Analysis on extended ant routing algorithms for network management

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

Hong Shen
School of Computing and Information Technology
Griffith University, Nathan,
QLD 4111, Australia;

Gilbert H. Young
Department of Computing
Hong Kong Polytechnic University,
Hung Hom, KLN, Hong Kong

Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University,
Baco Raton, Fl33432, USA.

Abstract

Recent advance in the agent research has brought in a new method for network routing, the ant routing algorithm. In a related work, we have derived some preliminary results regarding the population growth property and the jumping behavior of the ant routing algorithm. The focus was on the expected number of agents in a node. In practice, the number of propagating agents 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 of propagating agents based on these algorithms, both at nodes (hosts) and edges (channels) of the network.

Keywords

Internet, mobile agents, routing algorithms.

John Sum is the corresponding author. Address: PQ712, Department of Computing, Hong Kong Polytechnic University, Hung Hom, KLN, Hong Kong; Email: cspfsum@comp.polyu.edu.hk; Tel: (852) 2766 7246.

I. Introduction

To meet the requirement of rapid growth of the network-centric programming [6], [8], [12], [15] [22], [29] and applications due to the widespread availability of the Internet and popularity of the WWW [19], [20], [30], use of mobile agents appears to be one of the most promising new techniques evolved recently. 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. The main tasks of the mobile agent are determined by the user specified agent application which can range from online shopping and distributed computation to real-time device control. Successful examples using mobile agents can be seen in many new program paradigms such as Aglets [16], [18], Voyager [21], Agent Tcl [10], Tacoma [14], Knowbots [13] and Telescript [31]. As the communication between two agents (for instance the server agent and the user agent) is established within one single host and does not involve the message transferred between the host machines of these agents (the server machine and the user machine), one advantage of using agent programming is that the network usage for transferring messages can largely be reduced [30].

Examples using mobile agent technology include network management [3], [25], [32] and fault diagnosis [9]. The server first dispatches to the network the delegated mobile agents which are on behalf of the network manager. The agents then wonder 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. As the reports are given to the server only when their trips are over, during the course of travel there are very few communications between the agents and the server. Thus the network traffic can be relatively light.

Network routing is another example using mobile agent technology. Conventionally, once a packet of a message has to be sent to a destination (unicast) or to multiple destinations (multicast), the routers in the network will have to collectively select the best path (such as the shortest path) and send out the message. The basic assumption of this technique is that the network is stationary. In a faulty or mobile network, searching for the optimal path using this conventional approach will become very difficult. Ant routing (or ant-like routing method) [5], [7], [32], [33], [34], [35] is a recent proposed mobile agent based network routing algorithm particular for use in these environments.

The idea of ant routing algorithm mimics the path searching process of an ant. Once a request for sending a message is received from a server, the server will generate a number of mobile agents (ants). These agents will then move out from the server and search for the corresponding destination host. When an agent has reached the destination, it traverses back to the server, following the path searched and leave marks (just like the pheromone) on the hosts along the path. While a certain amount of agents have been back (other may have been dead or are still in the searching process), the source host will evaluate the cost of each path collected and pick up the path of minimum cost for the sending the message. The server then broadcasts the message along the path of minimum cost. If the request is more than sending a message but making a connected communication, the server will send out an allocator agent reserving resources from the hosts along the path of minimum cost.

It can be seen that in either fault management or message routing, agents have to be generated frequently and dispatched to the network. Thus will surely eat up certain amount of computation resources in the network hosts, even though they do not introduce much network traffic. In the worst are when there are too many agents in the network, this will introduce too much overhead on the host machines. Eventually, the host machines will always be busy and indirectly block the network traffic. Therefore, the analysis on the agents' population change and its growth behavior is necessary.

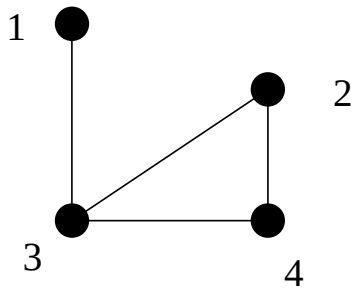


Fig. 1. Simple network structure.

An overview of [26] mention the importance of analysis on workload at each channel. The objective of this paper is two folds. First, a new ant routing algorithm will be introduced. Second, the results obtained in [26] will be extended to analyze the agent propagation behavior.

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 interchangably, as well as edges and channels.

II. Network model and the 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 & Oppacher [33] proposed an adaptive mobile agents algorithm for network routing and connection management. The essential idea of the algorithm can be sketched as follows.

1. Once a unicast connection[1] has been requested, 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.

[1]For multicast connection, the idea is essential the same.

4. In the source node, each return-back[2] explorer agent will compare its explored path with other return-back explorer agent. For each return-back 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.
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 take Figure 1 as an example. Suppose an explorer agent is sent out from Node-2 to Node-4. By chance, it moves to Node-3 in the first jump and then moves to Node-1 in the second jump. As the explorer agent can only move forward, it finally gets stuck in Node-1.

To alleviate the problem of being-stuck, we modify the algorithm presented in [33] by letting the ant agent randomly select a neighbor host and go. Besides, we also let the ant agent die out at any intermediate hop in order to prevent explosive growth of the agents in the network.

A. Algorithm I

Suppose a request for sending 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 neighbor host with probability

$$P(j|i) = P(i \to j) = \frac{1}{|\Omega_i| + 1},$$

and go. The parameter $|\Omega_i|$ is the total number of neighbor host of the i^{th} host. Let say 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 one neighbor host of the j^{th} host and go or select dying out at the i^{th} host. Note that the survival rate of an agent after jumping in the i^{th} host is $\frac{|\Omega_i|}{|\Omega_i|+1}$ while the dying rate is $\frac{1}{|\Omega_i|+1}$. 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.

B. Algorithm II

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 neighbor host with probability

$$P(j|i) = P(i \to j) = \frac{1 - \alpha}{|\Omega_i|},$$

and go. The parameter $|\Omega_i|$ is the total number of neighbor host of the i^{th} host and the parameter $\alpha (0 < \alpha < 1)$ is the dying rate of an agent. Compared with Algorithm I, this dying rate is independent on the number of neighbor hosts. Suppose 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 one neighbor 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]In White-Pagurek-Oppacher algorithm, if an explorer agent cannot reach the destination node within a predefined number of move, the agent will die automatically. Therefore, not all agent can return to the source node.

C. Remark

It should be noted that the essential idea of Algorithm II looks more closer to what White, Pagurek & Oppacher [33] proposed if we set a maximum allowable searching step, t_0, for each agent. One approach is to set t_0 in terms of the expected number of jumping step. 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 $\lfloor \sum_{k=1}^{\infty} k\alpha(1-\alpha)^{k-1} \rfloor$, which is equal to constant value $\lfloor \alpha^{-1} \rfloor$. Therefore, one can design the maximum searching step to be say $\lfloor \alpha^{-1} \rfloor$. Another approach is to set t_0 in terms of the probability that the number of jumping steps is larger than t, i.e. $P\{steps \geq t\}$. Since

$$\begin{aligned} P\{steps \geq t\} &= \sum_{k=t}^{\infty} \alpha(1-\alpha)^{k-1} \\ &= (1-\alpha)^{t-1}, \end{aligned}$$

we can set a small threshold ϵ, such that $(1-\alpha)^{t-1} \leq \epsilon$. Using this inequality, one can design the maximium allowable searching step t_0 as $\arg\max\{P\{steps \geq t\} \leq \epsilon\}$ which is equal to $\left\lfloor \frac{\log \epsilon}{\log(1-\alpha)} \right\rfloor + 1$.

III. Agents population : Synchronized mode

In synchronized mode, we assume that the propagation delay of an agent jumping from one hop to any of its neighbor is constant. Let $p_i(t)$ be the number of agents running in the i^{th} host at time t and $r_i(t)$ be the number of requests initiated at time t in the i^{th} host. Besides, we assume that the network is a connected graph. The dynamical change of the agents in the network can be given by the following equations :

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

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

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

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

A. Algorithm I

In accordance with the Algorithm I,

$$P(\delta_{jil}(t) = 1) = P(j|i) = \frac{1}{|\Omega_i| + 1}$$

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 the ant routing follows Markov property,

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

where

$$\vec{p}(t) = \begin{bmatrix} p_1(t) \\ p_2(t) \\ \cdots \\ p_N(t) \end{bmatrix} \quad \vec{r}(t) = \begin{bmatrix} r_1(t) \\ r_2(t) \\ \cdots \\ r_N(t) \end{bmatrix}.$$

In 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

$$a_{ji} = \begin{cases} \frac{1}{|\Omega_i|+1} & \text{if } j \in \Omega_i,\ j \neq i \\ 0 & \text{otherwise.} \end{cases}$$

Suppose 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.

$$\begin{aligned} \hat{\vec{p}}(t+1) &= A\hat{\vec{p}}(t) + k\hat{\vec{r}}(t) &(5)\\ &= A\hat{\vec{p}}(t) + km\vec{e}, &(6)\end{aligned}$$

where $\vec{e} = (1,1,\cdots,1)^T$. In [26], we have proved the following property.

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

We can furthermore show the following theorems.

Theorem 2: Using Algorithm I, the number of agents propagating through any edge is less than or equal to km.

Proof : The proof is accomplished by Induction. Consider the j^{th} row in Equation (6),

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

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 :

$$\hat{f}_j(t+1) = \frac{1}{1+|\Omega_j|}\left(km + \sum_{i\in\Omega_j}\hat{f}_i(t)\right). \tag{7}$$

Obviously, $\hat{f}_i(t) \leq km$ for all $i = 1,\cdots,N$ 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 $\frac{km}{1+|\Omega_i|}$ (which is smaller than km). Therefore, by the principle of M.I., the proof is completed. **Q.E.D.**

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

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

Theorem 3 shows that the upper bounds of the expected number of agents in the node is depended on the local parameter $|\Omega_i|$, which is the size of the neighbor set. It is different from the result presented in [26]

which depends on the global parameter $\max_i\{|\Omega_i|\}$. If a node has to be added in the network, there is no need to check for the global parameter $\max_i\{|\Omega_i|\}$ to determine value k. Simply check the computational power of the machine and decide the number of neighbors. Another beauty is the proof of the the number of agent being propagating out a node is less than or equal to km. Therefore, simple rules for the controlling of agent flowing in the network and the number of agent being processed in the network machines can be obtained.

B. Algorithm II

In accordance with Algorithm II,

$$P(\delta_{jil}(t)=1)=P(j|i)=\frac{1-\alpha}{|\Omega_i|} \tag{8}$$

for all $l=1,2,\cdots,p_i(t)$.

Taking the expectation on both sides of Equation (1) with the definition of $P(\delta_{jil}(t)=1)$ as in Equation (8), it is readily shown that the evolution behavior of this ant routing can be expressed by the following equation.

$$\hat{\vec{p}}(t+1)=A_\alpha\hat{\vec{p}}(t)+km\vec{e}, \tag{9}$$

where $A_\alpha=(a_{ji})_{N\times N}$ and

$$a_{ji}=\begin{cases}\frac{1-\alpha}{|\Omega_i|} & \text{if } j\in\Omega_i,\ j\neq i\\ 0 & \text{otherwise.}\end{cases}$$

Similar to that of Algorithm I, we are able to analyze the growth behavior of the agents in the nodes and edges. The first result on the expected number of agents in a host can be stated as the following theorem.

Theorem 4: Using Algorithm II, the expected number of agents running in the i^{th} host is less than or equal to $\alpha^{-1}km$.

Proof : Since $A_\alpha\trianglelefteq(1-\alpha)I$, $\hat{\vec{p}}(t)$ 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

$$\lim_{t\to\infty}\hat{\vec{p}}(t)=\alpha^{-1}km\vec{e}$$

and thus the expected number of agents running in the i^{th} host is less 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: Using Algorithm II, the number of agents propagating from the i^{th} node is less than or equal to

$$\frac{(1-\alpha)km}{\alpha|\Omega_i|}.$$

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 beauty

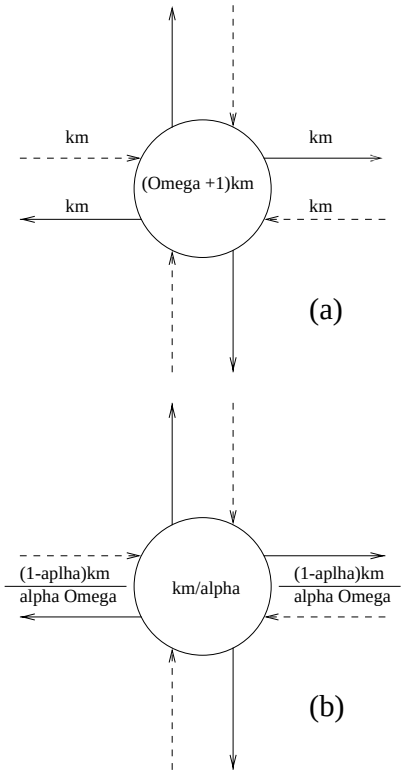


Fig. 2. The schematic diagram showing the agent populations in a node and its adjacent channels. (a) Algorithm I and (b) Algorithm II. The solid-line arrow corresponds to an outgoing channel while the dash-line arrow corresponds to an incoming channel. The values shown in the figure are the bound values.

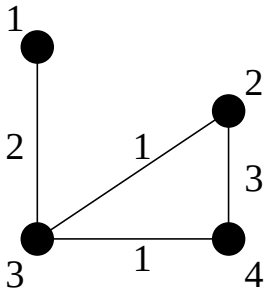


Fig. 3. A simple network with uneven transition delays.

is the proof of the the number of agent being propagating out a node is less than or equal to $km/(\alpha|\Omega_i|)$. It is again independent of the global parameter $\max_i\{|\Omega_i|\}$. Therefore, simple rules for the controlling of agent flowing in the network and the number of agent being processed in the network machines can be designed obtained. Figure 2 shows the differences between the results obtained for Algorithm I and Algorithm II.

IV. Agents population : Asynchronized mode

In asynchronized mode, we assume that the propagation delay of an agent jumping from one hop to any of its neighbor is not constant. Analysis the asynchronized mode behavior is similar to the approach in [26]. This might happen when there are uneven transition delays. To simplify the discussion, we first give a simple example to show our approach to analyze. An example is shown in Figure 3. The transition delay between any two neighbor hosts is depicted by the number labeled on their edge.

Let $\hat{p}_i(t)$ be the expectation of the number of agents taking over the random variables δ_{jil}s in the i^{th} host

$(i = 1, \cdots, 4)$ at the t^{th} step, given $\hat{p}_1(\tau), \cdots, \hat{p}_4(\tau)$ for $\tau < t$.

$$\hat{p}_1(t+1) = kr_1(t) + \frac{\hat{p}_3(t-1)}{|\Omega_3|+1} \tag{10}$$

$$\hat{p}_2(t+1) = kr_2(t) + \frac{\hat{p}_3(t)}{|\Omega_3|+1} + \frac{\hat{p}_4(t-2)}{|\Omega_4|+1} \tag{11}$$

$$\hat{p}_3(t+1) = kr_3(t) + \frac{\hat{p}_1(t-1)}{|\Omega_1|+1} + \frac{\hat{p}_2(t)}{|\Omega_2|+1} + \frac{\hat{p}_4(t)}{|\Omega_4|+1} \tag{12}$$

$$\hat{p}_4(t+1) = kr_4(t) + \frac{\hat{p}_2(t-2)}{|\Omega_2|+1} + \frac{\hat{p}_3(t)}{|\Omega_3|+1} \tag{13}$$

Next, let us define $\hat{\vec{p}}_A = (\hat{p}_1, \hat{p}_2, \hat{p}_3, \hat{p}_4)^T$, Equations (10) to (13) can be rewritten as follows :

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

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 $\vec{r}(t)$, $\vec{r}(t-1)$, $\cdots$, $\vec{r}(1)$, it is readily to obtain the following recursive equation :

$$\hat{\vec{p}}_A(t+1) = T_{11}\hat{\vec{p}}_A(t) + T_{12}\hat{\vec{p}}_A(t-1) + T_{13}\hat{\vec{p}}_A(t-2) + km\vec{e}, \tag{15}$$

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

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

Hence

$$\lim_{t \to \infty} \|\hat{\vec{p}}_A(t+1) - \hat{\vec{p}}_A(t)\| = 0$$

which means that limit exists for $\lim_{t\to\infty} \hat{\vec{p}}_A(t+1)$. Let this limiting vector be $\hat{p}_0$, using Equation (15) it can readily be shown that

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

Using this approach, we can see that for any finite propagation delay τ, an equation similar to that of Equation (15) can be written as follows :

$$\hat{\vec{p}}_A(t+1) = \sum_{i=1}^{\tau} T_{1i}\hat{\vec{p}}_A(t-(i-1)) + km\vec{e}, \tag{16}$$

$$\sum_{i=1}^{\tau} T_{1i} = A, \tag{17}$$

where the matrix $A = (a_{ji})_{N\times N}$ is defined in a similar way as in the synchronized mode, i.e.

$$a_{ji} = \begin{cases} \frac{1}{|\Omega_i|+1} & \text{if } j \in \Omega_i, j \neq i \\ 0 & \text{otherwise.} \end{cases}$$

Thus the following theorem can be stated for the general case when τ is any finite integer.

Theorem 6: Using Algorithm I in the asynchronized mode, the expected number of agents running in each host is less 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 (16) and by definition that $\hat{f}_i(t) = \frac{1}{1+|\Omega_i|}\hat{p}_i(t)$, one can readily obtain that

$$\hat{f}_j(t+1) = \sum_{i\in\Omega_j} \frac{1}{1+|\Omega_i|}\hat{p}_i(t-d(i,j)) + km\vec{e}, \tag{18}$$

$$= \sum_{i\in\Omega_j} \hat{f}_i(t-d(i,j)) + km\vec{e}, \tag{19}$$

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

Theorem 7: Using Algorithm I in an asynchronized 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 the similar technique for Algorithm II, we can obtain results similar to that of Theorems 4 and 5. As the steps are more or less the same as the proof for Theorem 7, the details are omitted.

Theorem 8: Using Algorithm II in an asynchronized 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 $km/(\alpha|\Omega_i|)$.

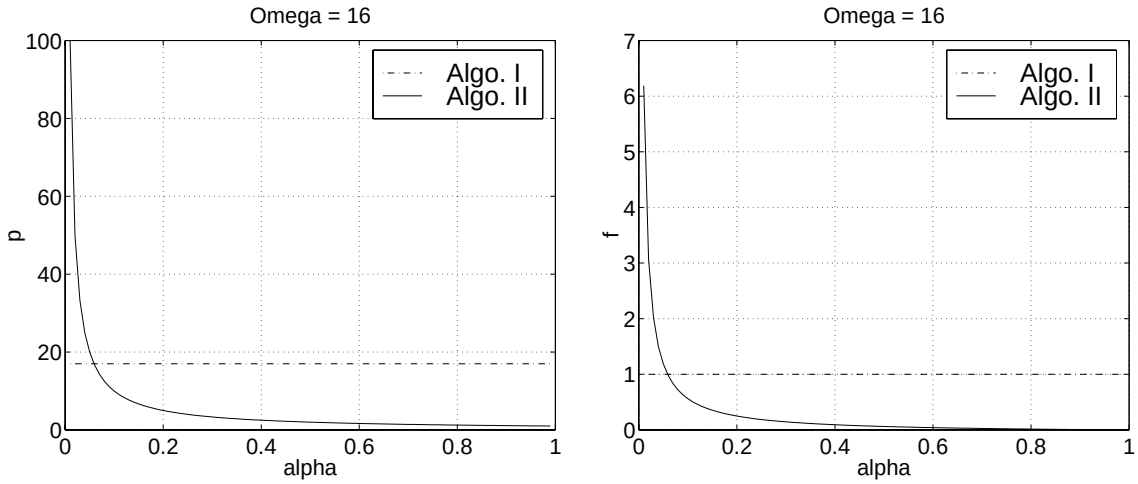
V. Conclusion

This paper has analyzed on the growth behavior of agent based network routing and management algorithms. The theorems being proved in this paper are summarized in Table I. The subscripts i, j are dropped for simplicity. 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 expectation of the expected number of agents in each host server and the number of agents propagating out from any host machine must also be finite, independent of the network being operated

TABLE I

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.



(a) (b)

Fig. 4. Comparing the values of (a) $\hat{p}$ and (b) $\hat{f}$ for Algorithm I and Algorithm II respectively. Here we assume that the values of $|\Omega_i|$ are all equal to 16.

in synchronized or asynchronized mode. 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, see Figure 4. In case more information about the quality of the paths being searched and the value of α could be obtained, it should be able to compare the performance of Algorithm I and Algorithm II. In the meantime, these results proved in this paper simply provide a guideline for the design of an agent based network routing and management system. The results are extremely useful when the computational power of the host servers is limited which is unable to handle large amount of processing requests and/or the network channel capacity is limited for large volume of mobile agents propagating in the network.

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, *Proc. 1st ACM International Conf. on Mobile Computing and Networking (Mobicom)*, Nov., 1995.

[3] A. Bieszczad, T. White, B. Pagurek. Mobile Agents for Network Management. *IEEE Communications Surveys*, September, 1998.

[4] Bronson R., *Matrix Operation*, Schaum's Outline Series, McGraw Hill, 1989.

[5] Bullnheimer B., R.F.Hartl and C.Strauss. Applying the ant system to the vehicle routing problem. *Proceedings of the 2nd Metaheuristics International Conference* (MIC-97), Sophia-Antipolis, France.

[6] Luca Cardelli. A Language with Distributed Scope. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, 286-297, 1995.

[7] Di Caro G. and M. Dorigo. AntNet: A mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, Université Libre de Bruxelles, Belgium, 1997.

[8] Fred Douglis and John 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. To be presented at the Sixth IFIP/IEEE International Symposium on Integrated Network Management, May, 1999.

[10] Robert S. Gray. Agent Tcl: A flexible and secure mobile-agent system. In *Proceedings of the Fourth Annual Tcl/Tk Workshop* (TCL '96), July 1996.

[11] Garey M.R. and Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: Freeman, 1979.

[12] Colin G. Harrison, David M. Chess, and Aaron 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] Jeremy Hylton, Ken Manheimer, Fred L. Drake Jr., Barry Warsaw, Roger Masse, and Guido van Rossum. Knowbot programming: System support for mobile agents. In *Proceedings of the Fifth International Workshop on Object Orientation in Operating Systems* (IWOOOS '96), October 1996.

[14] Dag Johansen, Robbert van Renesse, and Fred B. Schneider. An Introduction to the TACOMA Distributed System. Technical Report 95-23, Department of Computer Science, University of Tromso, June 1995.

[15] Eric Jul, Henry Levy, Norman Hutchinson, and Andrew Black. Fine-Grained Mobility in the Emerald System. *ACM Transactions on Computer Systems*, 6(1):109-133, February 1988.

[16] Gunter Karjoth, Danny Lange, and Mitsuru Oshima. A Security Model for Aglets. *IEEE Internet Computing*, 68-77, July-August 1997.

[17] Neeran M. Karnik and Anand R. Tripathi Design issues in mobile-agent programming systems, *IEEE Concurrency*, 6(3), pp. 52-61, July - sep 1998.

[18] Danny B. Lange and Mitsuru Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Addison Wesley, 1998.

[19] Anselm Lingnau, Oswald Drobnik, and Peter Domel. An HTTP-based Infrastructure for Mobile Agents. In *Proceedings of the Fall 1995 WWW Conference*, December 1995.

[20] Craig Munday, Jirapun Dangedej, Tim Cross, Dickson Lukose. Motivation and Perception Mechanisms in Mobil Agent for Electronic Commerce. *Proceedings of the First Australian Workshop on Distributed Artificial Intelligence*, LNAI, Vol. 1087, pp. 144-158, Springer, November 13 1996.

[21] ObjectSpace. ObjectSpace Voyager Core Package Technical Overview. Technical report, ObjectSpace, Inc., July 1997.

[22] M. Ranganathan, Anurag Acharya, Shamik Sharma, and Joel Saltz. Network-aware Mobile Programs. In *Proceedings of USENIX* '97, Winter 1997.

[23] Perkins C., Providing continuous network access to mobile hosts using TCP/IP. *Computer Networks and ISDN Systems*, Vol.26, 357-370, Nov. 1993.

[24] James D. Soloman, *Mobile IP: The Internet Unplugged*, Prentice Hall, 1998.

[25] Schramm, C., Bieszczad, A. and Pagurek, B. (1998), 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] John Sum, Hong Shen and Gilbert H. Young, Analysis on an intelligence mobile agent based algorithm for network routing and management, submitted for publication.

[27] Tanenbaum A.S., *Computer Networks*, 3rd Ed., Prentice hall, 1996.

[28] M. Taylor, W. Waung and M. Banan, *Internetworking Mobility: The CDPD Approach*, Prentice Hall, 1997.

[29] Tommy Thorn. Programming Languages for Mobile Code. *ACM Computing Surveys*, 29(3):213-239, September 1997.

[30] Wayner P., Agents away, *Byte*, May, 1994.

[31] James E. White, Mobile Agents. Technical report, General Magic,
Available at URL http://www.genmagic.com/Telescript/, October 1995.

[32] White, T., Routing with Swarm Intelligence, Technical Report SCE-97-15, Systems and Computer Engineering, Carleton University, September 1997.

[33] White T., Pagurek B. and Oppacher F. (1998), ASGA: Improving the Ant System by Integration with Genetic Algorithms. *Proceedings of the 3rd Conference on Genetic Programming (GP/SGA '98)*, 610-617, July, 1998.

[34] White T., Pagurek B. and Oppacher F., Connection management using adaptive agents. *Proceedings of PDPTA'98*, 802-809, July, 1998.

[35] 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), 333-340, July, 1998.