# An Improved Adaptive Probabilistic Search In Unstructured Peer To Peer Network

Rajani S.N
Dept. Computer Science and Engineering
AMC Engineering College
Bangalore, India
Rajaninagraj89@gmail.com

Bharathi R
Dept. Computer Science and Engineering
AMC Engineering College
Bangalore, India
Bharathi_saravanan@hotmail.com

*Abstract*-- **Peer-to-peer (P2P) networks are gaining increased attention from both the scientific community and the larger Internet user community. Data retrieval algorithms lie at the center of P2P networks, and this paper addresses the problem of efficiently searching for files in unstructured P2P systems. We propose an Improved Adaptive Probabilistic Search (IAPS) algorithm that is fully distributed and bandwidth efficient .IAPS uses ant-colony optimization and takes file types into consideration in order to search for file container nodes with a high probability of success. We have performed extensive simulations to study the performance of IAPS, and we compare it with the Random Walk and Adaptive Probabilistic Search algorithms. Our experimental results show that IAPS achieves high success rates, high response rates, and significant message reduction**

***Key words:*** ***Unstructured peer-to-peer, Adaptive probabilistic search , Ant colony , Success rate ,Information retrieval.***

## I. Introduction

A peer-to-peer (P2P) network is a distributed system in which computing nodes employ distributed resources to perform a critical function in a decentralized manner. Nodes in a P2P network normally play equal roles and are therefore called peers. Since P2P systems are capable of searching and communicating across the globe, they have become phenomenally popular in recent years [1]. Examples of P2P applications are distributed file-sharing systems, event notification services, and chat services [2–5].

P2P networks can generally be classified, based on the control over where data are located and the network topology, as structured or unstructured [6]. In an unstructured P2P network such as Gnutella peers are typically connected to a random set of neighbors. There are no rules that strictly define where data is stored and which nodes are neighbors

In contrast, structured P2P networks have well defined neighbor links and can be further classified as loosely structured or highly structured. In highly structured P2P networks such as the network in Chord [7], the neighbors of nodes, the network architecture, and the locations for storing data are precisely specified. In loosely structured P2P networks such as Freenet [8] and Symphony [9], the overlay structure and data locations are not precisely determined. In Freenet [8], both the overlay topology and the data location are determined based on hints, and the network topology eventually evolves into some intended structure. In

Symphony, the overlay topology is determined probabilistically but data locations are precisely defined.

The principal requirement in any peer-to-peer network is efficient searching for desired resources such as data or files. Heuristic and meta-heuristic methods have proven to be efficient for solving many hard network search problems. They are also demonstrating their usefulness in the P2P network domain, especially for unstructured P2P networks. Ant Colony Optimization (ACO) [10] is inspired by observations of the foraging behavior of real ant colonies. While moving from food sources to the nest and vice versa, ants deposit a substance called pheromone on the ground, forming a pheromone trail. When choosing their way, ants can smell pheromone and tend to choose paths marked by strongly probable pheromone concentrations. While an isolated ant moves practically at random, an ant encountering a previously laid trail can detect it and decide to follow it with high probability, thus reinforcing the trail with its own pheromone. The collective behavior that emerges is a form of autocatalytic process where the more the ants follow a trail, the more attractive that trail becomes for following. The process is thus characterized by a positive feedback loop, where the probability with which an ant chooses a path increases with the number of ants that have previously chosen the same path.This method is one of the approaches that scholars have applied to unstructured P2P networks in the last decade, including AntNet [11], Anthill [12], AntSearch [13], SemAnt [14],ACO-based Search [15], and APS [16]. In the next section, the last three approaches are discussed in more detail.

In this paper, here presents a novel Improved Adaptive Probabilistic Search (IAPS) algorithm for locating files in unstructured P2P networks. We focus on unstructured P2P networks because most popular P2P applications operate on unstructured networks. The highly transient node populations and ad hoc organization of unstructured P2P networks make it much more challenging to locate desired resources in these networks than in structured P2P networks. Conventional search methods for unstructured P2P networks use flooding and its variations or various types of indices that are quite expensive to maintain, may not scale well [17], and can incur heavy overhead. In contrast, IAPS considers the file`s type and a score based on previous searches for this file type to minimize the search space and, as a result, the search overhead

## II. Related Work

Many search algorithms for unstructured P2P networks have been proposed in recent years. The algorithms can be classified as deterministic or probabilistic. In a deterministic approach query forwarding is deterministic, while in a probabilistic approach query forwarding is probabilistic, random, or based on resource ranking. Searching schemes in unstructured P2P systems can also be categorized as regular-grained or coarse-grained. In a regular-grained approach, all nodes participate in query forwarding. In a coarse grained scheme, query forwarding is performed by only a subset of the nodes in the network. Existing search methods can also be classified into blind searches and informed searches. In blind searches, nodes do not store information about data locations, while in informed searches nodes store some metadata that facilitates the search.

Flooding-based searching is the most commonly used method to locate specific data items .Flooding involves a breadth-first search (BFS) of the overlay network graph, with the querying node contacting nodes that are reachable within a specified time-to-live (TTL) number of hops.

Table 1. Query table status indexed for each node

| *File type* | *File Format* | *Number Of Copies* | *Score Of files* |
|---|---|---|---|
| Video | *.3gp | 500 | 80 |
| Audio | *.mp3 | 300 | 200 |
| Document | *.doc | 200 | 1300 |

TABLE 2. FILE-TYPE PROBABILITY FOR EACH NODE

| *File type* | *File Format* | *File Scores* | *Probability* |
|---|---|---|---|
| Video | *.3gp | 80 | 80/(80+200+1200) |
| Audio | *.mp3 | 200 | 80/(80+200+1200) |

## III. System model

In the Improved Adaptive Probabilistic Search (IAPS) algorithm, each peer maintains information about files and shares this information with its single-hop neighbors. Table 1 (the Query Table) shows the information that a node maintains about each file. Recall that, in the search algorithm for a P2P network, the destination node is unknown and the contents of search packets are variable, so the content of a query, which is expressed by a keyword and syntax, should be considered in the design of the routing or query table. The Query Table has four columns representing the file type, the file format, the number of copies of the file, and the scores of the file. Initially, all tables' entries are commenced with the same amount of file-types files. entries in its index table to its neighbors in a uniform and balanced style. This ensures that the attenuated bloom filters(data structures that reside at each node in the system) in its index table are still available while the node is offline. When a node rejoins the network, it asks its neighbors to distribute some entries in their index tables to fill in its own index table. A query (request) Q consists of one or more keywords k1,…,kn, which are connected by the Boolean operator OR or AND. At startup, all table entries for each node (peer) are initialized with the same small value for each file type`s scores.

IAPS maintains a score for each file based on the results of previous searches for the file; the score is an indication of how many times the file has been referenced. score in Table 1 for a file with a.mp3 extension indicates that there have been 200 references to that file. The file format score maintained in each node is increased or decreased based on the search request and the placement of the node along the path traveled by the file request. The type of request depends on the awareness of the data structure by the user and on the user expertise. We use requests . If a node has the requested file, the node increases its score for the file by two points (we have selected this amount to show how the pheromone of the forward ant can calculate the nodes that will cause the query to reach the destination node), and all other nodes along the route to the destination node, excepting the node that originated the request, increase their file score by one point. If there is more than one path from the destination node to the requesting node, the path with fewer hops is selected, and the nodes in the other paths are rewarded the same points as the nodes of the selected path. However, if the requested file was not found or the route comes back to the requesting node, one point is deducted from the score of each of the nodes that assessed the request.

Figure 2 illustrates how the score for the file is computed .In this figure, we consider two walkers, and node G is the originating node for a request for a.mp3 file that is hosted by node A. The figure also displays next to each node its index table with its current state (state a).Node G sends a request message to all of its neighbors (broadcasting request message (state b). We search the index tables of the unvisited neighbors of each node simultaneously. Then nodes B, D, and H search their databases concurrently. If a node finds the requested file, it sends a response to node G. Otherwise, the node sends the request to its unvisited neighbors. For example, node D is not visited because of the path D→G (state c). In state (d), node B sends a request message to all of its neighbors that have not already received the request. Node A searches its database and finds the file. Now, the score phase to update the score for all nodes engaged in the search begins (state e). The nodes along the path A→B→G, except the requesting node, are rewarded with additional points, while all other nodes that engaged in the search process but were not successful in locating the file in their database have one point deducted. Thus, node A is rewarded twice compared to the other nodes that receive additional points. Finally, node A sends the requested file back to node G along the selected path (state f).

In this example, we used two walkers. If, instead, we consider three walkers, the nodes that are rewarded will change. This is because node C will send a request to node A and receive its response. The reward for node A will then be four points, while the rewards for each of nodes B, C, and D will be one point. Also, we will have two alternative paths (A→B→G and A→C→D→G) for sending the response back to node G. We do not consider shared file types in this paper, because there is no supplementary data about shared file types among nodes. In the following section, we discuss IAPS in greater detail.

## IV. IMPROVED ADAPTIVE PROBABILISTIC SEARCH (IAPS)

Our proposed Improved Adaptive Probabilistic Search (IAPS) method consists of four phases: (1) the search phase, (2) the neighbor selection phase, (3) the scoring phase, and (4) the flow control phase.

### *A.* Search phase

The search actions are as follows. There are two states in which a search can be conducted for a given file

- First Search: If this is the first request for the file, our method is different from the APS method, because it emphasizes the specific file type when randomly selecting the nodes to which the query will be sent.
- Next Search: If this is not the first request for the file, its path has been indexed in the index table.

If a node receives a request message for a file, it searches in its local database for the requested file. If the file exists, the node sends a response message by generating a forward ant at starting time Tstart to the querying node, along the reverse of the path that the file request traveled by generating the backward ant. The node considers the TTL parameter Tfinal and also the number of walkers, as, in searching the network, starting by searching its available neighbors. Tfinal and the number of walkers serve to prevent ants from infinitely running forward in the network. Our method uses ICMP packets to avoid network congestion and prevent the possibility that a requested service is not available

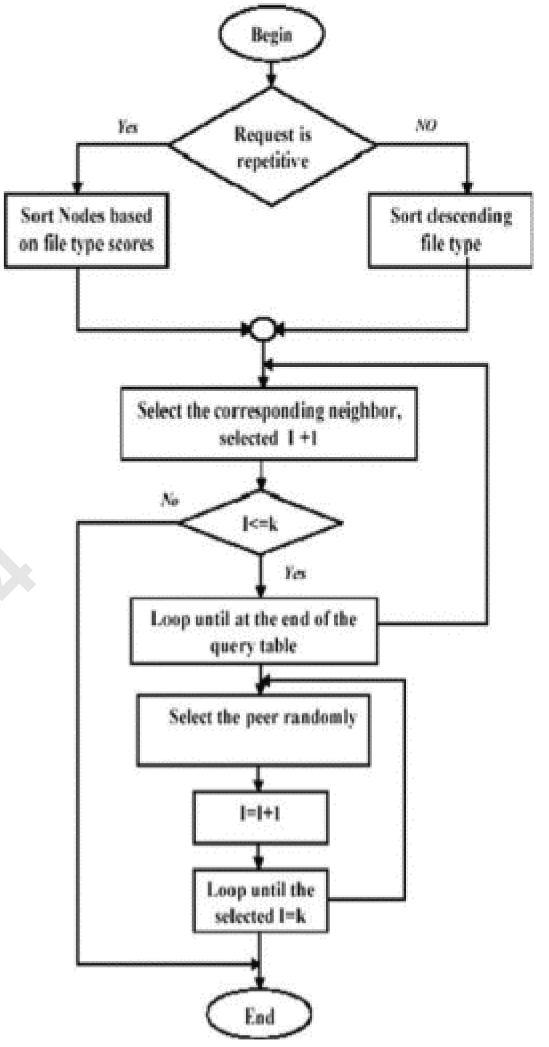
Begin
Request is repetitive
Yes
NO
Sort Nodes based on file type scores
Sort descending file type
Select the corresponding neighbor, selected I +1
I<=k
No
Yes
Loop until at the end of the query table
Select the peer randomly
I=I+1
Loop until the selected I=k
End

Fig. 1 Flowchart for the IAPS neighbor selection phase

or that a host or neighbor peers cannot be reached, so that only available peers are visited in our query search. In the sequel, if the file does not exist, the node forwards the request to its neighbors by its forward ant. We use a common policy for preventing the forward ant from engaging in a cycle or loop when searching: If a forward ant detects a cycle occurring (i.e., it is about to search the nest peers), the forward ant is forced to return to an already visited node, the cycle's nodes are popped from the ant's stack, all memory about them is destroyed, and the forward ant continues searching. But if the cycle lasts longer than the Tfinal value of the forward ant, it is destroyed as in [10].

### B. Neighbour Selection Phase

Neighbor selection for request forwarding is based on information in the nodes' index tables (illustrated in Section 3), such as the number of successful requests (file scores), file types, and the file-type probability table τ . The file-type probability table τ stores the probability rate for each peer, a value between 0 and 1, based on the file scores that it has. Table 2 shows the probability information that a node maintains about each file type. The querying node selects the neighbors that have the highest scores for the requested file type. Figure 2 describes the IAPS selection phase in a flowchart. For example, if we want to find "mike.mp3," we identify the nodes that have high scores for files with .mp3 extensions. When a request is issued, we sort the nodes in descending order of the scores, using the Counting Sort algorithm . We then select k nodes with the highest scores from the sorted neighbor list and send the request message to these nodes. This process is repeated for the unvisited and available neighbors of those nodes, and so on, until the requested file is found or all the walkers cross paths and the corresponding file is not found.

### C. Scoring Phase

At the outset, each node is given a score of one point for each file of a given type that it contains in is database and zero points for file types that are absent. Figure 3 illustrates the initial scores for nodes before searching commences. Nodes F, G, and H have scores of zero for the file types that are not contained in their database.

When a node in P2P network sends a file request to its neighbors, the neighbors search their databases for the requested file. If a neighbor node finds the file in its database, it sends a reply message to the requesting node by generating a backward ant and increases its score for the file type by three points. Otherwise, the node sends a request for the file to k neighbors in Tfinal, where k is the number of walkers. If one of the neighbors finds the file in its database (index tables), the neighbor node will be rewarded with two points for the file type and updates Tables 1 and 2. In addition, the scores of the nodes along the path from the requesting node and the responding node are increased by one point excluding the requesting node (i.e., the forward ant has a stack into which it pushes visited peers; Reward). If the neighbor node does not have the requested file in its database, it will similarly send a request message to its unvisited neighbors by its forward ant. In the worst-case situation, if none of the nodes in the network have the requested file or the TTL (Tfinal) of all of the walkers reaches zero, the scores of all visited notes are decreased by one point (i.e., the backward ant performs a pop and punishes that visited peer; Punishment). On the other hand, if the search is successful, then the scores of all visited nodes that are not on the successful search path are also decreased by one point.

The scoring process is depicted below, in Figs. 2 and 3, for two walkers. In both figures, Node G requests "a.mp3" file that is has failed to find in its database. In Fig. 2, Node G sorts its neighbors and selects two of these: nodes B and D. It sends request message to these nodes. Based on Fig. 3, in the next walk both node B and node D select node C (because of its high score) and send C a file request. Node C sends the request to nodes F and A (i.e. Node C does not send the request message to Node D, because, it saves the previous request message already came from Node D in its database, so, no need to send request to Node D), and node A has the file. Figure 3 shows the steps involved in rewarding and deducting score points. All the nodes participating in the path connecting A to G are rewarded according to the policies we have presented (Fig. 3). In contrast, all visited nodes (light red color nodes) except the nodes that were rewarded are punished under the scoring policy discussed above. Thus, a reward fortifies the location of the file extension in the network and enhances the probability of selecting that node for the next search . We use an ant colony to simulate the scoring phase in our programming, with a pheromone parameter that rewards a node based on the number of times per minute a file type is found on that node. Pheromone parameter is updated according to the information gathered by the forward ant by altering the routing table of each visited node similar. If a file type is requested more than 10 times per minute and is repeatedly found on one node, then the node is rewarded with three points the 11th time the file type is found there, and every node along the path from that node to the requesting node is rewarded two points (except of the requesting node-here node . For example (Figs. 2 and 3), if a file is searched for more than 10 times in a minute with the same results, node A is rewarded three points so that its new score will be 11+3=14 points, and other nodes along the successful route, such as node B in this example, are rewarded two points, so that node B's newscore will be 7+2=9 (i.e. this figure is not shown here). Figure 2 to 3 show ants share the negative data. It gives negative points to the nodes that were participated as intermediate nodes to find the corresponding file until reach to the requested node.

Each peer applies the evaporation rule shown in 2 in a predefined interval Te for each link to a neighbor peer, where the amount of evaporating pheromone is controlled by the parameter ρ 2 [0,1]

### D. Flow Control Phase

The flow control phase is used to control and limit the number of messages received at each time step based on the nodes' abilities and capabilities. We use a least-recently-used (LRU) scheme to control the flow; this involves throwing away some unimportant messages by ICMP controlling packets and thus increasing the network's capacity to serve important new messages in the near future.

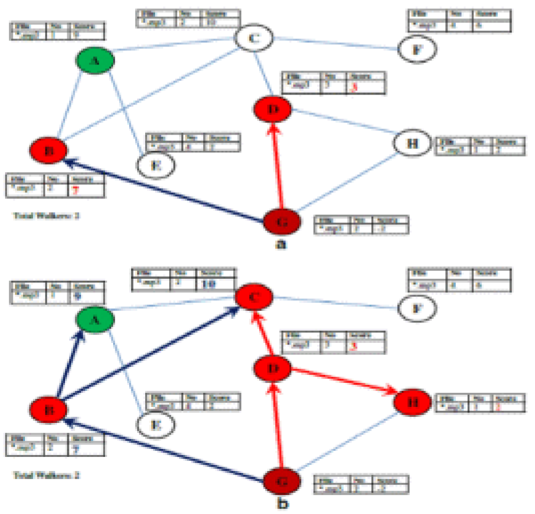
Fig.2 Selecting Routes

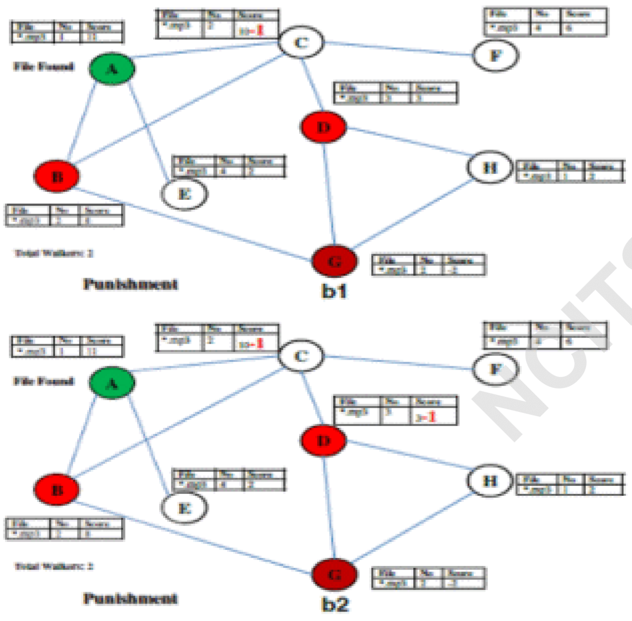
Fig.3 Rewards and Punishments

## REFERENCES

[1] Iskandar I, Naomie S (2009) Selective flooding based on relevant nearest-neighbor using query feedback and similarity across unstructured peer-to-peer networks. Journal of Computer Science 3(5):184–190, ISSN 1549–3636

[2] Balakrishnan H, Kaashoek MF, Karger D, Morris R, Stoica I (2003) Looking up data in P2P systems. Communications of ACM 46(2):43–48

[3] Barkai D (2002) Technologies for sharing and collaborating on the net. Proceeding of the 1st International Workshop on Peer-to-Peer Computing (IPTPS'02), ISBN: 0-7695-1503-7, 13–28. doi: 10.1109/P2P.2001.990419

[4] Daswani N, Garcia-molina H, Yang B (2003) Open problems in datasharing peer-to-peer systems. Proc . of the 9th International Conference on Database Theory (ICDT'03) 1–15

[5] Milojicic DS, Kalogeraki V, Lukose R, Nagaraja K, Pruyne J, RichardB, Rollins M, Xu Z (2002) Peer-to-peer computing, HP Lab technical report, HPL-2002-57 www.hpl.hp.com/techreports/2002/HPL-2002-57R1.pdf

[6] Lv Q, Cao P, Cohen E, Li K, Shenker S (2002) Search and replication in unstructured peer to-peer network's. Proceeding of the 16th ACM International Conference on Supercomputing (ACM ICS'02) 258–259. doi: 10.1145/514191.514206

[7] Stoica I, Morris R, Karger D, Frans M, Kaashoek, Balakrishnan H (2001) Chord: A scalable peer-to-peer lookup service for internet applications. Proceeding of the 2001 ACM Annual Conference of the Special Interest Group on Data Communication (ACM SIGCOMM'01) 149–160.http://pdos.csail.mit.edu/6.824/papers/stoica-chord.pdf

[8] Clarke I, Sandberg O, Theodore BW, Hong W (2001) Free net: A distributed anonymous information storage and retrieval system. Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability 46–66. www.cs.cornell.edu/people/egs/615/freenet.pdf

[9] Manku GS, Bawa M, Raghavan P (2003) Verity Inc, Symphony: Distributed hashing in a small world. Proceeding of 4th USENIX Symposium on Internet Technology and Systems (USITS'03)127–140. www.infolab.stanford.edu/~bawa/Pub/symphony.p

[10] Dorigo M, Gambardella LM (1997) Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1):53–66

[11] Caro GD, Dorigo M (1998) AntNet: Distributed stigmergy control for communications networks. Journal of Artificial Intelligence Research 9:317–365

[12] Babaoglu O, Meling H, Montresor A (2002) Anthill: A framework for the development of agent-based peer-to-peer systems. In Proceedings of the 22nd International Conference on Distributed Computing Systems

[13] Wu C, Yang K, Ho J (2006) AntSearch: An ant search algorithm in unstructured peer-to-peer networks. In Proceedings of the 11th IEEE Symposium on Computers and Communications

[14] Michlmayr E (2006) Ant algorithms for search in unstructured peer to peer networks, Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW '06). IEEE Computer Society, Washington, pp 142–146

[15] Tang D, Lu X, Yang L (2011) ACO-based search algorithm in unstructured P2P Network. In Proceedings of the 2011 International Conference of Information Technology, Computer Engineering and Management Sciences, 1 (ICM '11), IEEE Computer Society, Washington, 143–146

[16] Tsoumakos D, Roussopoulos N (2003) Adaptive probabilistic search in peer-to-peer networks. Technical Report, CS-TR-4451

[17] Tsoumakos D, Roussopoulos N (2003) Adaptive probabilistic search for peer-to-peer networks. Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P 2003) 102–109Peer-to-Peer Network Appl.