An online transaction always retrieves a large amount of information before making decisions. Currently, the parallel methods for retrieving such information can only provide a similar performance to serial methods. In this paper we first perform an analysis to determine the factors that affect the performance of existing methods, i.e., HQR and EHQR, and show that the several of these factors are not considered by these methods. Motivated by this, we propose a new dispatch scheme called AEHQR, which takes into account the features of parallel dispatching. In addition, we provide cost models that determine the optimal performance achievable by any parallel dispatching method. Using experimental comparison, we illustrate that the AEHQR is significantly outperforms the HQR and EHQR under all conditions.

Keywords: Information retrieval, mobile agents, clone, dispatch, parallel

1. INTRODUCTION

Information retrieval that retrieves user specified information is becoming increasingly important due to the explosive popularization of the WWW and the dramatically increasing amount of data available via the Internet [11]. In particular, a user query for retrieving some kind of information (user query, for short) specifies a query region \( q_R \) and a constraint \( q_c \), and retrieves the set of objects that will intersect \( q_R \) under the constraint \( q_c \). While the majority of research (see [25] for a survey) in mobile agent-based information retrieval aims at user queries on individual objects that satisfy certain predicates, the motivations of this work is that many queries have multiple objectives are possible decomposed into a set of semantically independent sub-queries.

Given a query \( q_R \) and \( q_c \), if \( q_R \) can be decomposed into a set of independent sub-queries \( q_{i} (i = 1, 2, \cdots, N) \), the query is called a complex query; otherwise, it is a simple query. For example, a query \( q_R \) "find all houses that are less than $300,000 in Shinagawa-ku or Tokyo-ku" is a complex query since it can be decomposed into two sub-queries, i.e., \( q_{1} = \) "find all houses that are less than $300,000 in Shinagawa-ku" and \( q_{2} = \) "find all houses that are less than $300,000 in Tokyo-ku". In this work we focus on complex queries, i.e., we assume that an user query can be decomposed into several semantically independent sub-queries and the number of remote hosts that should be visited is large [13].

Mobile agents are programs that can transport its state from one environment to another and resume its performing appropriately in the new environment [9, 10]. In a mobile agent-based information retrieval algorithm, each sub-query is performed by a mobile agent so that all sub-queries are performed in parallel. Instead of recording the retrieval processes to all objective hosts, parallel retrieval process are represented by an integration of the retrieval processes of sub-queries. Specifically, each sub-query has the form \( \{S_{q}, id., S_{q}(A)\} \), where the first field is a tag indicating "sub-query", the second denotes the locations of the objective hosts to be visited for the \( S_{q} \), and \( S_{q}(A) \) corresponds to the set of the remaining attributes of the tuple. Accordingly, a retrieval process has the form \( \{N, T, M, q(A)\} \), where \( N \) defines the number of sub-queries, \( T \) denotes time interval for completing the user query, \( M \) indicates the number of generated agents, and \( q(A) \) corresponds other attributes of the retrieval process.

- Motivation
Existing mobile agent-based parallel information retrieval techniques focus on the self-clone ability of mobile agents. The best existing algorithm, EHQR [25], consumes $\sum_{i=1}^{N} T_g(i)$ generating times for agents for $N$ sub-queries, where $T_g(i)$ is the generating time of the $i$-th generated agent. As demonstrated in our analysis, in spite of its utilization of clone agents, the round trip time of this algorithm is similar as that of an earlier proposed algorithm HQR, hampering its usefulness for practical queries.

Furthermore, the EHQR approach (as well as most other methods for this problem) is currently inapplicable in practice since its dispatching process of mobile agents is intrinsically serial that does not really make use of the self-clone ability of mobile agents. A possible solution exists to reduce the round trip time of an user query by better utilization of mobile agents' self-clone ability.

**Contribution**

This paper settles the above problems. Specifically:

- We derive the first evaluation model that accurately estimates the generating time of mobile agents for a fixed number of sub-queries.
- We analyze, using this model, the dispatching cost of existing algorithms. The analytical results show that the current approaches may be significantly improved.
- We propose the AEHQR approach, which utilize the self-clone ability of mobile agents to enhance performance.
- We prove, through extensive experiments, that the AEHQR is consistently outperforms the HQR and EHQR approaches.

The rest of the paper is organized as follows. Section 2 describes the structure of a mobile agent-based e-commerce system. Section 3 presents the working mechanism of dispatched mobile agents. Section 4 analyzes the performance of HQR and EHQR, and provide the AEHQR approach. Section 5 contains an extensive experimental evaluation, and Section 6 concludes the paper.

2. **STRUCTURE OF MOBILE AGENT-BASED E-COMMERCE SYSTEM**

This Section describes the structure of our agent-driven E-commerce system. The Mole infrastructure [2] is adopted in our model in which two different kinds of agents are provided: client agents and work agents. Client agents are static and usually interface components to users and local resources. Work agents are generated by client agents for user requests. They execute on a node as a "foreigner" with limited rights as long as they can not convince the local client agent. Although some works assume that the resources might be replaced dynamically [12], we assume the resources are pre-allocated in the market systems.

As shown in Figure 1, the E-commerce system is constructed by a number of service servers and a set of market places. On each service server, there is a client agent which is responsible for contacting with users and generating work agents. Once a system user submits his request for finding a product to the system through an user interface of a service server, the client agent on the service server will make a response to the user and take responsibility for further processing on the user request. Firstly, the user request is analyzed and decomposed into a number of sub-tasks such that each sub-task requires visit only one e-market place. Then, the same number of search agents are generated and each agent is assigned one sub-task. The search agents migrate to the market places with assigned tasks, execute on the market places, and report their search result to the client agent. The client agent will collect all the search results, rank the results, and provide the user a summarized report on the request.

On each market place, there are a number of e-shops running simultaneously and there is a market manager in each market place which is responsible for information maintaining of the market place and security check of incoming search agents. The information maintained by the market managers includes the domain names, IP addresses of the marketplace and e-shops, product information of e-shops, etc. Once a search agent arrives at a market place, it firstly contacts with the market manager. After having passed through the security check, the search agent checks the goods catalogues. If the search agent finds from the catalogues some products that might be relevant to its task, the agent goes to the e-shop that has the product for further information. On each e-shop, there is a shop manager that is responsible for local issues, including maintaining and protecting the local database, e.g., the shop information and goods information, communicating can with incoming search agents, and providing a negotiating and purchase interface to purchase agents.

3. **THE WORKING MECHANISM OF DISPATCHED MOBILE AGENTS**

The execution process of a dispatched agent, no matter a search agent or a purchase agent, can be mainly decomposed as five stages, namely generating, sending, executing, returning, and reporting. In most occasions, the stages sending, executing and returning can be merged into one stage, namely working stage.

computer systems science & engineering
3.1 Generating

In this stage, the dispatched agent is created by the client agent on the service server. Some arguments are encapsulated into the dispatched agent, including the task, the code for accomplishing the task, and the address information of the service server for sending results back. The time consumption for this stage is mainly depends on the computation speed of the service server, the size of the mobile agent, etc. In this paper, we use a random variable $T_g(i)$ to express the time consumption for generating the $i$-th dispatched agent.

3.2 Sending

After the dispatched agent has been generated, the client agent will dispatch it to the market places to fulfill the user requests. The time consumption for the agent leaves the service server and arrives at an e-market place is mainly depends on the bandwidth and traffic state of the network, the size of the dispatched agent, and the sending algorithm. In this paper, we use a random variable $T_{Sg}$ to describe the time consumption for the $i$-th dispatched agent to move from the service server to the assigned node.

3.3 Executing

If the dispatched agent has been successfully arrived at an e-market place, it begins to execute and access local data to accomplish its task. Due to the characteristics of the task, the dispatched agent can communicate with the market manager/shop managers or access the local data. This period of time depends mainly on the nature of the task and the current state of the local processor. We use a random variable $T_{E}$, to show the execution time of the $i$-th dispatched agent.

3.4 Returning

After the dispatched agent has completed its task, a report of the completed task should be sent back to the client agent. This job either done by either the dispatched agent itself carries the result back or a message to send the results back. As an agent may contain more codes than a message, sending the results back through a message might be faster. On the other hand, sending the results back by the dispatched agent might be more suitable for a network that the connection is dynamically changed. This stage can be viewed as a reverse stage of the sending stage but the time consumption of these two stages might be different. Both the size of the dispatched agent/message and the current state of the network might be changed during the execution process. A random variable $T_{Rg}$ is used to describe the time consumption of this stage for the $i$-th dispatched agent to return back to the service server.

3.5 Reporting

In this stage, the dispatched agent will submit a report on the completed task to the client agent. For a search agent, the route it has passed is also reported to the client agent for the possible utilization of a purchase agent. The time consumption for this stage is also mainly depends on the computation speed of the service server. In this paper, we use a random variable $T_r(i)$ to express the reporting time of the $i$-th dispatched agent to the client agent.

4. DISPATCHING MODEL OF MOBILE AGENTS

It is possible that a user request can be decomposed into several simple tasks that can be accomplished by visiting only one remote host. A dispatched agent can be assigned a simple task or a set of simple tasks. In case that there is a set of tasks that are semantically dependent and should only be finished serially, only one dispatched agent will be generated and execute these tasks in a serial way. Otherwise, if these tasks are semantically independent and the number of remote hosts that should be visited is large, distributing these tasks to multiple agents might be a good choice. As each dispatched agent has only one or a small number of simple tasks which will not take a long time to accomplish, the user can get the feedback from the client agent in a short time since these dispatched agents can execute in parallel over different processors. This section describes our parallel dispatch algorithm. For a better understanding, we firstly give a brief look on two parallel dispatch algorithms, HQR and EHQR [25].

4.1 HQR

HQR approach is based on the self-clone ability of mobile agents. There are three kinds of agents in HQR approach, i.e., the client agent, the clone agent, and the dispatched agent. Once a user request is arrived at a service server through a user interface, a client agent is generated and take responsibility for the user request. The client agent firstly decomposes the user request into a set of simple tasks that each task can be accomplished by visiting a single place in the system. Then the client agent generates a number of clone agents one by one. Assume that each agent generates $\beta$ replica agents each generation. Then as shown in Figure 2, the client agent generates $\beta$ clone agents at the first generation and each clone agent generates $\beta$ clone agents at the second generation. This process will continue until there are enough clone agents have been generated. Then, each clone agent of the last generation generates $\beta$ dispatched agents and distributes the decomposed tasks to the dispatched agents. After the dispatched agents have been generated and assigned a task, they are dispatched to a remote node for accomplishing the task. Once a dispatched agent accomplished its task, it sends the results back to the client agent or a clone agent. The client agent collects all the results from the dispatched agents and provide a report to the user. HQR approach is in fact a serial dispatching model, the client agent generates a set of clone agents one by one and the clone agents generate dispatched agents one by one after all clone agents are generated. From the working mechanism shown in Figure 3, the total number of generated agents for generating $N = \beta^n$
dispatched agents is

\[ \beta(\text{the 1st generation}) + \beta \cdot \beta(\text{the 2nd generation}) + \cdots + \beta^m(\text{the } m\text{-th generation}) = \sum_{i=1}^{m} \beta^i = \frac{\beta}{\beta - 1} \cdot (\beta^m - 1) \]  

(1)

Before the first dispatched agent is generated, the time for generating all the clone agents is

\[ \sum_{j=1}^{m} T_g(j). \]  

(2)

Similarly, the time for generating all dispatched agents, denoted by \( D_{HQR} \), can be evaluated as

\[ D_{HQR} = \sum_{j=1}^{m} T_g(j). \]  

(3)

If the dispatched agent sets off immediately after its generation, the time it returns back to the client agent can be expressed as

\[ \sum_{j=1}^{m} \beta^j + 1 \sum_{j=1}^{m} T_g(j) + T_s + T_e + T_b + T_r. \]  

(4)

Similarly, the return time of the \( k \)-th dispatched agent can be expressed as

\[ t(k) = \sum_{j=1}^{m} \beta^j + k T_g(j) + T_s + T_e + T_b + T_r. \]  

(5)

Thus, based on the above discussion, we can further analyze the round trip time of HQR.

**Definition 1** The round trip time of an approach, denoted by \( T_{r} \), is defined as the time consumption from the first agent is generated by the client agent for a user request to the last agent submits its task report about the user request to the client agent.

From 4 and 5, the round trip time of HQR, denoted by \( T_{HQR} \), can be expressed as follows:

\[ T_{HQR} = \max_{1 \leq k \leq m} |t(k)|. \]  

(6)

Taking the expectation on both side of 3, it is ready to show the average generating time is

\[ E[D_{HQR}] = E \left[ \sum_{j=1}^{m} T_g(j) \right] = \sum_{j=1}^{m} \beta^j \cdot E[T_g(j)]. \]  

(7)

Furthermore, taking the expectation on both side of 6, the average round trip time of HQR satisfies

\[ E[T_{HQR}] = E[T_s] \left[ E_T \left[ \sum_{j=1}^{m} \beta^j \cdot E[T_g(j)] \right] \right] + E[T_b] + E[T_e] + E[T_r]. \]  

(8)

### 4.2 EHQR

In EHQR, a clone agent will be transferred to a search agent after the clone task is completed. As shown in Figure 4, at the first generation, the client agent generates \( \beta \) clone agents. At the second generation, each clone agent clones \( \beta - 1 \) times. As the clone agent of the first generation will join the clone agents of the second generation, the total number of clone agents for generating the third generation clone agents is \( \beta^2 \), which is the same as that of HQR approach. This process continues until enough clone agents have been generated at the \( m-1 \)-th generation, that is, \( \beta^{m-1} \) clone agents in total. Then, at the \( m \)-th generation, each
clone agent at the $m-1$-th generation generates $\beta - 1$ dispatched agents and totally there are $\beta^{m-1}(\beta - 1)$ dispatched agents generated. Thus, together with the clone agents, there are $N = \beta^m$ agents generated. Now, each generated clone agents will also take on a task and set off to accomplish the task together with the dispatched agents. The working mechanism of the EHQR approach is shown in Figure 5.

Before the first dispatched agent is generated, the time for generating all the clone agents is

$$\sum_{j=1}^{\beta^{m-1}} T_g(j).$$

Similarly, the time for generating all dispatched agents, denoted by $D_{EHQR}$, can be evaluated as

$$D_{EHQR} = \sum_{j=1}^{\beta^m} T_g(j).$$

Since it is complex to analyze the accurate set off time of the clone agents which are generated at the previous generations, we simplified the problem by an assumption that all the clone agents will set off after all the dispatched agents have been generated. Thus, the time that a dispatched agent returns back to the client agent can be expressed as

$$t(k) = \sum_{j=1}^{\beta^{m-1}} T_g(j) + T_s + T_e + T_k,$$

where $\beta^{m-1} + 1 \leq j \leq \beta^m$. On the other hand, the time that a clone agent returns back to the client agent can be expressed as

$$t(\kappa) = \sum_{j=1}^{\beta^{m-1}} T_g(j) + T_s + T_e + T_k,$$

where $1 \leq \kappa \leq \beta^{m-1}$. Therefore, the round trip time of EHQR, denoted by $T_{EHQR}$, can be expressed as follows:

$$T_{EHQR} = \max_{1 \leq \kappa \leq \beta^{m-1}} \{t(\kappa), t(k)\}.$$

Taking the expectation on both side of 10, it is ready to show the average generating time of EHQR is

$$E[D_{EHQR}] = E\left[\sum_{j=1}^{\beta^m} T_g(j)\right] = \beta^m \cdot E[T_g(j)].$$

Furthermore, taking the expectation on both side of 13, the average round trip time of EHQR satisfies

$$E[T_{EHQR}] = E[T_g] \cdot \max_{1 \leq \kappa \leq \beta^{m-1}} \{t(\kappa), t(k)\},$$

$$= \beta^m \cdot E[T_g(j)] + E[T_\kappa] + E[T_k] + E[T_{\beta^m-1}] + E[T_\beta^m].$$

4.3 AEHQR

In both HQR and EHQR, the clone behavior acts in a serial fashion. In our proposed dispatching model, the clone behavior acts in parallel. As shown in Figure 6, each clone agent begins generating new agents immediately after its birth without waiting for other clone agents' birth. These clone agents may generate new agents on the service server where the user request is received or move to other servers first before generating so as to reduce the computation load of the service server. In each circle, the numbers on the upper line indicates the sequence of generated agents and the numbers on the lower line indicates the sequence of birth of those generated agents. At the $\ell$-th generation, there will be $\beta^\ell - \beta^{\ell-1}$ agents generated. The total number of generated agents, including clone agents and dispatched agents, is $N = \beta^m$ which is the same as $EHQR$. Also, each generated clone agent will take on a task and set off to accomplish the task together with the dispatched agents. The working mechanism is shown in Figure 7.

The generating time of the first $\beta - 1$ agents at the $m$-th generation can be expressed as

$$T_g(1) + \sum_{j=1}^{\beta^m-1} T_g(j) + \sum_{j=1}^{\beta^{m-1}+1} T_g(j)$$

where $1 \leq s \leq \beta - 1$, the generating time of the second $\beta - 1$ agents at the $m$-th generation can be expressed as

$$T_g(1) + \sum_{j=1}^{\beta^{m-1}} T_g(j) + T_g(\beta^{m-2} + 1)$$

$$+ \sum_{j=1}^{\beta^{m-2} + 2} T_g(j).$$

Figure 6 AEHQR Generation Tree
where $1 \leq s \leq \beta - 1$, and the generating time for the last agent at the $m$-th generation, i.e., the generating time for all dispatched agents, denoted by $D_{AEHQQR}$, is

$$D_{AEHQQR} = Tg(1) + \sum_{i=1}^{m} \sum_{j=\beta-\beta+2}^{\beta} Tg(j)$$  \hspace{1cm} (18)

As the departure time of agents is complex, it is difficult to calculate the detail dispatching time of each agent. Here, we will not present the accurate description on the round trip time of AEHQQR but gives an upper bound of it. An obvious upper bound of the round trip time of AEHQQR can be expressed as

$$T_{EHQR} \leq U_{EHQR} = \max_{1 \leq i \leq \beta-1} \{[\beta^{-1} + \rho(\beta - 1)] \} + \max_{1 \leq k \leq \rho_{m}} \{T_s(k) + Te(k) + Tb(k) + Tr(k)\}.$$  \hspace{1cm} (19)

Taking the expectation on both side of 18, it is ready to show the average generating time of AEHQQR is

$$E[D_{AEHQQR}] = E[Tg(1) + \sum_{i=1}^{m} \sum_{j=\beta-\beta+2}^{\beta} Tg(j)]$$
$$= [m(\beta - 1) + 1] \cdot E[Tg(j)].$$  \hspace{1cm} (20)

Furthermore, taking the expectation on both side of 19, the average round trip time of AEHQQR satisfies

$$E[T_{AEHQQR}] = [m(\beta - 1) + 1] \cdot E[Tg(j)] + E[Te(k)] + E[Tb(k)] + E[Tr(k)].$$  \hspace{1cm} (21)

5. EXPERIMENTS

The retrieval process is evaluated by the round trip time which is defined as the time from the client agent generates clone agents to the dispatched agents submit their report to the client agent (see Figure 1). The evaluations on the quality of the query results are not discussed here.

In our experiments, the number of generated agents for a query retrieval process using the HQR, EHQR, and AEHQQR approaches have been investigated where $\beta$ is set to 4. Figure 8-(a) shows the results. From Fig 8-(a) we can see that the EHQR and AEHQQR approaches generate a same number of agents as the number of sub-queries decomposed from the user request by the client agent. The number of mobile agent to be generated in the HQR approach is the highest. Although the number of agents generated in EHQR approach and AEHQQR approach are same, the working mechanisms are different. In the EHQR approach, there are a number of clone agents which help the client agent to generate dispatched agents. After the generating process of mobile agents finishes, the clone agents turn into work agents and set off for executing a sub-task of the user task. The HQR approach has a similar hierarchical generating process. However, the HQR approach generates more agents for a same number of places to be visited is that the clone agents in the HQR approach are responsible to generating agents only and not take part in the execution of an user task. On each generation, EHQR performs a serial generation of mobile agents while the AEHQQR performs a real parallel generation.

We also compared the generating time of the number of agents of the HQR, EHQR, and AEHQQR approaches. The comparison results are reported reported in Figure 8-(b) where $\beta = 4$. From Figure 8-(b), it can be seen that when the number of dispatched agents is small, the performances of these three approaches are similar. With the number of dispatched agents increases, the HQR approach consumes much time than the other two approaches. As the AEHQQR adopts a parallel dispatching mechanism, it consumes less time for generating a same number of agents than the EHQR approach.

In the experiments for the round trip time each experiment, a randomly selected node is set to be the service server and a number of other randomly selected nodes are picked out as the destination servers/workplaces of mobile agents. The size of a newly generated mobile agent is set to be 48 bytes and the size of a sub-task or a report is set to be 5KB, 50KB, 300KB, or 1024KB for different implementations. A time unit is defined and $Tg(i)$, $Te(i)$, and $Tr(i)$ are set to be 1 time unit. $Ts(i)$ and $Tb(i)$ are decided by the hops from the service server to the destination server and the size of the agent. Fig. 9 shows experimental results of the round trip time for the analyzed approaches as a function on the number of destination servers. Fig. 10 shows experimental results of the round trip time for the analyzed approaches as a function on the size of a sub-task/report that an agent carries.

From Fig. 9 it can be seen that the round trip time of all approaches is a monotonously increasing function on the number of destination servers. The increasing rate of the HQR approach is the fastest among all approaches. AEHQQR achieves the best performance.

From Fig. 10 it can be seen that the round trip time of all approaches is a monotonously increasing function on the size of a sub-task/report. From the figures it can be seen that the HQR approach is the most time consuming. Again, the AEHQQR approach achieve the best performance.
6. THE OPTIMAL DISPATCH STRATEGY

In this section, we will provide some simple analysis on the optimal dispatch strategy and the time complexity for each approach. The notations used in this section are described as follows:

1. $N$ is the number of subtasks that the client agent has decomposed from the user request. Each subtask requires the dispatched agent visits only one workplace in the system. Without loss of generality, we assume that $N \geq 2$.

2. $\beta$ is the width of a generation tree. That is, $\beta$ agents will be generated by one mother agent. A mother agent should be the client agent or a clone agent. Without loss of generality, we assume that $\beta \geq 2$.

3. $m$ is the height of a generation tree.

4. $t = E(T_{g(t)}) > 0$ is the average time consumption for generating an agent.

**Theorem 1** For the HQR approach, the minimum time for dispatching all agents to $N$ workplaces is $Nt$.

**Proof** From 3, the minimization problem of the dispatching time can be expressed as follows:

$$\min \quad T = \sum_{i=1}^{m} \beta^i \cdot t; \quad \text{where} \quad \beta^m = N. \quad (22)$$

Substituting $\beta^m = N$ in function $T$, we have

$$T = \sum_{i=1}^{m} \beta^i \cdot t = \frac{\beta^m - 1}{\beta - 1} = (N - 1) \cdot \frac{\beta^m - 1}{\beta - 1} = (N - 1) \cdot \frac{\beta - 1}{\beta - 1} = (N - 1) \cdot \frac{\beta - 1}{\beta - 1} \quad (23)$$

Based on the K-T condition, we take differentiation on $\beta$ on both side of 23 and have

$$T' = -\frac{(N - 1)t}{(\beta - 1)^2} < 0, \quad (24)$$

which means $T$ a monotonically decreasing function on $\beta$. Substituting the maximum value of $\beta$, i.e., $N$, to 22, we have

$$\max T = (N - 1)t \cdot \left(1 + \frac{1}{\beta - 1}\right) = Nt, \quad m = 1. \quad (25)$$

Hence, the theorem is proven. \(\Box\)

**Corollary 1** For the HQR approach, the maximum time for dispatching all agents to $N$ workplaces is $2(N - 1)t$.

**Proof** From 24, it is ready to see that when $\beta$ takes the minimum value, i.e., $\beta = 2$, the time for dispatching all agents is the maximum.

$$T = \left(1 + \frac{1}{2 - 1}\right)(N - 1)t = 2(N - 1)t, \quad m = \log_2 N. \quad (26)$$

Based on the above discussion, it is ready to have the following corollary:

**Corollary 2** The time complexity of the HQR approach is $O(N)$.

**Theorem 2** For the EHQR approach, the minimum time for dispatching all agents to $N$ workplaces is $Nt$.

**Proof** From 10, the minimization problem of the dispatching time can be expressed as follows:

$$\min \quad T = \beta^m \cdot t; \quad \text{where} \quad \beta^m = N. \quad (27)$$

Substituting $\beta^m = N$ in function $T$, we have $T = Nt$. Hence, the theorem is proven. \(\Box\) Therefore, the following two corollaries are also hold.

**Corollary 3** For the EHQR approach, the maximum time for dispatching all agents to $N$ workplaces is $Nt$.

**Corollary 4** The time complexity of the EHQR approach is $O(N)$.

**Theorem 3** For the AEHQR approach, the minimum time for dispatching all agents to $N$ workplaces is $(\log_2 N + 1)t$. 

vol 24 no 5 September 2009
Proof From 18, the minimization problem of the dispatching time can be expressed as follows:

\[
\min \ T = [m(β - 1) + 1] \tau; \\
\text{where} \quad β^* = N.
\] (28)

Substituting $β = N^{1/n}$ into function $T$ and take differentiation on both side of the new function, we have

\[
T' = \left\lceil m \left( N^{\frac{1}{N}} - 1 \right) + 1 \right\rceil \tau' = \left( 1 + \frac{1}{N} \right) N^{\frac{1}{N}} > 0
\] (29)

which means $T$ a monotonically increasing function on $m$ and therefore a monotonically decreasing function on $β$. Substituting the minimum value of $β$, i.e., $β = 2$, into 28, we have

\[
m = \log_2 N, T = (\log_2 N + 1) \tau.
\] (30)

Hence, the theorem is proven. Therefore, the following two corollaries are also hold.

Corollary 5 For the AEHQ approach, the maximum time for dispatching all agents to $N$ workplaces is $N$.

Proof Substituting the maximum value of $β$, i.e., $β = N$, into 28, we have

\[
m = \log_2 N - 1, T = N\tau.
\] (31)

Hence, the Corollary is proven.

Corollary 6 The time complexity of the EHQR approach is $O(\log_2 N)$.

7. CONCLUSION

In this paper, we first analyzed two existing parallel dispatching models of mobile agents for the application of information retrieval. Then, we proposed a new dispatching model of mobile agent-based information retrieval. Experiments were conducted on the performance of these three approaches. Theoretical analysis on the time complexity was also provided. Both experimental results and analytical results showed that our proposed approach is prior to the existing approaches. Our approach well utilized mobile agents’ inherent merits and provided useful tools for efficient information retrieval.

Acknowledgement

This work was co-supported by National Natural Science Foundation of China (90818002 and 90718030) and Scientific Research Foundation for the Returned Overseas Chinese Scholars.

REFERENCES
