Locating Mobile Agents on Predictable Search Paths

Hung-Chin Jang, Jyh-Shyan Huang, and Yao-Nan Lien
Department of Computer Science, National Chengchi University, Taiwan, R.O.C.
jang@cs.nccu.edu.tw

Abstract

In this paper, we explain why traditional search methods are not applicable to locating mobile agents on predictable search paths. We thereby propose mechanisms of Residing Probability Estimation Mechanism, Maximum Excluded Servers Mechanism, and Dynamic Switch Mechanism to make an efficient Predictable Path Search algorithm. At last, we propose an Evaluation Function of Expected Probe Number to evaluate the expected probe number which is a crucial indicator of agent search performance.

1. Introduction

A mobile agent is a self-contained and identifiable computer program either procedural or declarative. It can be executed by recipient server on behalf of the originating client and instruct the server to forward message to another server continuously in a pipeline fashion. Main goal of using mobile agents is to reduce network traffic and interact in asynchronous manner [8].

In a mobile computing environment that supports mobile agents, after a client submits a service agent into a service network [4], either the client or network manager may want to know its current location in order to either acquire its status or control its execution, etc. Tracking the locations of service agents becomes a critical problem in managing a mobile agent service network. An efficient agent search method is needed to send out another agent, called search agent, to track the originating service agent, called target agent [1,5,6,7]. When a target agent moves from server to server, each visited server will record the relevant agent information like agent ID, arrival time, departure time, and the next hop of target agent, etc. as tracking information for latter use.

In this paper, we assume routing paths of target agents are predictable which may be either deterministic or non-deterministic. A deterministic path spans a list and a non-deterministic path spans a tree. We embed mechanisms of Residing Probability Estimation Mechanism, Maximum Excluded Servers Mechanism, and Dynamic Switch Mechanism in agent search algorithm to speed up search process.

The paper is organized as follows. In section 2, we define predictable search paths and classify them into deterministic paths and non-deterministic paths. Section 3 proposes three mechanisms to be embedded into search algorithm to speed up search process. The predictable path search algorithm is introduced in section 4. In section 5, we propose an evaluation function to evaluate the expected probe number. Conclusion and future research are in the final section.

2. Predictable Search Paths

When a target agent is on its itinerary, if not only the servers to be visited but also the routing paths are known then this path is said to be a predictable search path. Predictable search paths can be classified into either deterministic or non-deterministic paths according to whether each hop is deterministic or not.

2.1 Deterministic Path

As an agent moves to a server and if its next hop is deterministic, the current server is said to be a deterministic server. A non-deterministic server is defined in a similar way. If all the servers to be visited by a target agent are deterministic servers then the routing path spans a list. Figure 1 shows a routing sequence of a target agent spans a list : server 7→server 12→server 8, etc.

S_7	S_{12}	S_8	S_2	S_{14}	S_1	S_{10}	S_4	S_{11}	S_6	S_3	S_5	S_{13}	S_9

Figure 1: A routing path spans a list.

As a real world example, a professor in Taipei plans to go to New York for a conference. Given that he wants to reside in a specific hotel and rent a car from a specific car-rental company. He commissions a service agent with the following task. The service agent should first book a ticket from an airline agency information server (AAIS) in Taipei; rent a car from a car-rental information server (CRIS) close to New York airport; and, at last, reserve a hotel from a hotel reservation information server (HRIS). In this case, the routing sequence of this service agent is AAIS→CRIS →HRIS, a predictable deterministic path.

2.2 Non-Deterministic Path

If some of the servers to be visited by a target agent are non-deterministic then the routing path spans a spanning tree. Figure 2 shows an example of spanning tree where servers S_2, S_5 and S_6 are non-deterministic.

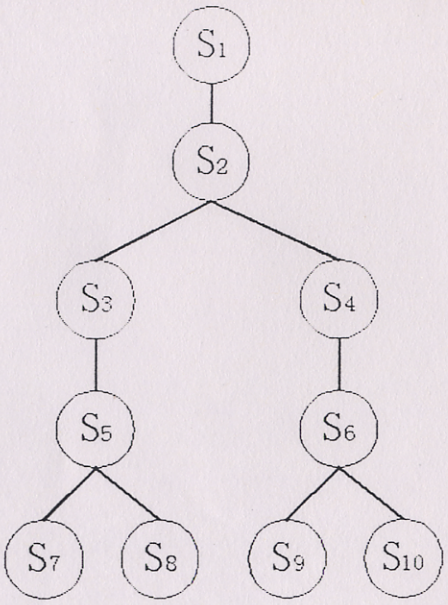

Figure 2: A routing path spans a spanning tree.

Continuing the previous example, if both hotel and car-rental companies are not bound to those specific ones. The routing path of the service agent becomes non-deterministic since many other hotels and car-rental companies are possible candidates. This path spans no more a sequential list but a spanning tree of all alternative servers.

3. Mechanisms Used to Speed Up Agent Search

Before we propose Predictable Path Search (PPS) algorithm (in section 4) for mobile agent search, we introduce three mechanisms to be embedded in PPS to speed up search process. During a search, a "probe" means to send a search agent to a certain server to locate the target agent. The criteria used to select a proper server to probe bases on maximizing those excluded (not-to-visit) servers and estimations of target agents residing probabilities. After each probe, PPS uses dynamic switch mechanism to determine whether switching intelligent search to sequential search. Intelligent search means to use PREM and MESM mechanisms in doing search. These mechanisms will be introduced in sections 3.

The main difference between traditional search and mobile agent search is the "mobility" of mobile agents. Doing with traditional search, we exclude those servers next to a probed server that are not visited yet. However, a mobile agent keeps moving while being searched. It is very possible that a target agent slips out of the search window and into one of those excluded servers. It turns out that

we loose the chance to find out target agent anymore. This is called a "slip out" problem. PPS prevents this problem by not excluding any unvisited servers after each probe.

To clearly explain our methodology, we define a couple of denotations for the rest of the paper in subsection 3.1. Three mechanisms used to speed up mobile agent search are introduced in subsections 3.2, 3.3 and 3.4, respectively.

3.1 Denotations

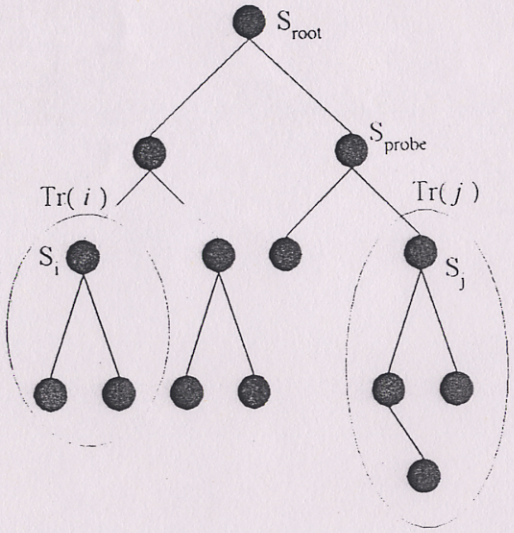

Figure 3: Illustration of slipped tree and residing tree.

In what follows, we define a set of denotations in Table 1 with Figure 3 as a reference figure.

Tr(i)	■ Tr(i) = a spanning tree with S_i as the root. ■ A special case while i = *root*, i.e. Tr(i) = Tr(*root*), where *root* is the root of the whole tree. Target agent visits the spanning tree from *root* and goes downwardly.
S_{probe}	■ The probed server.
Slipped Tree (ST) **and** **Residing Tree (RT)**	■ Either a slipped tree or a residing tree is defined upon a specific target agent. ■ Initially, slipped tree ST = NULL and residing tree RT = Tr(*root*). ■ Slipped tree consists of those servers where target agent is not in. ■ If a search agent probes S_{probe} and it is unvisited then target agent should not reside in tree Tr(S_{probe}). Thus Tr(S_{probe}) is added to ST. ■ Residing tree is defined to consist of those servers that target agent might possibly reside in. RT = Tr(*root*) – ST.

	■ For example, if S_i and S_j in Figure 3 are unvisited when probed. The target agent is not in both Tr(i) and Tr(j). Then Tr(i) and Tr(j) are added to ST. RT = Tr($root$)–Tr(i)–Tr(j).
θ	■ If S_{probe} is probed and found to be unvisited then Tr($probe$) shouldn't contain target agent and will then be added to slipped tree. ■ Target agent may slip into these subtrees after their roots are probed. ■ The residing probability of target agent that will be found in these subtrees is set to θ, and the probability of that be found in the residing list to 1–θ. ■ The values of θ are varied according to distinct slipped trees.
P(Tr(i))	■ The residing probability of spanning tree Tr(i).
N(Tr(i))	■ Number of servers in Tr(i).
P(S_i)	■ The residing probability of server S_i.
T_D	■ The time elapsed since target agent dispatched from originating server.
Ts_k	■ The service time (or duration time) that target agent served at server S_k.

Table 1: Denotations.

In Figure 2, tree Tr(1) has four leaves S_7, S_8, S_9 and S_{10}. It means the target agent has four possible paths. They are lists {S_1, S_2, S_3, S_5, S_7}, {S_1, S_2, S_3, S_5, S_8}, {S_1, S_2, S_4, S_6, S_9} and {S_1, S_2, S_4, S_6, S_{10}}. As an abbreviation, we use (S_1, S_7) to represent the list consisting of all servers on the routing path from S_1 to leaf node S_7. Here we have (S_1, S_7) = {S_1, S_2, S_3, S_5, S_7}.

We make the following two assumptions :

(A1) Target agents have equal probability to traverse Tr(1) by following each alternative lists from root to leaves.

(A2) Residing probabilities of each server in a list are equal at the beginning.

Continuing the example, we have probability 0.25 for each list from (A1). An example of (A2) is that S_5 is included in both lists (S_1, S_7) and (S_1, S_8) which consist of five servers, respectively. Thus the residing probability of S_5 in Tr(1) can be calculated accordingly.

$P(S_5)$ =

[prob. that a target agent traverses list (S_1, S_7)] × [residing prob. of S_5 in list (S_1, S_7)] +
[prob. that target agent traverses list (S_1, S_8)] × [residing prob. of S_5 in list (S_1, S_8)]
= 0.25×0.2 + 0.25×0.2 = 0.1

As a search agent probes an unvisited server S_{probe}, tree Tr(1) is divided into slipped tree Tr(*probe*) and residing tree [Tr(1)–Tr(*probe*)]. Tr(*probe*) is the subtree of Tr(1) with S_{probe} as its root. For example, if *probe* = 5 then Tr(5) contains servers S_5, S_7 and S_8. [Tr(1)–Tr(*probe*)] represents subtree Tr(*probe*) is excluded from Tr(1). If next probed server S_{next_probe} is also an unvisited server then slipped tree becomes [Tr(*probe*)+Tr(*next_probe*)] and residing tree becomes [Tr(1)–Tr(*probe*)–Tr(*next_probe*)]. The residing probabilities of servers in slipped tree and residing tree are θ and 1–θ, respectively.

3.2 Residing Probability Estimation Mechanism (RPEM)

One way to efficiently reduce the number of probes is to probe the server that has the largest residing probability. This can be realized through Residing Probability Estimation Mechanism. The residing probabilities of the servers in a list can be calculated as follows.

Assuming that an agent visits a set of servers, S_1, S_2,⋯, S_n, in sequence and stays at server S_k a duration time Ts_k. Given an elapsed time T_D, we define $P(S_c)$ to be the probability the target agent resides in *Sc*. $P(S_c)$ is determined by the following formula:

$$P(S_c) = P\left[\sum_{i=1}^{c-1} Ts_i \le T_D \le \sum_{i=1}^{c} Ts_i\right] = P\left[\sum_{i=1}^{c-1} Ts_i \le T_D\right] - P\left[\sum_{i=1}^{c} Ts_i \le T_D\right]$$

Let $f_i(t_i)$ be the independent probability density function of Ts_i where i = 1 to c. Then

$$P\left[\sum_{i=1}^{c} Ts_i \le T_D\right] = \iiint_{\sum_{i=1}^{c} t_i \le T_D} f_1(t_1) \times \ldots \times f_c(t_c)\, dt_1 \ldots dt_c$$

Assuming the duration time of target agent staying at each server follows normal distribution, i.e. $Ts_i \sim N(u_i, \delta_i)$ where i = 1 to c. Then

$$Ts_1 + \ldots + Ts_c \sim N\left(\sum_{i=1}^{c} \mu_i, \sqrt{\sum_{i=1}^{c} \delta_i^2}\right)$$

and $P\left(\sum_{i=1}^{c} Ts_i \le T_D\right)$ is simplified as follows.

$$P\left(\sum_{i=1}^{c} Ts_i \le T_D\right) = P\left(\frac{\sum_{i=1}^{c} Ts_i - \sum_{i=1}^{c} u_i}{\sqrt{\sum_{i=1}^{c} \delta_i^2}} \le \frac{T_D - \sum_{i=1}^{c} u_i}{\sqrt{\sum_{i=1}^{c} \delta_i^2}}\right) = P\left(\frac{\sum_{i=1}^{c} Ts_i - \sum_{i=1}^{c} u_i}{\sqrt{\sum_{i=1}^{c} \delta_i^2}} \le z\right)$$

= the area under the normal curve when x ranges from $-\infty$ to z

Since the probabilities of those lists to be traversed by target agent and the residing probabilities of those servers in these lists are known. The residing probability of every server in a spanning tree can be determined. With all these probabilities, RPEM is used to select the probed server that has the largest residing probabilities.

3.3 Maximum Excluded Servers Mechanism (MESM)

Once the residing probability is given, we then calculate the expected number of servers that can be excluded after each probe. The more servers excluded the faster search algorithm we get. Assuming that a search agent probes S_{probe} in Tr(*root*), if S_{probe} is visited then we know the target agent resides in Tr(*probe*). The probability that S_{probe} is visited is P(Tr(*probe*)). The other servers, [Tr(*root*)–Tr(*probe*)], would then be excluded. It is because that the target agent won't reside in [Tr*(root)*–Tr(*probe*)] any more. Hence, the expected number of excluded servers, *ENES*(Tr(*root*), *probe*), can be expressed as follows.

$$ENES(\mathrm{Tr}(root), probe) = \mathrm{P}(\mathrm{Tr}(probe) \times \mathrm{N}(\mathrm{Tr}(root) - \mathrm{Tr}(probe)))$$

where P(Tr(*probe*)) equals to the sum of the residing probabilities of all those servers in Tr(*probe*).

Maximum Excluded Servers Mechanism is realized through the following two steps:

[Step 1] Using *ENES*(Tr(*root*), *probe*) to calculate the expected number of excluded servers of all servers.

[Step 2] Selecting to probe the server that causes largest expected number of servers excluded.

3.4 Dynamic Switch Mechanism (DSM)

As a search agent comes to a probed server, it can read the timestamp of a target agent from this server to know when and how long the agent had ever been there. If it shows the target agent just visited and left then it should reside in a nearby server. In this case, it might be more efficient if we dynamically switch from current (intelligent) search to sequential search. The decision of making a dynamic switch is based upon comparing probe number threshold and the timestamp of target agent left the probed server, S_{probe}. Probe number threshold is defined to be the expected number of probing target agent.

PPS calculates the residing probabilities of those servers next to S_{probe} that target agent might reside in. It then uses this information to evaluate the expected number of probes using both sequential search and intelligent search. If the number of sequential search is less than that of intelligent search, search is then switched to sequential search. Otherwise, it remains intelligent search.

4. Predictable Path Search (PPS) Algorithm

In this section, we propose a Predictable Path Search (PPS) algorithm to show how to do mobile agent search on predictable search paths by embedding those mechanisms in section 3. At the beginning we have a spanning tree Tr(*root*), a residing tree Tr(*root*), and a slipped tree NULL. The search algorithm goes as follows.

Step 1 : Using RPEM and MESM, search agent selects a certain S_{probe} to maximize the expected number of *ENES*(Tr(*root*), *probe*).

Step 2 : Search agent probes S_{probe}. If target agent is found in S_{probe} then the search is done. Else, there are two cases.

[Case 1] S_{probe} is visited

Tr(*root*) = Tr(*probe*). Slipped tree = slipped tree $\cap$ Tr(*probe*). Residing tree = residing tree $\cap$ Tr(*probe*).

[Case 2] S_{probe} is unvisited

Tr(*root*) = Tr(*root*). Slipped tree = slipped tree $\cup$ Tr(*probe*). Residing tree = residing tree – Tr(*probe*).

Step 3 : Using DSM to determine whether switching to sequential search. If yes, go for sequential search. Otherwise, go to step 1.

In [Case 1], if S_{probe} is visited then target agent should reside in Tr(*probe*). Those servers that are not in Tr(*probe*) are to be excluded. Hence, Tr(*root*) is reset to Tr(*probe*). Those servers in the slipped tree but not in Tr(*probe*) are excluded. The servers in residing tree are excluded similarly. Therefore, the slipped tree is set to slipped tree $\cap$ Tr(*probe*) and residing tree to residing tree $\cap$ Tr(*probe*).

In [Case 2], if S_{probe} is unvisited then target agent should not reside in Tr(*probe*). They are not excluded from Tr(*root*), but added into slipped tree. Slipped tree = slipped tree $\cup$ Tr(*probe*). Residing tree = residing tree – Tr(*probe*).

As an illustration, we assume a given tree with root = S_{root} and slipped tree = Tr(i) + Tr(j) in Figure 3. When target agent probes S_{probe}, there are two cases.

[Case 1] S_{probe} *is visited* (Figure 4)

Tr(*root*) = Tr(*probe*). Slipped tree = slipped tree $\cap$ Tr(*probe*) = Tr(j). Residing tree = residing tree $\cap$ Tr(*probe*) = Tr(probe)–Tr(j).

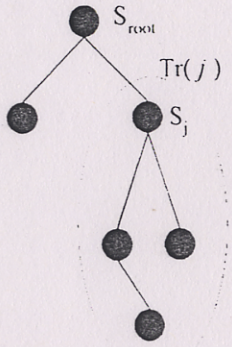

Figure 4: S_{probe} is visited.

[Case 2] S_{probe} *is unvisited* (Figure 5)

Tr(*root*) = Tr(*root*). Slipped tree = slipped tree $\cup$ Tr(*probe*) = Tr(i) + Tr(*probe*). Residing tree = residing tree – Tr(*probe*) = Tr(*root*)–Tr(i)–Tr(*probe*).

Example:

In Figure 6, we assume that there is a target agent routing path S with S_1 and S_n as head and end, respectively. After a number of probes, path S is divided into slipped tree and residing tree. Now, we have head = S_{head}, residing tree = List(S_{head}, S_{slip-1}), and slipped tree = List(S_{slip}, S_n).

Step 1 : Using RPEM and MESM, search agent selects S_{probe} to maximize the expected number of *ENES*(Tr(*root*), *probe*).

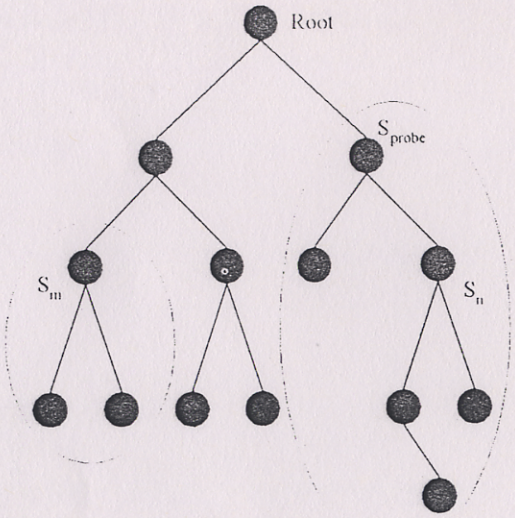

Figure 5: S_{probe} is unvisited.

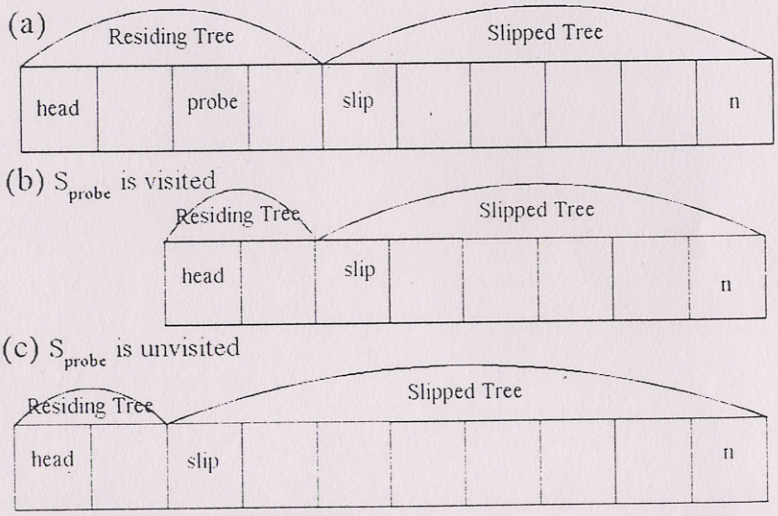

Figure 6: Search a target agent using PPS.

Step 2 : Search agent probes S_{probe}. If target agent is found in S_{probe} then search is done. Else, there are two cases.

[Case 1] S_{probe} is visited

Tr(*root*) = Tr(*probe*). Slipped tree = slipped tree ∩ Tr(*probe*). Residing tree = residing tree ∩ Tr(*probe*). S = (S_{probed}, S_n). Slipped tree = (S_{slip}, S_n) ∩ (S_{probed}, S_n) = ($S_{max(slip, probed)}$, S_n). Residing tree = (S_{head}, S_{slip-1}) ∩ (S_{probed}, S_n) = (S_{probed}, S_{slip-1}).

[Case 2] S_{probe} is unvisited

Tr(*root*) = Tr(*root*). Slipped tree = slipped tree ∪ Tr(*probe*). Residing tree = residing tree – Tr(*probe*). Slipped tree = $(S_{slip}, S_n) \cup (S_{probe}, S_n) = (S_{min(slip, probe)}, S_n)$. Reside tree = $(S_{head}, S_{slip-1}) - (S_{probe}, S_n) = (S_{head}, S_{min(slip-1, probe-1)})$

Step 3 : Using DSM to determine whether switching to sequential search. If yes, go for sequential search. Otherwise, go to step 1.

5. Evaluation Function of Expected Probe Number (EFEPN)

In this section, we propose an Evaluation Function of Expected Probe Number (EFEPN) to calculate the expected number of probes. We define *f*(Tr(*probe*), *slipped tree*, *residing tree*, *θ*, *probe*) to be expected probe number of Tr(*root*) where S_{probe} is the probed server. Initial setting of parameters are as follows: root =1, residing tree = Tr(1), slipped tree = NULL, residing probability of slipped tree (θ) = 0, residing probability of residing tree = 1. RPEM and MESM mechanisms are used to choose a server, say S_{probe}, to probe. In what follows, we illustrate how to come out EFEPN from distinct S_{probe}.

(1) If $S_{probe} \in$ *residing tree*

[Case 1] *Target agent is found*

The recursive function stops and returns 0. Residing probability is $P(S_{probe})$.

[Case 2] S_{probe} *has been visited*

Target agent resides in Tr(*probe*)–S_{probe}. Residing probability is P(Tr(*probe*))–$P(S_{probe})$. Tr(root) = Tr(probe). Slipped tree = slipped tree ∩ Tr(*probe*). Residing tree = residing tree ∩ Tr(*probe*).

[Case 3] S_{probe} *is unvisited*

Target agent is not in Tr(*probe*). Residing probability is 1–P(Tr(*probe*)). Tr(*root*) = Tr(*root*). Slipped tree = slipped tree ∪ Tr(*probe*). Residing tree = residing tree – Tr(*probe*).

To summarize the above, we have:

If $S_{probe} \in$ *residing tree*

f(*root*, *slipped tree*, *residing tree*, θ, *probe*)
= 1 + [P(Tr(*probe*))–P(S_{probe})] × f(Tr(*probe*), *slipped tree* ∩ Tr(*probe*), *residing tree* ∩ Tr(*probe*), θ, *next_probe*) + [P(Tr(*probe*))] × f(Tr(*root*), slipped tree ∪ Tr(*probe*), residing tree – Tr(*probe*), θ, *next_probe*)

(2) If $S_{probe} \in$ *slipped tree*

[Case 1] *Target agent is found*

The recursive function stops and returns 0. Residing probability is P(S_{probe}).

[Case 2] S_{probe} *has been visited*

Target agent resides in Tr(*probe*)–S_{probe}. Residing probability is P(Tr(*probe*))–P(S_{probe}). Tr(root) = Tr(probe). Since target agent slips into slipped tree, we reset residing tree = Tr(*probe*) and slipped tree = NULL.

[Case 3] S_{probe} *is unvisited*

Target agent is not in Tr(*probe*). Residing probability is 1–P(Tr(*probe*)). Tr(*root*) = Tr(*root*). Slipped tree = slipped tree ∪ Tr(*probe*). Residing tree = residing tree – Tr(*probe*).

To summarize the above, we have:

If $S_{probe} \in$ *slipped tree*

f(*root*, *slipped tree*, *residing tree*, θ, *probe*)
= 1 + [P(Tr(*probe*))–P(S_{probe})] × f(Tr(*probe*), *NULL*, Tr(*probe*), θ, *next_probe*)
+ P(Tr(*probe*)) × f(Tr(*root*), slipped tree ∪ Tr(*probe*), residing tree–Tr(*probe*), θ, *next_probe*)

Example: Given a tree in Figure 3, EFEPN is expressed as f(*root*, Tr(i) + Tr(j), Tr(*root*) – Tr(i) – Tr(j), θ, *probe*).

$S_{probe} \in$ *residing tree*

[Case 1] *Target agent is found*

Residing probability is $P(S_{probe})$.

[Case 2] *S_{probe} has been visited*

Residing probability is $P(Tr(probe))–P(S_{probe})$. The tree in Figure 3 reduces to that in Figure 4. Evaluating function of that tree becomes $f(probe, Tr(j), Tr(probe) - Tr(j), \theta, next\ probe)$.

[Case 3] *S_{probe} is unvisited*

Residing probability is $1–P(Tr(probe))$. The tree in Figure 3 becomes that in Figure 5. Evaluating function of that tree becomes $f(root, Tr(i) + Tr(probe), Tr(root) - Tr(j) - Tr(probe), \theta, next\ probe)$.

To summarize the above, we have:

$$
\begin{aligned}
&f(probe, Tr(j), Tr(probe)–Tr(j), \theta, next\ probe) \\
&= 1 + [P(Tr(probe)) - P(S_{probe})] \times f(probe, Tr(j), Tr(probe) - Tr(j), \theta, next\ probe). \\
&\quad + [\ 1 - P(Tr(probe))] \times f(root, Tr(i) + Tr(probe), Tr(root) - Tr(j) - Tr(probe), \theta, next\ probe).
\end{aligned}
$$

6. Conclusion

In this paper, we propose a Predictable Path Search algorithm to locate mobile agents on predictable search paths. This algorithm consists of three modules : Residing Probability Estimation Mechanism, Maximum Excluded Servers Mechanism, and Dynamic Switch Mechanism. These mechanisms could efficiently speed up search process. This work can be extended to non-predictable path which spans a graph [2,3]. In which, those servers to be visited are known, however, routing paths of service agents are non-predictable.

References

[1] Hung-Chin Jang, Yao-Nan Lien, Jyh-Shyan Huang, and Fu-Han Liu "New Intelligent Search of Mobile Agents", First Agent Technology Workshop, Taipei, Taiwan, Dec. 4, 1997.

[2] Hung-Chin Jang, Yao-Nan Lien, Jyh-Shyan Huang, and Fu-Han Liu, "Non-Deterministic Binary Search of Mobile Agents", 1997 National Computer Symposium (NSC '97), Taiwan, R.O.C., Dec. 22-23, pp.89-94, 1997.

[3] Hung-Chin Jang, Yao-Nan Lien, and Jyh-Shyan Huang, "Using Cut Vertices Search to Search Mobile Agents on a Non-Deterministic Path", the 4th Mobile Computing Workshop, National Chiao Tung Unvi., Taiwan, R.O.C., March 25-26, 1998.

[4] Yao-Nan Lien, "An Open Intelligent Messaging Network Infrastructure for Ubiquitous Information Service," Proc. of the First Workshop on Mobile Computing, Hsing-Chu, Taiwan, April 1995, pp. 2-9.

[5] Yao-Nan Lien and Chun-Wu Leng, "On the Search of Mobile Agents," Proc. of the IEEE Personal, Indoor, and Mobile Radio Conference, Taiwan, Oct. 1996, pp. 703-707.

[6] Yao-Nan Lien, Fuhan Liu, Chun-Wu Leng and Wen-Shyan Chen, "Intelligent Search of Mobile Agents", 1997 International Conference on Computer System Technology for Industrial Applications, April, 1997, pp. 110-116.

[7] Yao-Nan Lien, Fuhan Liu, Wen-Shyen Chen and Chun-Wu Leng, "Asymmetric Binary Search of Mobile Agents", 1997 International Symposium on Multimedia Information Processing, Dec. 1997, pp. 294-299.

[8] Maes, "Agents that reduce work and information overload", CACM, July 1994, pp. 30-41.