Optimal Resource Discovery Paths of Gnutella2

Mikko Vapa, Annemari Auvinen, Yevgeniy Ivanchenko, Niko Kotilainen and Jarkko Vuori
Department of Mathematical Information Technology
P.O.Box 35 (Agora), 40014 University of Jyväskylä, Finland
firstname.lastname@jyu.fi

Abstract

This paper shows that the performance of peer-to-peer resource discovery algorithms is upper bounded by a k-Steiner minimum tree and proposes an algorithm locating near-optimal query paths for the peer-to-peer resource discovery problem. Global knowledge of the topology and the resources from the peer-to-peer network are required as an input to the algorithm. The algorithm provides an objective measure for defining how good local search algorithms are. The performance is evaluated in simulated peer-to-peer scenarios and in the measured Gnutella2 P2P network topology with four local search algorithms: breadth-first search, self-avoiding random walker, highest degree search and Dynamic Query Protocol.

Keywords - peer-to-peer; P2P; resource discovery; k-Steiner minimum tree; optimal paths; Gnutella2;

1. Introduction

Peer-to-Peer networks (P2P) are distributed systems, which consist of resource sharing processes. A typical use case for a P2P network is the file sharing, where users can share the files located in their computers to other users in the network. The shared files can be found by executing a query, which locates the instances of the queried file and returns the information for downloading them. Thus the processes connected to the P2P network act both as a client and a server consuming and offering resources.

Locating resources is an essential problem in peer-to-peer networks, because there is no centralized point or index from which the information about the resources could be found. Therefore developing efficient resource discovery algorithms is crucial.

In the *peer-to-peer resource discovery problem*[1], any node can possess resources and query resources from other nodes in the network. The problem consists of network with nodes, links and resources. Resources are identified by unique IDs and nodes may contain any number of resources. One node knows only the resources it is currently hosting. Any node in the network can start a query, which means that some of the links are traversed based on the local forwarding decisions in the network. Whenever the query reaches a node which has the resource, the node replies. The goal is to locate a predetermined amount of resource instances with a given ID using as few query packets as possible.

The problem can be solved using a distributed search algorithm, in which the querying node sends a query to its neighbors, who in turn forward the query further until the algorithm stops. Whenever a queried resource is located, a reply message is relayed back using the query path. Such an algorithm works optimally if the query is forwarded only to the neighbors, who either provide the queried resource, or can provide a minimal cost path to a set of nodes containing the queried resource.

With the global information about the topology and the resources the problem can be formulated as a Steiner minimum tree problem in graphs [19], giving an upper bound for the performance of resource discovery algorithms. In the Steiner tree problem, given a graph containing the vertices and the edges and a terminal set containing the vertices, the task is to compute a spanning tree containing all vertices in the terminal set. Steiner minimum tree is the tree with minimum length of all such spanning trees. The terminal set contains the node, which starts the query and the matching resource instances that can be located in the network.

The peer-to-peer resource discovery problem can be mapped to the Steiner minimum tree problem only if the number of needed resource instances is the same as the size of the terminal set minus one (because the query originator also needs to be in the set). However, it is often sufficient to locate for example half of the available resources, because the query originator may use, e.g. download, only some of the located resources. Also locating only one instance is not always a feasible solution, because there can be many different resources matching the query keyword, but only some of them represent the resource the query initiator is interested in.

[1] Note that peer-to-peer resource discovery problem differs from the resource discovery problem described in [4] because only one node needs to discover the other nodes containing resources. Peer-to-peer resource discovery problem has also other names such as the resource-location problem [12].

Usually locating only a portion of resource instances reduces the amount of query traffic significantly. This is beneficial especially in mobile and wireless peer-to-peer networks, where the use of battery power and therefore the amount of forwarded query packets should be minimized. Also, as was seen in the first version of Gnutella [18] the scalability of the peer-to-peer network weakens in wired networks when the resource discovery algorithm is not properly designed.

In this paper we show that the peer-to-peer resource discovery problem with global knowledge is identical to the Steiner tree problem when all resources need to be found and therefore can be used to find optimal paths for the peer-to-peer resource discovery problem. Also, to enable only a part of the resources to be discovered we modify the original Steiner minimum tree problem to *Rooted k-Steiner minimum tree* problem, where *k* represents the number of resources that needs to be located and present an approximation algorithm for solving the problem.

The approximation is needed because *k*-Steiner minimum tree problem is known to be *NP*-hard and thus no efficient polynomial algorithm exists for practically solving the Steiner minimum tree problem in large graphs. To demonstrate the use of the proposed algorithm we present an analysis of different peer-to-peer scenarios including the topology recently crawled from Gnutella2 network. As a comparison algorithms we use breadth-first search, self-avoiding random walk and highest degree search and the proposed minimum spanning tree *k*-Steiner algorithm (MST *k*-Steiner) as an approximation of optimal using global knowledge of network topology and resources. The results show that there is a significant gap between the performance of local search algorithms and the optimal solution.

2. Related Work

Peer-to-Peer resource discovery problem has been investigated extensively in the research literature [1,4,6,8,9,10,16,20,22,23,25].

Adamic et al. [1] propose High-Degree Seeking algorithm for finding one node in a graph by forwarding query to the highest degree neighbor, which has not yet been visited. They evaluate the performance of their algorithm in random graphs, power-law graphs and a snapshot of Gnutella P2P network. Compared to Random Walker, where query is forwarded to a randomly selected neighbor, the traffic reduction is in the order of magnitude.

Lv et al. [12] evaluate the use of multiple Random Walkers and Expanding Ring algorithm against Breadth-First Search (BFS) in random graphs, power-law graphs and a regular two-dimensional grid graph as well as in a snapshot of Gnutella. Traffic reductions of one or two orders of magnitude are gained with multiple Random Walkers compared to the BFS.

Crespo and Garcia-Molina [4] propose routing indices, which provide shortcuts for random walkers to locate the resources. As an evaluation graphs they use trees, trees with additional cycles and power-law graphs. Compared to random walkers routing indices reduce the traffic up to 50% and compared to BFS the traffic reduction is in the order of one or two magnitudes with uniform resource distributions.

Yang and Garcia-Molina [25] propose Directed BFS, which selects the first neighbor based on heuristics and further uses BFS for forwarding the query. They also propose the use of Local Indices for replicating resources to a certain radius of hops from a node. Evaluations are conducted on a snapshot of Gnutella and the performance of these algorithms are compared to the BFS. The Directed BFS reduces traffic to 38% while locating significantly less resources than the BFS. Local Indices, however, locates similar numbers of resources as the BFS with 39% traffic generated by the BFS.

Kalogeraki et al. [8] propose Modified Random Breadth-First Search as an improvement to the BFS algorithm. In their algorithm only a subset of neighbors are selected for forwarding. Also, they propose an Intelligent Search Mechanism, which stores the performance of past queries for each neighbor and thus can direct further queries to the neighbors, which are likely to have the queried resource. For evaluation they use randomly connected P2P network and reduce traffic to 35% compared to the BFS.

Menascé [19] follows the ideas of Kalogeraki et al. and propose a modification of BFS, where only a subset of neighbors are randomly selected for forwarding. Evaluations are done in a random graph without a comparison algorithm.

Tsoumakos and Roussopoulos [22] propose Adaptive Probabilistic Search, where the feedback from previous queries is used to tune probabilities for the further forwarding of random walkers. The algorithm is evaluated in random graphs and power-law graphs against Lv et al.'s multiple Random Walkers and Gnutella's UDP extension for scalable searches [5]. While keeping approximately the same level of traffic, APS doubles the success rate of queries compared to multiple Random Walkers.

Sarshar et al. [20] propose Percolation Search algorithm for power-law networks. The idea is to replicate copies of resources to sufficient number of nodes and thus ensure that the algorithm locates at least one replica of the resource. The algorithm's performance is evaluated in power-law graphs and a snapshot of Gnutella P2P network without a comparison algorithm.

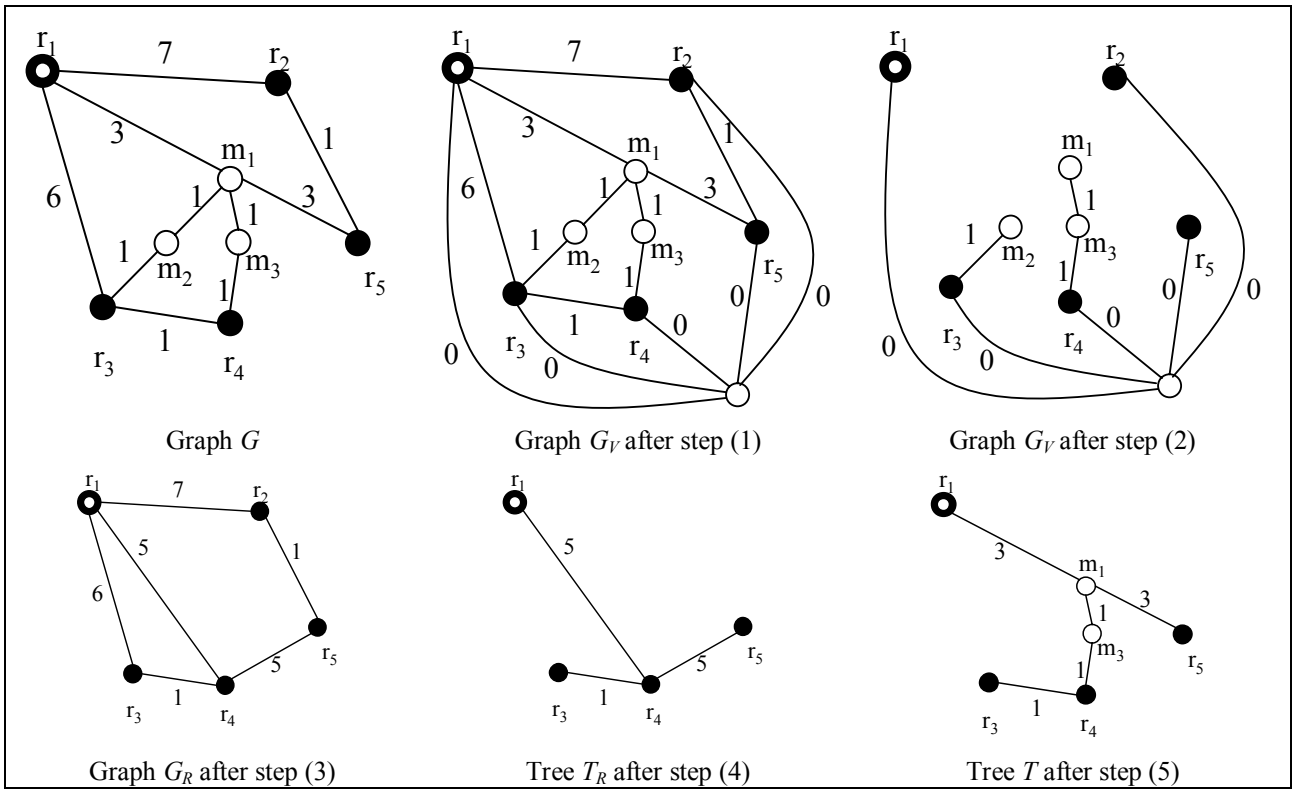

Figure 1. Execution of MST *k*-Steiner Algorithm with *k*=4

Fisk [6] proposes Dynamic Query Protocol (DQP), which has now been implemented in Gnutella2 peers. DQP executes first a probe query to estimate how rare the resource is and based on the obtained results calculates proper TTL and number of neighbors, which the query will be forwarded. The query is terminated when 150 resource instances has been located, there are no connections left to query or when the theoretical horizon of the query has hit the limit of 200,000 peers.

Vapa et al. [23] propose NeuroSearch, which is a neural network based resource discovery algorithm. In NeuroSearch a neural network is given a set of heuristics and by calculating the output of the neural network the algorithm can decide which of the neighbor nodes will receive the query. The evaluations are done in small power-law graphs and the traffic is reduced approximately to 80% from the BFS.

The main theme of all the papers reviewed in this section has been to introduce new algorithm(s) and to compare their performance to other algorithms of a similar type. However, the level of performance is not properly identified if the optimal performance is not measured in the simulations. The algorithm proposed later in this paper aims to overcome this problem.

3. Steiner Minimum Tree Problem

Let $G = (V,E)$ be an undirected graph, where V is a set of vertices and E is a set of edges having edge costs. Given a terminal set $R \subseteq V$, a Steiner minimum tree (SMT) is a tree $T \subseteq G$ such that T contains all vertices of R and the length $w(T)$ is minimum among all Steiner trees. $w(T)$ is defined as a sum of all edge costs $e \in E$ contained in T.

Compared to a minimum spanning tree, which contains all vertices of a graph, SMT spans only a subset of vertices and thus if the cardinality of the terminal set $|R| = |V|$ these problems are equivalent. Also, if $|R| = 2$, SMT reduces to solving a shortest-path problem.

In SMT the vertices are divided into two sets: terminal vertices and non-terminal vertices. Terminal vertices belong to a set, which has to be included in the solution, whereas non-terminal vertices can shorten the length of the solution.

SMT is known to be *NP*-complete problem [9]. Being in complexity class *NP* means that there exists a polynomial time algorithm to check whether the given solution is a correct Steiner tree and whether the length of a given solution is less than a given bound B, but there is no polynomial algorithm (unless *P*=*NP*) that would find such a Steiner tree. Therefore exact solving of the problem is not practical with large graphs. Also, when a problem is classified as *NP*-complete it means that the problem is the hardest among all problems contained in *NP*. More information about the *NP*-completeness of the Steiner tree problem can be found in [19].

Because SMT is *NP*-complete, approximation needs to be used. An approximated solution is not guaranteed

to locate the Steiner minimum tree, but it can give guarantees that the length of a located solution is within certain range from the optimal solution.

4. Peer-to-Peer Resource Discovery As Steiner Tree Problem

As was described earlier the peer-to-peer resource discovery problem does not map to the Steiner tree problem if only part of the resources needs to be found. Therefore we introduce *k*-Steiner Minimum Tree problem as described in [3] with an addition of a root vertex to the solution set. In Rooted *k*-Steiner Minimum Tree problem (Rooted *k*-SMT) it suffices to select only *k* terminal vertices from *R* to be included in the Steiner minimum tree starting from the root vertex *r*. Also we propose an approximation algorithm for solving the Rooted *k*-SMT problem.

4.1. Rooted *k*-Steiner Minimum Tree

Problem: `Rooted k-Steiner Minimum Tree`

Given: A connected graph $G = (V,E)$, a terminal set $R \subseteq V$, a root vertex $r \in R$ and $2 \le k \le |R|$

Find: A Steiner tree T for R in G rooted to vertex r and containing k terminal vertices, such that $w(T) = \min \{|w(T')| \mid T'$ is a Steiner tree for k vertices in $R\}$

The Rooted *k*-SMT becomes equivalent to the SMT by selecting $k=|R|$ and as a root any vertex in *R*. The SMT thus reduces to a special case of the Rooted *k*-SMT and therefore Rooted *k*-SMT for all *k* is at least as hard as SMT. When applied to the resource discovery problem the terminal set *R* is formed of query originator as root vertex and $|R|$-1 resource instances.

4.2. Approximation of Rooted *k*-Steiner Minimum Tree Problem With Minimum Spanning Tree

A well-known method for approximating the SMT is the use of a minimum spanning tree (MST) [19,24]. The MST *k*-Steiner Minimum Tree algorithm (MST *k*-Steiner) proposed here uses the same principles as MST-approximation algorithm to locate a solution for Rooted *k*-SMT.

MST *k*-Steiner starts by computing Voronoi regions of each terminal node. Voronoi region of a terminal node contains all the nodes which are closer to that terminal node than to other terminal node. Voronoi regions can be computed by adding one node in the graph *G* and connecting this node to all terminal nodes of *R* with edge length 0. Let G_V denote this graph. Then by executing a minimum spanning tree on G_V the Voronoi regions are obtained. This also gives the distance of each non-terminal node to its closest terminal node. The technique used here was introduced by Mehlhorn in [15].

Next, the Voronoi regions are used to compute the shortest distance graph G_R of vertices in *R*. Let $l(u,v)$ denote the edge cost of the edge between nodes *u* and *v*. Let $l(u)$ denote the distance of node *u* from the closest terminal node. Let $t(u)$ denote the closest terminal node of node *u*. Shortest distance graph G_R is obtained by going through each edge (u,v), $u,v \in E$, $u \ne v$ and computing the two triplets $(t(u), t(v), l(u)+l(u,v)+l(v))$ and $(t(v), t(u), l(u)+l(u,v)+l(v))$. These triplets are collected in a list and only those where $t(u) \ne t(v)$ and $l(u)+l(u,v)+l(v)$ is the shortest are kept in the list. This list is used to create the graph G_R by associating two terminal nodes *u* and *v* if they have a corresponding triplet in the list and setting the edge cost to be the third value of the triplet.

Then a *k*-minimum spanning tree approximation T_R containing *k* vertices is located greedily from G_R by selecting the closest node to the spanning tree starting from the vertex *r* and decomposed back to the original graph by replacing the edges with their shortest paths.

Algorithm: `MST k-Steiner Minimum Tree`

Input: A connected graph $G = (V,E)$, a terminal set $R \subseteq V$, a root vertex $r \in R$ and $2 \le k \le |R|$

Output: A Steiner tree *T* for *R* in *G* rooted to the vertex *r* containing *k* terminal vertices.

(1) Add one node to the graph *G* and connect it to all terminal nodes contained in *R* with an edge having cost 0. The result is denoted as graph G_V.

(2) Replace G_V with the minimum spanning tree of G_V.

(3) Compute the shortest path between two terminal nodes by iterating all edges of *E* in *G* and constructing the corresponding triplets. Transform the resulting triplets to graph G_R.

(4) Compute a *k*-minimum spanning tree approximation T_R from G_R rooted to the vertex *r* and containing *k* vertices of *R*.

(5) Transform T_R into subtree *T* of *G* by replacing each edge of T_R by the corresponding shortest path.

An example execution of the MST *k*-Steiner algorithm when $k=4$ and $|R|=5$ is shown in the Figure 1. In the figure a graph *G* is given with the terminal set $R = \{r_i \mid 1 \le i \le 5\}$ (denoted as ● including root vertex r_1, which is denoted as ◉) and the non-terminal nodes $m_i, 1 \le i \le 3$ (denoted as ○). Integers associated to the edges represent the edge costs.

5. Time Complexity

MST *k*-Steiner executes MST algorithm once in step (2) and once in step (4) stopping when *k* nodes have been reached. The transformation of the graph in step (3) using bucket sort [19] requires at maximum $|V|\log|V|+|E|$

steps, where $|V|$ is the number of vertices in the input graph G and $|E|$ is the number of edges in input graph G. Therefore the time complexity required for the algorithm is:

$$MST + MST_k + |V|\log|V| + |E|, \quad (5.1)$$

where MST denotes the time required for executing the Minimum Spanning Tree and MST_k denotes the time required for executing the Minimum Spanning Tree for k nodes. Certainly $MST_k \leq MST$ and $|V| \leq |E|\text{-}1 \leq |E|$, bounding the time complexity to:

$$2*MST+|E|\log|E|+|E|. \quad (5.2)$$

Minimum Spanning Tree can be implemented for example using the Kruskal's algorithm [24] having $O(|E|\log|E|)$ time complexity. Therefore MST k-Steiner algorithm's time complexity is $O(|E|\log|E|)$, which allows the algorithm to be used also in large graphs.

6. Approximation Ratio

Approximation ratio of an algorithm is computed as a ratio between the worst case performance and the optimal performance. For $k = 2$ the approximation ratio is 1, because the shortest path to the nearest resource is always selected. Also when $k = |R|$, MST k-Steiner reduces to a well-known MST-approximation algorithm [19,24] for Steiner Minimum Tree problem having approximation ratio 2. So, it still remains to determine what the approximation ratio is when $2 < k < |R|$.

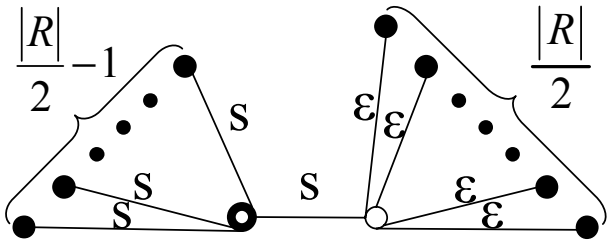

Figure 2. A graph where MST *k*-Steiner makes a large approximation error

A difficult case for MST k-Steiner is a graph shown in Figure 2. In the scenario, the root node is located within S distance from $\frac{|R|}{2}-1$ terminal nodes and within $S + \varepsilon$ distance from the other half of terminal nodes. The difference between these distances is that on the left hand side discovering each terminal node requires travelling S distance and on the right hand side discovering the first terminal node requires travelling $S + \varepsilon$ distance, but then the other terminal nodes can be discovered with ε distance.

Without a loss of generality the analysis can be restricted to cases where $|R|$ is even. Now the approximation ratio α between the discovered path and the optimal path can be calculated for $k = \frac{|R|}{2}$ as:

$$\alpha = \frac{\left(\frac{|R|}{2}-1\right)S + S + \varepsilon}{S + \frac{|R|}{2}\varepsilon} = \frac{\frac{|R|}{2}S + \varepsilon}{S + \frac{|R|}{2}\varepsilon} \quad (6.1)$$

Considering $\varepsilon \approx 0$ the approximation ratio becomes:

$$\alpha = \frac{|R|}{2} \quad (6.2)$$

This implies that when the size of the terminal set grows and the number of discovered terminals k is close to $\frac{|R|}{2}$ the approximation ratio can become large. Still, the approximation ratio seems to be bounded to $\frac{|R|}{2}$, because adding terminal node on the left hand side and removing one terminal node from the right hand side makes the optimal path longer while keeping the discovered path almost the same (decreased by ε). In contrast by adding a terminal node on the right hand side and removing one terminal node from the left hand side makes the discovered path shorter while keeping the optimal path the same. Also decreasing k from $\frac{|R|}{2}$ will decrease the length of the discovered path faster than the optimal path thus giving a lower approximation ratio. Increasing k will lengthen the optimal path faster than the discovered path resulting in a lower approximation ratio than $\frac{|R|}{2}$.

As a summary, the approximation ratio of the algorithm depends on the number of available resources and can be no less than $\frac{|R|}{2}$. It is still left for future work to show that the ratio could not be even worse. There are however approximation algorithms for k-Steiner Minimum Tree, which achieve constant factor approximation ratios [3]. They rely on integer programming and by relaxing the constraints to a linear program sustain approximation ratio guarantees.

7. Simulations

In this section we present an analysis of five algorithms: Breadth-First Search (BFS) [13], Self-avoiding Random Walk (RWSA), Highest Degree Search (HDS) [1,10], Dynamic Query Protocol (DQP) [6] and MST k-Steiner Minimum Tree. The simulations were conducted in P2PRealm network simulator.

7.1. Peer-to-Peer Network Scenarios

As simulation scenarios we used power-law graphs, normal distributed random graphs and a recently measured topology of Gnutella2 P2P network [21] with an edge cost 1 for all edges. Power-law graphs were generated using Barabási-Albert model [2]. In power-law network few hub nodes have many neighbors

and many nodes have only few neighbors. Gnutella2 topology was obtained by extracting the largest connected component from the topology data of 02/02/05 presented in [21] and removing those nodes whose edges were not referenced by other nodes. Finally those edges whose one end point was missing were removed.

Resource instances were allocated for power-law and random graphs based on the number of neighbors each node had such that the number of different resource instances in a node was the same as the number of neighbors the node had. This means that in the power-law graphs the hubs were more likely to contain the queried resource. Resources were allocated to nodes by randomly sampling from a uniform distribution. The queried resources and the querying nodes were selected also randomly from a uniform distribution for each query.

In Gnutella2 topology the resources were allocated based on the measured resource distributions of Gnutella network in September 2003 [14]. The number of different resources was selected to be 10, so the topology files could be kept small enough, but the number of resource instances for each resource was sampled from the resource distribution of [14] which produced 43216 different resources instances. These resource instances were allocated randomly to nodes following the measured distribution of shared files in nodes [14] such that one node could not have multiple instances of the same resource. Now when 100 queries were executed each resource was queried multiple times, but from a different location, which was randomly selected. The queried resource was selected according to the peer keyword distribution of [14].

Table 1 illustrates the characteristics of each scenario used in the simulations.

Table 1. Simulation Scenarios

Scenario	PL10000	N10000	Gnutella2
Distribution	Power-Law	Normal	-
Nodes	10000	10000	74297
Edges	19997	19997	609036
Largest hub	161	11	360
Resources	1000	1000	10
Res. instances	39994	39994	43216
Queries	100	100	100
Diameter	8	10	12

7.2. Results

The tests were conducted by varying the target amount of resource instances that was needed to be found by the algorithms. The target percentage of the discovered resource instances determines the amount how many resource instances of a certain resource needs to be discovered out of all resource instances of that resource and represents the *k* parameter of the Rooted *k*-SMT problem. The measured variables were the average number of query packets used in a query as shown in figures 3, 4 and 5 and the average number of maximum hops as shown in figures 6, 7 and 8.

As can be seen from Figure 3 in power-law graphs MST *k*-Steiner algorithm produces query paths between one and two orders of magnitude shorter than local search algorithms. Also, the approximation error of MST *k*-Steiner in the scenario is at most $\alpha = 2$, because the theoretical optimum is *k*-1 query packets when each node can have only one instance of the queried resource. *k*-1 represents a situation that each forwarded query packet would locate one new resource instance and the query originator does not have the queried resource.

The performance of HDS is close to the paths of MST *k*-Steiner algorithm when only one or two instances of resources needs to be located (resource percentage < 3%). This is a bit surprising even though the scenario is designed directly for HDS type of algorithms. The resources are discovered more probably in the center of the network and as noted in [1] HDS travels those nodes early in the search process. However, when more resources needs to be discovered HDS travels multiple times to the central nodes and sometimes randomly forward the query packet decreasing the performance. Compared to RWSA and BFS, HDS performs significantly better when half of the available resource instances needs to be located and after that RWSA becomes a better algorithm. BFS in turn is at the same level with RWSA when less than 40% of resources needs to be located having TTL values between 1 and 4. With TTL values 5-7 BFS cannot keep up with RWSA. DQP is significantly less performing than BFS when small amount of resource instances needs to be located, because DQP requires always executing a two hop query first. DQP however reaches the same level with BFS when 40% or more resources needs to be located. Because of maximum TTL restrictions DQP cannot locate more than 60% of available resource instances.

In normal distributed graphs, as shown in Figure 4, MST *k*-Steiner retains its characteristics having largest approximation error at most $\alpha = 4$. Normal distributed graphs have larger diameter than power law distributed graphs and therefore estimating the optimal performance with *k* is too pessimistic. This argument is supported by the fact, that when 100% of resource instances needs to be discovered, the approximation ratio is at maximum $\alpha = 2$ as discussed in Section 6. It is therefore not clear, whether as short paths as *k* would exist in the normal distributed graph and presumably the real approximation error is at a similar range as in power-law graphs. Thus we conclude that the approximation ratio derived in section 6 highly overestimates the optimal performance in power-law and normal distributed P2P scenarios.

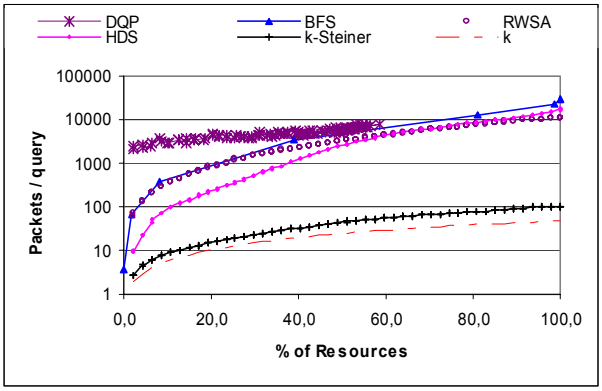

Figure 3. Query packets in PL10000

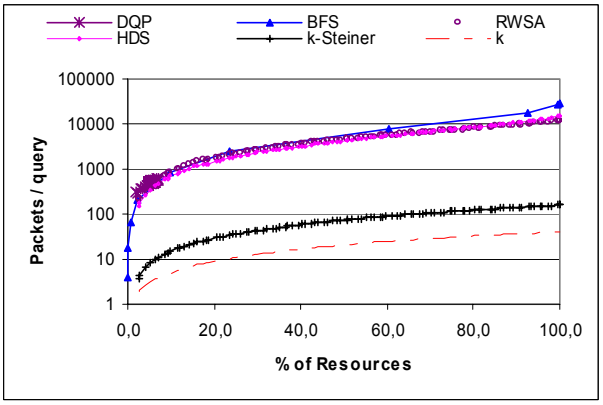

Figure 4. Query packets in N10000

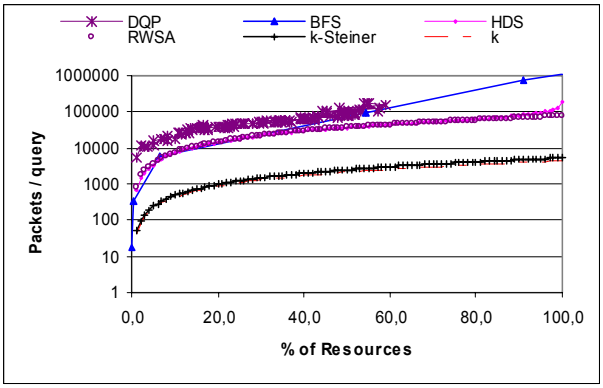

Figure 5. Query packets in Gnutella2

The difference between local search algorithms and MST *k*-Steiner paths is again in the order of one or two magnitudes. In contrast to power-law graphs, the local search algorithms in normal distributed graphs have similar performance when less than half of available resource instances needs to be located. After that RWSA and HDS outperform BFS. Random graph does not contain hub nodes and therefore HDS does not benefit from its ability to travel to high degree nodes. Basically, HDS appears as a self-avoiding random walker, because all the neighbors are almost equally connected. The large diameter of normal distributed graph restricts DQP

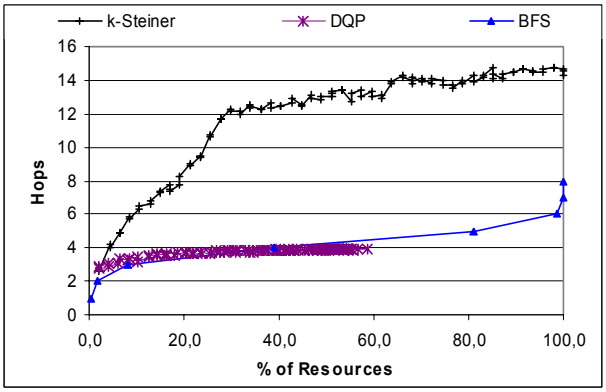

Figure 6. Maximum number of hops in PL10000

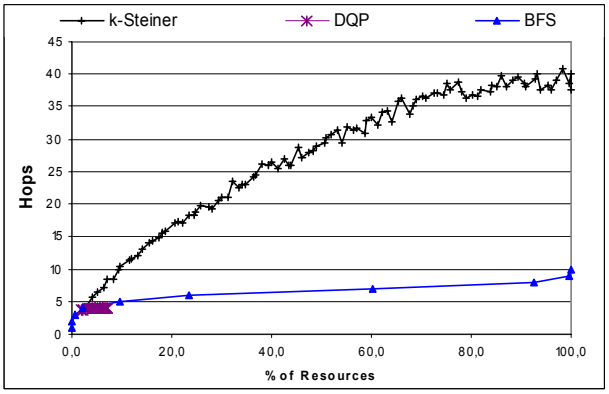

Figure 7. Maximum number of hops in N10000

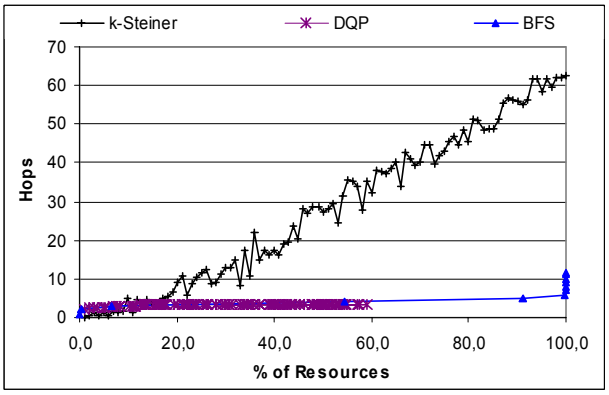

Figure 8. Maximum number of hops in Gnutella2

to locate only 7% of resource instances with time-to-live 4.

In Gnutella2 topology, as shown in Figure 5, MST *k*-Steiner does not seem to make any approximation error suggesting that Gnutella2 topology is highly connected and thus allowing each hop of a query to locate a new resource instance. The difference between MST *k*-Steiner paths and local search algorithms is in the order of a magnitude. HDS and RWSA perform equally well and BFS can keep up with them to 40% of resource instances. Then BFS departs to the level of DQP, which can locate at maximum 60% of resource instances.

The average of maximum hops for MST *k*-Steiner, BFS and DQP is plotted in Figures 6, 7 and 8. HDS and RWSA are omitted as their number of hops is shown in Figures 3, 4 and 5. Because HDS and RWSA forward to only one direction at a time their maximum hops are in different scale than what MST *k*-Steiner, BFS and DQP are using. Therefore if low latency in the network is critical, HDS and RWSA may not be suitable as local search algorithms. From the Figures 6 and 7, it can be seen that BFS and DQP require in N10000 two or three hops more than in PL10000 to locate the same amount of resource instances. BFS locates the shortest paths to resources and therefore has a small latency. However, MST *k*-Steiner does not seem to be using these paths. Reason for this is that the shortest paths do not necessarily contain resources along the path and therefore collecting some resources using a longer route may lead to a path which is more efficient. The latency in power-law graphs also stays comparable to BFS, but in normal distributed graphs the length of query paths grows significantly. This is, however, in completely different scale than the hops used by HDS and RWSA.

8. Conclusion

The Rooted *k*-Steiner Minimum Tree problem connects the resource discovery problem to a solid foundation of graph theory providing means to calculate near-optimal query paths in a graph. The MST *k*-Steiner algorithm computes an approximation of the shortest tree between the querying node and the nodes having the queried resource instances and thus is an upper bound for the performance of local search algorithms. The algorithm can be used in cases, where nodes contain only one instance of queried resource and the problem has to be further extended if multiple resource instances in a node is to be supported. In overall, the results presented here show that local search algorithms commonly used in P2P networks are far from optimal paths.

Based on the findings in Gnutella2 topology, DQP has slightly lower performance than BFS, but because of automatic adaptation of time-to-live parameter it can be feasibly used in current P2P networks. HDS and RWSA suffer from implementation problems because to avoid already visited nodes they need to keep record of visited nodes and therefore the size of the query packet grows in large graphs limiting their use.

What makes the resource discovery problem hard in P2P networks is that only local information is available. It would be interesting to know how close to the optimum can algorithms get using local knowledge. A record of the global network topology is used in Open Shortest Path First [17] IP routing protocol and Dijkstra's algorithm for computing the shortest paths suggesting possibilities that MST *k*-Steiner tree algorithm could be adapted to distributed P2P networks. In this case, information about the resources needs to be at least partially cached in the nodes. This, however, needs further research.

References

[1] L. A. Adamic, R. M. Lukose, B. A. Huberman, "Local Search in Unstructured Networks", Handbook of Graphs and Networks: From the Genome to the Internet, Wiley-VCH, 2003, pp. 295-317.

[2] A.-L. Barabási, R. Albert, "Emergence of Scaling in Random Networks", Science 286, 1999, pp. 509-512.

[3] F. A. Chudak, T. Roughgarden, D. P. Williamson, "Approximate k-MSTs and k-Steiner Trees via the Primal-Dual Method and Lagrangean Relaxation", Proceedings of the 8th International Integer Programming and Combinatorial Optimization Conference (IPCO), Springer, 2001, pp. 60-70.

[4] A. Crespo, H. Garcia-Molina, "Routing Indices For Peer-to-Peer Systems", Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02), IEEE Press, 2002, pp. 23-33.

[5] S. Daswani, A. Fisk, "Gnutella UDP extension for scalable searches (GUESS) v. 0.1".

[6] A. Fisk, "Gnutella Dynamic Query Protocol v0.1", Gnutella Developer's Forum, May 2003.

[7] M. Harchol-Balter, T. Leighton, D. Lewin, "Resource Discovery in Distributed Networks", 18th Annual ACM-SIGACT/SIGOPS Symposium on Principles of Distributed Computing (PODC'99), Atlanta, 1999.

[8] V. Kalogeraki, D. Gunopulos, D. Zeinalipour-Yatzi, "A Local Search Mechanism for Peer-to-Peer Networks", Proceedings of the 11th International Conference on Information and Knowledge Management, ACM Press, 2002, pp. 300-307.

[9] R. M. Karp, "Reducibility Among Combinatorial Problems", Complexity of Computer Computations, Plenum Press, New York, 1975, pp. 85-103.

[10] B. J. Kim, C. N. Yoon, S. K. Han, H. Jeong, "Path finding strategies in scale-free networks", Physical Review E 65, 2002.

[11] N. Kotilainen, M. Vapa, A. Auvinen, T. Keltanen, J. Vuori, "P2PRealm – Peer-to-Peer Network Simulator", 11th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, IEEE Communications Society, pp. 93-99, Trento, Italy, 2006.

[12] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks", Proceedings of the 16th International Conference on Supercomputing, ACM Press, 2002, pp. 84-95.

[13] N. Lynch, "Distributed Algorithms", Morgan Kauffmann Publishers, 1996.

[14] P. Makosiej, G. Sakaryan, H. Unger, "Measurement Study of Shared Content and User Request Structure in Peer-to-Peer Gnutella Network", Proceedings of Design, Analysis, and Simulation of Distributed Systems (DASD 2004), Arlington, USA, April 2004. pp. 115-124.

[15] K. Mehlhorn, "A faster approximation algorithm for the Steiner problem in graphs", Information Processing Letters, vol. 27 issue 3, 1988, p. 125-128.

[16] D. A. Menascé, "Scalable P2P Search", IEEE Internet Computing, Vol. 7, No. 2, March-April 2003, pp. 83-87.

[17] J. Moy, "OSPF Version 2", RFC 2328, The Internet Society, April 1998.

[18] A. Oram, "Peer-to-Peer: Harnessing the Power of Disruptive Technologies", O'Reilly & Associates, March 2001.

[19] H.-J. Prömel, A. Steger, "The Steiner Tree Problem: A Tour through Graphs, Algorithms, and Complexity", Advanced Lectures in Mathematics, Vieweg Verlag, 2002.

[20] N. Sarshar, P. O. Boykin, V. P. Roychowdhury, "Percolation Search in Power Law Networks: Making Unstructured Peer-to-Peer Networks Scalable", Proceedings of the Fourth International Conference on P2P Computing (P2P'04), IEEE Press, 2004, pp. 2-9.

[21] D. Stutzbach, R. Rejaie, S. Sen, "Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems", Proceedings of the ACM SIGCOMM Internet Measurement Conference, Berkeley, October 2005.

[22] D. Tsoumakos, N. Roussopoulos, "Adaptive Probabilistic Search for Peer-to-Peer Networks", Proceedings of the Third International Conference on P2P Computing (P2P'03), IEEE Press, 2003, pp. 102-109.

[23] M. Vapa, N. Kotilainen, A. Auvinen, H. Kainulainen, J. Vuori, "Resource Discovery in P2P Networks Using Evolutionary Neural Networks", International Conference on Advances in Intelligent Systems – Theory and Applications (AISTA 2004), 2004.

[24] B. Y. Wu, K.-M. Chao, "Spanning Trees and Optimization Problems", Discrete Mathematics and Its Applications, Chapman & Hall/CRC, 2004.

[25] B. Yang, H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks", Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02), IEEE Press, 2002, pp. 5-14.