligent decision-making tasks, and is capable of migrating autonomously from node to node in a network. Usually, the main task of a mobile agent is determined by user specified applications, which can range from online shopping and distributed computation to real-time device control. Successful

examples can be seen in many new program paradigms such as Aglets [16, 18], Voyager [21], Agent Tcl [10], Tacoma [14], Knowbots [13] and Telescript [30].

As the communication between two agents (for instance the server agent and the user agent) is established within one single host, and it does not involve message transmission between the host machines of these agents (the server machine and the user machine), use of agent programming can significantly reduce the network (bandwidth) usage for message transmission [29]. Due to the tremendous growth of the size of the Internet and the latest development of mobile computing [1, 2, 23, 24, 27], the demand for low-usage network management technologies, such as mobile agent based network management, increases drastically.

In recent years, many intelligent mobile agent based network management techniques have been proposed and implemented [3, 9, 25, 31]. In a typical agent based network management system, the server first dispatches the delegated mobile agents to the network on behalf of the network manager. The agents then wander around the network and gather the information about the current status of the network. Once the agents have traversed back to the server, they hand-in their summary reports for network management. As the reports are given to the server only when their trips are over, there is little communication between the agents, and the server during the whole process, making the network traffic relatively light.

Network routing is another problem in network management [26] using mobile agent techniques. Ant routing (or ant-like routing method) [5, 7, 31–34] is a recently proposed mobile agent based network routing algorithm for use in these environments. The idea mimics the path searching process of an ant. Once a request for sending a message is received by a server, the server will generate a number of mobile agents like ants. These agents will then move out from the server to search for the corresponding destination host. When an agent has reached the destination, it traverses back to the source host, the server, following the path searched, and leave marks (just like the pheromone) on the hosts along the path. When a certain number of agents have been back (others may have been dead, or are still in the searching process), the source host will evaluate the costs of the paths collected, and pick up the optimal path for sending the message. If a connection is needed, the server will then send out an allocator agent to reserve resources in the hosts along the path.

It can be seen that for both network management and routing, agents have to be generated frequently in the network. When there are too many agents in the network, it will introduce too much overhead on host machines, which will eventually become always busy, and thus, indirectly block the network traffic. Therefore, an analysis on the agents' population change, and its growth behavior is necessary.

The objective of this paper is two-fold. First, two extended ant routing algorithms to our previous preliminary analysis will be introduced. Second, the agent propagation behaviors in the ant routing algorithms in both synchronous and asynchronous modes will be analyzed.

The paper is organized as follows. In the next section, the network model and two variant ant routing algorithms will be introduced. To simplify the analysis, we first analyze the behavior of these algorithms in a synchronous mode in Section 3.

Then we extend the results to an asynchronous mode in Section 4. Finally, the conclusion is presented in Section 5. Throughout the paper, nodes and hosts are used interchangeably, as well as edges and channels.

2. Network model and ant routing

Let $G = \{V, E\}$ be a graph corresponding to a fixed network, where V is the set of hosts, and E is the edge set. The connectivity of the graph is described by a connection matrix C. For example, suppose the graphical structure of a network is shown in Figure 1, the connection matrix C will be given as follows:

$$C = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

Throughout the paper, we assume that the topology of a network is a connected graph in order to ensure that communication is able to be made between any two host machines.

White, Pagurek and Oppacher [32] proposed an adaptive mobile agents algorithm for network routing and connection management. The essential idea of their algorithm can be sketched as follows.

1. Once a unicast[1] request has been request, m ant agents (the *explorer* agents) are created and sent out into the network.
2. The explorer agents traverse the network from the source to destination. At each immediate node, the explorer agent will select randomly a neighbor node to move forward. For the example in Figure 1, if the agent has moved from node A to node B, and node C and node D are connected to node B, the allowable moves will be either $A \to B \to C$ or $A \to B \to D$, but not $A \to B \to A$.
3. Once the destination node has been reached, the explorer agent will traverse backward to the source node.
4. In the source node, each returned explorer agent[2] will compare its explored path with other returned explorer agent. For each returned explorer agent, if the cost of the corresponding traverse path is acceptable, then an *allocator* agent will thus be sent out immediately, and allocate network resources on the nodes and links used in the path.

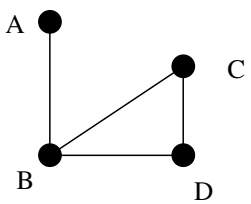

Figure 1. A simple network structure.

5. When the path is no longer required, a *deallocator* agent traverses the path, and de-allocates the network resources used on the nodes and links.

It should be noted that the performance of this ant routing algorithm is determined by the costs of the paths being searched by the explorer agents. Besides, it should be noted the forward-only move can make the explorer agent get stuck in any terminal node. Let us take Figure 1 as an example. Suppose an explorer agent is sent out from node B to node D. By chance, it moves to node C in the first jump and then to node A in the second jump. As the explorer agent can only move forward, it finally gets stuck in node A.

In order to alleviate the problem of being-stuck, we modify the algorithm presented in [32] as follows.

2.1. *Algorithm I*

Suppose a request for sending a message is received in the i^{th} host at time t, the host will thus generate k ant agents. Then each agent will randomly select one neighboring host with probability $P(j|i) = P(i \to j) = 1/(|\Omega_i| + 1)$, and go. The parameter $|\Omega_i|$ is the total number of neighboring hosts of the i^{th} host. Assume that when an ant has reached the j^{th} host, it will check whether the host is its destination. It will then turn back to the source host and report the path being searched if the j^{th} host is its destination. Otherwise, the agent will select randomly, one neighboring host of the j^{th} host and go. In the source machine, the server will pick up the path which is of minimum number of hops. Then, the message is sent along this path to the destination machine.

2.2. *Algorithm II*

Suppose that a request for sending a message is received in the i^{th} host at time t, the host will thus generate k ant agents. Then each agent will randomly select a neighboring host with probability $P(j|i) = P(i \to j) = 1 - \alpha/|\Omega_i|$, and go. The parameter $|\Omega_i|$ is the total number of neighboring hosts of the i^{th} host, and the parameter $\alpha(0 < \alpha < 1)$ is the dying rate of an agent. Similar to Algorithm I, the agent can die at any intermediate hop at a constant dying rate. While an ant agent has reached the j^{th} host, it will check whether the host is its destination. It will then turn back to the source host, and report the path being searched if the j^{th} host is its destination. Otherwise, the agent will select randomly a neighboring host of the j^{th} host and go. In the source machine, the server will pick up the path which is of minimum number of hops. Then, the message is sent along this path to the destination machine.

It should be noted that the essential idea of this algorithm looks similar to what White, Pagurek and Oppacher in [32] proposed. As in each jumping step, there is only a $(1 - \alpha)$ chance for an agent to survive. Therefore, the expected number of steps an agent can jump is equal to $\sum_{k=1}^{\infty} k(1 - \alpha)^k$, which is equal to constant value

$1 - \alpha/\alpha^2$. Using the same technique, it is able to determine the probability of the event that $\{steps \geq t\}$.

$$P\{steps \geq t\} = \sum_{k=t}^{\infty} \alpha(1-\alpha)^k = (1-\alpha)^t.$$

One can design the allowable search step t_0 as follows:

$$t_0 = \min_{t \geq 0}\{P\{steps \geq t\} \geq \epsilon\}$$
$$= \left\lfloor \frac{\log \epsilon}{\log \alpha} \right\rfloor,$$

where ϵ is a small positive number.

3. Agent population: synchronous mode

In the synchronous mode, we assume that the propagation delay of an agent jumping from one hop to any of its neighbors is constant. We will use the following notation:

$p_i(t)$: the number of agents running in the i^{th} host at time t
$r_i(t)$: the number of requests initiated at time t in the i^{th} host
k: the number of agents being generated once a request has been received
m: the expected number of requests received in a host

Besides, we assume that the network is a connected graph. The dynamical change of the agent population in the network can be given by the following equations:

$$p_j(t+1) = kr_j(t) + \sum_{i \in \Omega_j} \sum_{l=1}^{p_i(t)} \delta_{jil}(t); \tag{1}$$

$$\delta_{jil}(t) = \begin{cases} 1 & \text{if the } l^{th} \text{ agent selects the } j^{th} \text{ host to go} \\ 0 & \text{otherwise;} \end{cases} \tag{2}$$

$$\sum_{j \in \Omega_i} \delta_{jil}(t) = 1, \tag{3}$$

where $\delta_{jil}(t)$ indicates the selection of the l^{th} agent in the i^{th} host.

3.1. Algorithm I

In accordance with Algorithm I, $P(\delta_{jil}(t) = 1) = P(j|i) = 1/|\Omega_i| + 1$, for all $l = 1, 2, \cdots, p_i(t)$. Taking the expectation on both sides of Equation (1), it is readily shown that the evolution behavior of ant routing follows Markov property,

$$E\{p_j(t+1)|\vec{p}(t), \vec{r}(t)\} = kr_j(t) + \sum_{i \in \Omega_j} \frac{1}{|\Omega_i| + 1} p_i(t),$$

where $\vec{p}(t) = [p_1(t), p_2(t), \ldots, p_N(t),]^T$, $\vec{r}(t) = [r_1(t), r_2(t), \ldots, r_N(t)]^T$. In the compact form, it can be rewritten as follows:

$$E\{\vec{p}(t+1)|\vec{p}(t), \vec{r}(t)\} = A\vec{p}(t) + k\vec{r}(t), \tag{4}$$

where $A = (a_{ji})_{N\times N}$ and $a_{ji} = 1/(|\Omega_i| + 1)$ if $j \in \Omega_i$, $j \neq i$, and otherwise zero. Suppose that a survival agent, is an agent running after the current step of transition, a_{ji} is the expected delivery rate of a survival agent transmitted from i to $j \in \Omega_i$. By defining the following recurrent relation, $\hat{\vec{p}}(t+1) = E\{\vec{p}(t+1)|\hat{\vec{p}}(t), \hat{\vec{r}}(t)\}$, the dynamical equation for the expected number of agent in the network, can be obtained.

$$\hat{\vec{p}}(t+1) = A\hat{\vec{p}}(t) + k\hat{\vec{r}}(t) = A\hat{\vec{p}}(t) + km\vec{e}, \tag{5}$$

where $\vec{e} = (1, 1, \ldots, 1)^T$. In our previous preliminary analysis, we have proved the following property.

Theorem 1 *In Algorithm I, the expected number of agents running in each host is smaller than or equal to* $(\max_i\{|\Omega_i|\} + 1)km$.

Proof: Since matrix A can be decomposed into UDU^{-1} with D being a diagonal matrix whose elements are the eigenvalues of matrix A, $[I - A]$ can be expressed as $U[I - D]U$. Thus, $\|[I - A]\| \geq (1 - \sigma(A))$ and

$$\begin{aligned}\|[I - A]^{-1}\| &\leq (1 - \sigma(A))^{-1} \\ &\leq (1 + \max_i\{|\Omega_i|\}).\end{aligned}$$

Here, $\|.\|$ means infinity norm of a matrix. Since $\|km[I - A]^{-1}\vec{e}\| \leq \max_i\{1 + |\Omega_i|\}km$, the expected number of agents running in each host is then smaller than or equal to $\max_i\{1 + |\Omega_i|\}km$; the proof is thus completed. Q.E.D.

We can furthermore show the following theorems.

Theorem 2 *In Algorithm I, the number of agents propagating through any edge is smaller than or equal to* km.

Proof: The proof is accomplished by mathematical induction. Consider the j^{th} row in Equation (5),

$$\hat{p}_j(t+1) = km + \sum_{i\in\Omega_j} \frac{\hat{p}_i(t)}{1 + |\Omega_i|}.$$

Let $\hat{f}_i(t)$ be the average number of agents propagating out from the i^{th} host at time t, we can thus rewrite the above equation as follows:

$$\hat{f}_j(t+1) = \frac{1}{1 + |\Omega_j|}\left(km + \sum_{i\in\Omega_j} \hat{f}_i(t)\right). \tag{6}$$

Obviously, $\hat{f}_i(t) \leq km$ for all $i = 1, \ldots, N$ implies $\hat{f}_j(t+1) \leq km$. Since the initial value of $\hat{f}_i(0)$ is zero, and $\hat{f}_i(1)$ is equal to $km/(1 + |\Omega_i|)$ (which is smaller

than km). Therefore, by the principle of mathematical induction, the proof is completed. Q.E.D.

With this result and by definition—$\hat{f}_i(t) = \hat{p}_i(t)/(1 + |\Omega_i|)$, it can readily be shown that $\hat{p}_i(t) \leq (1 + |\Omega_i|)km$ and the following Theorem is true.

Theorem 3 *In Algorithm I, the expected number of agents jumping in the i^{th} host is smaller than or equal to* $(|\Omega_i| + 1)km$.

Theorem 3 shows that the upper bound of the expected number of agents in a node depends on the local parameter $|\Omega_i|$, which is the size of the neighbor set. If a node is added to the network, determining value k does not need to check the global parameter $\max_i\{|\Omega_i|\}$. It can be done by simply checking the computational power of the machine. Another nice outcome is the proof that the number of agents propagating out a node is smaller than or equal to km. Therefore, simple rules for controlling the agents flowing in the network, and computing the number of agents being processed in the network can be obtained.

3.2. Algorithm II

In accordance with Algorithm II,

$$P(\delta_{jil}(t) = 1) = P(j|i) = \frac{1 - \alpha}{|\Omega_i|} \tag{7}$$

for all $l = 1, 2, \ldots, p_i(t)$. Taking the expectation on both sides of Equation (1) with the definition of $P(\delta_{jil}(t) = 1)$ as in Equation (7), it is readily shown that the evolution behavior of this ant routing algorithm can be expressed by the following equation.

$$\hat{\vec{p}}(t + 1) = A_\alpha \hat{\vec{p}}(t) + km\vec{e}, \tag{8}$$

where $A_\alpha = (a_{ji})_{N\times N}$ and $a_{ji} = (1 - \alpha)/|\Omega_i|$ if $j \in \Omega_i$, $j \neq i$, and zero otherwise. Similar to that of Algorithm I, we are able to analyze the growth behavior of agents in the nodes and edges. The first result on the expected number of agents in a host can be stated by the following theorem.

Theorem 4 *If $P(\delta_{jil}(t) = 1) = (1 - \alpha)/|\Omega_i|$ for all $l = 1, 2, \ldots, p_i(t)$, the expected number of agents running in the i^{th} host is smaller than or equal to $\alpha^{-1}km$.*

Proof: Since $A_\alpha \trianglelefteq (1 - \alpha)I$, i.e. $A - (1 - \alpha)I$ is negatively semidefinite, $\hat{\vec{p}}(t)$ will converge. The limit of $\hat{\vec{p}}(t)$ is expressed by the following relation: $\lim_{t\to\infty} \hat{\vec{p}}(t) = [I - A_\alpha]^{-1}km\vec{e}$. Observe that $[I - A_\alpha]^{-1} \trianglelefteq \alpha^{-1}I$, it is readily shown that $\lim_{t\to\infty} \vec{p}(t) = \alpha^{-1}km\vec{e}$, and thus the expected number of agents running in the i^{th} host is smaller than or equal to $\alpha^{-1}km$. Q.E.D.

Similarly, let $\hat{f}_i(t)$ be the expected number of agents propagating from the i^{th} node, i.e.

$$\hat{f}_i(t) = \frac{1-\alpha}{|\Omega_i|}\hat{p}_i(t)$$

we can further obtain the following theorem.

Theorem 5 *If $P(\delta_{jil}(t) = 1) = 1 - \alpha/|\Omega_i|$ for all $l = 1, 2, \ldots, p_i(t)$, the number of agents propagating from the i^{th} node is smaller than or equal to $(1-\alpha)km/(\alpha|\Omega_i|)$.*

Similar to that of Algorithm I, one beauty of these results is that the upper bounds of the expected number of agents in the node, which is $\alpha^{-1}km$, is independent of the global parameter $\max_i\{|\Omega_i|\}$. Another nice outcome is the proof that the number of agent being propagating out a node, is smaller than or equal to $km/(\alpha|\Omega_i|)$. It is also independent of the global parameter $\max_i\{|\Omega_i|\}$. Therefore, simple rules for controlling the agents flowing in the network, and computing the number of agents being processed in the network, can be obtained. Figure 2 shows the differences

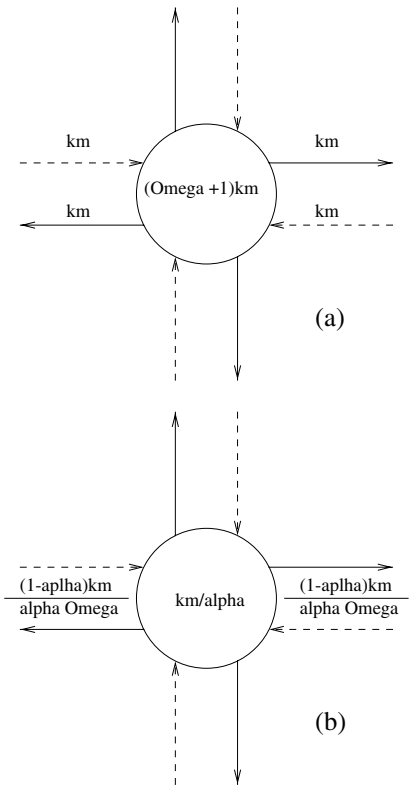

Figure 2. A schematic diagram showing the agent populations in a node and its adjacent channels: (a) Algorithm I, and (b) Algorithm II.

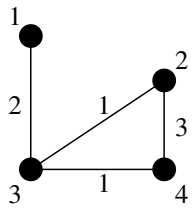

Figure 3. A simple network with uneven transition delays.

between the results obtained for Algorithm I and for Algorithm II. The solid-line arrow corresponds to an outgoing channel while the dashed-line arrow corresponds to an incoming channel. The values shown in the figure are the bound values.

4. Agent population: Asynchronous mode

In the asynchronous mode, we assume that the propagation delay of an agent jumping from one hop to any of its neighbors, is not constant. This might happen when there are uneven transition delays. To simplify the discussion, we first give a simple example to show our approach to analysis. An example is shown in Figure 3. The transition delay between any two neighboring hosts is depicted by the number labelled on their edge. For notational convenience, the nodes are labelled by numbers.

Let $\hat{p}_i(t)$ be the expected number of agents taking over the random variables δ_{jil}s in the i^{th} host ($i = 1, \ldots, 4$) at the t^{th} step, given $\hat{p}_1(\tau), \ldots, \hat{p}_4(\tau)$ for $\tau < t$.

$$\hat{p}_1(t+1) = kr_1(t) + \frac{\hat{p}_3(t-1)}{|\Omega_3|+1} \tag{9}$$

$$\hat{p}_2(t+1) = kr_2(t) + \frac{\hat{p}_3(t)}{|\Omega_3|+1} + \frac{\hat{p}_4(t-2)}{|\Omega_4|+1} \tag{10}$$

$$\hat{p}_3(t+1) = kr_3(t) + \frac{\hat{p}_1(t-1)}{|\Omega_1|+1} + \frac{\hat{p}_2(t)}{|\Omega_2|+1} + \frac{\hat{p}_4(t)}{|\Omega_4|+1} \tag{11}$$

$$\hat{p}_4(t+1) = kr_4(t) + \frac{\hat{p}_2(t-2)}{|\Omega_2|+1} + \frac{\hat{p}_3(t)}{|\Omega_3|+1} \tag{12}$$

Next, let us define $\hat{\vec{p}}_A = (\hat{p}_1, \hat{p}_2, \hat{p}_3, \hat{p}_4)^T$, Equations (9) to (12) can be rewritten as follows:

$$\hat{\vec{p}}_A(t+1) = T_{11}\hat{\vec{p}}_A(t) + T_{12}\hat{\vec{p}}_A(t-1) + T_{13}\hat{\vec{p}}_A(t-2) + k\vec{r}(t), \tag{13}$$

where

$$T_{11} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{|\Omega_3|+1} & 0 \\ 0 & \frac{1}{|\Omega_2|+1} & 0 & \frac{1}{|\Omega_4|+1} \\ 0 & 0 & \frac{1}{|\Omega_3|+1} & 0 \end{bmatrix},$$

$$T_{12} = \begin{bmatrix} 0 & 0 & \frac{1}{|\Omega_3|+1} & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{|\Omega_1|+1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$T_{13} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{|\Omega_4|+1} \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{|\Omega_2|+1} & 0 & 0 \end{bmatrix}.$$

Taking the expectation on both sides with respect to the random vectors $\vec{r}(t)$, $\vec{r}(t-1), \ldots, \vec{r}(1)$, it is readily to obtain the following recursive equation:

$$\hat{\vec{p}}_A(t+1) = T_{11}\hat{\vec{p}}_A(t) + T_{12}\hat{\vec{p}}_A(t-1) + T_{13}\hat{\vec{p}}_A(t-2) + km\vec{e}, \tag{14}$$

for all $t \geq 2$. As T_{11}, T_{12} and T_{13} are all non-negative, it can readily be derived that

$$\begin{aligned} &\|\hat{\vec{p}}_A(t+1) - \hat{\vec{p}}_A(t)\| \\ &\quad \leq (T_{11} + T_{12} + T_{13}) \max_{1\leq\tau\leq 3} \left\{ \|\hat{\vec{p}}_A(t-(\tau-1)) - \hat{\vec{p}}_A(t-(\tau)))_r\| \right\}. \end{aligned}$$

Here $\|p\|$ is the infinity norm of the vector p. Hence $\lim_{t\to\infty} \|\hat{\vec{p}}_A(t+1) - \hat{\vec{p}}_A(t)\| = 0$, indicating the existence of $\lim_{t\to\infty} \hat{\vec{p}}_A(t+1)$. Let this limiting vector be $\hat{p}_0$, using Equation (14) it can readily be shown that

$$\begin{aligned} \lim_{t\to\infty} \|\hat{\vec{p}}_A(t+1)\| &= \|km[I - T_{11} - T_{12} - T_{13}]^{-1}\vec{e}\| \\ &= \|km[I - A]^{-1}\vec{e}\| \\ &\leq km(\max_i\{|\Omega_i|\} + 1)\vec{e}. \end{aligned}$$

Using this approach, we can see that for any finite propagation delay τ, an equation similar to that of Equation (14) can be written as follows:

$$\hat{\vec{p}}_A(t+1) = \sum_{i=1}^{\tau} T_{1i}\hat{\vec{p}}_A(t-(i-1)) + km\vec{e}, \tag{15}$$

$$\sum_{i=1}^{\tau} T_{1i} = A, \tag{16}$$

where the matrix $A = (a_{ji})_{N\times N}$ is defined in a similar way as in the synchronous mode, i.e. $a_{ji} = 1/(|\Omega_i| + 1)$ if $j \in \Omega_i$, $j \neq i$ and otherwise zero. Thus the following theorem can be stated for the general case when τ is any finite integer.

Theorem 6 *In Algorithm I in the asynchronous mode, the expected number of agents running in each host is smaller than or equal to* $(1 + \max_i\{|\Omega_i|\})km$.

Next, we are going to show that for all $t \geq 0$, $\hat{p}_i(t) \leq (1 + |\Omega_i|)km$ and $\hat{f}_i(t) \leq km$. The technique is the same as the one used to prove Theorem 3. Considering

Equation (15) and by definition that $\hat{f}_i(t) = 1/(1 + |\Omega_i|)\hat{p}_i(t)$, we can readily obtain that

$$\hat{f}_j(t+1) = \sum_{i \in \Omega_j} \frac{1}{1+|\Omega_i|} \hat{p}_i(t - d(i,j)) + km\vec{e}, \tag{17}$$

$$= \sum_{i \in \Omega_j} \hat{f}_i(t - d(i,j)) + km\vec{e}, \tag{18}$$

where $d(i, j)$ is the propagation delay of an agent moving from the i^{th} host to the j^{th} host. Therefore, if $\hat{f}_i(t - d) \leq km$ holds all time before $t + 1$ for all $i = 1, 2, \ldots, N$, $\hat{f}_j(t + 1) \leq km$ holds for all $j = 1, 2, \ldots, N$. By the definition of $\hat{f}_i(t)$, it is also proved that $\hat{p}_i(t) \leq (1 + |\Omega_i|)km$ for all $i = 1, \ldots, N$. The results can be stated by the following theorem.

Theorem 7 *For Algorithm I in an asynchronous network, the expected number of agents running in the i^{th} host is no more than $(|\Omega_i| + 1)km$, and the number of agents propagating out the host cannot be larger than km.*

Using a similar technique to that for Algorithm II, we can obtain results similar to that of Theorems 4 and 5 in the asynchronous network. They are given in the following theorem, whose derivation is almost the same as the proof for Theorem 7.

Theorem 8 *For Algorithm II in an asynchronous network, the expected number of agents running in the i^{th} host is no more than $\alpha^{-1}km$, and the number of agents propagating out the host cannot be larger than $km/(\alpha|\Omega_i|)$.*

5. Conclusion

This paper has analyzed the agent population growth behavior for agent based network routing and management algorithms. The theorems proved in this paper are summarized in Table 1, where the subscripts i and j are dropped for simplicity.

Table 1. Summary of the theorems being proved in this paper

Theorem	Algo.	Result	Mode
1	I	$\hat{p} \leq \max_i\{\|1 + \Omega_i\|\}km$	Syn.
2	I	$\hat{f} \leq km$	Syn.
3	I	$\hat{p} \leq (1 + \|\Omega_i\|)km$	Syn.
4	II	$\hat{p} \leq \alpha^{-1}km$	Syn.
5	II	$\hat{f} \leq (1 - \alpha)km/(\alpha\|\Omega_i\|)$	Syn.
6	I	$\hat{p} \leq \max_i\{1 + \|\Omega_i\|\}km$	Asyn.
7	I	$\hat{f} \leq km$	Asyn.
	I	$\hat{p} \leq (1 + \|\Omega_i\|)km$	Asyn.
8	II	$\hat{p} \leq \alpha^{-1}km$	Asyn.
	II	$\hat{f} \leq (1 - \alpha)km/(\alpha\|\Omega_i\|)$	Asyn.

It is found that as long as the network topology is fixed, and the number of agents being created for each request is finite, the expected number of agents in each host server, and the number of agents propagating out from any host machine, must also be finite for networks operated in both synchronous and asynchronous modes. Furthermore, one might recognize that (by comparing Theorem 7 and Theorem 8) once $\alpha < (1 + |\Omega_i|)^{-1}$, the values of $\hat{p}_i$ and $\hat{f}_i$ using Algorithm I will be smaller than that of using Algorithm II. On the contrary, if $\alpha > (1 + |\Omega_i|)^{-1}$, the situation will be reversed, as illustrated in Figure 4.

In case if more information about the quality of the paths being searched, and the value of α could be obtained, it should be possible to compare the performance of Algorithm I and Algorithm II quantitatively. In the meantime, the theorems proved

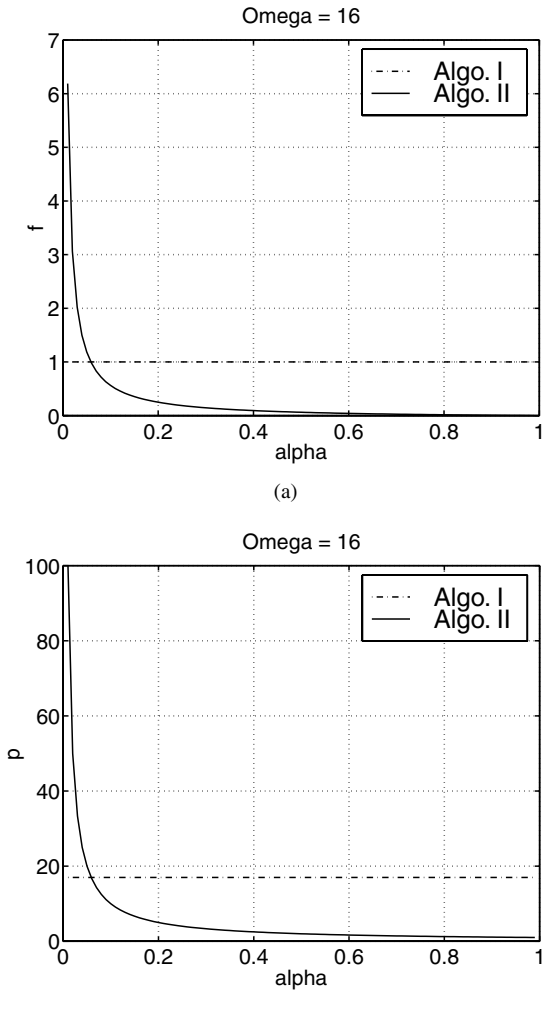

Figure 4. Comparison of the values of (a) $\hat{p}$, and (b) $\hat{f}$ for Algorithm I and Algorithm II, respectively.

in this paper simply provide some guidelines for the design of an agent based network routing and management system, particularly when the computational power of the host servers or the network channel capacity is limited, and unable to handle the large amount of processing requests due to the large volume of mobile agents propagating in the network.

Notes

1. For multi-cast, the idea is essentially the same.
2. In White-Pagurek-Oppacher algorithm, if an explorer agent cannot reach the destination node within a predefined number of moves, the agent will die automatically.

References

1. B. Bakshi, P. Krishna, N. Vaidya and D. Pradhan. Improving performance of TCP over wireless networks, Department of Computer Science, Texas A&M University, Technical Report 96-014, May 1996.
2. H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP performance over wireless networks, In *Proc. 1st ACM International Conf. on Mobile Computing and Networking (Mobicom)*, pp. 2–11, Nov. 1995.
3. A. Bieszczad, T. White, and B. Pagurek. Mobile agents for network management. *IEEE Communications Surveys*, 1(1):2–9, September 1998.
4. R. Bronson. *Matrix Operation*, Schaum's Outline Series, McGraw Hill, 1989.
5. B. Bullnheimer, R. F. Hartl, and C. Strauss. Applying the ant system to the vehicle routing problem. In *Proceedings of the 2nd Metaheuristics International Conference* (MIC-97), Sophia-Antipolis, France.
6. L. Cardelli, A Language with distributed scope. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, pp. 286–297, 1995.
7. G. Di Caro and M. Dorigo. AntNet: A mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, Université Libre de Bruxelles, Belgium, 1997.
8. F. Douglis and J. Ousterhout. Transparent process migration: design alternatives and the sprite implementation. *Software Practice and Experience*, 21(8):757–785, August 1991.
9. M. El-Darieby and A. Bieszczad. Intelligence mobile agents: Towards network fault management automation. In *Proceedings of Sixth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 611–622, May 1999.
10. R. Gray. Agent Tcl: A flexible and secure mobile-agent system. In *Proceedings of the Fourth Annual Tcl/Tk Workshop* (TCL '96), pp. 9–23, July 1996.
11. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
12. C. Harrison, D. M. Chess, and A. Kershenbaum. Mobile agents: are they a good idea? Technical Report, IBM Research Division, T. J. Watson Research Center, March 1995. (http://www.research.ibm.com/massdist/mobag.ps).
13. J. Hylton, K. Manheimer, F. L. Drake Jr., B. Warsaw, R. Masse, and G. van Rossum. Knowbot programming: System support for mobile agents. In *Proceedings of the Fifth International Workshop on Object Orientation in Operating Systems* (IWOOOS '96), pp. 8–13, October 1996.
14. D. Johansen, R. van Renesse, and F. B. Schneider. An introduction to the TACOMA distributed system. Technical Report 95-23, Department of Computer Science, University of Tromso, June 1995.
15. E. Jul, H. Levy, N. Hutchinson, and A. Black. Fine-grained mobility in the emerald system. *ACM Transactions on Computer Systems*, 6(1):109–133, February 1988.
16. G. Karjoth, D. Lange, and M. Oshima. A security model for Aglets. *IEEE Internet Computing*, 1(4):68–77, July–August 1997.

17. N. Karnik and R. Tripathi. Design issues in mobile-agent programming systems, *IEEE Concurrency*, 6(3):52–61, July–Sept. 1998.
18. D. Lange and M. Oshima. *Programming and Deploying Java Mobile Agents with Aglets*, Addison Wesley, 1998.
19. A. Lingnau, O. Drobnik, and P. Domel. An HTTP-based infrastructure for mobile agents. In *Proceedings of the Fall 1995 WWW Conference*, pp. 461–471, December 1995.
20. C. Munday, J. Dangedej, T. Cross, D. Lukose. Motivation and perception mechanisms in mobil agent for electronic commerce. In *Proceedings of the First Australian Workshop on Distributed Artificial Intelligence*, LNAI, Vol. 1087, pp. 144–158, Springer, November 1996.
21. ObjectSpace. ObjectSpace voyager core package technical overview. Technical Report, ObjectSpace, Inc., July 1997.
22. M. Ranganathan, A. Acharya, S. Sharma, and J. Saltz. Network-aware mobile programs. In *Proceedings of USENIX*'97, 91–103, January 1997.
23. C. Perkins. Providing continuous network access to mobile hosts using TCP/IP. *Computer Networks and ISDN Systems*, 26(3):357–369, November 1993.
24. J. Soloman. *Mobile IP: The Internet Unplugged*, Prentice Hall, 1998.
25. C. Schramm, A. Bieszczad, and B. Pagurek. Application-oriented network modeling with Mobile agents. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium* NOMS'98, New Orleans, Louisiana, February 1998.
26. A. S. Tanenbaum. *Computer Networks*, 3rd Ed., Prentice Hall, 1996.
27. M. Taylor, W. Waung and M. Banan. *Internetworking Mobility: The CDPD Approach*, Prentice Hall, 1997.
28. T. Thorn. Programming languages for mobile code. *ACM Computing Surveys*, 29(3):213–239, September 1997.
29. P. Wayner. Agents away, *Byte*, May, 1994.
30. J. White. Mobile Agents. Technical report, General Magic. Available at URL http://www.genmagic.com/Telescript/, October 1995.
31. T. White. Routing with swarm intelligence, Technical Report SCE-97-15, Systems and Computer Engineering, Carleton University, September 1997.
32. T. White, B. Pagurek, and F. Oppacher. ASGA: Improving the ant system by integration with genetic algorithms. In *Proceedings of the 3rd Conference on Genetic Programming (GP/SGA '98)*, pp. 610–617, July 1998.
33. T. White, B. Pagurek, and F. Oppacher. Connection management using adaptive agents. In *Proceedings of 1998 International Conference on Parallel and Distributed Processing Techniques and Applications* (PDPTA'98), pp. 802–809, July 1998.
34. T. White and B. Pagurek. Towards multi-swarm problem solving in networks. In *Proceedings of the 3rd International Conference on Multi-Agent Systems* (ICMAS '98), pp. 333–340, July 1998.