調適型網路精靈搜尋法
Adaptive Sarch of Mobile Agents

連耀南* 劉富翰* 陳文賢** 冷俊武*
Yao-Nan Lien, Fuhan Liu, Wen-Shyen Chen, Chun-Wu Leng

* Department of Computer Science, Natinal Chengchi University
國立政治大學資訊科學系
** Department of Computer Science, Natinal Chung-Hsing University
國立中興大學資訊科學系

摘要

在行動計算環境中，使用者可送出一個網路精靈於網路上依指令循序到各伺服器上取得服務。因此，『網路精靈之搜尋』顯然是管理行動資訊系統的一個重要技術。在本篇研究報告中，我們提出一個異於傳統二元搜尋方式的全新策略。在此演算法中當搜尋精靈於伺服器尋找上目標精靈，而其卻已離開伺服器時，搜尋精靈可運用目標精靈離開某伺服器的時間，以更精確估算目標精靈的落點，如預估目標精靈離開未久，搜尋精靈便可由原搜尋法轉換為線性搜尋方式，依序就近搜尋目標精靈。在我們實驗中發現本演算法比先前所提出的盲目搜尋或智慧型搜尋法，可增強搜尋的效益。

關鍵字：行動計算，網路精靈

Abstract

In a mobile computing environment that supports mobile agents, a client can send an agent to visit a sequence of servers in the network. Tracking the location of agents becomes a critical problem in managing a mobile agent service network. In this research, we propose a new search strategy that can change the basic algorithm during the search of a mobile agent. This strategy makes use of the up-to-date information obtained form the servers that have been visited by the target agent. By knowing the departure time of the target agent from a server, the search agent can either recalculate the location prediction or simply switch from the original search strategy to the sequential search. When applying this strategy to the previously proposed blind and intelligent searches, we can obtain a significant improvement on the search efficiency.

Keywords: Mobile Computing, Mobile Agent.

§ This work is partially supported by ITRI/CCL grant under MOEA Distributed Information Processing Program (Contract number: G3-87064).

1. Introduction

1.1 Agent and Agent Mobility

A ubiquitous information service network provides information to the users any time anywhere [1,8,9]. Client users can access to the network through a wireless mobile communication network. In addition, the service network must provide some mechanisms allowing client users access to various information resources conveniently, which is referred to as *mobile computing* [1,2,3].

Because the distributed computing technology is not sufficiently mature to support mobile computing in such a scale, client users have to access network resources in a prescriptive fashion by interacting with individual servers one by one to accomplish a complicated task. However, in most mobile computing environments, the nature of communications is intermittent and the battery energy is limited so that it is very difficult and expensive to accomplish a complicated task. The *mobile agent* paradigm, which allows clients to interact with multiple servers in a dynamic fashion has been brought up to cope with this problem [1,2,3,8,10].

Simply speaking, a *mobile agent* is an electronic message that carries a computer program, whether procedural or declarative, which can be executed by the receiving servers on behalf of the originating client. The program in the message can also instruct a receiving server to forward automatically the message itself to another server, on which the program is executed continuously in a pipeline fashion. Good examples can be found in [3].

Since an agent may move continuely in a service network, the originating client may not be able to trace or control its operation directly. A service network must provide some mechanisms allowing its clients to trace and control these agents. This problem is referred to as the *agent mobility management*.

1.2 Open service network architecture

To establish a large scale service network that can support mobile agents, we proposed an open service network architecture in [3], which separates service networks from transport networks. It also allows

services of any scale and any quality to be introduced into the network easily. Readers are referred to [3] for details.

On the top of the open architecture mentioned above, we also proposed a hybrid operation mode in [2] that allows a service provider to offer both centralized and distributed operation modes to its subscribers. Subscribers can choose to use their own Internet facility, called *Home Base Node* (HBN), to share the OA&M (Operation, Administration, and Maintenance) duties. (Typical OA&M functionalities are billing, client location register, agent tracking and control, agent status holder, etc.) At their own expense, subscribers can also designate some OA&M duties to the centralized facility managed by the service providers. In this paper, we assume that a service network that supports mobile agents is established based on the proposed open architecture and managed using the proposed operation infrastructure [2,3,5].

1.3 Search of a mobile agent

After an agent is dispatched into a service network, the client or the network manager may need to know its current location in order to inquire its status, or to control its execution, etc. A simple way is to send another agent, called *search agent*, to search the original agent along the original path, or to broadcast a message to every server where the agent might have visited. There are some problems associated with these straightforward solutions:

1. Sending many messages over a wireless network may be too expensive.
2. A sequential search may take too much time.

Therefore, better ways to locate an agent are in need. In [4], we proposed and analyzed several search strategies, which will be described in the next section. Among the proposed strategies, the intelligent search is the most promising one if the service time of each individual task can be estimated using prior statistics. To reduce the computation time that is required to estimate the best possible location of a target agent, we studied various ways based on the aggregated statistical properties to improve the computational performance of the intelligent search strategy (IBS) proposed in [4]. They will be summarized in Section 2.

1.4 Adaptive Search

The search strategies previously proposed are all static, which means that the search strategy is fixed when the search agent is dispatched to the network. For a particular search strategy, the search route is completely determined by the path taken by the target agent. They do not make use of other information, such as the departure time when the target agent left a server, to adjust the search sequence.

One way to improve the proposed search strategies is to recalculate the residing probability, based on the departure time when the target agent left, each time after a search agent probes a server. For instance, if an IBS search agent finds out that the target agent has already visited and left the currently probed server, it will know that the estimation is not accurate. The search agent can use this departure time information to recalculate the Most Probable Server (MPS). This may be better than blindly searching forward along the original planed path.

The rest of this paper is organized as follows: Section 2 will review the search strategies proposed in [4,6,7]; Section 3 will discuss the various adaptive search strategies; Section 4 will illustrate our experiments; finally, Section 5 will discuss some complicated issues and future research directions.

2. Previous Work

The strategies proposed in [4] can be categorized into two types: blind and intelligent searches. The intelligent search strategies make use of prior knowledge about the execution of all tasks, while blind search strategies don't.

The following notations will be used in the following sections:

- $S = \{S_1, S_2, \ldots, S_n\}$: the set of distinct servers visited by an agent.
- T: the elapsed time since the target agent was originated.
- T_k: the time duration that the target agent stayed at server S_k. It is called the *service time* at server S_k.

In this paper, we assume that the target agent visits $\{S_1, S_2, \ldots, S_n\}$ sequentially and nonrecursively. Further, the time for the target agent to move from one server to another is considered nominal and is neglected.

Note that a recursive execution sequence can be easily expanded to a nonrecursive one by treating each visit (i.e. the target agent visits a server) a new server rather than a revisit. Further, if the time for an agent to move from one server to another is too long to be ignored, the agent migration can be modeled as a server so that our algorithms and analytical results can be applied without any modification.

2.1 Chase-from-holder Algorithms

In the hybrid operational infrastructure we proposed in [2], users are encouraged to use their own HBNs to participate in the management of their agents. One possible usage of HBNs is to store the current status of agents, including agents' locations. Thus, the current reported location of an agent can be obtained right in the status holder (i.g. HBN) of its originating user. If the accurate current location is needed, the search agent can visit the reported location first and proceed with a sequential search right from there if the target agent had passed that server. This type of algorithms is called the *Chase-from-holder algorithms*. In face, it is a

combination of tracking and searching methods.

Obviously, one major problem with the Chase-from-holder algorithms is the extra expense to update status periodically. The update cost has at least two major components: the induced network traffic and the resource consumption in the status holder. The system resources could be very expensive if the status holder is designated to the network management center. Thus, the trade-off between update cost and status availability must be carefully balanced. Depending on its status inquiry frequency, a client may choose to command an agent report its status either completely or selectively. The following search strategies are useful when the current location of an agent is not available in its status holder.

2.2 Binary Search Algorithms

The *Basic Binary Search* algorithm (BBS) is similar to the binary search in searching a data object in a sorted list. The search agent probes the middle server in the search list and excludes half of the servers out of the search list at a time. The search is performed recursively until the target agent is found or the list is exhausted. In average, the number of probes required to locate the target agent is in the order of $log(n)$, where n is the number of servers in the search list. BBS might be a very good search strategy for blind searches. However, BBS may fail to locate the target agent if the target agent continues to move during the search. During the course of search, some unvisited servers may be excluded out of the search list after a server is probed. Unfortunately, the target agent may *slip through* the search window so that the servers it visits afterward are not included in the search list. As a result, the search agent will fail to locate the target agent. Although BBS is naive and faulty, it provides a basis to mutate into other search strategies as well as a baseline for performance comparison.

The *Extended Binary Search* (EBS) algorithm corrects the slip-through problem by not excluding any unvisited server out of the search list at the cost of demanding more search probes. Fortunately, the average number of probes required to locate the target agent is still in the order of $log(n)$, only with a larger coefficiency.

In [7], we found a very surprising fact that in Extended Binary Search algorithm, the best pioint (server) to probe is not right on the middle of the serach list. This is because forward and backward probes are treated differently. Probing a visited server can exclude some servers out of the search list, while probing an unvisited server can't. In our simulation study, we found that the best probing range is between 0.25 and 0.4 of the search list starting from the head.

Above search strategies are classified as *blind search* because they do not use prior knowledge about the status of the target agent and servers.

2.3 Intelligent Search

If a client has a good estimation on the service time in each server, he/she may be able to guess the approximate location of an agent. By using this information in a search, we will be able to reduce the search time significantly. The term *intelligent search* here reflects the fact that these algorithms make use of service time statistics and thus, is non-blind.

We assume that an agent visits a set of servers, $S_1, S_2, \ldots, S_n$, in sequence and it stays at each server, says S_k, for a time duration of T_k. At any elapsed time T, the current location of the agent, S_c, can be determined by the following formula:

$$\sum_{i=1}^{c-1} T_i \le T < \sum_{i=1}^{c} T_i.$$

However, it is impractical to know in advance exactly how long the target agent will stay at each server. The service time in each server is most likely probabilistic and can be pre-estimated through either sample collection or experiments. Furthermore, for security and privacy reasons, it may be more practical to obtain statistics rather than detailed execution time records from servers. As a result, the location of the agent is probabilistic. We can calculate the location of the target agent with the highest probability, next highest, etc., and then search the target agent according to the order of probability. Assuming that the service time of each task is uniformly distributed, we derived a formula to calculate for each server the probability that a target agent might reside [4]. For simplicity, this probability is called the *residing probability*. In our proposed Intelligent Binary Search algorithm (IBS), the search list is presorted according to the calculated probabilities. Then, the search agent probes the servers in the search list sequentially.

Intelligent search algorithms are much better than blind search algorithms because they make use of prior knowledge about the service time of each task in each server. However, it requires the service time to be predictable. In other words, the algorithm assumes the target agent and the service network work normally without any exception. Thus, these algorithms have better be used under healthy operation conditions. They may not be appropriate to be used in handling exceptional cases.

Another problem associated with the intelligent search strategy is its quadratic computation time to calculate each individual residing probability. Even if the computation overhead can be compensated by using faster processors, the collection of the entire service time distribution from every server will still consume significant system and network resources. To reduce computation time, we proposed to use aggregated statistical properties to predict the location of a mobile agent [6]. Our analysis and experiments showed that, under a variety of circumstances, the mean service time is a reliable index to predict the location of a mobile agent. Readers are referred to [6] for details.

3. Adaptive Search

As mentioned in Section 1.4, the search strategies proposed so far are all static. An adaptive search strategy makes use of the information available in all servers that have been probed to improve the search efficiency. Considering the situation that the target agent just left a server when the search agent is probing the server, the search agent can go directly to the "next" server instead of continuing the original blind search. There is a high probability catching the target agent within a few sequential probes. Further, in an intelligent search, the search agent can recalculate the residing probability, based on the departure time when the target agent left, each time after the search agent probes a server. For instance, if an IBS search agent finds out that the target agent has already visited and left the currently probed server, it will know that the estimation of the MPS is inaccurate. The search agent can use this departure time information to recalculate the MPS. This may be better than blindly searching forward along the original planed search sequence.

This adaptive technique may improve the performance of blind search strategies by a significant margin. In this paper we will focus on applying this adaptivity to the blind searches.

3.1 Binary-Sequential Search (BSS) Algorithm

We assume that every server maintains an execution log that stores various execution status including the departure time when the target agent moved to the next server. By making use of this information, a search agent can choose the best search strategy dynamically. We first investigate the performance impact on the EBS when the adaptivity is applied. In the following Binary-Sequential Search algorithm, the search is divided into two phases: *binary phase* and *sequential phase*. The search agent first uses EBS to search the target agent in the binary phase, and then switches to the sequential phase using a sequential search if the target agent "just" left the currently probed server not long ago.

Binary-Sequential Search (BSS) Algorithm

```
Main {
  SrhSet = {S1,S2, . . . ,Sn}
  Sc = Middle Server in the SrhSet
  BSSrh(target, SrhSet, Sc )
  }
Procedure BSSrh (target, SrhSet, Sc )
 (1) if (SrhSet = empty)
 (2) then return (NOT_FOUND)
 (3) if (target in Sc)  return (Sc)
 (4) if (target had visited Sc)
 (5) then {
 (6)    SrhSet = {Sc+1, . . . ,Sn}
 (7)    if ( target is near )
 (8)    then {
          while ( SrhSet is not empty) {
 (9)        Sc = head of SrhSet
(10)        if (target in Sc) return(FOUND)
(11)        SrhSet -= Sc
        } } }
(12) else  {
(13)    h = index of head of SrhSet
(14)    c = h + 1/2×(c−h+1)
      }
(15) return ( BSSrh(target, SrhSet, Sc ))
```

There are some parameters yet to be determined. The most critical one is the proper condition to have the search agent switch from a search strategy to another (Line 7 of BSS). One simple condition is to use the elapsed time since the target agent left the currently probed server as a measure. It requires an intensive experiment to determine the proper values.

The second parameter to be investigated is the best probing range to be used in EBS. As mentioned in Section 2, the best server to probe may not be right on the middle of the search list due to the different treatment of forward and backward probes. Intuitively, the adaptivity may have impact on the probing range as well. Because the search agent can make use of the log information to obtain a more accurate prediction, it may prefer probing a visited server. Therefore, we could adjust the range of search probes further to increase the possibility of probing a visited server.

3.2 The ABSS algorithm

The above BSS algorithm is then modified as follows to take this idea (asymmetric probing range) into account:

Asymmetric Binary Sequential Search (ABSS) Algorithm

```
Main {
  SrhSet = {S1,S2, . . . ,Sn}
  c = n × θ
  ABSSrh(target, SrhSet, Sc )
  }
Procedure ABSSrh (target, SrhSet, Sc )
 (1) if (SrhSet = empty) return (NOT_FOUND)
 (2) if (target in Sc) return (Sc)
 (3) elseif (target had visited Sc) {
 (4)  SrhSet = {Sc+1, . . . ,Sn}
 (5)  if (target is near) SeqSrh(target,SrhSet)
 (6)  else {
 (7)    c = c+(n−c+1) × θ
       }}
 (8) else  {
 (9)  h = index of head of SrhSet
(10)  c = h + θ×(c−h+1)
      }
(11) return (ABSSrh(target, SrhSet, Sc))
```

The parameter θ in ABSS is a value to be researched. The procedure *SeqSrh* is a sequential search algorithm. (The details are ignored. It is the same as the block from line (6) to (11) in BSS). Similar to BSS, the most critical decision is line (5) which is to decide the proper condition to switch to the sequential search.

It is not difficult to figure out that it is more beneficial switching from the EBS to the sequential search if the target agent is within the range of $\log_2$(length of the search list) since the average number of search probes for the EBS is in the order of $O(log_2N)$. In other words, if the target agent is within the range of $\log_2$(length of the search list) away from the currently probed server, the search agent had better switch to the sequential search. The main problem to carry out this idea is that the search agent might not know exactly whether the target agent is within that range or not. One possible way is to make a prediction and accept the possible penalty caused by the estimation errors. Thus, it is a challenge to find a practical method that can minimize this penalty. Since the theoretical analysis is non-trivial, we proceeded with a simulation study for the initial observation.

3.3 Gambling Range

Although the elapsed time is a good measure to determine whether to switch from the EBS to the sequential search, it depends on the execution time of the target agent in each server and, thus, is inconvenient to be used in simulation studies. Therefore, we propose the following model to facilitate the simulation study. We first define the following terms:

- *target distance:*
  the distance (number of servers) between the currently probed server and the server where the target agent currently resides;
- *estimated target distance:*
  the target distance that the search agent estimates;
- *gambling range:*
  the farthest estimated target distance that the search agent is willing to switch from EBS to the sequential search.

Since there is no practical way to know exactly the target distance, a search agent can only estimate it and make a "gamble". The choice of gambling range depends on the accuracy of the estimation and the trade-off between the "win" and "lose". A winning gamble will reduce the number of search probes, while a losing gamble will increase it. An aggressive search strategy could take a longer gambling range, while a less aggressive search strategy could take a shorter one. In the following section, we will present a series of simulation studies that, hopefully, can offer some knowledge to help users in determining their best gambling strategies.

4. Simulation Experiments

In our simulation experiments, two types of gambling strategies are compared. A *static gambling strategy* uses a fixed gambling range regradless the length of the search list; while a *dynamic gambling* strategy changes its gambling range according to the current length of the search list. To simplify the simulation, the search agent doesn't make any prediction. Instead, it switches to the sequential search directly when the target agent is within the gambling range. In other words, the decision is made by the simulator, not the search agent in the simulation. Thus, the performance of a search algorithm that uses a dynamic gambling range represents a theoretical upper bound, in which no penalty is incurred due to any estimation error. On the other hands, a search algorithm that uses a static gambling range will be closer to the reality since it uses a less aggressive gambling strategy for its lack of prediction accuracy.

4.1 Binary-Sequential Search

We first studied the performance of symmetric BSS where the search agent always probes the middle server of the search list in the binary search phase. The number of servers is set at 30. The BSS was simulated with the following gambling ranges: 0, 1, 2, 3, and dynamic. When the gambling range is set at 0, the BSS is actually a regular EBS without any adaptivity at all. The simulation results are shown in Figure 1. As we can see that the performance of the one with dynamic gambling range is the best representing an upper bound, while the one with 0 gambling range is the worst. The performance of those with a non-zero gambling range are all better than EBS. The adaptivity actually offers a significant performance enhancement.

4.2 Asymmetric Binary-Sequential Search

We then repeated the simulations for the ABSS. All parameters but the probing range are the same. The probing ranges are first set at 0.4 as suggested by our previous study [7]. The simulation results are shown in Figure 2. The adaptivity actually offers some performance enhancement to ABSS. As we have expected, the performance of ABSS is better than BSS due to the performance gain obtained from the asymmetric probing range. The comparison is shown in Figure 3 where the probing range is set at 0.4.

4.3 Observations

From the simulation results shown above, we can see that the adaptivity actually reduces the number of search probes for the EBS. It also prefers an asymmetric probing range in the binary phase because probing a visited server may obtained some helpful information in predicting the location of the target agent. Furthermore, an aggressive search strategy that chooses a longer gambling range may not always better than a less aggressive one due to the estimation errors.

5. Concluding Remarks

To locate an agent in a mobile agent service network is a critical problem. In this paper, we demonstrated a new technique, adaptivity, that can be applied to the most existing search algorithms to reduce the number of search probes. An adaptive search strategy will take different search strategies adapting to the most current

estimation based on the most current knowledge about the target agent. In this paper, a search agent can use the elapsed time since the target agent left a server to predict the current location of the target agent. Due to a lack of knowledge about the exact execution time in each server, this location prediction is restricted within the near neighborhood of the currently probed server. Our simulation studies show that the adaptivity actually reduces the number of search probes for the EBS.

References

1. T. Imielinski and B. R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Communication of ACM*, August 1994.
2. Yao-Nan Lien, "Client and Agent Mobility Management," *Proc. of the Second Workshop on Mobile Computing*, Hsing-Chu, Taiwan, March 1996, pp. 141-152.
3. 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.
4. 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.
5. Yao-Nan Lien, et. al., "FlyingCloud: A Mobile Agent Service Network", *Proceedings of the International Conference on Distributed Systems, Software Engineering, and Database Systems*, Dec. 1996, pp. 177-183.
6. Yao-Nan Lien, Fuhan Liu, Chun-Wu Leng and Wen-Shyen Chen, "Intelligent Search of Mobile Agents", *Proceedings of the 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 ", *Submitted to the 1997 International Symposium on Multimedia Information Processing.*
8. P. Maes, "Agents that reduce work and information overload", *CACM*, July 1994, pp. 30-41.
9. M. Weiser, "The computer for the 21st century", *Scientific America*, 1992, pp. 94-104.
10. James E. White, "Telescript Technology: The Foundation for the Electronic Marketplace", General Magic, Inc.

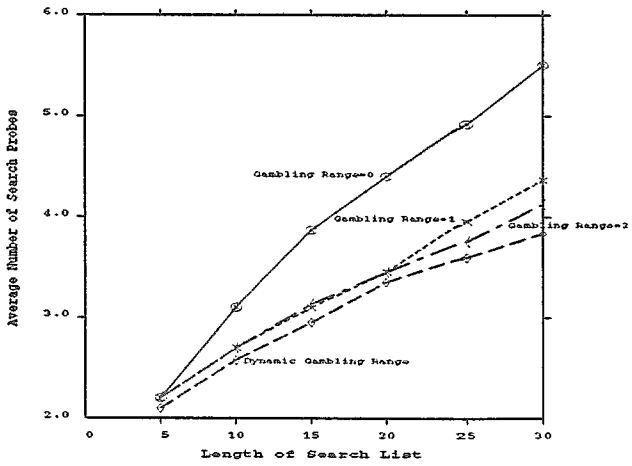

Figure 1. Simulation results for BBS algorithm

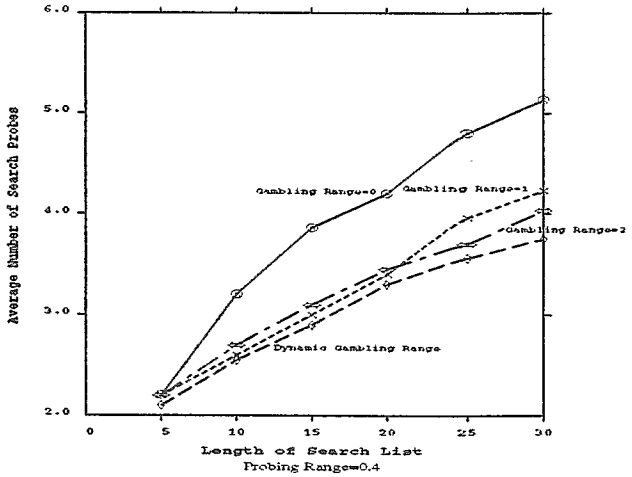

Figure 2. Simulation results for ABSS algorithm

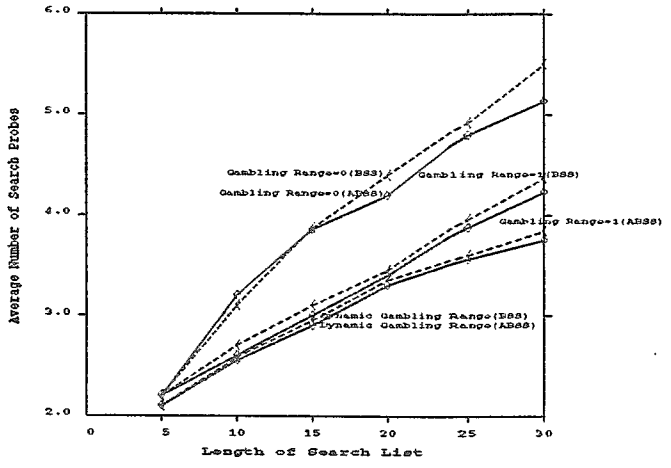

Figure 3. BSS vs. ABSS