Distributed Contents Discovery based on PageRank

Guidance

Associate Professor	Tetsuya TAKINE
Professor	Masao FUKUSHIMA

Daisuke ASAHARA

2002 Graduate Course

in

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University

Feburary, 2004

Abstract

Today network technology is experiencing great growth in peer-to-peer networks. We believe this growth should make innovative network systems, and users will be able to take information through networks wherever and whenever they stay. In these networks, a large number of objects, such as users, sensor devices, cars, and so on, will be connected. We cannot control the whole network at a centralized server because many objects offer many kinds of contents, and act in their own way. In other words, these networks must be purely peer-to-peer without centralized servers, and they highly requires the ability to discover contents matching given query criteria such as a searched keyword.

An existing broadcast-based discovery mechanism used in Gnutetlla includes the following two problems. One is that discovery queries impact network traffic heavily. The other is that query result quality is not guaranteed, namely, a user has to select what he or she needs from a huge number of received query results containing the futile ones. In this study, we combine the following two methodologies to solve these issues, making desirable improvements.

First, we adopt the PageRank system, used as one of the most important ranking factor in the web search engine Google, to rank the value of query results. Because the original PageRank is used only in centralized systems, we develop a new algorithm, named distributed PageRank, in order to apply the PageRank system into our distributed environments. We consider this distributed PageRank algorithm as one of the significant achievement in this study. Our discovery mechanism using distributed PageRank automatically ranks the value of the contents offered by network objects.

In the next place, we introduce a learning algorithm using pheromone, which we call keyword pheromone. In our scheme, based on distributed PageRank, a query originator object selects the best query result from among all the received query results and returns a reward, which means to deposit keyword pheromone, only to the reverse route of the forwarding path to that best query result. By tracing deposited keyword pheromone, network objects are able to forward a query more likely to lead towards relevant contents.

This paper concludes that our proposed discovery mechanism shows superior performance through simulations, compared with an existing broadcast-based discovery mechanism in terms of the above two issues, network efficiency and a guarantee of query result quality.

Contents

1 Introduction

1.1 Contents discovery in peer-to-peer networks

Today network technology is experiencing great growth in peer-to-peer networks and used in various fields. We believe this growth should make innovative network applications, and then people will be able to take information through networks wherever and whenever they stay. As for such a network, the following networks are suggested; Jack-in-the-Net (Ja-Net) [17, 16] and Bio-net [13, 12] that includes the feature of biological behavior in a computer network.

In these networks, a huge number of objects are connected, such as not only users and computers but many kinds of devices like sensors, home electric appliances, cars and so on. These objects offer some contents such as a web page or many kinds of service. We cannot control the whole network using a centralized server because many objects exist and act in their own way. In other words, this network must be purely distributed peer-to-peer without any directory. A commonly needed functionality in such a network is the ability to discover network objects or contents that match particular query criteria. To develop a discovery mechanism in such a network, care must be taken of next two issues.

First of all, discovery must not impact network traffic and object resource efficiency. Secondly, only objects having preferred or good contents out of matching query criteria must be discovered. It is not useful discovery that users receive a huge number of discovery results including both good results and poor ones. Therefore, discovery results must be narrowed down to only good ones.

To mention the existing discovery mechanisms [1] used in such a distributed peer-to-peer environment, however, there are no discovery mechanisms which satisfy the above both issues. As a representative example, we discuss a broadcast-based discovery mechanism used in Gnutella [5], which is widely used in file sharing applications. In Gnutella, a user who initiates discovery process inputs a keyword(s) to represent contents, offered by network objects, such as music or movie files which a user wants to search for, and then a network node (i.e., PC) which a user is logging on broadcasts a query message containing the keyword(s) to neighbor network nodes (i.e., other user's PC) which can communicate with the node issuing the query. If neighbor nodes do not have the contents which the user is searching for, the neighbor nodes broadcast the query to their neighbor nodes again. A query is forwarded from one network node to another in this way until it eventually reaches the target network node which has the contents matching searched keyword, or the query satisfies the condition of discovery termination such as time to live (TTL) [4]. This broadcast-based mechanism is simple and guarantees high success rate of discovery of target objects within a limited search area. However, it has some major downsides as follows. The broadcast-based discovery mechanism is not scalable because it causes heavy traffic. Forwarding of a large number of query messages, many of which do not contribute to successful discovery, consume bandwidth of the network [10, 3, 11]. As a matter of fact, scalability is the most serious issue to be solved in Gnutella. There is another problem about TTL (Time to Live) value. Usually in broadcast-based discovery mechanism, the same TTL value is set in every query message, and decremented by 1 whenever a query message is forwarded from one node to another. TTL restrains the amount of traffic generated by query forwarding by limiting forwarding hop counts of queries. At the same time, TTL value restricts discovery coverage area where a query can be forwarded. Because of this trade off relationship between traffic and discovery coverage area, it is impossible to find appropriate value of TTL before discovery originates.

The other severe problem is that this broadcast-based mechanism cannot distinguish the quality of contents which match the same keywords. In such a network we consider, it is expected that usefulness or relevancy of contents is considered to be different even if they offer similar contents represented by the same keyword. Some objects may satisfy query originators more likely, whereas other objects may disappoint query originators. Under this condition, the broadcast-based mechanism, not taking the contents relevancy into consideration, usually returns too many results including useless ones for users. Consequently, users need to make an extra effort to choose useful information from the heap of the results.

The existing discovery mechanism does not fulfill our requirements described in the (i.e., network efficiency and a guarantee of query result quality). In this study, we develop a new discovery mechanism which satisfies our requirements in a peer-to-peer network.

1.2 Organization of this paper

The remainder of this paper is organized as follows. Section 2 describes the general discovery procedure. In Section 3, we explain our discovery mechanism outline, and develop distributed PageRank algorithm, which is necessary for our proposed discovery mechanism to be realized. In the next place, we explain our discovery procedure in detail. Section 4 shows the simulation model, results and observation. Then we conclude this paper and mention future work in Section 5.

2 Basic discovery mechanisim

2.1 Network model

Objects, each of which is labeled its unique number as an object ID, are distributed in a purely peer-to-peer network (i.e., no centralized server), and they act at the will of themselves, like migrating, replicating, or collaborating with other network objects. In this study, we assume that each object offers some contents and has a keyword describing its contents.

A network object establishes relationship with other network objects, which we call relationship partners. Relationship has the following three features; First, relationship is a logical pointer and independent of physical distance between network objects. Secondly, relationship partners are selected according to the object's own reasons such that a relationship partner offers useful contents, similar contents to itself, new contents and so on. In the last place, relationship is not necessarily in bi-directions. Namely, even if object A has relationship to object B, object B does not necessarily have relationship to object A.

An object stores such information as relationship partners' ID and location in its relationship table. Then, the object is always able to access its relationship partners by using not any centralized servers but referring its relationship table. Note that a network object cannot access any other objects except for relationship partners.

2.2 Discovery procedure

We consider the situation that users are searching for contents offered by network objects. A user logs on a network object, which originates a query having a unique query ID. A query contains a searched keyword which represents searched contents.

We develop a discovery mechanism to discover contents matching a searched keyword. Our discovery mechanism makes use of only relationship, tracing network objects in the next two phases: query forwarding and query result returning. The query forwarding process begins at a discovery originator, and queries move through relationships, seeking contents relevant to the query. The query result returning process gives a discovery originator discovery results.

Upon receiving a query, a network object first examines whether it has already received the same query by referring to query ID, and if it already has, the query is discarded. The network object then checks whether it can return the query result for the query or not. When the network object itself is not relevant to the searched keyword in the query, the network object decides to which network object to forward the discovery query. This decision is made based on its discovery policy explained in section 3.4. For example, in the case of broadcasting, a received query is forwarded to all the relationship partners. The discovery policy in our discovery mechanism is shown in Section 3.

Query results are returned back to the discovery originator following the reverse route of the forwarding path to the network object that returned its contents as a query result. In order to store the back propagation path information, each network object, during the query forwarding phase, keeps a record of the unique query ID, the network object from which the query is received in the query forwarding phase (i.e., the network object to which query results should be sent during query result returning), and the network objects to which the query is forwarded during the query forwarding phase (i.e., the network objects from which query results are expected to be received).

3 Proposed discovery mechanism based on PageRank

3.1 Outline

3.1.1 Objective

Our discovery mechanism aims at the following two issues; reducing wasted queries and discovering only good results. In other words, our mechanism overcomes the problem of broadcasting way, a large number of wasted queries and too many results containing the futile ones.

3.1.2 Approach

We explain our proposed mechanism roughly in this subsection and strictly in the following part.

First, we discuss how to rank contents that a network object offers. The method of using user preference is already proposed [14]. We think that method may ask a bother, to evaluate the results and return it to a network, of users, or first user's preference may have strong affect till after. Then, we introduce the matter of the web search engine, Google [6], developing the ranking system of contents objects offer in a distributed environment. We define the value of contents an object X offers for a searched keyword A as $Value^*(keywordA, objectX)$ which is higher value in the case the query result is better.

In the next place, to reduce wasted queries, we adopt learning algorithm using pheromone which guides queries more likely towards a network object relevant to that query. Our mechanism deposit pheromone, named keyword pheromone, in rewarding process our mechanism implements besides query forwarding process and query result returning process.

In the last place, we develop a new discovery policy (according to which a network object selects its partner object(s) to forward a query) by making use of it.

3.2 Value of contents

3.2.1 Definition

We consider two concepts to calculate $Value^*(keywordA, objectX)$.

One is the PageRank value of object X, used in Google [6], which is denoted $PR^*(X)$. PageRank value of an object means how much other network objects trust that object. We explain calculation of $PR^*(X)$ and more details in the following subsection. Note that PageRank value calculated independent of a searched keyword.

The other is the relevance between a searched keyword A and the contents of object X, which is denoted $Fitness^*(keywordA, objectX)$. $Fitness^*(keywordA, objectX)$ is based on many manners, such as how many times a searched keyword appears in the contents of object X, which are beyond the scope of this paper. Then, we simply assume that $Fitness^*(keywordA, objectX)$ indicates the follwoing;

$$Fitness^*(keywordA, objectX) = \begin{cases} 1, & \text{if the contents of object X has a searched keyword A,} \\ 0, & \text{otherwise.} \end{cases}$$

Lastly, we define $Value^*(keywordA, objectX)$ as follows;

$$Value^*(keywordA, objectX) = PR^*(X) \times Fitness^*(keywordA, objectX). \tag{1}$$

3.2.2 PageRank overview

We explain the original form of PageRank, and then propose distributed PageRank. L. Page et al. [9] developed PageRank to measure the PageRank value of web pages. They gave an intuitive description of PageRank; a page has a high rank if the sum of the ranks of its backlinks is high. This covers both the cases when a page has many backlinks and when a page has a few highly ranked backlinks. The PageRank value of a web page i, denoted $PR^*(i)$, is equivalent to the number of users on page i after infinite steps of a random walk which proceeds at each step as follows. Users move uniformly at random with probability $(1-d)$ to one of the pages linked from the current page, and users jump with probability d to some other page at random without tracing the linked pages. The value of PageRank is defined recursively according to the equation,

$$PR^*(i) = d + (1-d) \sum_{j \in \mathcal{M}(i)} \frac{PR^*(j)}{N(j)} \ , \tag{2}$$

where $\mathcal{M}(i)$ is a set of the web pages which have a link to web page i, $N(j)$ is the total number of links originating from page j, and $0 \leq d < 1$. Intuitively, the value of $PR^*(i)$ expresses the relevancy of the web page i. This ranking is used as one factor of the Google search engine to determine how to order the pages returned by a web search query [6].

For instance, consider the following environment shown in Figure 1.

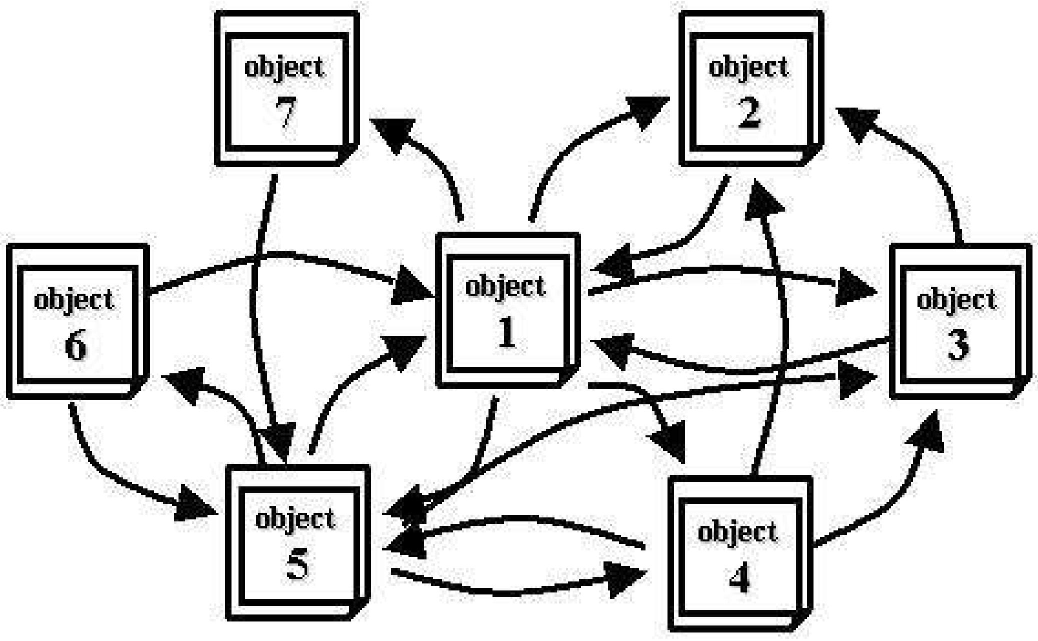

Figure 1: Sample environment.

Each square represents an object, and an arrow from object i to object j indicates that object i has a relationship to object j. We set $d = 0$ in eq. (2), and apply it to the relationship structure in Figure 1 for all objects i $(i = 1, 2, \ldots, 7)$. Adding the normalization condition $\sum_{i=1}^{7} PR^*(i) = 1$, we get PageRank values as below,

$$\begin{pmatrix} PR^*(1) \\ PR^*(2) \\ PR^*(3) \\ PR^*(4) \\ PR^*(5) \\ PR^*(6) \\ PR^*(7) \end{pmatrix} = \begin{pmatrix} 0.303514\cdots \\ 0.166134\cdots \\ 0.140575\cdots \\ 0.105431\cdots \\ 0.178914\cdots \\ 0.044728\cdots \\ 0.060703\cdots \end{pmatrix}. \quad (3)$$

Note that eq. (3) means that $PR^*(i)$ represents the steady state probability of state i in a Markov chain described in Figure 2 [7].

In the rest of this paper, we define $PR^*(i)$ as the PageRank value of object i setting $d = 0$ in eq. (2). One caveat of this setting is that link structure graph must be strongly connected, i.e., any web page can reach all the other web pages by tracing links [9].

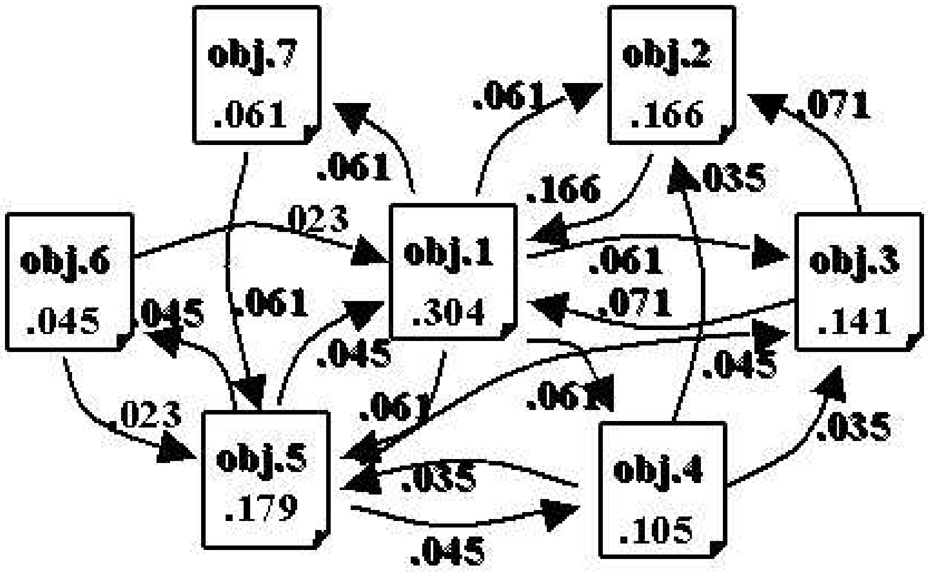

Figure 2: PageRank state chart.

3.2.3 Distributed PageRank

We regard relationship between objects in our scheme as a link between web pages. In our scheme, relationship from object A to object B implies that object A trusts object B. This concept is equal to links in web pages. Hence, we regard relationship between objects in our scheme as a link between web pages, and introduce PageRank to our scheme.

In Google, robots crawl web pages, and then store their information into the database to calculate the PageRank value. Therefore, Google is considered as a centralized system. To apply PageRank to our peer-to-peer environment, we develop a new algorithm, named distributed PageRank, to calculate the PageRank value. Distributed PageRank requires neither any centralized calculation server nor a crawling robot. The distributed PageRank algorithm does not require any special messages made only for this calculation. Instead, we utilize discovery queries. When an object receives a discovery query, it updates its PageRank value without synchronizing with all the other network objects, forwarding a discovery query put updated PageRank information in. In this point, ours is different from other related work, i.e., ours is considered to be more efficient for network traffic [8].

We assume that each object stores information about PageRank values in its relationship table. Considering the situation in Figure 2, the relationship table of object 1 stores PageRank information in its relationship table given in Table 1.

The inlink PageRank column indicates the values from all the objects that have a relationship to object 1. Elements of the inlink PageRank column are the conveyed value by the latest query from a corresponding object. Let $PR(i,t)$ denote PageRank value which object i derives as the PageRank value of itself at time t. $PR(1,t)$ is calculated as the sum of inlink PageRank column at time t. The values of the outlink PageRank column are equal to the value, $PR(1,t)$, divided by the number of its relationship partners.

Next, we explain the distributed PageRank, which enables each object i to reach $PR^*(i)$

Table 1: PageRank information on relationship table of object 1.

Partner's ID	inlink PageRank value	outlink PageRank value
object 2	0.166	0.061
object 3	0.071	0.061
object 4	NA	0.061
object 5	0.045	0.061
object 6	0.023	NA
object 7	NA	0.061

after normalization, $\sum_i PR(i,t) = 1$, where t is large enough. The procedure is described below.

A1. When object A makes a query,

- **(1)** object A selects one or more object(s) B out of A's partners (the way to select objects depends on discovery policy), and then,
- **(2)** marking up the value corresponding to B on A's outlink PageRank column in a query, and forwarding it to B (note that no other partner except for B is not informed of the value on A's outlink PageRank column).

A2. When an object receives a query from a partner, it performs either case (a) or case (b) depending on the situations.

- **(i)** If this object offers contents the originating object is searching for, the object performs the procedure "table update," which includes the following three procedures: updating its inlink PageRank column according to the value conveyed by a query from a partner, calculating its own PageRank based on its inlink PageRank column, and updating its outlink PageRank column. Then, the object finishes the query forwarding process.
- **(ii)** Otherwise, this object completes "table update," and forwarding this query to one or more object(s) with the updated value.

3.2.4 Example of distributed PageRank

To demonstrate the effectiveness of distributed PageRank, we conduct simulation experiments with the model in Figure 1. The simulation details are as follows:

B1. Each object has an initial PageRank value of 1, making its outlink PageRank column. To make an initial inlink PageRank column, let each object get the values from corresponding objects' outlink PageRank column. After that, each object performs "table update." These operations define initial values of each object's inlink PageRank column, outlink PageRank columln, and PageRank.

B2. Object 1 starts the first discovery using a single message at time 0. No object can start the discovery except for case (a). Object 1 chooses object X among its partners, forwarding a query with the value corresponding to X in the outlink PageRank column of peer 1. Each object receiving a query, such as object X, follows either case (a) with probability q, or case (b) with probability $(1 - q)$.

case (a) Object X offers searched contents and returns a query result. Object X performs "table update," and finishes this discovery processing. In the next place, we randomly choose an object among network objects, which starts the next new discovery in the same way as the originator object 1 acted in the first discovery.

case (b) Object X does not offer the contents the originator object is searching. The "table update" procedure is completed. In the next step, object X chooses a peer from among its partners with equal probability, forwarding a query with the updated value on the outlink PageRank column of object X.

We performed simulations assuming that time is divided into slots whose lengths are equal to a unit time which is taken for a query to hop between objects. A typical simulation result for $q = 0.2$ is shown in Table 2, and Figures 3 and 4. Table 2 shows $PR(i, 515)$ $(i = 1, 2, \ldots, 7)$. The vertical scale of the Figure 3 expresses the time series normalized PageRank value of object 1. The vertical scale of Figure 4 expresses the time series sum of the PageRank value that all the network objects have.

Table 2: Pagerank value of each object at time 515.

	before normalization	after normalization
$PR(1, 515)$	2.351770	0.303514
$PR(2, 515)$	1.287284	0.166134
$PR(3, 515)$	1.089241	0.140575
$PR(4, 515)$	0.816930	0.105431
$PR(5, 515)$	1.386306	0.178914
$PR(6, 515)$	0.346577	0.044728
$PR(7, 515)$	0.470354	0.060703

From Table 2, we see the PageRank value of each peer converges to the original PageRank value denoted in (3) after normalization. Figure 3 shows the normalized PageRank value of a object 1 is heading to the original PageRank value, 0.303514. From Figure 4, we see the sum of PageRank value is static after each peer's PageRank value has converged. We confirmed these features in every performed simulation including the ones performed by setting different initial values.

3.2.5 Mathematical model of distributed PageRank

We describe simulation scenario mathematically so that we can analyze the simulation result. We consider the next eight issues in mathematical analysis of the simulation result.

(ASS 1) There are N objects in a network, labeled from 1 to N as object ID.

(ASS 2) Object 1 makes a query at time 0. There still exists a query on object1 at time 0.

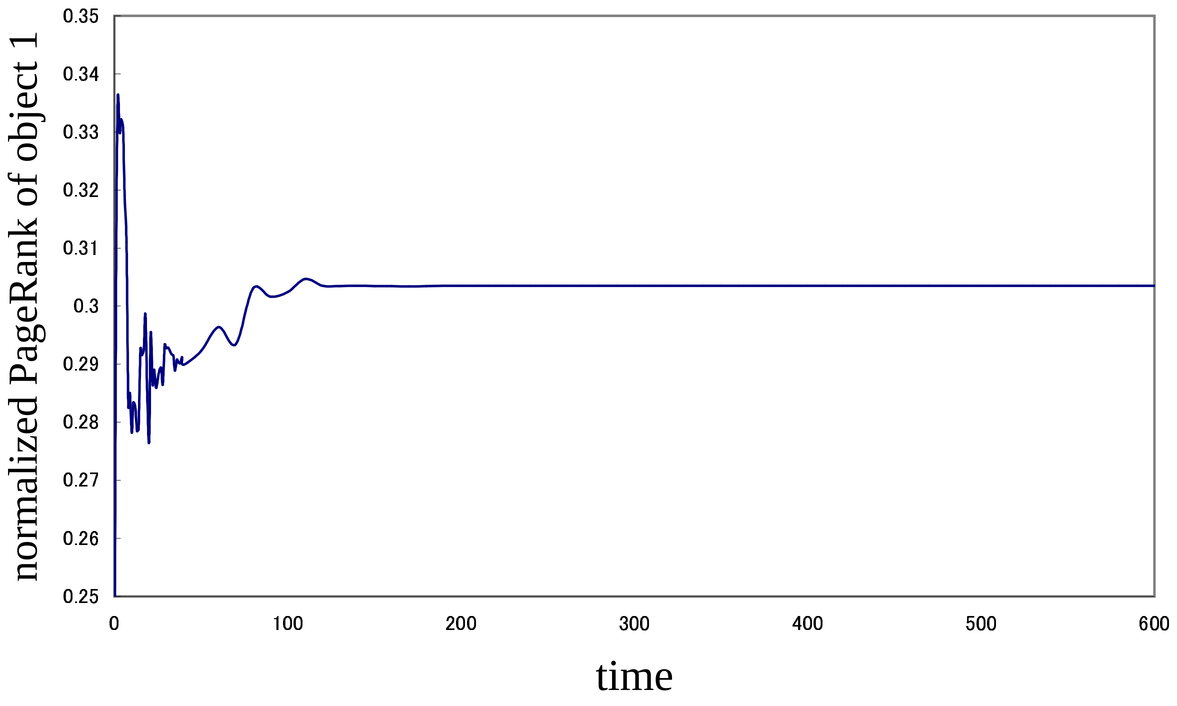

Figure 3: Variation of the normalized PageRank value of object 1, $PR(1, t)$.

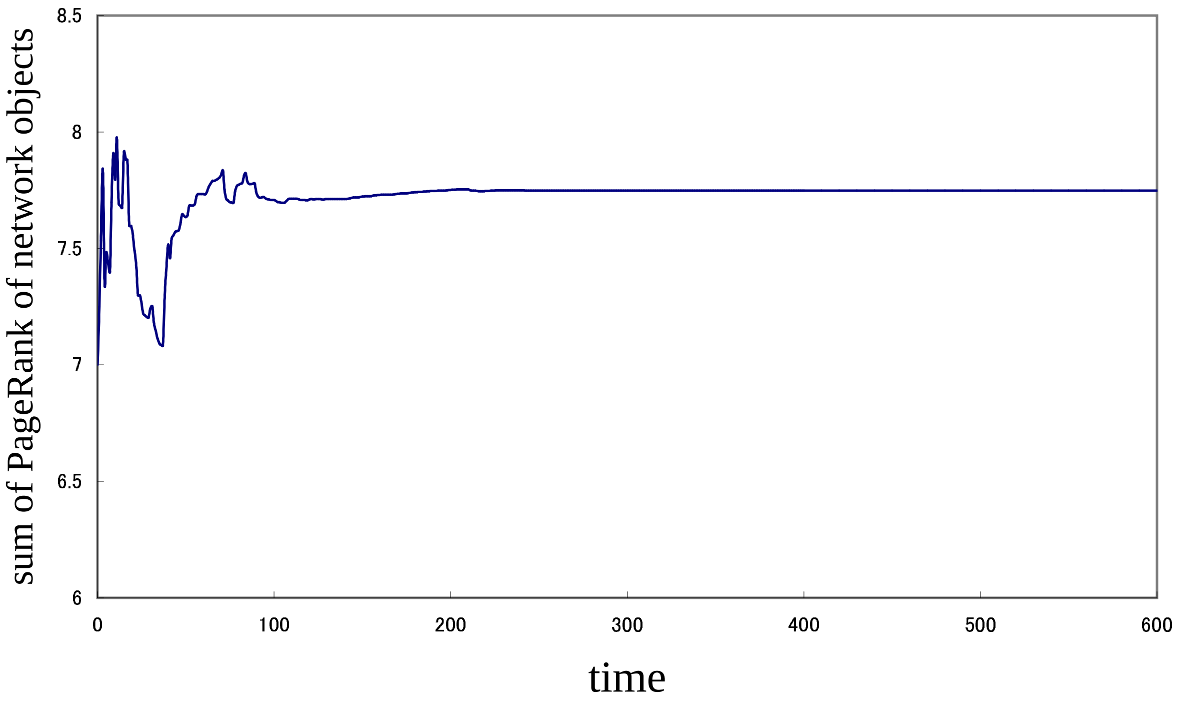

Figure 4: Total sum of PageRank.

(ASS 3) Objects receiving a query act in the following two ways.

a. They do the next operation with probability d. They return a query result, and do not forward a query. Query results do not update any PageRank value.

b. They do the next two operations with probability $(1-d)$. First, they update PageRank value for each partner on their relationship table. Secondly, after they choose a partner out of their all partners with equal probability, they forward a query to it and also convey the updated PageRank value.

(ASS 4) An object is randomly chosen out of the all network objects when an object returns a query result (ASS 3.a). Namely, a unit time after a query result is made by some object, there exsits a new originated query in a network. This new query acts in the same way as the first query described in (ASS 2).

(ASS 5) We define $\boldsymbol{A}$ as an $N \times N$ matrix that describes the network topology. The (i, j)th element of $\boldsymbol{A}$ is denoted by a_{ij}, which is defined as,

$$\begin{aligned} a_{ij} &= 1, \text{ if object } i \text{ has relationship to object } j, \\ a_{ij} &= 0, \text{ otherwise.} \end{aligned}$$

We consider an $N \times N$ matrix $\boldsymbol{P}$. The (i, j)th element of $\boldsymbol{P}$ is denoted by p_{ij}, which is defined as,

$$p_{ij} = \frac{a_{ij}}{\sum_{j=1}^{N} a_{ij}} \ .$$

$\boldsymbol{P}$ expresses the transition matrix from each object to its partner objects. The diagonal elements of $\boldsymbol{P}$ are 0 because we do not consider relationship from an object to itself. Assume a matrix $\boldsymbol{P}$ is irreducible.

(ASS 6) We describe PageRank of each object by a $1 \times N$ vector $\boldsymbol{r}_i(t)$. We define $r_{ij}(t)$, the ith element of $\boldsymbol{r}_i$, as the PageRank value which object i convey to object j through the latest query at time t. We define a $1 \times N^2$ vector $\boldsymbol{r}(t)$ as,

$$\boldsymbol{r}(t) = \left(\boldsymbol{r}_1(t), \boldsymbol{r}_2(t), \ldots, \boldsymbol{r}_n(t)\right) \ .$$

PageRank value of object j is expressed as a sum of PageRank values from all the objects that have relationship to object j, i.e., $PR(j,t) = \sum_{i=1}^{N} r_{ij}(t)$. We define the initial PageRank values as $PR(i,0) = 0$ $(i = 1, 2, \ldots, N)$, and distribute PageRank value of each object to its partner based on a matrix $\boldsymbol{P}$ in the following way. For all i $(i = 1, 2, \ldots, N)$ and for all j $(j = 1, 2, \ldots, N)$,

$$\begin{aligned} r_{ij}(0) &= PR(i,0) \times p_{ij} \\ &= p_{ij} \ , \end{aligned}$$

(ASS 7) Let $\boldsymbol{P}'$ denote an $N \times N$ stochastic matrix used in query forwarding, and described as,

$$\boldsymbol{P}' = (1-d)\,\boldsymbol{P} + \frac{d}{N}\boldsymbol{U} \ ,$$

where $\boldsymbol{U}$ denotes an $N \times N$ matrix, each of whose element is equal to one.

(ASS 8) We describe PageRank update algorithm, i.e. how to update $\boldsymbol{r}(t)$ below,

$$r_{ij}(t) = \begin{cases} \sum_{k=1}^{N} r_{ki}(t-1)\, p_{ij}, & \text{in the case that there exists a query on object } i \\ & \text{at time } t-1 \text{ and this query is forwarded to object } j \\ & \text{with probability } (1-d), \\ r_{ij}(t-1), & \text{otherwise.} \end{cases} \tag{4}$$

3.2.6 Mathematical analysis of distributed PageRank

The simulation result indicates that the PageRank value of each object converges to the original PageRank value after normalizing the sum of PageRank values from all the objects to be one, i.e., we can expect the following.

Conjecture 3.2.6
Considering $\boldsymbol{r}(t)$ $(t = 1, 2, \ldots, N)$ as the stochastic process, there exists some constant C for each sample, and $\boldsymbol{r}(t)$ converges to $C\pi_i p_{ij}$ with probability 1.

The following three issues are necessary conditions of the above:

1. a. Each element of $\boldsymbol{r}(t)$ is greater than or equal to 0 for all t $(t = 0, 1, 2, \ldots)$.
   b. $PageRank(i, t)$ is greater than 0 for all i $(i = 1, 2, \ldots, N)$ and for all t $(t = 0, 1, 2, \ldots)$.
2. $\{C\pi_i p_{ij} \mid i, j = 1, 2, \ldots, N\}$ is a unique fixed point of (4), up to a multiplicative constant C.
3. No element of $\boldsymbol{r}(t)$ diverges with probability 1.

We first prove 1.a.

For all i $(i = 1, 2, \ldots, N)$ and for all j $(j = 1, 2, \ldots, N)$,
$r_{ij}(0) = p_{ij}(0)$ is greater than or equal to 0 from (ASS 6).

Assume that each element of $\boldsymbol{r}(t)$ is greater than or equal to 0.
Each element of $\boldsymbol{r}(t+1)$ is greater than or equal to 0 from (ASS 8), which completes the proof of 1.a.

In the next place, we prove 1.b.

There exists some positive $r_{ij}(0)$ for all j $(j = 1, 2, \ldots, N)$ because a matrix $\boldsymbol{P}$ is irreducible from (ASS 5), and $\boldsymbol{r}(t)$ is greater than or equal to 0 from 1.a. Therefore, for all j $(j = 1, 2, \ldots, N)$,

$$\begin{aligned} PR(j, 0) &= \sum_{k=1}^{N} r_{kj}(0) \\ &> 0\ . \end{aligned}$$

Assume that $PR(j,t)$ is positive for all j $(j=1,2,\ldots,N)$, i.e. for all j $(j=1,2,\ldots,N)$,

$$\begin{aligned} PR(j,t) &= \sum_{k=1}^{N} r_{kj}(t) \\ &> 0 \ . \end{aligned} \tag{5}$$

We consider object a has a query at time t, and forward it to object b, resulting in the situation that there is a query on object b at time $t+1$.
In this situation, for $i=a$ and $j=b$,

$$\begin{aligned} PR(j,t+1) &= \sum_{k=1}^{N} r_{kj}(t+1) \\ &= \sum_{k=1}^{N} r_{kb}(t+1) \\ &\geq r_{ab}(t+1) \ \ (\text{from } \boldsymbol{r}(t+1) \geq 0) \\ &= \sum_{k=1}^{N} r_{ka}(t)\, p_{ab} \\ &> 0 \ , \end{aligned}$$

where the last inequality follows from eq. (5) and $p_{ab} > 0$. For any other i and j,

$$\begin{aligned} PR(j,t+1) &= \sum_{k=1}^{N} r_{kj}(t+1) \\ &= \sum_{k=1}^{N} r_{kj}(t) \\ &> 0 \ , \end{aligned}$$

where the second equality follows from $\boldsymbol{r}(t+1) = \boldsymbol{r}(t)$ according to (ASS 8), and the inequality follows from eq. (5), which completes the proof of 1.b.

Next we prove $\{C\pi_i p_{ij} \mid i,j=1,2,\ldots,N\}$ is a fixed point of (4) and, in the next place, this is a unique fix point of (4), up to a multiplicative constant C.

Assume for all i $(i=1,2,\ldots,N)$ and for all j $(j=1,2,\ldots,N)$,

$$r_{ij}(t) = C\pi_i p_{ij} \ . \tag{6}$$

We consider object a has a query at time t, and forward it to object b, resulting in the situation that there is a query on object b at time $t+1$.

In this situation, for $i = a$ and $j = b$,

$$\begin{aligned} r_{ij}(t+1) &= \sum_{k=1}^{N} r_{ki}(t)\, p_{ij} \\ &= \sum_{k=1}^{N} C\pi_k p_{ki} p_{ij} \quad \text{(from eq. (6))} \\ &= \sum_{k=1}^{N} C\pi_i p_{ij} \\ &= r_{ij}(t) \ . \end{aligned} \tag{7}$$

For any other i and j,

$$r_{ij}(t+1) = r_{ij}(t) \ . \tag{8}$$

Iterating (7) and (8), $\boldsymbol{r}(s) = \boldsymbol{r}(t)$ for any integers $s > t$.
That is to say, we proved that $\{C\pi_i p_{ij} \mid i, j = 1, 2, \ldots, N\}$ is a fixed point of eq. (4).

Let r_{ij} denote a fixed point of eq. (4) for all i $(i = 1, 2, \ldots, N)$ and for all j $(j = 1, 2, \ldots, N)$.
For all i $(i = 1, 2, \ldots, N)$ and all j $(j = 1, 2, \ldots, N)$,

$$r_{ij} = \sum_{k=1}^{N} r_{ki} p_{ij} \ . \tag{9}$$

Taking the sum of both sides of eq. (9) with respect to i, we have

$$\sum_{i=1}^{N} r_{ij} = \sum_{i=1}^{N} \sum_{k=1}^{N} r_{ki} p_{ij} \ . \tag{10}$$

Define g_j as $\sum_{i=1}^{N} r_{ij}$ for all j $(j = 1, 2, \ldots, N)$. We see from (1.b.) that g_j is positive for all j $(j = 1, 2, \ldots, N)$. From eq. (10),

$$g_j = \sum_{i=1}^{N} g_i p_{ij} \ . \tag{11}$$

This implies that g_j is unique up to a multiplicative constant because a steady state probability vector is unique for an irreducible stochastic matrix [15]. Consequently, from eq. (9), r_{ij} is also unique for all i $(i = 1, 2, \ldots, N)$ and for all j $(j = 1, 2, \ldots, N)$.

The above discussion completes the proof of 2.

As for 3, we cannot prove it yet, but we believe 3 holds in our environment from simulation results. We think 1, 2, and 3 are the strong reason why conjecture 3.2.6 can be true.

3.3 Keyword pheromone

We introduce keyword pheromone, deposited at the route which returned good results, into our scheme so that it can be reflected on discovery policy. Here we reconsider the value of contents to evaluate returned query results since object i does not know $PR^*(i)$ used in eq. (1). Let $Value(keywordA, objectX, t)$ be the value of a query result which object X returned for a searched keywrod A at time t. An originator object uses $Value(keywordA, objectX, t)$ calculated as below instead of $Value^*(keywordA, objectX)$,

$$Value(keywordA, objectX, t) = PR(X, t) \times Fitness^*(keywordA, objectX).$$

Keyword pheromone evaporates with rate ρ $(0 < \rho < 1)$ per unit time to adapt the change of networks.

Keyword pheromone is different between searched keywords, and each object stores keyword pheromone information for its relationship partners in its relationship table. Table 3 shows an example of keyword pheromone an object i stores.

Table 3: Keyword pheromone on a relationship table of an object i.

Partner's ID	Keyword A	Keyword B	$\cdots$
object j	$X_{ij}(A)$	$X_{ij}(B)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
object k	$X_{ik}(A)$	$X_{ik}(B)$	$\cdots$

To deposit keyword pheromone, we build a new process, rewarding process, in our mechanism besides query forwarding process and query result returning process. Rewarding process is performed after query returning process. To be more precise, a query originator evaluates query results based on $Value(keyword, object, t)$, and returns a reward propagated through the good query returning path (i.e. the one returned the highest value of $Value(keyword, object, t)$). This procedure is illustrated in Figure 5, where object A is a query originator, object E returned the best result, and object D returned a worse result. Note that a query originator does not return any reward to the query returning route which does not return the best result.

When an intermediate object on the path receives a reward, it adds a reward to keyword pheromone in its relationship table. A reward, the increased amount of keyword pheromone in the rewarding process, is set to be $value(object, keyword, t)$ in this study.

3.4 Discovery policy

We propose a discovery policy using keyword pheromone aiming at reducing wasted queries and discovering only good results.

Each object forwards a query according to our proposed discovery policy described in the following;

When object i originates or forwards a query including searched keyword A,

step 1 object i selects one partner randomly with probability p $(0 < p < 1)$, and using keyword pheromone with probability $(1 - p)$. Letting $X_{ij}(A)$ and $forward(i, A)$ be keyword

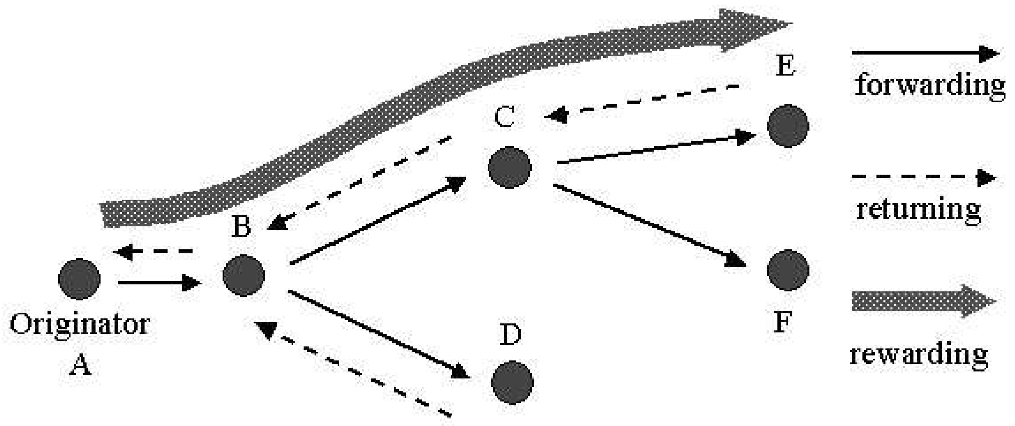

Figure 5: Rewarding process.

pheromone object i has for its partner object j as for keyword A and the partner object that object i selects in forwarding a query containing a searched keyword A, respectively,

$$Pr(forward(i, A) = j) = p \times \frac{1}{N(i)} + (1 - p) \times \frac{X_{ik}(A)}{\sum_{j \in \mathcal{M}(i)} X_{ij}(A)} ,$$

where $\mathcal{M}(i)$ is a set of object i's partner object ID, and $N(i)$ is the number of object i's partners.

step 2 Object i repeats step 1 independently $N(i)$ times. Note that bobject i is allowed to select the same object as already selected.

step 3 Object i forwards a query to all the partner objects selected. object i forwards only one query even if the same object is selected more than once.

4 Simulation

4.1 Model

We conduct simulation experiments of our discovery mechanism to examine its behavior. In addition, we also conduct simulation experiments of broadcast-based mechanism under the same assumption for the purpose of performance comparison. We adopt a broadcast-based mechanism in Gnutella, where queries are broadcasted within limited hop counts defined by TTL (time to live) value and an object having contents matching a keyword returns a query result which backtracks the path which the query is forwarded on. TTL is set to 5 all simulation experiments.

We deploy 1000 objects whose ID are labeled from 1 to 1000 in a peer-to-peer network. Object i has relationship with object $(i + 1)$ so that network objects in simulation environment can be strongly connected. An object has relationship with 8.3 partner objects on average (See Figure 6). Recall that relationship is not necessarily in bi-directions. Each object has a keyword representing its offering contents from 30 keywords, $keyword\ 1, keyword\ 2, \ldots, keyword\ 30$. Figure 7 shows the frequency distribution of keywords existing in the simulation network.

One discovery trial consists of the following; An object is chosen as a query originator randomly. That originator object originates a discovery query containing single searched keyword selected from 30 keywords with equal probability. The originator object receives all the query results for the issued query, and then returns a reward along the path leading towards the object that returned the best query result, as explained Section 3.3.

We assume that only one discovery is underway at any time, i.e., after one discovery trial is done, the next starts. Queries are assumed to take a unit time to move from an object to another object at any discovery trials.

4.2 Results

This section explains the simulation results. We conducted simulation experiments setting $p = 0.05$ and $\rho = 0.001$.

We first investigate whether our mechanism equipped with distributed PageRank can calculate the original PageRank value, $PR^*(object)$. Let $S(n)$ denote the absolute error between $PR(object, time)$ and $PR^*(object)$,

$$S(n) = \sum_{i=1}^{1000} |PR^*(i) - PR(i, T(n))| \ ,$$

where n denotes the number of discovery trials, and $T(n)$ denotes the time instant when the nth discovery trial completes.

Figure 8 and Figure 9 show $S(n)$ and $PR(1, T(n))$, respectively, as a function of n. These Figures show that there exists $n = n^*$ such that $S(n)$ is zero, and $PR(1, T(n))$ is stable at $PR^*(1) = 0.001045994$, for all $n > n^*$. This observation concludes that our discovery mechanism derives $PR^*(object)$.

Next we compare our discovery mechanism with a broadcast-based discovery mechanism in the same situation. Concretely, we take samples of the discovery trials in which a searched keyword is set to be $keyword\ 1$.

Table 4: Top 5 of objects offering contents represented by keyword 1.

ranking	object ID	value
1	828	0.002371179
2	480	0.001861129
3	524	0.001556443
4	677	0.001555417
5	52	0.001350562

Figure 10 and Figure 11 show the maximum value of received query results in our mechanism and the broadcast-based mechanism, respectively, as functions of the number of trials. At every discovery trial for $keyword\ 1$, our mechanism discovers the contents of object 828 whose value is 0.002371179, the maximum value for keyword 1 existing in a network as shown in Table 4.

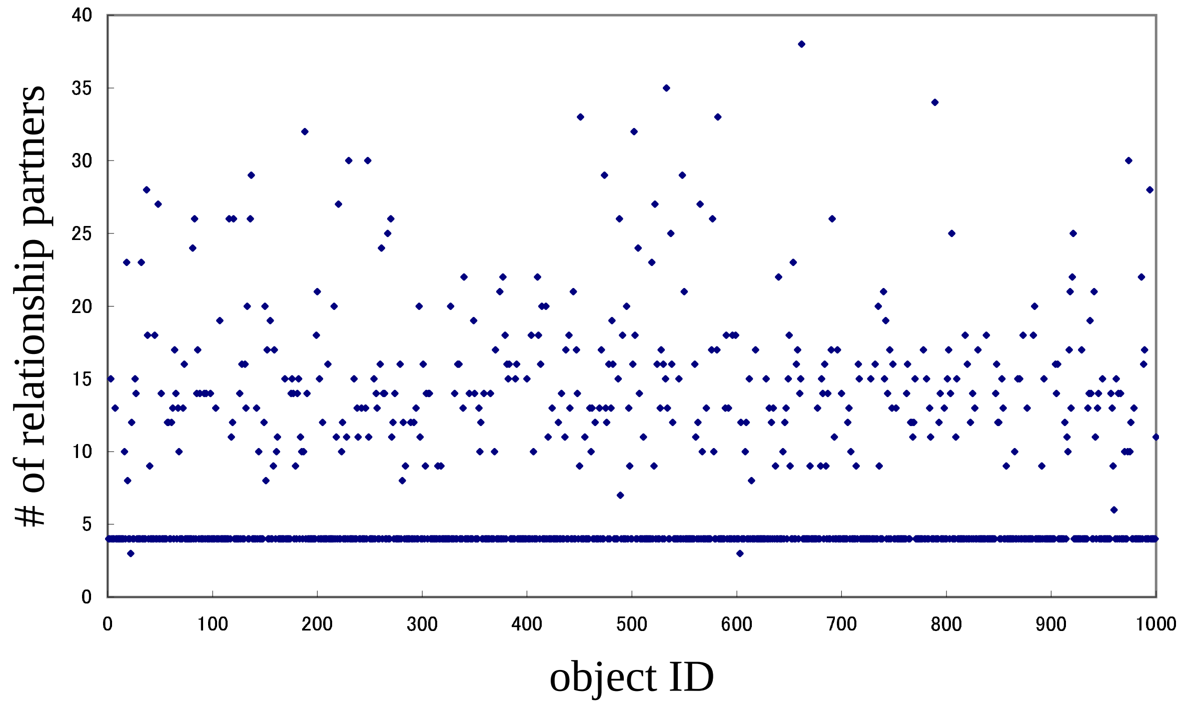

Figure 6: Number of relationship partners for each object.

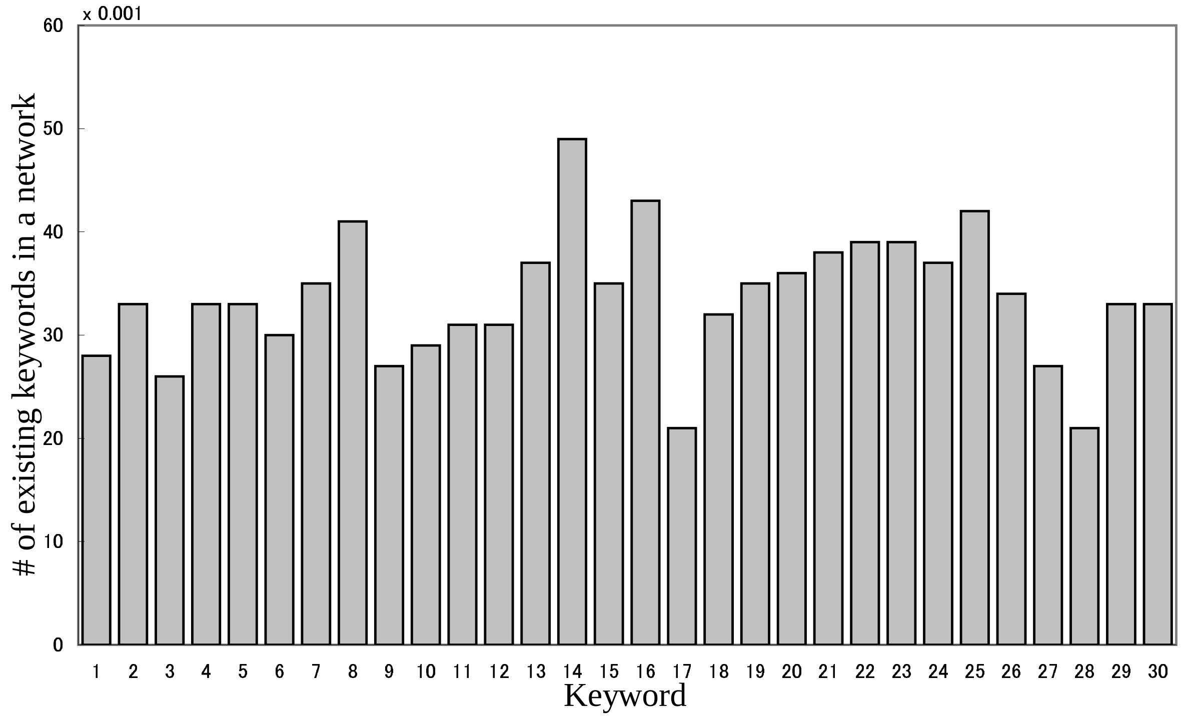

Figure 7: Frequency distribution of keywords.

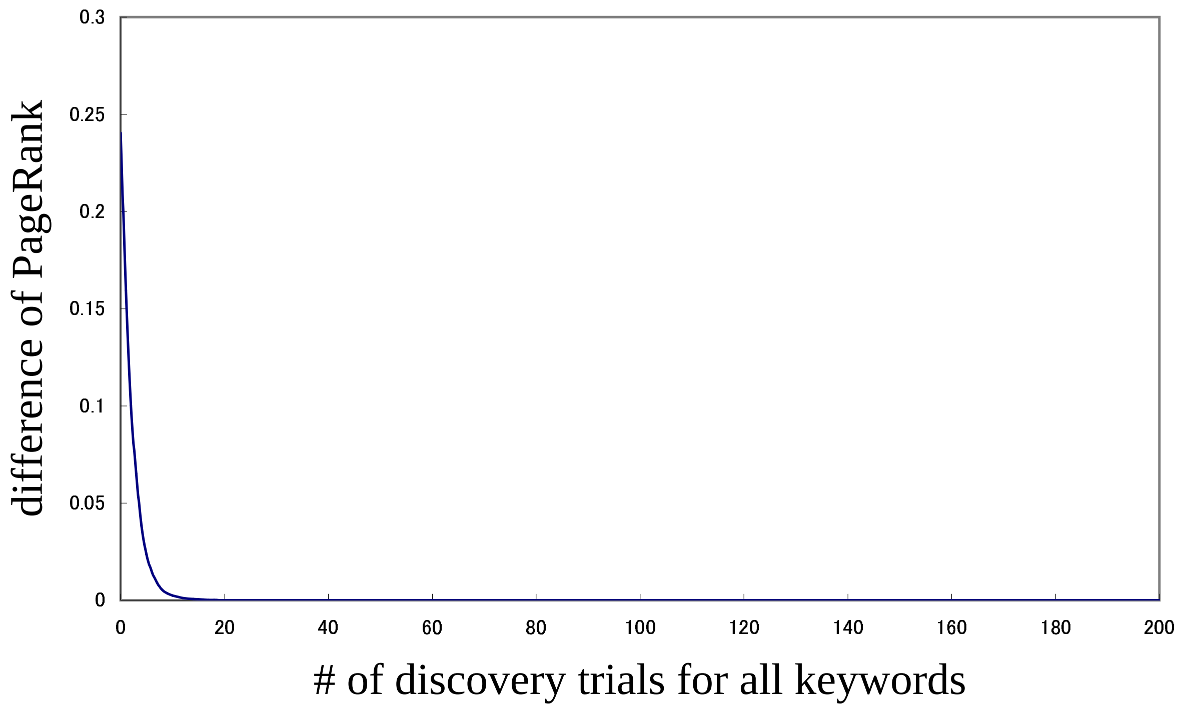

Figure 8: Variation of $S(T(n))$.

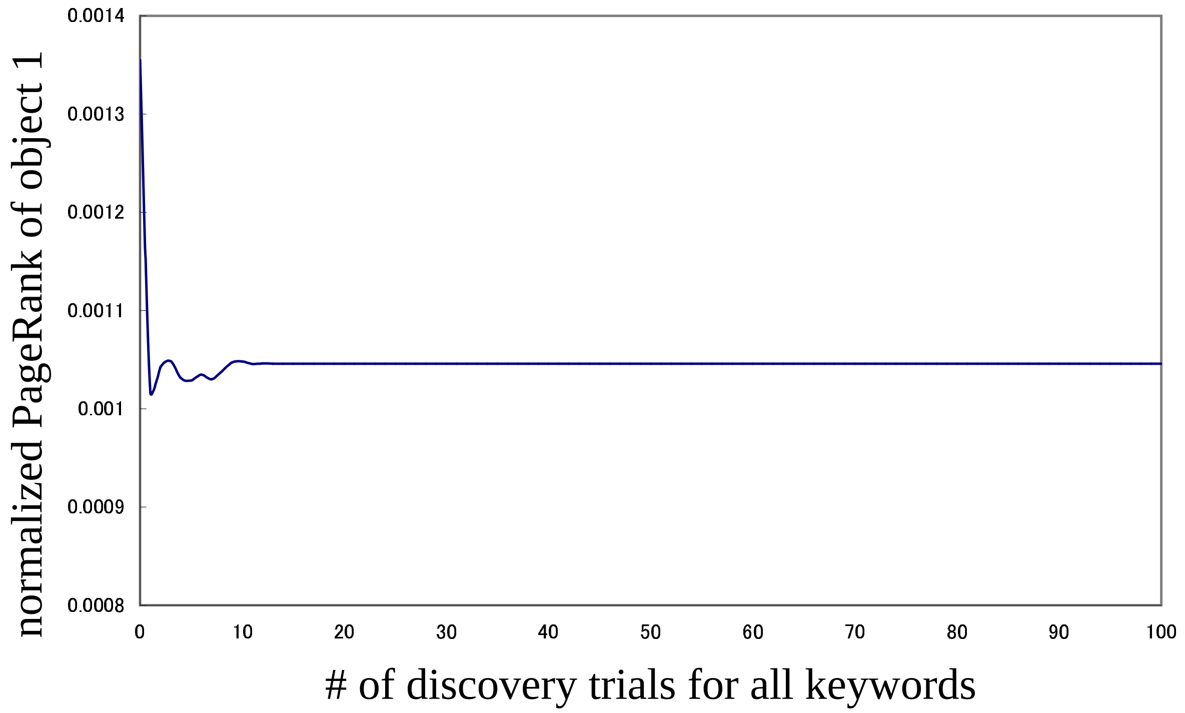

Figure 9: Variation of the normalized PageRank of object 1, $PR(1, T(n))$.

On the other hand, the broadcast-based mechanism does not always discover the best contents in the network. This may come from the fact that the TTL value is fixed to five, which is too small for some originator objects.

Figure 12 and Figure 13 plot the average value of all the received query results in our mechanism and the broadcast-based mechanism, respectively. Figure 12 indicates that our discovery mechanism guarantees the quality of received query results. Namely, our discovery mechanism learns to discover results with higher value on average than the broadcast-based mechanism, illustrated in Figure 13, as discovery trials are repeated.

Figure 14 and Figure 15 plot the number of the received query results in our mechanism and a broadcast-based mechanism, respectively. Figure 14 implies that our mechanism learns the path to relevant contents, resulting in less received results. The broadcast-based mechanism always returns many results, both good ones and poor ones, to originators. This is the reason why our mechanism shows the higher average value in received queries although the maximum value is the almost same as the broadcast-based mechanism.

Figure 16 and Figure 17 plot the number of messages, i.e., query packets, query result packets, and reward packets, used in our mechanism and the broadcast-based mechanism, respectively. The number of overhead is getting smaller in our mechanism as the number of discovery trials increases, but the broadcast-based mechanism does not reduce overhead. This feature in our mechanism assures network efficiency.

5 Conclusion

From simulation results, we conclude that our proposed discovery mechanism shows superior performance, compared with an existing broadcast-based discovery mechanism in terms of network efficiency and a guarantee of query result quality. In addition, this paper is the first official literature of our proposed distributed PageRank, which is expected to be applied into various fields, such as a reputation model in a peer-to-peer [2]. Our future work is to prove Conjecture 3.2.6 and to examine the adaptability of our discovery mechanism to network changing.

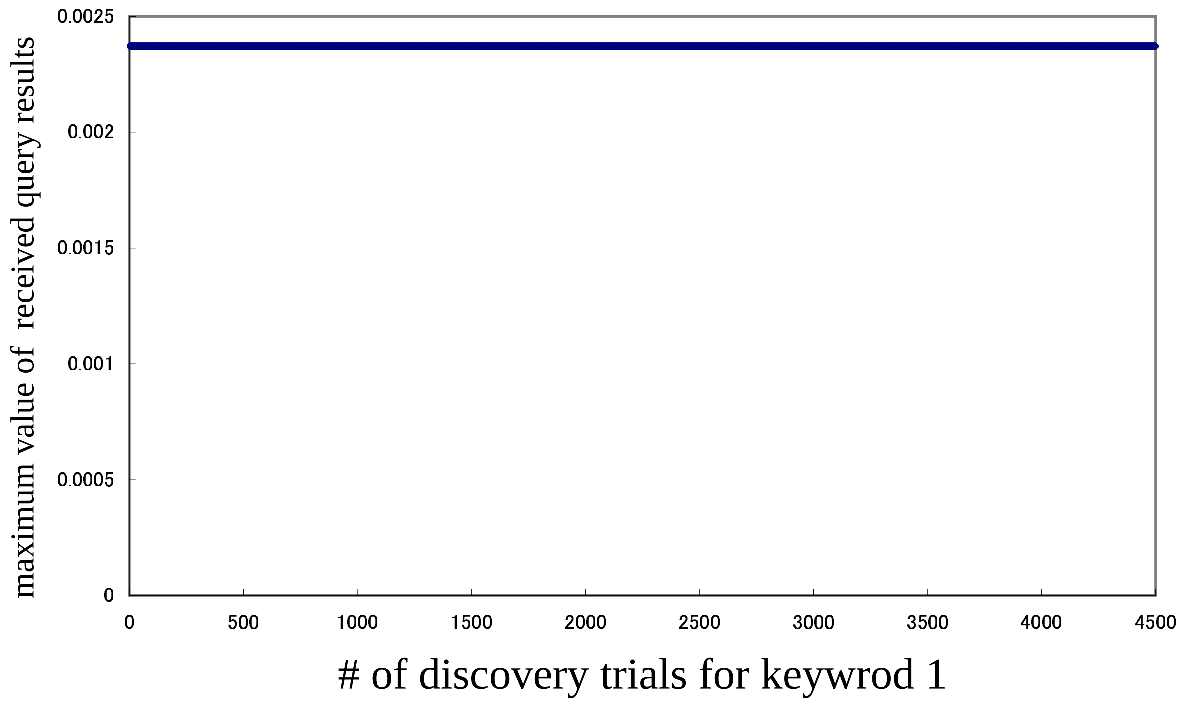

Figure 10: Maximum value of received query results in our proposed discovery mechanism.

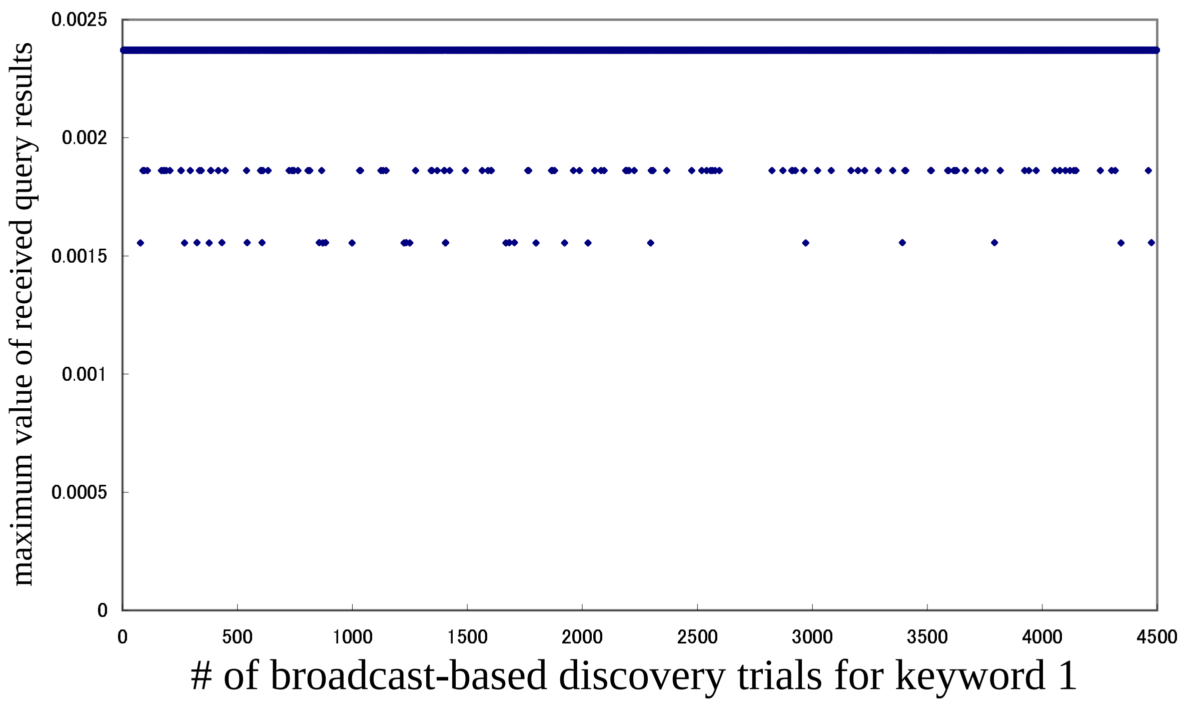

Figure 11: Maximum value of received query results in a broadcast mechanism.

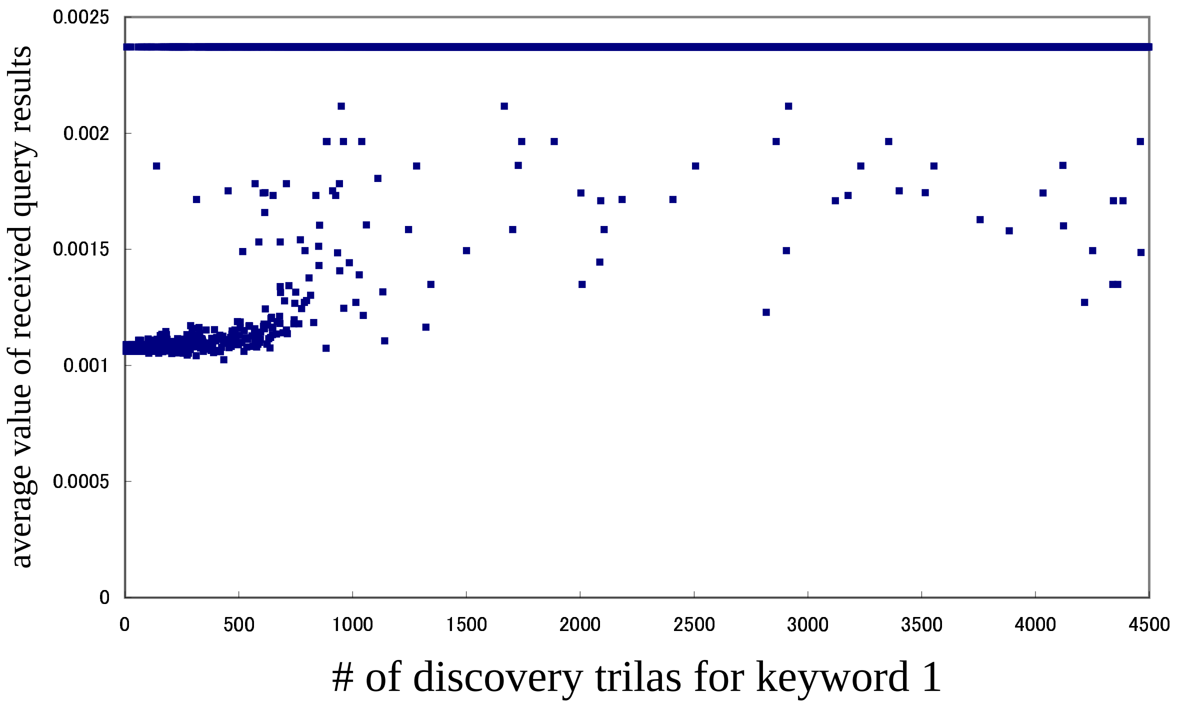

Figure 12: Average value of received query results in our proposed discovery mechanism.

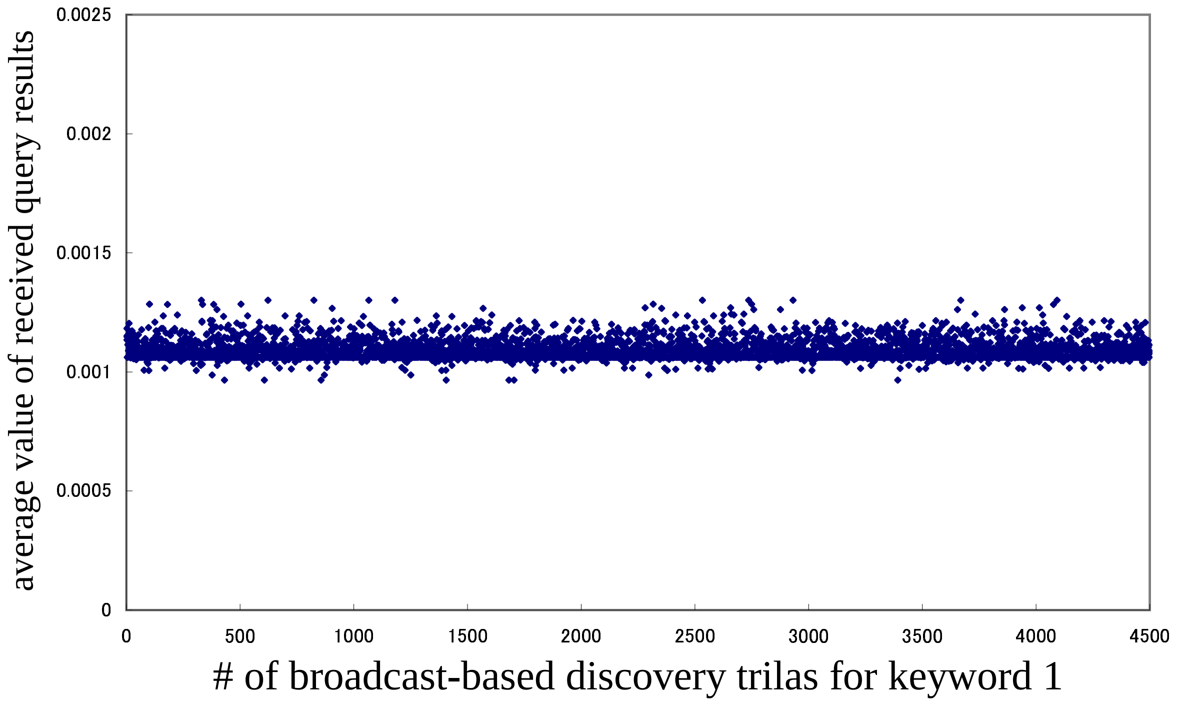

Figure 13: Average value of received query results in a broadcast mechanism.

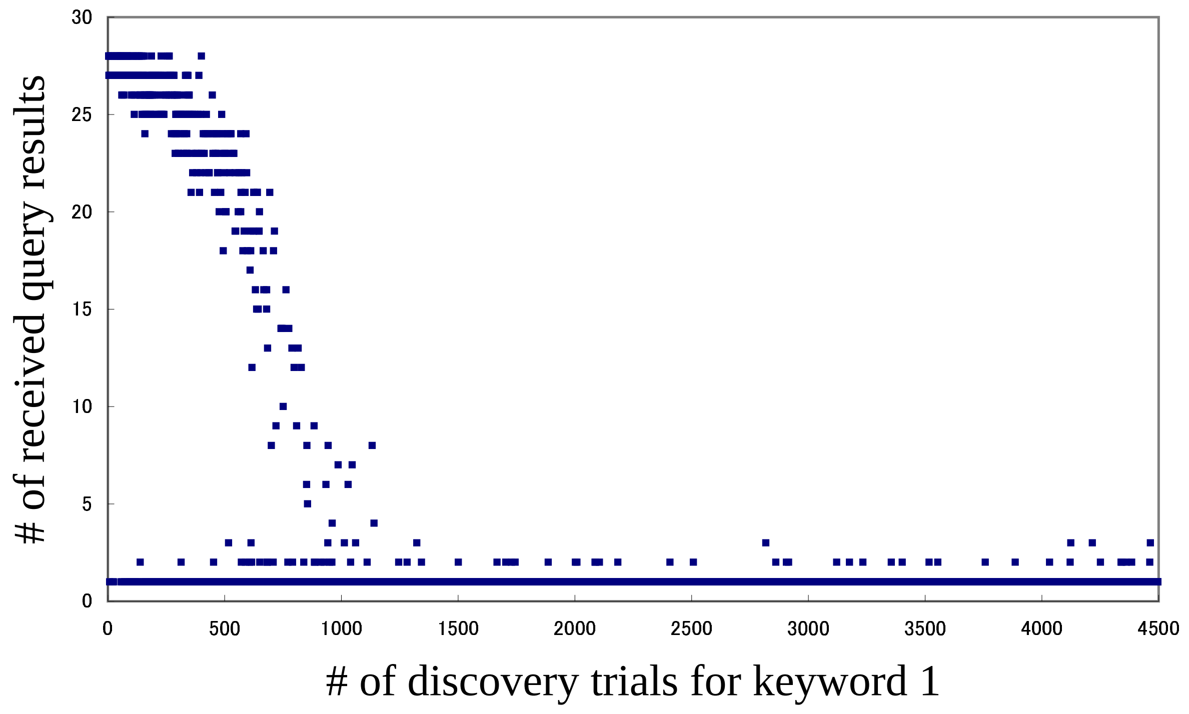

Figure 14: Number of received query results in our proposed discovery mechanism.

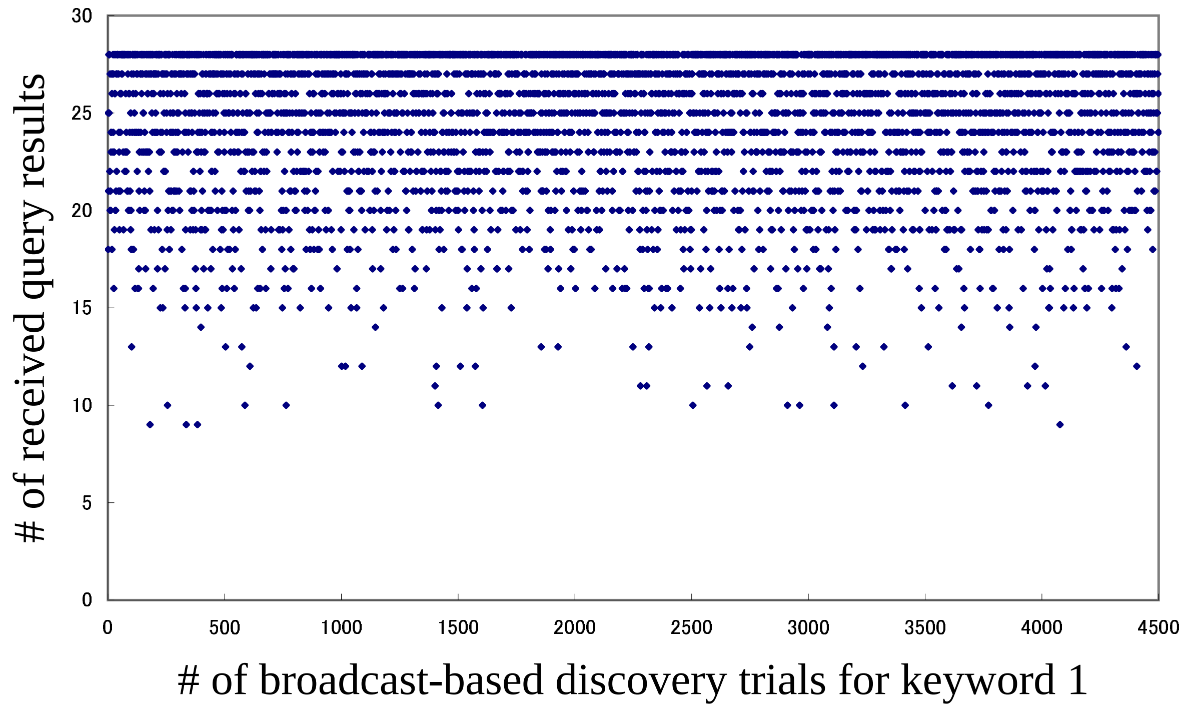

Figure 15: Number of received query results in a broadcast mechanism.

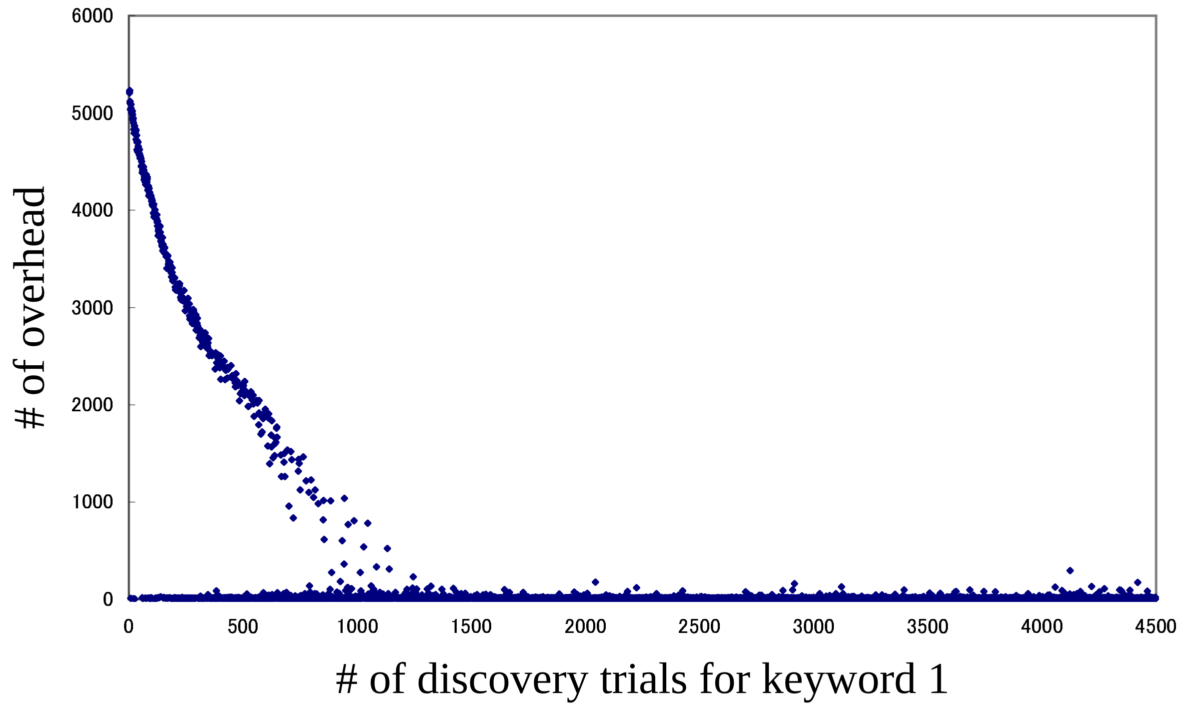

Figure 16: Number of overhead (all the messages a discovery) in our proposed discovery mechanism.

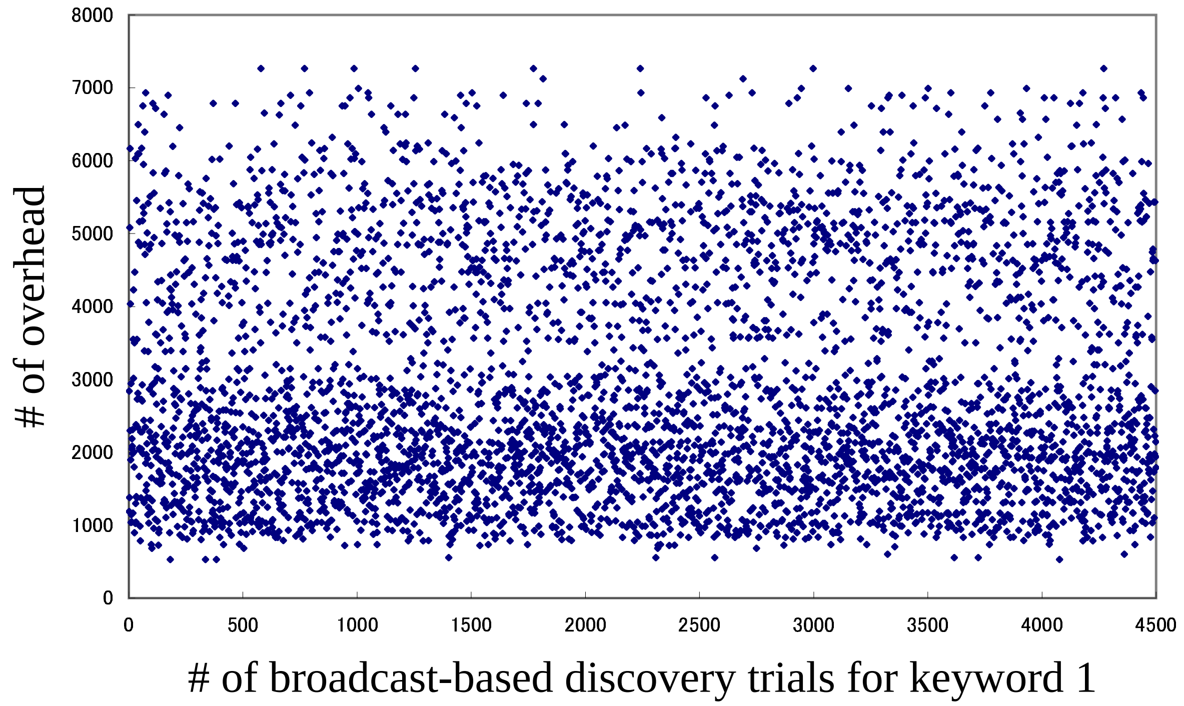

Figure 17: Number of overhead (all the messages a discovery) in a broadcast mechanism.

Acknowledgement

I express my heartfelt gratitude to all the people helping and encouraging me.

I sincerely appreciate Prof. Tatsuya Suda at the University of California, Irvine (UCI) for his patient instruction. He gave me the chance to study at UCI as a visiting researcher. This study has been inspired and brushed up through many discussions with him.

I acknowledge Associate Prof. Tetsuya Takine with my honest gratitude. He is always open to discussion and, undoubtedly, success of this study owes his guidance. I am indebted to Prof. Masao Fukushima not only for this study but also for giving me the opportunities to study in Canada and U.S. I am sincerely honored to have them as my advisor. I express my appreciation to all the Fukushima Lab members as well. Thanks to their cooperation and understanding, I was able to research abroad and complete this thesis.

I appreciate Prof. K. W. Hipel at the University of Waterloo, Canada, for coordinating Exchange Program. He kindly helped us, exchange students, with the life at UW, which is an absolutely critical experience for me. I am also thankful for the coordination of Prof. Norio Okada in this Exchange Program.

Special thanks go to the following friends, Dr. Jun Suzuki, Dr. Tadashi Nakano, Keita Fujii, Ryota Egashira, Ariffin Yahaya, Michael Moore, Yi pang, Akihiro Enomoto, Haruko Takeda, Mark Golinski, Go Suzuki, Naoko Ito and her family, Megumi Obata, Haruka Shoji, Kiyoshi Kaji, Liz Thai, Karmen Too, konja, and UW exchange students. I owe them pleasant days during Master's program, my English skills, or living place as well as brilliant advice of this paper. Their words, still remaining in my brain though, brought a new discovery to my life philosophy.

I thank Satoshi Tanaka, Tomoko Itao, and Atsushi Yamamoto at NTT for applying distributed PageRank developed in this study into their application. I also appreciate Hirotoshi Iwasaki and Hiroshi Sasaki at Denso IT Laboratory for discussions at California and Tokyo.

Last of all, I take this opportunity to acknowledge my family from the bottom of my heart. Living abroad lets me know about the priceless and eternal love of Family.

This notice is the greatest discovery result I received throughout the research, " Discovery. "

Daisuke Asahara
February, 2004

References

[1] A. Oram, Peer-to-Peer Hanessing the power of disruptive technologies, O ’ REILLY, 2001

[2] A. Yamamoto, D. Asahara, T. Itao, S. Tanaka, and T. Suda, “Distributed Pagerank:A Distributed Reputation Model for Open Peer-to-Peer Networks,” *Proc. of the 2004 Symposium on Applications and the Internet Workshops (SAINT2004 Workshops)*, January 26 - 30, 2004.

[3] E. Adar and B. A. Huberman, “Free riding in Gnutella,” *Xerox PARC report*, Oct2000, available at www.parc.xerox.com/istl/groups/iea/papaers/gnutella

[4] Gnutella protocol specification v0.4, http://www.clip2.com/GnutellaProtocol04.pdf

[5] Gnutella. The Gnutella web site, http://www.gnutella.com 2003

[6] Google search engine, http://www.google.com

[7] H. Baba, “The secret of PageRank,”
http://www.kusastro.kyoto-u.ac.jp/˜baba/wais/pagerank.html

[8] K. Sankaralingam, S. Sethumadhavan, and J. C. Browne. Distributed pagerank for p2p systems. *Proc. of the IEEE HPDC-12*, 2003.

[9] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: Bringing order to the web,” *Stanford Digital Libraries Working Paper*, 1998.

[10] M. A. Jovanovic, F. S. Annexstein, and K. A. Berman, “Scalability issues in large peer to peer networks - A case study of Gnutella,” *Tech.Report. Univ.of Cincinati*,2001

[11] M.Ripeanu, “Peer-to-Peer architecture case study:Gnutella network,” *Proc. of International Conference on Peer-to-Peer Computing*, August, 2001.

[12] M. Wang and T. Suda, “The Bio-Networking Architecture: A Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications,” *ICS Technical Report, 05-TR00-03, University of California*, Irvine, 2002.

[13] Netgroup website, http://netresearch.ics.uci.edu/bionet/.

[14] R. Egashira, A. Enomoto, and T. Suda, “Distributed and Adaptive Discovery Using Preference,” *Proc. of the 2004 Symposium on Applications and the Internet Workshops (SAINT2004 Workshops)*, January 26 - 30, 2004.

[15] Robert G. Gallager, Discrete Stochastic Processes, *Kluwer Academic Publishers, Boston*, 1996.

[16] T. Itao, T. Suda, T. Nakamura, and M. Matsuo, “Adaptive Networking Architecture for Service Emergence,” Proc. of SAINT2001, pp.9–16, 2001.

[17] T. Suda, T. Itao, T. Nakamura and M. Matsuo, “A Network for Service Evolution and Emergence : Jack-in-the-Net,” *The Trancactions of the IEICE, Vol.J84-b, No.3, pp.310–320*, 2001.