Improve Searching by Reinforcement Learning in Unstructured P2Ps

Xiuqi Li and Jie Wu
Florida Atlantic University
Boca Raton, Florida 33431
{xli, jie}@cse.fau.edu
Shi Zhong
Yahoo! Inc.
Sunnyvale, California 94089
frank.shizhong@gmail.com

***Abstract*— Existing searching schemes in unstructured P2Ps can be categorized as either blind or informed. The quality of query results in blind schemes is low. Informed schemes use simple heuristics that lack the theoretical background to support the simulation results. In this paper, we propose to improve searching by reinforcement learning (RL), which has been proven in artificial intelligence to be able to learn the best sequence of actions in order to achieve a certain goal. Our approach, ISRL, aims at locating the best path to desired files at low cost. It explores new paths by forwarding queries to randomly chosen neighbors. It also exploits the paths that have been discovered to reduce the cumulative query cost. Two models of ISRL are proposed: the basic ISRL for finding one desired file, and MP-ISRL for finding multiple desired files. ISRL outperforms existing searching approaches in unstructured P2Ps by achieving higher query quality with less query traffic. The experimental result supports the performance improvement of ISRL.**

Keywords: ***Hint-based search, intelligent search, peer-to-peer networks, reinforcement learning, unstructured P2P.***

I. Introduction

An unstructured peer-to-peer (P2P) network is an overlay network where all nodes play equal roles, and the topology and data location do not follow restrictive rules. It is often used for each user to share local files with other users in the same network [1]. Searching is the basic operation in unstructured P2Ps. Existing searching approaches can be classified as *blind* or *informed*. An informed search, such as routing indices [2], utilizes hints to facilitate query forwarding. A blind search, e.g. the random walk [3], blindly sends queries. Informed schemes usually achieve a higher search quality than blind ones.

Existing informed approaches can be categorized into either proactive or reactive. Proactive approaches, such as routing indices [2], propagate hints before files are queried whereas reactive approaches, for example intelligent search [4], collect hints after query processing. Reactive approaches do not waste resource in propagating hints about unpopular files. Routing indices spreads hints about document topics. Intelligent search collects past similar queries. All these informed schemes employ simple heuristics and lack theoretical support for their effectiveness. In addition, reactive approaches do not try to improve the quality of collected hints.

This work was supported in part by NSF grants CCR 9900646, CCR 0329741, ANI 0073736, and EIA 0130806.

In this paper, we propose to systematically learn the best route to desired files through reinforcement learning (RL) [5]. RL addresses the general issue of how a learner that interacts with its environment can learn the optimal actions to achieve its goal. Trial-and-error learning and the delayed reward mechanism are the most important characteristics of RL. Whenever a learner takes an action, it receives an immediate reward and the environment changes its state. The learner discovers which actions yield the highest cumulative reward by trying them. An action may affect both the immediate reward and all future rewards. RL has been proven to converge to the best sequence of actions with the maximum cumulative reward.

When applying RL to P2P searching, each node is a distributed learner and the P2P network is its environment. An action is a query forwarding. Each query path is a sequential process. The goal is to reach a node that hosts a desired file. All nodes (learners) work together to learn the best query path to hosting nodes. Each learner iteratively estimates which next-hop neighbor is the best one to forward a given query by trying them. When a query is successfully resolved, rewards are propagated along the reverse query paths. After receiving rewards, each node updates its estimation to reinforce better paths. Each learner can also adapt itself to the network dynamics through this trial-and-error mechanism.

RL has been used in [6] to adapt P2P topologies to peer interests. This paper applies RL to P2P searching without topology adaptation. We propose two models: basic ISRL and MP-ISRL. The basic version is designed for locating one desired file efficiently. Each node learns the best path to one desired file by exploring new potential paths and exploiting the already discovered paths. To balance exploration and exploitation, each node adapts its exploration coarsely or in a fine-grained manner based on the quality of the learned paths. MP-ISRL aims at finding more than one file efficiently. Each node learns the top-k best paths to desired files.

The following are contributed in this paper.

1) We put forward a searching scheme, ISRL, that systematically learns the best path to a desired file by reinforcement learning and adapts itself to system dy-

namics. To the best of our knowledge, this is the first work that applies RL to P2P searching without topology adaptation.

2) We design two models of ISRL, the basic version for locating the best path, and MP-ISRL for finding the top k best paths. Important design issues, such as balancing exploration and exploitation are also discussed.
3) We conduct simulations and compare results with existing searching schemes.

The remainder of this paper is organized as follows. In Section II, we review the related work. In Section III and IV, we introduce the basic ISRL and discuss MP-ISRL respectively. Simulation results are presented in Section V. Our work is summarized in Section VI.

II. Related work

Existing searching schemes in unstructured P2Ps can be classified into blind or informed. Flooding, Depth First Search (DFS), random walks, and their variations are blind searches. Flooding [7] is a Breadth First Search (BFS) of the overlay graph. DiCAS [8] uses grouping to restrict flooding among a subset of nodes in the P2P. Freenet [9] uses a DFS of the P2P overlay. Random walks [3] [4] try to reduce message redundancy in flooding. k-walker random walks [3] deploy k random walkers at the querying source. Modified random BFS [4] forwards a query to a random subset of neighbors at each receiving node.

Informed searches include intelligent search [4], APS [10], hybrid [11], routing indices [2], and ESS [12]. Intelligent search directs a query to a subset of neighbors that answered similar queries previously. APS forwards a single file lookup query probabilistically based on the query history and the guesses of query sources. APS can be viewed as an ad-hoc application of reinforcement learning in specific single file lookup queries. Our ISRL approach is more general and theoretically sound. Hybrid probes and forwards queries based on simple heuristics such as the number of files. Routing indices forwards a keyword query probabilistically based on the propagated keyword information. ESS tries to resolve semantic queries efficiently by converting the unstructured P2P overlay to a semantic overlay dynamically.

Q-learning [13] is one simple and widely used RL scheme. It defines a Q function for each state-action pair (s,a). $Q(s,a)$ represents the expected discounted cumulative reward obtained by taking action a at state s. A learner estimates $Q(s,a)$ values iteratively based on experiences as follows. At current state s, take a selected action a, receive an immediate reward r, and observe the new state $s^{'}$. Then the learner updates $Q(s,a)$ according the following formula,

$$Q(s,a) = (1-\eta)Q(s,a) + \eta(r + \gamma max_b Q(s^{'}, b)),$$

where η is the learning rate and γ is the discount factor. Both are in the range $[0,1]$. $Q(s,a)$ can be implemented as a simple table or a trainable parameterized function. RL has been used in [6] to adapt the P2P overlay topology to peer interests. RL has also been used in packet routing in switch networks [14].

III. The basic ISRL

The goal of the basic ISRL is to deliver a query through the best path to a node hosting the desired file. It explores new paths by forwarding query q to a random neighbor with probability p_q. It exploits existing paths by sending q through the best-so-far path with probability $1-p_q$. The newly explored better path replaces the existing path during path updates. We consider semantic queries for files with similar semantic content.

A. Semantic content representation

The vector space model in IR is used to represent the semantic content of documents and queries. Each document is represented by a semantic vector, where each dimension is the weight of a term appearing in a document. Term weights indicate the importance of terms in describing the semantic content of documents. The size of semantic vectors is equal to the size of the vocabulary for the document collection. Stemming is often used to reduce vector dimensions. A user can issue a semantic query described in a natural language. The system extracts a semantic vector for such a query just like extracting a semantic vector for a document [15]. Each query vector or document vector is normalized such that its Euclidean vector norm is 1 before similarity computation.

Many term weighting schemes have been proposed in the IR literature [15]. We calculate the weight of a term t, denoted by w_t, in a document d according to the following formula.

$$w_t = 1 + \log(tf_t),$$

where tf_t refers to the number of occurrences of t in d. This scheme does not require global knowledge of the document collection in a P2P network, and has been demonstrated to be effective in document clustering [16].

B. Path entries

To facilitate learning the best path during query processing, each node, say x, keeps one path entry for each query vector that was resolved successfully through x. Table I lists the items in an entry. q represents the query vector. z is the best neighbor for finding files similar to q that x has discovered so far. Q_{xq} refers to the path cost in terms of the number of overlay hops. It corresponds to the Q value in Q-learning but needs to be minimized. p_q represents the probability of exploring new paths for q. $cntU_q$ counts the consecutive number of updates to Q_{xq} that are smaller than a threshold. These path entries are created when desired files were found the first time. They are continuously monitored during path exploration and update process, which will be discussed later in this section.

C. 1-thread semantic search

All nodes handle queries similarly. When node x receives from node y a Query message for query vector q initiated at node s, it takes the following actions.

1) If a local file f is semantically similar to q, x replies to y a QueryResponse message that includes the query, the detailed description about f, and the cost $Q_{xq} = 0$.

Notation	Meaning
q	The query semantic vector
z	The neighbor on the best-so-far path for q
Q_{xq}	The cost of the best-so-far path for q
p_q	The path exploration probability for q
$cntU_q$	The consecutive number of minor updates to Q_{xq}

TABLE I
THE QUERY-PATH ENTRY FOR QUERY VECTOR q AT NODE x IN BASIC ISRL

The query will not be forwarded further. If no desired file exists on x, go to step 2).

2) If x has a path entry that contains vector q, then with probability p_q, the query is forwarded to a randomly chosen neighbor other than z (exploration). With probability $(1-p_q)$, the query is sent to the best neighbor z discovered so far (exploitation).
3) If x does not have an entry for q, it then directs the query to a neighbor chosen randomly.

A query response message is sent along the reverse query path and terminates at the querying source. p_q is an important design parameter. It will be discussed in detail in the next subsection. To avoid searching loops (duplicate queries), each Query message carries all node IDs on the query path so far.

The cosine similarity model is selected for evaluating the similarity between a query vector and a document vector or between two query vectors. This model is widely used in IR community. Given two m-dimensional semantic vectors, $a = (a_1, a_2, ..., a_m)^T$, and $b = (b_1, b_2, ..., b_m)^T$, their semantic similarity, $Sim(a,b)$, is the cosine of the angle between them

$$Sim(a,b) = \frac{\sum_{i=1}^{m} a_i b_i}{\sqrt{\sum_{i=1}^{m} a_i^2}\sqrt{\sum_{i=1}^{m} b_i^2}}.$$

Because we normalize a query vector and a document vector before the similarity computation, the denominator in the formula is 1. The larger $Sim(a,b)$, the semantically closer the two vectors a and b. We use a threshold to determine whether two semantic vectors are similar.

D. Path update

Path entries are updated when queried files are found. The new path information is carried in the QueryResponse message and transferred along the reverse query path. All nodes on the reverse query path update their related entries accordingly. Specifically, When a node x receives a QueryResponse message for query source s and query vector q from node y, x performs as shown in Figure 1. If the discovered path via y is the first path for q, x adds this new path. If the cost of the new path via y is lower than that via the currently known best neighbor z, then replace z by y. Otherwise, keep z and reset the update to Q_{xq}, ΔQ_{xq}, to zero. If the update to Q_{xq} is trivial, then increase the counter $cntU_q$. Otherwise, reset $cntU_q$. x always propagates the updated Q_{xq} using a new QueryResponse message.

E. Path exploration

Path exploration is controlled by the system parameter p_q, which determines what neighbor to send query q. The

```
Path update at node $x$ when getting $qr_{qs}$ from $y$:
    // Basic ISRL; $qr_{qs}$: QueryResponse message for $qs$
 1. $if$ (the first path for $q$)
 2.     $Q_{xq} = Q_{yq} + 1$;
 3.     $z = y$;
 4. $else\ if$ $(Q_{yq} + 1 < Q_{xq})$
 5.     $\Delta Q_{xq} = Q_{xq} - (Q_{yq} + 1)$;
 6.     $Q_{xq} = Q_{yq} + 1$;
 7.     $z = y$;
 8. $else$
 9.     $\Delta Q_{xq} = 0$.
10. $if$ $(\Delta Q_{xq} < \epsilon_1)$
11.     $++cntU_q$;
12. $else$
13.     $cntU_q = 0$;
14. Adjust $p_q$ based on $cntU_q$;
15. Include $(q, Q_{xq})$ in the new QueryResponse message;
```

Fig. 1. Path update algorithm at node x when receiving QueryResponse message qr_{qs} from node y in basic ISRL.

neighbors attempted by a node x serve as the training samples for x to learn the best path for q. x faces a tradeoff between exploration and exploitation. Large p_q favors gathering new information by exploring unknown paths. Small p_q prefers utilizing already discovered good paths so as to reduce the total path cost. Typically, exploration is favored initially and exploitation is preferred later. In this paper, we consider two design options for setting p_q.

p_q **design 1: coarse adaptation.** In this option, we use two constants: one large value α_{high} for the initial progressive exploration; one small value α_{low} ($< \alpha_{high}$) for later lazy exploration. Each node, say x, initializes p_q to α_{high} to gather new information aggressively. x changes p_q to α_{low} when the best path discovered so far is close to the actual best path. This can be estimated by a large ($\geq$ a threshold w_1) number of consecutive minor updates to path cost Q_{xq}. $cntU_q$ is used to count such updates. The detailed function is shown below.

$$p_q = \begin{cases} \alpha_{high} & \text{if } cntU_q < w_1 \\ \alpha_{low} & \text{otherwise.} \end{cases}$$

p_q **design 2: fine tuning.** Adjust p gradually. Each node, say x, initializes p_q to a large value. Each time when certain number of consecutive minor updates is observed at x, reduce p_q by a constant amount μ. When forwarding a query for q next time, x explores less. p_q decreases as the best-so-far path approaches the actual best path.

$$p_q = p_q - \mu,\ if\ cntU_q > w_2.$$

F. Illustration

Figure 2 shows an example of the basic ISRL search. The query source is A, denoted by an empty square. The desired file is only hosted in node E, represented by a solid circle. The arrows indicate query forwarding directions. The number next to each arrowed line refers to the experiment sequence. A - F are node IDs. The number within curly brackets next to each node ID is the Q value for this query. The first trial via B is successful, which causes nodes D, C, B, A to add new entries for this query with Q values as shown in the figure. The second and third trials fail. No update to Q values occur

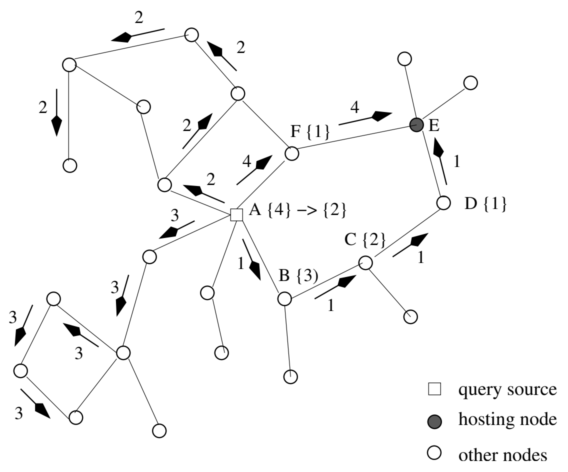


Fig. 2. An example of the basic ISRL search.

on the reverse query path. The fourth experiment succeeds and makes node F add a new entry with cost 1 for the query. A replaces the entry to B by the new entry to F with lower cost 2. Therefore A learns the best path to hosting node E through trials.

The basic ISRL corresponds to Q-learning with the following settings: $r = 1$, $\gamma = 1$, and $\eta = 1$. Thus we have

$$Q(s, a) = 1 + min_b Q(s', b)).$$

Because $Q(s, a)$ represents the path cost, it is to be minimized. r is set to 1 considering that the path cost is in terms of overlay hops. γ is set to 1 because $Q(s, a)$ represents path costs and discounts are unnecessary. η is set to 1 because we can adjust learning through the exploration probability.

IV. MP-ISRL (Multi-Path ISRL)

The basic ISRL keeps only one path for each successful query vector. This may not suffice when we try to find more than one desired files. In this extended version, MP-ISRL, each node keeps multiple (k) paths for each query vector. Both exploration and exploitation are implemented as k-thread forwarding. An exploring message for vector q is always sent to k neighbors chosen randomly. An exploiting one always utilizes k discovered paths for q. If a node does not have k known paths, it replaces unknown paths by random forwarding. The exploration probability for q is similar to basic ISRL. A newly discovered path is always added before k paths have been found, and replaces the worst path thereafter if it is better. Path evaluation can be based on cost alone or discounted reward considering both cost and similarity scores of desired files.

A. Path entries

Each node keeps the top k best-so-far paths for each successful query vector. Table II lists items in the path entry for query vector q kept at node x. (z_j, Q_{xqj}) represents the j^{th} path. z_j is the next-hop neighbor on this path and Q_{xqj} the path evaluation score. p_q has the same meaning as that in basic ISRL. To determine that the top k best paths have been found, $cntU_q$ counts the consecutive number of minor updates to any path for q.

Notation	Meaning
q	The query vector
z_j	The neighbor on the j^{th} path for q, $j \in [1, k]$
Q_{xqj}	The evaluation score of the j^{th} path for q, $j \in [1, k]$
p_q	The exploration probability for q
$cntU_q$	The consecutive number of minor updates to any Q_{xqj}, $j \in [1, k]$

TABLE II
THE PATH ENTRY FOR VECTOR q AT NODE x IN MP-ISRL

```
Path update at node $x$ when getting $qr_{qs}$ from $y$:
    // MP-ISRL; path cost as evaluation score.
    // $qr_{qs}$: QueryResponse message for $qs$
    // $qs$: Query message for vector $q$ and source $s$
    // $k_c$: the number of currently known paths for $q$
 1. $if$ ($y == z_i$ in the $i^{th}$ path for $q$) //existing next-hop
 2.     $if$ ($Q_{yq} + 1 < Q_{xqi}$) // a better one
 3.         $Q_{xqi} = Q_{yq} + 1$;
 4.         $\Delta Q_{xq} = Q_{xqi} - (Q_{yq} + 1)$;
 5.     $else$
 6.         $\Delta Q_{xq} = 0$;
 7. $else\ if$ ($k_c < k$) //a new next-hop; not found $k$ paths yet
 8.     $Q_{xqj} = Q_{yq} + 1$;
 9.     $\Delta Q_{xq} = Q_{xqj}$;
10.     $z_j = y$; $j = j + 1$;
11. $else$ //found $k$ paths already
12.     Find the worst existing path $(z_w, Q_{xqw})$;
13.     $if$ ($Q_{yq} + 1 < Q_{xqw}$)
14.         $\Delta Q_{xq} = Q_{xqw} - (Q_{yq} + 1)$;
15.         $Q_{xqw} = Q_{yq} + 1$;
16.         $z_w = y$;
17.     $else$
18.         $\Delta Q_{xq} = 0$.
19. $if$ ($\Delta Q_{xq} < \epsilon_2$)
20.     $++cntU_q$;
21. $else$
22.     $cntU_q = 0$;
23. Adjust $p_q$ based on $cntU_q$;
```

Fig. 3. Path update at node x when receiving from node y QueryResponse message qr_{qs} in MP-ISRL; path cost as evaluation score.

B. k-thread semantic search

When a desired file is found, a node x in MP-ISRL acts similarly to nodes in basic ISRL. MP-ISRL differs from basic ISRL in query forwarding. When node x receives a semantic Query message from node y that contains the query vector q initiated at query source s, x acts as follows.

1) If x has one or more paths for q, then with probability p_q, q is forwarded to k randomly chosen neighbors other than those in the known paths (exploration). With probability $(1 - p_q)$, x dispatches the query along k already discovered paths for q.
2) If no existing path for q is kept at x, then send the query to k neighbors chosen uniformly at random.

C. Path exploration and update

Like basic ISRL, MP-ISRL sets the exploration probability p_q for query vector q using a similar formula according to the quality of already discovered paths. Unlike basic ISRL, MP-ISRL gathers new information more aggressively by using k-thread explorations because MP-ISRL aims to locate the top k best paths. When node x decides to explore new paths for any query vector q, it dispatches q to k randomly chosen neighbors that do not appear in known paths.

As for path updates in MP-ISRL, like the basic version, updates occur only when a query result is found. Path information is propagated along the reverse query path. We evaluate paths in two ways. The first one is by path cost like the basic version. The second one is a new model, called discounted reward, which is designed for the scenario where we care about the similarity scores of desired files with respect to the same query.

Figure 3 shows the path update in this scenario. When a node x receives a QueryResponse message qr_{qs} for query vector q and source s from node y, if y is the next-hop neighbor on the existing i^{th} path for q, x records the new cost Q_{xqi} if y brings a smaller cost. If y is a new next-hop neighbor and x hasn't discovered all k paths yet, add y as a new path for q. If there are k existing paths for q, replace the worst existing path if the new path via y is better. If the update to any existing path of q is trivial, the counter $cntU_q$ is increased. Otherwise, $cntU_q$ is reset.

In summary, both basic ISRL and MP-ISRL apply Q-learning to P2P searching. Each node learns the best next-hop neighbor to forward a given query in order to follow the best path with the lowest cumulative cost or the highest cumulative reward. The learning process iterates through continuous exploration and exploitation. The learning rate is adjusted via the exploration probability. Basic ISRL keeps track of the best path while MP-ISRL maintains the top k best paths.

V. Evaluation

In this section, we will describe the experimental setup and compare ISRL to existing searches in unstructured P2Ps.

A. Simulation setup

We simulated the algorithms using random graphs that have 2000 and 5000 nodes and average degree of 5. The document collection in the P2P network consists of 6000 documents chosen from disk 1&2 in the TREC data set [17]. All documents that are semantically similar are distributed to randomly selected neighborhoods in the P2P overlay. 100 document vectors are selected as semantic query vectors. Queries arrive sequentially. Each query consists of a query source node and a query vector. 100 nodes are randomly chosen as query sources. The query distribution is uniform. The semantic similarity threshold is selected as 0.43 based on the chosen document collection. The performance metrics are the average query messages per query for measuring query traffic, the average query success rate, and the average number of desired documents found as an indication of query quality. A query is considered successful if at least one desired document is found.

B. Evaluation of the basic ISRL

The basic ISRL is evaluated against 1-walker random walks where each node randomly forwards a query to one neighbor. Figure 4 illustrates their performance differences with varying TTLs in random networks with 2000/5000 nodes and the average degree 5. The basic ISRL uses the path update policy: fine tuning with parameters, p_q being 0.05 and μ being 0.01. The fine exploration is better than coarse exploration in the experiments. The results are not included in the paper due to space limitation.

It is observed from Figure 4(a) that in both schemes the message consumption increases as TTL increases. This is because the query can travel further and the query path is longer at larger TTL values. However, random walks incur much more traffic than the basic ISRL at higher TTLs because the random walk does not utilize any hints and many messages are wasted. The basic ISRL reduces traffic by exploiting discovered good paths and exploring better paths.

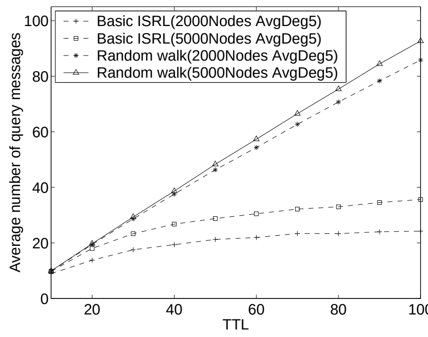
Figure 4(b) shows that the success rate in both schemes also increases with increasing TTLs because the desired files far from the querying source can be found at larger TTLs. In a given network, the basic ISRL improves the query success rate by 300% to 400% over random walks because random walks does not keep already discovered paths. The success rate for the basic ISRL increases dramatically at smaller TTLs (< 40) and the increase slows down at larger TTLs. This is because when TTL is small, an increase in TTL causes exploration to be more successful and more queries can be satisfied by exploiting these already discovered paths.

It is also observed from Figure 4 that both the basic ISRL and the 1-walker random walk achieve a higher query success rate with less number of messages in 2000-node networks than in 5000-node networks. The basic ISRL outperforms the 1-walker random walk in both types of networks.

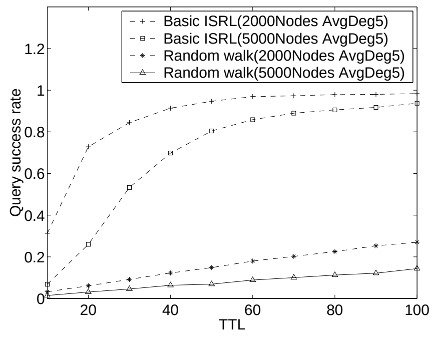
C. Evaluation of MP-ISRL

The MP-ISRL is compared to random walks where each node randomly forwards a received query message to k neighbors. It employs fine exploration with parameters, p_q being 0.05 and μ being 0.01. Figure 5 demonstrates the performance with varying k values in networks with the average degree 5 and variable number of nodes. The TTL value is chosen as 10 because it is large enough to show the performance differences between MP-ISRL and random walks. It is also chosen for faster simulation speed.

MP-ISRL achieves a higher success rate with much fewer messages starting from $k = 3$ than random walks because it can learn and utilize discovered paths. At large k values, the random walk can reach a significant number of nodes. Therefore its success rate is also high. As for the number of discovered documents, MP-ISRL can find more desired documents than random walks at mid-size k values due to its higher success rate. Both MP-ISRL and random walks deliver more queries successfully with less message load and find more desired documents in smaller networks (2000-node) than larger networks (5000-node). MP-ISRL is superior to random walks in both types of networks.

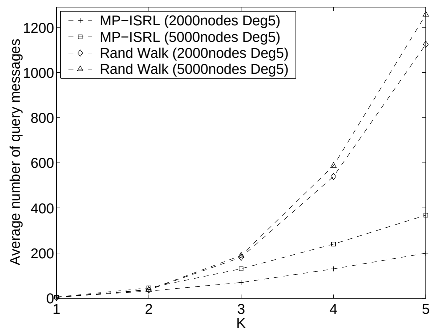


(a) Average number of query messages.

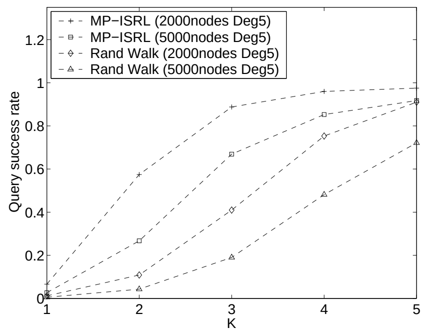


(b) Average query success rate.

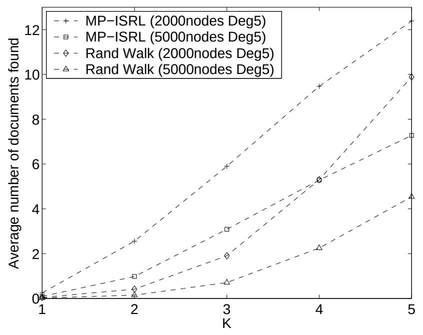
Fig. 4. Basic ISRL vs 1-thread random walks in 2000-node and 5000-node networks with the average degree 5.



(a) Average number of query messages.



(b) Average query success rate.



(c) Average number of documents found.

Fig. 5. MP-ISRL vs. Random walk in 2000-node and 5000-node networks with the average degree 5.

VI. Conclusion

In this paper, we have proposed to improve searching in unstructured P2Ps through reinforcement learning. Our approach, ISRL, systematically learns the best path to desired files by exploring new paths and exploiting existing explored paths. We have presented two design models, basic ISRL and MP-ISRL. The basic ISRL can be extended in many other ways. One approach that can reduce the memory storage overhead is to cluster query vectors based on semantic similarity and keep one vector for each query cluster. In the future, we will investigate this and other extensions.

References

[1] X. Li and J. Wu, "Searching techniques in peer-to-peer networks," in *Handbook of Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, Edited by J. Wu.* CRC Press, 2005.

[2] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," in *Proc. of the 22nd International Conference on Distributed Computing (IEEE ICDCS'02)*, 2002.

[3] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proc. of the 16th ACM International Conference on Supercomputing (ICS'02)*, 2002.

[4] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-yazti, "A local search mechanism for peer-to-peer networks," in *Proc. of the 11th ACM Conference on Information and Knowledge Management (CIKM'02)*, 2002.

[5] T. M. Mitchell, *Machine learning*. WCB / McGraw-Hill, 1997.

[6] L. Gatani, G. L. Re, A. Urso, and S. Gaglio, "Reinforcement learning for P2P searching," in *Proc. of the International Workshop on Computer Architecture for Machine Perception (CAMP'05)*, 2005.

[7] "Gnutella," http://www.gnutella.com.

[8] C. Wang, L. Xiao, Y. Liu, and P. Zheng, "Ditributed caching and adaptive search in multilayer P2P networks," in *Proc. of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, 2004.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Proc. of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[10] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search in peer-to-peer networks," in *Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.

[11] X. Li and J. Wu, "A hybrid searching scheme in unstructured P2P networks," in *Proc. of 2005 International Conference on Parallel Processing (ICPP'05)*, 2005.

[12] Y. Zhu and Y. Hu, "Ess: Efficient semantic search on gnutella-like P2P systems," in *Technical Report, Department of ECECS, University of Cincinnati*, 2004.

[13] C. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, 1992.

[14] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in Neural Information Processing Systems*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan Kaufmann Publishers, Inc., 1994.

[15] M. Berry and M. Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*. Society for Industrial and Applied Mathematics (SIAM), 1999.

[16] H. Schtze and C. Silverstein, "Projections for efficient document clustering," in *Proc. of ACM SIGIR'97*, 1997.

[17] "Trec dataset," http://www.trec.org.