Adaptive Double Routing Indices: Combining Effectiveness and Efficiency in P2P Systems

Stephane Bressan[1], Achmad Nizar Hidayanto[2], Chu Yee Liau[1], and Zainal A. Hasibuan[2]

[1] School of Computing,
Department of Computer Science,
National University of Singapore
{steph, liaucy}@comp.nus.edu.sg
[2] Faculty of Computer Science,
University of Indonesia
{nizar, zhasibua}@cs.ui.ac.id

Abstract. Unstructured peer-to-peer systems rely on strategies and data structures (Routing Indices) for the routing of requests in the network. For those requests corresponding to information retrieval queries, the emphasis can be either put on the effectiveness of the routing by privileging the relevance of the documents retrieved, or on the efficiency of the routing by privileging the response time. We propose in this paper a novel routing strategy based on adaptive Routing Indices. The Routing Indices are adaptive to the environment, i.e. network traffic, location, as well as relevance of the documents indexed, thanks to a reinforcement learning approach to their maintenance. The strategy can be used to tune the compromise between efficient and effective routing. It combines the estimation of the response time of routes with the estimation of the relevance of routes to keywords. We study performance and the tuning of the compromise offered by this novel strategy under various characteristics of the network and traffic.

1 Introduction

In unstructured peer-to-peer architectures such as those of [**?**,**?**] peers are connected to each other as they join the network and objects are generally not moved. Searching for an object requires the broadcasting of the request to all or some of the neighboring peers. The order in which the peers are contacted depends on the search strategy. The selection of the peers to which the request is forwarded can leverage routing information maintained, for instance, in local Routing Indices. Because of the autonomy of the peers, of the users, and of the underlying network, a routing strategy for a peer-to-peer system in general and for an unstructured peer-to-peer system in particular needs to deal with a continuously changing environment. It must be able to adapt simultaneously to several phenomena: changes in the amount of traffic of requests, responses,

system messages and possibly other packets on the network, availability of objects such as documents being stored at users discretion, peers freely joining and leaving the system, etc.

In this paper, we are interested in studying the routing of keyword-based requests for documents in an unstructured peer-to-peer system. We propose to extend the adaptive Routing Indices that we introduced in [**?**]. While in [**?**] we focused on routing of definite requests, i.e. request to retrieve documents by their identifier or name, we are now interested in search by keyword. Therefore while, as we have shown, the former strategy yielded efficient routing, the new strategy must combine efficiency and effectiveness. It must minimize the response time and maximize the relevance to keywords of the documents retrieved, or at least allow the control of the compromise between the two conflicting factors.

We present Routing Indices that are adaptive to the environment, i.e. network traffic, location, and relevance of the documents indexed, thanks to a reinforcement learning approach to their maintenance. The corresponding routing strategy combines the optimization of both relevance and response time routing thanks to a double dual-reinforcement learning. We study the tuning of the compromise offered by this strategy under various characteristics of the network and traffic.

In the next section we present a brief overview of the main related work on searching and routing strategies in unstructured peer-to-peer systems. We briefly recall the principles underlying reinforcement learning in section **??**. We present the new Routing Indices and the proposed routing strategy with its algorithms and variants in section **??**. In section **??** we evaluate and analyze the performance of the approach we propose under various parameters of the system and environment. In the last section we present our conclusions.

2 Related Work

Different routing strategies for unstructured peer-to-peer networks have been proposed. Gnutella is one of the most popular peer-to-peer systems. Gnutella belongs to the category of unstructured peer-to-peer systems. Its search algorithm simply requires every node to broadcast the search request to its neighbors. Matei presents a performance evaluation of Gnutella in [**?**].

Yang and Garcia-Molina [**?**] have studied various strategies to improve the performance of search in unstructured peer-to-peer systems and distributed information retrieval systems. They studied three techniques: Iterative Deepening, Directed BFS and Local Indices. Iterative deepening uses BFS at depths specified in the policy. The next iteration is applied when the request is not answered. Directed BFS selects a subset of the neighbors to broadcast the request using heuristics techniques. The authors also consider maintaining some routing information in Local Indices. The selection of the neighbors to which a request is forwarded is based on the Local Index. Local Indices maintain a table of pairs (object, neighbor) such that the object is reachable via the peer in the mini-

mum number of hops. The table contains entries for objects reachable within a maximum number of hops. This threshold limits the size of the index.

The Routing Indices by Arturo and Garcia-Molina [?] extend the idea of Local Indices to keyword or topic search. The authors propose several variants in which such information as the estimate of the number of documents per topic when routing to a given neighbor and the distance in hops to documents of a given topic is maintained. The authors demonstrate that Routing Indices yield significantly better performance than flooding or random forwarding strategies. The Routing Indices are created and maintained using (this is not acknowledged by the authors of [?]) the propagation principle of the Bellman-Ford algorithm.

Bellman-ford propagation limits the evaluation of the response time to the evaluation of number of hops ignoring the traffic and network load. Similarly to all algorithms estimating or computing the number of hops (shortest path), it is not capable of identifying and adapting to such phenomenon as traffic jams. In practice, it and all algorithms estimating or computing the number of hops even create traffic jams at bottlenecks by sending similar requests through the same routes. This has been studied in the context of packet routing in communication networks. In this context adaptive routing algorithms such as Q-routing [?] and its variants have been developed.

We have shown in [?] that such adaptive algorithms can be used in the context of peer-to-peer systems to maintain Local Indices, i.e. indices suitable for searching objects by reference. They yield a better performance than existing routing algorithms and adapt to various traffic conditions. We have also shown that Local Indices can be pruned more effectively, i.e. for a better routing performance, based on the popularity of the objects rather than on topological information such as furthest or nearest (as in [?]) located objects.

3 Reinforcement Learning

A reinforcement learning agent [?] learns from its interactions with its environment. In a given state, the reinforcement learning agent selects the next action according to a model it constantly constructs and updates based on the reward that it receives when trying the different actions. A routing agent could, for instance, favor an action that yields the minimum estimated time to destination. Notice that, in order to continuously learn about its environment, the agent must occasionally depart from its base strategy and try random actions.

Reinforcement learning has been successfully applied to many domains, among which network routing. There are numerous forms and variants of reinforcement learning that have been proposed for various network routing problems: Q-Routing [?], Dual Q-Routing [?], Confidence-based Q-Routing and Dual Confidence-based Q-Routing [?], etc. Dual Q-Routing extends the idea of Q-Routing by incorporating backward propagation that yields better performance. Confidence-based Q-Routing in other hand extends the idea of Q-Routing by adding a table that stores the confidence level of Q-value. Experiment in [?] showed that confidence-based yields the best performance among all of them

but requires large storage. For the sake of simplicity and scalability we chose Dual Reinforcement Q-Routing. Further refinement of the reinforcement learning strategy can only improve the results presented in this paper.

In Q-Routing, a node that receives a request for a resource and hosts the resource then provides the resource to the original requester otherwise forwards the request according to the reinforcement learning strategy. It adopts forward propagation strategy to estimate the status of the network by maintaining a table called Q-table that contains Q-value $Q(n, r)$ which quantifies the expected reinforcement when a request for a resource r is sent to n. Dual Q-routing incorporates the idea of propagation of Bellman-Ford algorithm into Q-routing. A node n not only forwards a request to resource r to its neighbor but also transmits the value of $Q(n, r)$ which is used by the neighbor to update its Q-table. The Q-table is updated by forward and backward propagation hence the name of *Dual Q-learning*.

Such algorithms can outperform shortest-paths algorithms such as Bellman-Ford for they are able to estimate routing time taking into account the traffic.

4 Design

The peer-to-peer model we study is following the architecture proposed by [**?**]: we consider an unstructured system in which peers store documents and hold Routing Indices used for by routing strategy. As in [**?**], we consider requests to documents by topics or keywords. We aim at a design that combines the optimization of the response time to a request with the optimization of the relevance of the retrieved documents to the topics in the request. For the relevance we assume that each peer embeds a local mechanism capable of evaluating the relevance of a document to a topic. We assume the availability of a function rf maps pairs of documents and list of topics to a number that we call the relevance. This function can, for instance, be the cosine similarity in a vector space model.

4.1 Routing Indices

We implement the Routing Index as a Q-table of a dual Q-routing algorithm. The Routing Index maintained by a peer o contains, for each neighbor n of the peer and for each topic t values denoted $T_o(n, t)$ (we call them T-values) which represent the estimated minimum time to retrieve a document relevant to t by forwarding a query to n. The Routing Index therefore estimates the response time instead of just the number of hops as in other systems. The Routing Index of each peer o also maintains for each neighbor n of the peer and for each topic t values denoted $R_o(n, t)$ (we call them R-values) which represents the estimated maximum relevance to t of a document obtained by forwarding a query to n.

Peers can seamlessly join and leave the network without necessity to tell to the whole network. The peers joining the network can initialize the content of the Routing Index to the default and initial values, which is 0. Peers leaving the

network inform their direct neighbors to delete the corresponding entries in the Routing Index. If a peer leaves the network on a failure, i.e. without informing its neighbors, its absence will be eventually discovered from its estimated performance in the Routing Index.

For a peer o each entry in the Routing Index is of the form:

$$(n, (T_o(n,t_1), R_o(n,t_1)), \ldots, (T_o(n,t_m), R_o(n,t_m)))$$

for each direct neighbor peer n, where m is the number of topics.

Values in the Routing Index, i.e. T- and R-values, are updated from both forward and backward propagation of the neighbors T- and R-values respectively upon sending or forwarding or receiving a query.

T-values for a peer o and a topic t are updated according to the following Q-routing formula:

$$T_o(n,t)^{new} = T_o(n,t)^{old} + l(T_n(t) + q_{o,n} - T_o(n,t)^{old})$$

where n is the neighbor of o transmitting its best T-value for t, namely $T_n(t) = \min_{p \in neighbor(n) \wedge p \neq o}(T_n(p,t))$, $q_{o,n}$ is the overhead communication cost from o to n, l is a number between zero and one which determines the learning rate. The bigger l, the more sensitive to changes the system. When l is set to 1, the equation becomes:

$$T_o(n,t)^{new} = T_n(t) + q_{o,n}$$

which updates the T-value without considering the old value.

R-values for a peer o and a topic t are updated according to the following formula:

$$R_o(n,t)^{new} = R_n(t)$$

where n is the neighbor of o transmitting its best R-value for t, namely

$$R_n(t) = max(\max_{p \in neighbor(n) \wedge p \neq o}(R_n(p,t)), \max_{d \in doc(n)}(rf(d,t)))$$

It is a learning process with a rate of 1 and an overhead cost of zero. The relevance function rf is used to compute the actual relevance (retrieval status value) of stored documents.

Clearly the estimated R-values are expected to be less subject to fluctuation than the T-values. Indeed, although both values depend on the network structure (peers leaving and joining) the T-value depends on the traffic (requests and responses) while the R-value depends on the documents content and location. We expect the traffic to be most dynamic element of the system.

As we have shown in [**?**] for Local Indices, we expect to be able to most effectively prune Routing Indices without damaging their routing performance using the popularity of information, i.e. in this case, the frequency of topics. We will not study this pruning in this paper to concentrate on the adaptive combination of response time and relevance indices.

4.2 Routing Strategy

The general routing strategy generally routes a request to the neighbor[3] with the smallest T-value for that request for a strategy seeking to optimize the response time and to the neighbor with the highest R-value (while the local R-value is smaller than the neighbors R-value) for that request for a strategy seeking to optimize the relevance. Occasionally, requests are randomly routed to allow the correction of the estimated values. Also, in practice requests are deleted from the system after they have travelled a predetermined number of hops known as the Time-to-Live or TTL.

A strategy seeking the combined optimization of the response with the relevance of the retrieved documents clearly calls for a trade-off. The more exhaustive, therefore the longer the search and the higher the chances to locate and retrieve more relevant documents. Such a strategy combines the T-values and the R-values into a single value obtained as follow.

First the T-and R-values are normalized as follow:

$$norm\_R_o(n,t) = \frac{R_o(n,t)}{\max\limits_{y \in neighbor(o)}(R_o(y,t))}$$

and

$$norm\_T_o(n,t) = 1 - \frac{T_o(n,t)}{\max\limits_{y \in neighbor(o)}(T_o(y,t))}$$

For every pair of T- and R-values in the Routing Index a weighted sum, $V(o,t)$, of their normalized values called the V-value or routing value is computed as follow:

$$V(o,t) = w \times norm\_T_o(n,t) + (1-w) \times norm\_R_o(n,t)$$

Queries are forwarded to the neighbor with the highest routing value. A value close to 0 for w emphasizes higher relevance, while a value close to 1 emphasizes better response time.

The reader notices that the weight w needs not be a parameter fixed globally to the system nor locally to the machines but can be associated to each individual request thus allowing users to indicate their preference for the combination of efficiency or response time and effectiveness or relevance.

In the general case, unless the TTL limit is reached, a document d at a peer n is returned as the answer to a query on a topic t and the search is stopped when its relevance is equal or greater than

$$(1-w) \times \max_{p \in neighbor(n)}(R_n(p,t))$$

[3] Without loss of generality we will evaluate the strategy in the simpler case in which queries are forwarded to a single neighbor. The scheme naturally extends to forwarding queries to several neighbors

The reader has noticed that we implicitly defined as relevant a document with non-zero relevance. This hypothesis can easily be changed to a minimum acceptable relevance. The model can also be generalized to allow a set of result to be returned and sorted.

4.3 Dynamically Controlling Effectiveness and Efficiency

Numerous and various functions can be used to combine the T- and the R-values into a single value used for the routing strategy. A simple weighted function can achieve some level of direct control but leave some risks of divergence of the response time (i.e. the response time increases as the elapsed time increases), as experiments presented in the next section show. To demonstrate the opportunity to define functions adaptive to various parameters of the environment lets us try and design a function which is adaptive to the network load.

Obtaining information about the network level of activity requires information from other peers participating in the system. However, the local activity may be a good estimate of the network activity as far as the routing of queries is concerned. We locally measure the load of each machine o as:

$$l_o = \frac{N_m}{C_p}$$

where N_m represents the number of messages (requests and system messages) in the queue of the peer o and C_p represents the number of messages that peer o can process in a unit of time (which for instance can be assumed to be commensurate to its processor speed and other hardware and system features).

We now define a function of l_o, returning the weight w. We want to design a function that takes the network load into account when choosing a compromise between efficiency and effectiveness or response time. When the network in low load the function should favor effectiveness or relevance with a small value of w: the function is decreasing. When the network in high load the function should favor efficiency with a large value of w: the function is increasing. In summary, we wish to design a function such that:

- The value of the function is between 0 and 1 (since the w expected is within this range)
- The function is continuous (to be able to display a gradual effect)
- The function follows l_o monotonically.

There are many such functions. As an example, we propose the following function:

$$w_{=} \begin{cases} 1, & if\ \frac{N_m}{C_p} > \theta + \varphi \\ 0, & if\ \frac{N_m}{C_p} < \theta - \varphi \\ \frac{\left(\frac{N_m}{C_p} - (\theta - \varphi)\right)}{2\varphi} & otherwise \end{cases}$$

As shown in figure **??**, θ is used to control the inflection point of the function. θ takes its values between zero and infinity. θ reflects the normal value of l_o. If

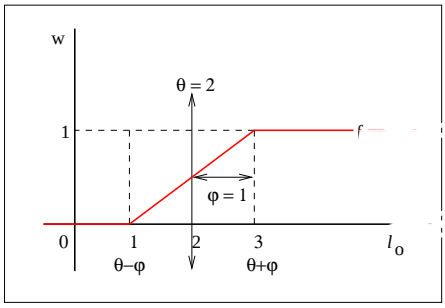

Fig. 1. A Function Adaptive to the Network and Machine Load

the value of l_o is equal to θ then w is equal to 0.5 therefore giving equal weight to efficiency and effectiveness. Setting θ to a large value relatively to l_o is expected to increase relevance because the w becomes small. φ is used to control the gradient of the function. The value of φ ranges between zero and infinity. When setting φ tends towards zero, the slope of the function near its inflection tends towards vertical. When the value of φ is large the slope of the function tends toward the horizontal making the function less sensitive to variation of l_o and thus striking a balance between effectiveness and efficiency.

Although the proposed function has now two parameters θ and φ instead of one w, our experiments (some of which are discussed in the next section) have shown that the proposed function is capable of adapting to the network and machine load. In that sense the two parameters are secondary fine tuning parameters. As with w the two parameters θ and φ can be set for individual queries and need not be global to the system or to a peer.

5 Performance Analysis

[4]We implemented a discrete event simulation of our proposed strategies to evaluate their performance. In our experiments, we use a re-wired ring lattice network topology as suggested by I. Clark et. al. in [**?**] for the simulation of a peer-to-peer network. A ring lattice is a graph with n vertices (peers), each of which is connected to its nearest k neighbors. In our experiments we use a k value of 4. The rewiring is done by randomly choosing m vertices to connect with each other. We use an m value of 10% of n. The network we use for these experiments contains 36 peers without loss of generality since we have shown in [**?**] it can be scaled to larger number of nodes.

We consider 500 documents and 1000 topics. For each document, we chose 10% of the topics. For each document and for each of these chosen topics, we randomly chose a relevance value. For each document d, we randomly choose 10% of the peers to store it. The number of queries on a given topic is determined by a

[4] The figures in this section can be viewed better with colors

uniform distribution (see [?] for non uniform distribution with a simpler routing strategy). For each query, the peer emitting the query is chosen randomly.

In the simulation, the logical time needed to transmit a message between two connected peers is 1 unit of time. Each peer in the network has the capability to process 5 messages in 1 unit of time. We recall from section ?? that

$$l_o = \frac{N_m}{C_p}$$

where N_m represents the number of messages in peer o and C_p represents the number of messages that peer o can process in a unit of time. Each set of experiment is done in three different network loads: low, medium and high. We define the different network loads by varying the number of new queries sent at each unit of time:

- for low traffic we send 10 queries per unit of time;
- for medium traffic we send 25 queries per unit of time;
- for high traffic we send 40 queries per unit of time;

The metrics we use in the sequel are the average routing time of a newly answered query (in units of simulation time) per elapsed simulation time, and the relevance of a newly retrieved document per elapsed simulation time. When not ambiguous, we refer to these metrics as routing time and relevance, respectively.

First we study the time taken to answer requests in routing strategies where full priority is given to time, relevance as well as random routing. When routing with full priority is given to time, a request is always routed to a neighboring peer expected to lead to a peer storing a relevant document within the shortest response time. As shown in the [?], the learning algorithm tries possible path and then learns the situation to get the faster path. This mechanism explains why in the beginning of learning process the curve is increasing and decreasing after some time point that reflecting the behavior of Q-Routing algorithm. Because we release a number of queries(messages) in each unit time, some queries wait in the queue of the node, which explains why in some cases the order of time is hundred because most of the life time of the queries spent in the queue. In routing with full priority given to relevance, a request is always routed to the peer expected to lead to a peer storing a document with the highest relevance. As a theoretical reference, we also use a random routing that routes requests to a neighbor peer chosen randomly.

Figure ?? shows the response time for the three strategies. We observe that, except when full priority is given to time, the response time diverges, i.e. it increases as the elapsed time increases. Indeed, the queue of each peer are unbounded in our simulation. The response time diverges without bound as the queue build up. The strategy giving full priority to relevance performs worse than a random strategy because it creates traffic jams at bottlenecks. Figure ?? shows the relevance for the three strategies. We observe that when full priority is given to relevance, the relevance converges towards its maximum and that, for the two other strategies, average relevance is obtained as we would expect

given the experimental set-up. Figure **??** and **??** confirm our design showing that w with extreme values favors efficiency or effectiveness exclusively. They demonstrate the need for a combined strategy motivating the research presented here.

On figures **??** and **??**, we can see that different values for the weight w indeed produces the expected gradual effect on routing time and relevance: routing time converges faster and to a smaller value as w varies from 0 to 1 and relevance varies from average (0.5) to maximum 1 as w varies from 1 to 0. This confirms that the strategy we propose can be tuned.

Let us now consider that, as we suggested, individual users choose a value of w corresponding to the balance they wish for an individual query (w is then transmitted to the peers together with the query). We have conducted experiments in which a different value of w is randomly generated for each query. The net-effect is similar to setting w to 0.5 while the effect for the subsets of queries with the same value of w follows the general trends displayed on figures **??** and **??**. This means that w can generally be used as a parameter of the query.

However it also generally difficult to set a satisfying value of w because network traffic and machine load may be of crucial influence on the response time. Indeed figure **??** shows that under medium load network already, a too strong emphasis on relevance can result in a diverging routing time. We do not show the corresponding graph but confirm that our experiments show that the phenomenon is accentuated in a high load network. This last observation compels more adaptive strategies, i.e. strategies that can automatically tune the value of w based on the network and machine loads thus avoiding divergence of the response time.

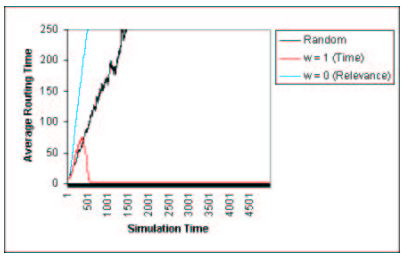

Fig. 2. Average routing time with different routing strategies for Medium Load

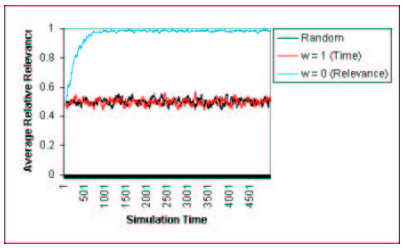

Fig. 3. Average relative relevance with different routing strategies for Medium Load

Figures **??** to **??** present the routing time and relevance performance for a strategy based on the function presented in section **??**. The results are presented for arbitrary values for the φ and θ parameters. It is beyond the scope of this paper to study the detailed effect of the values of the parameters on the performance since the purpose of the function is only to prove the feasibility of the automatic tuning. The results confirm the expected self-tuning effect. They

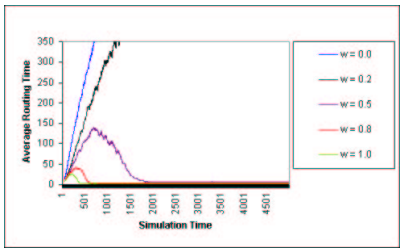

Fig. 4. Routing Time - Static w for Medium Load

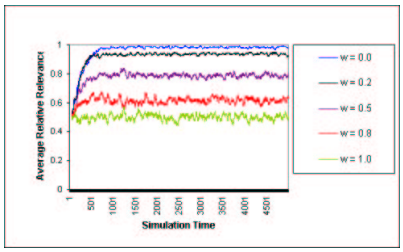

Fig. 5. Relevance - Static w for Medium Load

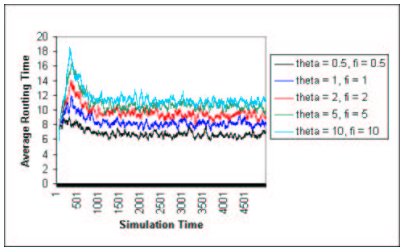

Fig. 6. Routing Time - Dynamic Controlling for Low Load

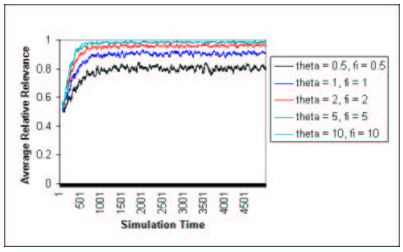

Fig. 7. Relevance - Dynamic Controlling for Low Load

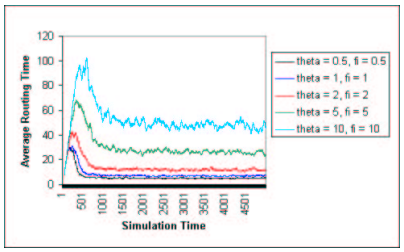

Fig. 8. Routing Time - Dynamic Controlling for Medium Load

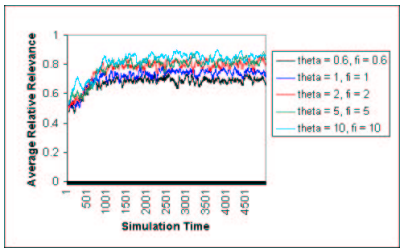

Fig. 9. Relevance - Dynamic Controlling for medium Load

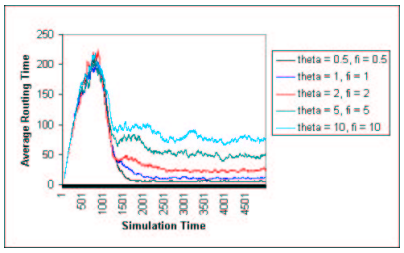

Fig. 10. Routing Time - Dynamic Controlling for High Load

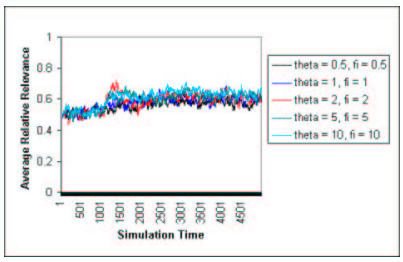

Fig. 11. Relevance - Dynamic Controlling for High Load

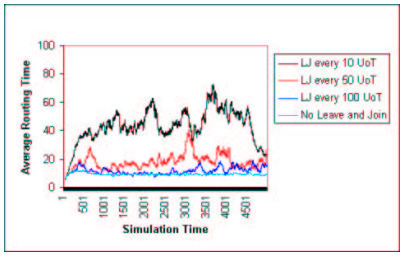

Fig. 12. Routing Time - Dynamic Leave and Join with Low Network Load

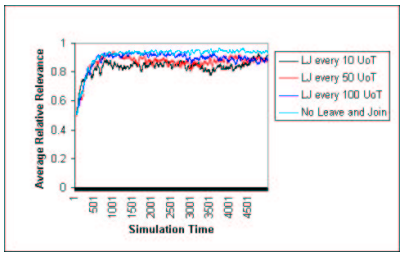

Fig. 13. Relevance - Dynamic Leave and Join with Low Network Load

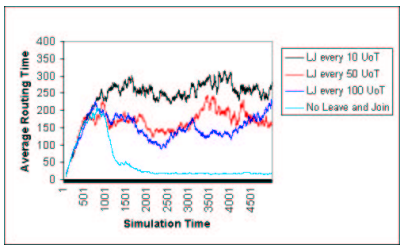

Fig. 14. Routing Time - Dynamic Leave and Join with High Network Load

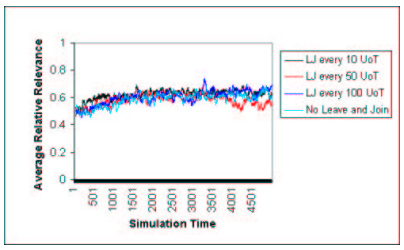

Fig. 15. Relevance - Dynamic Leave and Join with High Network Load

confirm the expected properties sought by design. The function is adaptive and can avoid diverging routing time including under high load as shown on figure **??**. Indeed as the queue sizes change at the different peers, the input l_o of the function varies locally for each peer and enables the adaptive behavior. In a high load network, as the queue sizes possibly increase, relevance improvement is sacrificed but relevance itself does stabilize between 0.5 and 0.7.

Finally we want to study the effect of a dynamic peer-to-peer network on the result. We did experiments reflecting the continuous changes that may occur in a peer-to-peer network. It only confirms that the strategy is robust to a sudden change provided enough time is available to stabilize. Let us evaluate how much continuous and incremental change is acceptable. For this experiment one peer leaves or join the network every t units of time. Figures **??** to **??** show the experiment results for the routing time and relevance with t varying from 10 to 100 and with low and high loads respectively. The reader will admit that a rate of one peer- joining or leaving every 10 units of time in a network of an average of 36 peers constitutes a very unstable environment. Relevance is not significantly affected by the changes in the network. Even when these changes are very frequent (a peer leaves or joins every 10 units of time) in a high load network, the relevance (see figure **??**) is maintained around its value in a static environment (see figure **??**). The routing time is more significantly affected yet

it does not diverge in any of our experiments including in an environment with frequent changes and under high load (see figure **??**).

6 Conclusion

Existing designs and proposals for unstructured peer-to-peer systems that rely on Routing Indices use shortest path algorithms that compute or estimate the number of hops to a document but fail to estimate the actual response time. Strategies combining the estimate of the number of hops and the relevance are straightforward but hide behind their simplicity the weakness of the prediction intrinsic to using the number of hops.

We have designed and evaluated an approach that not only adaptively estimates both response time and relevance but also adaptively combines them (on a per-query basis) into a single strategy. The adaptive estimation of the response time and relevance values in the Routing Index is achieved with two reinforcement learning algorithms, respectively. The adaptive combination of the response time and relevance values is achieved by a parameterized function that takes into account the local load (network and machine) at the peer. The parameters give to the user a fine grain control on the compromise between efficiency and effectiveness of the search for her request.

The empirical analysis that we have conducted confirms the efficiency of our proposed approach as well as its robustness to challenging conditions such as high network load and unstable networks.