Search Algorithms for Unstructured Peer-to-Peer Networks

Reza Dorrigiv
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada
Email: rdorrigiv@uwaterloo.ca

Alejandro López-Ortiz
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada
Email: alopez-o@uwaterloo.ca

Paweł Prałat
Department of Mathematics and Statistics
Dalhousie University
Halifax, NS, B3H 3J5, Canada
Email: pralat@mathstat.dal.ca

***Abstract*—We study the performance of several search algorithms on unstructured peer-to-peer networks, both using classic search algorithms such as flooding and random walk, as well as a new hybrid algorithm proposed in this paper. This hybrid algorithm first uses flooding to find sufficient number of nodes and then starts random walks from these nodes. We compare the performance of the search algorithms on several graphs corresponding to common topologies proposed for peer-to-peer networks. In particular, we consider binomial random graphs, regular random graphs, power-law graphs, and clustered topologies. Our experiments show that for binomial random graphs and regular random graphs all algorithms have similar performance. For power-law graphs, flooding is effective for small number of messages, but for large number of messages our hybrid algorithm outperforms it. Flooding is ineffective for clustered topologies in which random walk is the best algorithm. For these topologies, our hybrid algorithm provides a compromise between flooding and random walk. We also compare the proposed hybrid algorithm with the k-walker algorithm on power-law and clustered topologies. Our experiments show that while they have close performance on clustered topologies, the hybrid algorithm has much better performance on power-law graphs. We theoretically prove that flooding is effective for regular random graphs which is consistent with our experimental results.**

I. Introduction

Peer-to-peer networks are widely used for file sharing purposes. This type of usage tends to favour resilient, decentralized architectures over centralized solutions. However this comes at a penalty in ease of searching. At first, peer-to-peer systems addressed this shortcoming by incorporating a flooding mechanism for resource discovery. A node in the peer-to-peer network broadcasts a query message to its neighbours. The neighbours in turn are responsible for reporting any matches as well as forwarding the message to its neighbours, if necessary. This mechanism has been proven effective in practice for finding items which are prevalent across the peer-to-peer network, but otherwise ineffective and resource consuming.

As a consequence numerous alternatives search techniques have been proposed, ranging from variants in flooding algorithms, to structured distributed solutions to centralized indices [CS02]. Each of these techniques has its own merits and drawbacks.

In this paper we consider searching for an object in an unstructured decentralized peer-to-peer networks such as Gnutella or FastTrack. It is known that in this context flooding excels in the search of popular items which are widely available. On the other hand, traditional time-to-live (TTL) techniques fail to locate items that are not available locally. Gkantsidis et al. (INFOCOM'04) [GMS04] studied random walks as an alternative to TTL-based flood techniques. They showed that certain types of random walks can outperform flooding techniques provided that certain modifications in the topology are introduced as part of the search. This work was further extended by the same authors [GMS05] in INFOCOM'05. In that work, they consider the impact of a random walk followed by shallow flooding and observe that if we assume that edge criticality is known then the graph can be labeled in such a way as to obtain an effective search technique. A similar study is by Lv et al. [LCC$^+$02] where they consider k random walkers in random, power-law, grid and Gnutella graphs.

In this paper we systematically study the effect of random walk versus flooding in a variety of settings and topologies. We consider, in particular, binomial random graphs, d-regular graphs, power-law graphs and clustered topologies. We follow the methodology of Gkantsidis et al. [GMS04], [GMS05] in which a maximum number of messages is set and then we determine what percentage of the nodes in the peer-to-peer graph are visited using this number of messages. In a sense this is a measure of per-message effectiveness of the search method. As well, we consider the case in which changes to the topology are not feasible due to the unstructured nature of the peer-to-peer topology.

We proceed to describe the search algorithms which are of interest in our work.

Flooding. Flooding is one of the simplest search strategies and yet one of the most commonly used in peer-to-peer networks. For instance, it is the search method employed by the Gnutella network. The search proceeds as follows: the origin of the search sends "discovery" messages to all its neighbours, which in turn propagate the message to their own neighbours (except the neighbour from which they have received the message) and so on.

Flooding as described above is unrestricted in the sense that there is no constraint on the number of messages generated

by the search request. A simple way to limit the range of the flooding is to introduce the so-called *time-to-live* (TTL) parameter. More superficially, the origin of the search sends a discovery message to all its neighbours, along with the parameter TTL. The recipients decrease TTL by one and if TTL is still positive they propagate the message to their own neighbourhoods (except the neighbour from which they have received the message), and so fourth. In this way, it is guaranteed that the search is restricted within a ball of radius TTL from the origin of the search.

The popularity of flooding is mainly due to its simplicity; it is easy to implement and easy to support. Moreover, in its unrestricted version (or when TTL is set to a sufficiently large value) it will always succeed in the search. The most striking drawback of flooding is that it suffers from rather prohibitive message complexity.

Simulation of Random Walks. Random walks have emerged in the area of peer-to-peer networks as alternatives to flooding. The idea is that instead of forwarding a request to all the neighbours, the recipient sends it to a random neighbour. The appealing property of the random walk method is the small message complexity, which means the algorithm scales well with the size of the network. In addition, random walks have been studied extensively from a theoretical point of view. Naturally, we are interested in the time (or, equivalently, on the number of hops) the random walk needs to locate the host sought. An efficient random walk will have the property that this time is small.

There are variants of the random walk method which aim to reduce the search time. One variant involves the use of k *independent random walks*, all simulated in parallel from the origin. The search is successful once the file is located by any one of the individual random walks. We call this variant the k-walkers algorithm. A different variant introduces small *local flooding* into the random walk. More precisely, every node in the random walk initiates a (small) flooding, namely flooding with a small TTL value. This hybrid algorithm was proposed and studied in [GMS05].

Local flooding with k independent random walks. In this paper we propose and study a new hybrid algorithm, as a compromise between flooding and random walks. The idea is to first perform a (local) flooding starting from the origin of the search; the flooding continues until exactly k new outer nodes have been discovered, for some predefined value of k. In the case where one of these nodes has the file, the search is successful and the origin is notified. Otherwise, each of the k nodes initiates an independent random walk.

What is the motivation behind such a hybrid algorithm? Clearly, we are trying to exploit the positive characteristics of two widely different algorithms, namely flooding and random walks. If the file is located close to the origin, the local flooding would be sufficient to locate it fast, and with few messages exchanged. If the file is located away from the network, it is expected that it will be located by one of the random walks, but more importantly, since the flooding occurs only locally, the message complexity is small.

There exist situations in which one might anticipate that the two hybrid algorithms described above will not necessarily have similar performance. For instance consider a network which consists by several dense clusters (e.g., cliques) which are interconnected by means of a very sparse network (e.g., a tree that spans the representatives of each cluster). If we initiate a search from a certain cluster, assuming that the file is located in a different cluster, using random walks with local flooding, it may happen that the random walk remains "trapped" within the cluster of the origin of the search. On the other side, one would hope that local flooding combined with independent random walks may resolve this problem, since the flooding could discover vertices outside the origin's cluster at an earlier time.

II. Theoretical Results

In this section we prove that in d-regular graphs, flooding finds new vertices most of the time. Similar behaviour can be proved for $G_{n,d/n}$. Therefore we expect flooding to have a good performance on these topologies. Let $N_i(u)$ denote the set of vertices at distance at most i from u. Note that, in the early stages of the flooding process, the graph revealed from vertex u tend to be a tree, that is, the number n_i of elements in $N_i(u)$ is approximately $n_{i-1}+(d-1)(n_{i-1}-n_{i-2})$. Thus, after $i \sim \frac{3}{4}\log_{d-1} n$ steps, we expect $N_i(u)$ to contain about $n^{3/4}$ vertices.

Let f_i denote the total number of queries for flooding with $TTL = i$, that is, $f_i = d\sum_{j=0}^{i-1}(d-1)^j = \frac{d(d-1)^i-1)}{d-2}$, and $i_0 = \frac{1}{2}\log_{d-1} n$. An event is said to hold *with high probability* (w.h.p.), if it holds with probability $1-o(1)$ as $n \to \infty$.

Lemma 1: Let $\omega(n)$ be any function of n such that $\omega(n) \to \infty$ as $n \to \infty$. For $i \le i_0 - \omega(n)$ w.h.p. the cardinality n_i of $N_i(u)$ equals f_i. Moreover, for $i \le i_0 + \omega(n)$ w.h.p. $n_i = f_i - O\Big(\omega(n)(d-1)^{3(i-i_0)+\omega(n)}\Big)$.

Proof: Note that the expected number of "cross edges" that generate a new cycle(s) at step $i+1$ is equal to $O(n_i^2/n) = O(f_i^2/n) = O(d-1)^{2i}/n)$.

Consider $i_1 = \lfloor i_0 - \omega(n)\rfloor = \lfloor \frac{1}{2}\log_{d-1} n - \omega(n)\rfloor$. The expected number of "cross edges" found up to time-step i_1 is equal to $\sum_{j=0}^{i_1-1} O\big(d-1)^{2j}/n\big) = O\big(d-1)^{2i_1}/n\big) = o(1)$. Thus, from Markov inequality, until step i_1 w.h.p. there are no "cross edges", hence w.h.p. $N_{i_1}(u)$ is a tree and $n_i = f_i$ for $i \le i_1$.

Also, the expected number of "cross edges" added between step i_1+1 and step i, $i \le \lfloor i_0+\omega(n)\rfloor = \lfloor \frac{1}{2}\log_{d-1} n+\omega(n)\rfloor$ is equal to $\sum_{j=i_1}^{i-1} O\big((d-1)^{2j}/n\big) = O\big((d-1)^{2i}/n\big) = O\big((d-1)^{2(i-i_0)}\big)$. Thus, again from Markov inequality, w.h.p. the total number of "cross edges" at time i is at most $O\big(\omega(n)(d-1)^{2(i-i_0)}\big)$. Since one "cross edge" added at this time interval can destroy tree branch of size $O\big((d-1)^{i-i_0+\omega(n)}\big)$, the following equality holds w.h.p. $n_i = f_i - O\big(\omega(n)(d-1)^{2(i-i_0)}\big)\cdot O\big((d-1)^{i-i_0+\omega(n)}\big)$, which completes the proof of the theorem. ∎

The next theorem is a simple consequence of the Lemma 1.

Theorem 1: Let $\hat{\omega}(n)$ be any function such that $\hat{\omega}(n) \to \infty$ as $n \to \infty$. For $i \leq \hat{i} = \frac{3}{4}\log_{d-1} n - \hat{\omega}(n)$ w.h.p. $\frac{n_i}{f_i} = 1 - o(1)$.

Proof: Let $\omega(n) = o(\hat{\omega}(n))$. From Lemma 1, for any $i \leq \hat{i}$

$$\begin{aligned}\frac{n_i}{f_i} &= 1 - O\Big(\omega(n)(d-1)^{2i-3i_0+\omega(n)}\Big)\\ &= 1 - O\Big(\omega(n)(d-1)^{-2\hat{\omega}(n)+\omega(n)}\Big)\\ &= 1 - O\Big(d-1)^{-2(1+o(1))\hat{\omega}(n)}\Big)\\ &= 1 - o(1)\end{aligned}$$

and the assertion follows. ∎

III. Experimental Evaluation

In this section we present the results of the experimental evaluation of our algorithms. Subsection III-A provides details concerning the generation of inputs and the performance metrics used. Subsection III-B is a discussion of the graph topologies we are studying. Finally in Subsection III-C we present the results we obtained through experiments along with a discussion of their significance. We implemented the algorithms in C/C++, and run them in a cluster of clusters (see Section V for more details). The program code can be found at the following address: "http://www.mathstat.dal.ca/˜pralat/".

A. Methodology

For our experiments we generate 80 random graphs (for each topology) of 100,000 nodes each, and run the algorithms on each graph; this ensures that we do not, accidentally, choose an outlier graph. For each topology and each algorithm, we consider several aggregate functions (mean, median, standard deviation) on the results of running that algorithm on the corresponding 80 graphs.

On a similar note, for each of 80 input graphs, we consider every vertex as a possible origin for a search, and we evaluate the average performance of the algorithms over all origins (and all input graphs). This is particularly important for clustered topologies and power-law graphs in which such outlier vertices are not infrequent.

Naturally, we want to evaluate the algorithms on input graphs which are connected. However, the popular methods for randomly generating graphs (see Subsection III-B) do not necessarily produce connected graphs. We remedy this situation by the following process: First, we make every isolated vertex adjacent to a vertex in the graph, by adding an edge at random. Then, we find the connected components of the new graph (using BFS) and add a random edge between components. This guarantees the final graph is connected.

For a fair comparison of the various search algorithms, we require that they all use the same number of messages. For our experiments we look at values of 5^5, 5^6, and 10^5 for some topologies ($G_{n,\frac{5}{n}}$, 5-regular graphs, $\mathcal{P}(n,5,\eta)$, $\mathcal{C}(n,100,d)$) and 10^4 and 10^5 for other topologies.

The major performance metric we use is the vector $V = (v_0, v_1, \ldots, v_k)$, where v_i denotes the average number of nodes visited exactly i times, over all searches. Clearly, the most significant coordinate of V is v_0 since the number of distinct nodes visited during the execution of a search algorithm is $n - v_0$. An additional interpretation of the significance of v_0 is the following: Suppose c copies of a file we seek are placed over the network, uniformly at random. Then we expect $\frac{n-v_0}{n}c$ copies of this file to be discovered by the search algorithm. The remaining coordinates of V capture repetitive visits of nodes in the graph. Clearly, a good search algorithm would exhibit very small values of v_i's , especially for large values of i.

B. Topologies

We focus on topologies which have been studied extensively in the past, and in particular for topologies for which there is good empirical evidence that they model, at least to a satisfactory level, the structure of modern communication networks. Among the topologies we study, the binomial random graph and the d-regular random graph models are models of mostly theoretical interest whereas power-law graphs and clustered topologies come closer to capturing the behaviour of realistic networks.

Binomial random graphs. Given a real number p, $0 \leq p \leq 1$ and a collection of vertices V, a random binomial graph is generated by including, for every pair of vertices $v_i, v_j \in V$, an edge (v_i, v_j) with probability p; the random experiment is repeated independently for all $\binom{n}{2}$ pairs of vertices. This model has been studied extensively (see [JŁR00] for the definitive treatment of this topic). For instance, well-known theorems concerning this class of graphs characterize the threshold at which a giant component emerges or the random graph becomes connected.

For our evaluation, we are looking at graphs with expected degree 5 and 10.

d-regular random graphs. A d-regular graph is defined as a graph in which every vertex has degree d; a random d-regular graph is a graph chosen uniformly at random from the space of all d-regular graphs on n vertices. Random d-regular graphs have certain appealing properties such as low diameter and good connectivity and, as in the case of binomial random graphs, have also been studied extensively from a theoretical point of view.

Generating a random d-regular graph is a subtle issue. One approach is the so called *configurational model* which provides a framework for creating a random graph with a certain degree sequence (see e.g., [Wor99]). This approach is relatively simple, but has the drawback that it may yield a multigraph (i.e., multiedges and self-loops may be present) instead of a simple graph. Instead, we choose to implement the following d-process: we begin with n isolated vertices, to which edges are added randomly one by one so that the maximum degree of the induced graph is always at most d. It

is known [RW92] that with probability tending to 1 as $n \to \infty$, the result of this process is a d-regular graph (except for one vertex of degree $d-1$ when dn is odd).

For our evaluation, we are looking at 5-regular and 10-regular graphs.

Power-law graphs. In recent years substantial evidence has been presented that networks such as the Web graph exhibit a power-law with respect to the degrees of their nodes. More precisely, Broder et al. [BKM$^+$00] noticed that the distribution of degrees in the Web graph follows a power-law: the fraction of vertices with degree d is proportional to $d^{-\gamma}$, where γ is a constant independent of the size of the network (more precisely, $\gamma \sim 2.1$ for in-degrees, $\gamma \sim 2.7$ for out-degrees). As a reminder, the web graph is defined as the graph in which every node corresponds to a static web page and for every hyperlink between two pages there exists a directed edge incident to the corresponding nodes.

There are several known models for the web graph (see for example the general survey [Bon04]). We chose to implement a recently developed model, known as the *Protean Model*. Our choice is motivated by certain very attractive properties of such graphs: in particular, it has been shown (see [Pra06]) that the protean graph has one giant component which contains a positive fraction of all vertices and is of diameter $\Theta(\log n)$. This indicates that the protean graph captures the structure of small-world networks (of which real-life examples are web graph and other natural networks). Note also that the definition of protean process allows us to study the *recovery time*; an interesting and very important property which does not have its counterpart for the other models.

The formal definition of the protean graph $\mathcal{P}_n(d,\eta)$ is somewhat technical (we refer the reader to [ŁP06] for more details; see also [PW] for extended version of standard protean graph). Nevertheless the idea behind its definition is simple. This model takes into account an additional natural parameter of a vertex, its *age*, and predicts how it influences the degree of a vertex. We start with any graph G with vertex set $[n]$, and at each time-step we pick randomly one of the vertices v to be *renewed*. More precisely, we delete from G all edges incident to v; this corresponds to a removal of a random node from the network. Then we generate d new edges from v to existing vertices chosen randomly with weighted probabilities ('old' vertices have bigger probability of being chosen). Note that vertex v can be viewed as a new node which establishes connection with some nodes in the network. When all vertices are renewed at least once, the random graph is a protean graph $\mathcal{P}_n(d,\eta)$.

In [ŁP06] it is shown that the degrees of the $\mathcal{P}_n(d,\eta)$ are distributed according to a power-law. More precisely, the number of vertices of degree k decreases roughly as $k^{-1-1/\eta}$.

Clustered Topologies. These graphs are intended to model realistic networks which consist of relatively isolated clusters of nodes, connected through a backbone network. In particular, for our experiments we consider three models for clusters:

TABLE I
EXPERIMENTAL RESULTS FOR $G_{N,p}$ WITH $N = 100,000$ AND $p = 5/N$

A	B	Mean	Perc. of best	Minimum	Maximum	σ
5^5	$\mathcal{F}$	3072.177	100.000	3071.768	3072.548	0.137
5^5	$\mathcal{NF}$	3066.836	99.826	3066.157	3067.383	0.238
5^5	$\mathcal{RW}$	3034.931	98.788	3032.454	3036.592	0.718
5^5	$\mathcal{H}_1$	3035.026	98.791	3032.569	3036.718	0.728
5^5	$\mathcal{H}_2$	3035.282	98.799	3032.854	3036.947	0.697
5^5	$\mathcal{H}_3$	3046.035	99.149	3044.455	3047.162	0.490
5^6	$\mathcal{F}$	14215.424	99.811	14209.195	14222.816	2.854
5^6	$\mathcal{NF}$	14242.398	100.000	14237.605	14246.279	1.932
5^6	$\mathcal{RW}$	14120.694	99.145	14107.459	14129.181	3.885
5^6	$\mathcal{H}_1$	14120.759	99.146	14107.630	14129.214	3.875
5^6	$\mathcal{H}_2$	14121.114	99.148	14107.901	14129.619	3.854
5^6	$\mathcal{H}_3$	14122.017	99.155	14108.917	14130.281	3.774
N	$\mathcal{F}$	58867.496	99.231	58784.255	58936.885	30.072
N	$\mathcal{NF}$	59323.840	100.000	59248.742	59383.612	23.729
N	$\mathcal{RW}$	59297.404	99.955	59218.167	59351.545	25.516
N	$\mathcal{H}_1$	59297.513	99.956	59217.602	59351.357	25.495
N	$\mathcal{H}_2$	59297.580	99.956	59218.256	59351.393	25.468
N	$\mathcal{H}_3$	59298.196	99.957	59217.733	59352.119	25.401

TABLE II
EXPERIMENTAL RESULTS FOR $G_{N,p}$ WITH $N = 100,000$ AND $p = 10/N$

A	B	Mean	Perc. of best	Minimum	Maximum	σ
10^4	$\mathcal{F}$	9468.841	99.968	9467.126	9470.258	0.618
10^4	$\mathcal{NF}$	9471.838	100.000	9471.162	9472.433	0.282
10^4	$\mathcal{RW}$	9470.951	99.991	9470.313	9471.552	0.263
10^4	$\mathcal{H}_1$	9470.944	99.991	9470.304	9471.557	0.275
10^4	$\mathcal{H}_2$	9470.978	99.991	9470.453	9471.598	0.261
10^4	$\mathcal{H}_3$	9471.172	99.993	9470.518	9471.796	0.298
N	$\mathcal{F}$	61209.108	99.707	61179.880	61234.376	12.791
N	$\mathcal{NF}$	61386.856	99.997	61365.743	61405.355	9.185
N	$\mathcal{RW}$	61388.751	100.000	61367.928	61407.741	9.038
N	$\mathcal{H}_1$	61388.728	100.000	61367.282	61408.125	9.062
N	$\mathcal{H}_2$	61388.819	100.000	61367.340	61407.824	9.110
N	$\mathcal{H}_3$	61388.855	100.000	61367.372	61408.264	9.003

- K_l, i.e. clique of size l
- Random graph $G_{l,1/2}$
- Random graph $G_{l,1/5}$

Every cluster has a representative; we then connect the representatives by means of a 3-regular random graph. We consider clusters of size $l = 100$.

C. Experiments

Tables I-XI show the Mean/Minimum/Maximum of the average number of distinct nodes that are discovered by each algorithm on the 80 graphs produced according to different topologies. In each table, the first column (labeled A) shows the number of messages and the second column (labeled B) denotes the algorithm. The algorithms are denoted as follows: flooding by $\mathcal{F}$, normalized flooding by $\mathcal{NF}$, random walk by $\mathcal{RW}$, and hybrid algorithm with 10^i random walkers by $\mathcal{H}_i$. For each topology and a fixed number of messages the row that corresponds to the algorithm with maximum Mean is highlighted. Also the percentage of Mean of other algorithms to this algorithm's Mean is shown. The other columns show the Mean, Minimum, Maximum, and standard deviation.

a) Analysis of results for binomial random graphs: Tables I and II show the results for binomial random graphs. We observe that in this case all algorithms have similar performance. Also as the number of messages increases, the performance of flooding decreases; it is the best algorithm for 5^5 messages, but not for N messages. We note that $\mathcal{RW}$, $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$ behave almost the same.

TABLE III
EXPERIMENTAL RESULTS FOR 5-REGULAR GRAPHS WITH $N = 100,000$

A	B	Mean	Perc. of best	Minimum	Maximum	σ
5^5	$\mathcal{F}$	3077.056	100.000	3076.313	3077.660	0.266
5^5	$\mathcal{NF}$	3077.056	100.000	3076.313	3077.660	0.266
5^5	$\mathcal{RW}$	3076.770	99.991	3076.686	3076.867	0.038
5^5	$\mathcal{H}_1$	3076.768	99.991	3076.676	3076.839	0.034
5^5	$\mathcal{H}_2$	3076.776	99.991	3076.682	3076.856	0.034
5^5	$\mathcal{H}_3$	3076.974	99.997	3076.426	3077.435	0.204
5^6	$\mathcal{F}$	14467.513	100.000	14462.400	14471.077	1.719
5^6	$\mathcal{NF}$	14467.513	100.000	14462.400	14471.077	1.719
5^6	$\mathcal{RW}$	14465.866	99.989	14465.449	14466.116	0.132
5^6	$\mathcal{H}_1$	14465.865	99.989	14107.630	14129.214	3.875
5^6	$\mathcal{H}_2$	14465.888	99.989	14465.457	14466.261	0.152
5^6	$\mathcal{H}_3$	14466.101	99.990	14465.279	14466.774	0.304
N	$\mathcal{F}$	63218.463	100.000	63198.744	63234.677	7.315
N	$\mathcal{NF}$	63218.463	100.000	63198.744	63234.677	7.315
N	$\mathcal{RW}$	63213.218	99.992	63,212.362	63,214.303	0.413
N	$\mathcal{H}_1$	63213.175	99.992	63212.390	63213.939	0.374
N	$\mathcal{H}_2$	63213.080	99.991	63211.912	63213.840	0.440
N	$\mathcal{H}_3$	63213.291	99.992	63212.235	63214.413	0.487

TABLE IV
EXPERIMENTAL RESULTS FOR 10-REGULAR GRAPHS WITH $N = 100,000$

A	B	Mean	Perc. of best	Minimum	Maximum	σ
10^4	$\mathcal{F}$	9517.860	100.000	9516.843	9519.056	0.517
10^4	$\mathcal{NF}$	9516.986	99.991	9516.675	9517.306	0.151
10^4	$\mathcal{RW}$	9516.496	99.986	9516.257	9516.649	0.075
10^4	$\mathcal{H}_1$	9516.515	99.986	9516.329	9516.669	0.075
10^4	$\mathcal{H}_2$	9516.513	99.986	9516.314	9516.685	0.081
10^4	$\mathcal{H}_3$	9516.674	99.988	9516.395	9516.981	0.117
N	$\mathcal{F}$	63219.926	100.000	63212.771	63228.066	3.277
N	$\mathcal{NF}$	63215.058	99.992	63213.303	63216.856	0.706
N	$\mathcal{RW}$	63213.002	99.989	63212.023	63213.810	0.431
N	$\mathcal{H}_1$	63213.065	99.989	63212.228	63213.905	0.364
N	$\mathcal{H}_2$	63213.050	99.989	63211.844	63213.721	0.347
N	$\mathcal{H}_3$	63213.106	99.989	63211.802	63213.884	0.419

b) Analysis of results for d-regular random graphs: Tables III and IV show the results for d-regular random graphs. Again the performances are very close. The good performance of flooding on these graphs is consistent with theoretical results of Section II. We observe that the standard deviations of $\mathcal{F}$ and $\mathcal{NF}$ is worse (larger) than other algorithms.

c) Analysis of results for power-law graphs with $\lambda = 2.1$: Tables V and VI show the results for power-law graphs with $\lambda = 2.1$. For small number of messages, flooding is the best algorithm according to Mean. As the number of messages increases, the efficiency of flooding decreases. For example it is the worst algorithm when we use N messages. The standard deviation of flooding is much larger than other algorithms.

d) Analysis of results for power-law graphs with $\lambda = 2.7$: Tables VII and VIII show the results for power-law graphs with $\lambda = 2.7$. For average degree 5, $\mathcal{NF}$ is the best algorithm and for average degree 10, $\mathcal{H}_3$ is the best algorithm. Flooding is the worst algorithm for large number of messages, that is, N messages. Flooding has the largest standard deviation.

e) Analysis of results for clustered graphs: Tables IX, X, and XI show the results for clustered graphs. In general, all algorithms have bad performance according to Mean. $\mathcal{RW}$ is the best algorithm; $\mathcal{F}$ and $\mathcal{NF}$ are the worst algorithms. Our hybrid algorithms are between these two extreme behaviours.

f) Summary: For binomial and d-regular graphs, all algorithms have close performance. Therefore normalized flooding is a good choice in this case as it is faster than other algorithms. This means that it takes less time on average

TABLE V
EXPERIMENTAL RESULTS FOR POWER-LAW GRAPHS WITH $\lambda = 2.1$ AND AVERAGE DEGREE 5

A	B	Mean	Perc. of best	Minimum	Maximum	σ
5^5	$\mathcal{F}$	2987.723	100.000	2950.868	3015.776	14.893
5^5	$\mathcal{NF}$	2727.960	91.306	3076.313	3077.660	0.266
5^5	$\mathcal{RW}$	2571.427	86.066	2545.339	2606.159	15.165
5^5	$\mathcal{H}_1$	2574.199	86.159	2546.333	2611.409	16.086
5^5	$\mathcal{H}_2$	2585.539	86.539	2558.693	2621.724	15.446
5^5	$\mathcal{H}_3$	2724.836	91.201	2709.105	2748.038	8.242
5^6	$\mathcal{F}$	12851.139	100.000	12311.150	13398.720	291.761
5^6	$\mathcal{NF}$	11930.450	92.836	14462.400	14471.077	1.719
5^6	$\mathcal{RW}$	11284.211	87.807	11183.438	11400.340	51.740
5^6	$\mathcal{H}_1$	11289.183	87.846	11184.395	11404.517	54.289
5^6	$\mathcal{H}_2$	11303.555	87.958	11198.152	11418.896	54.056
5^6	$\mathcal{H}_3$	11414.221	88.819	11317.776	11519.615	47.704
N	$\mathcal{F}$	39074.911	76.818	35153.269	42861.851	1968.823
N	$\mathcal{NF}$	50866.582	100.000	50677.829	51087.124	98.176
N	$\mathcal{RW}$	48771.353	95.881	48653.596	48936.619	60.500
N	$\mathcal{H}_1$	48774.005	95.886	48658.002	48937.930	60.433
N	$\mathcal{H}_2$	48789.317	95.916	48665.956	48950.913	61.582
N	$\mathcal{H}_3$	48826.519	95.989	48703.350	48974.728	60.341

TABLE VI
EXPERIMENTAL RESULTS FOR POWER-LAW GRAPHS WITH $\lambda = 2.1$ AND AVERAGE DEGREE 10

A	B	Mean	Perc. of best	Minimum	Maximum	σ
10^4	$\mathcal{F}$	9297.479	100.000	8744.323	9557.168	141.577
10^4	$\mathcal{NF}$	7842.801	84.354	7691.151	8054.712	67.750
10^4	$\mathcal{RW}$	7786.536	83.749	7638.957	7992.578	65.924
10^4	$\mathcal{H}_1$	7786.141	83.745	7638.686	7992.406	65.904
10^4	$\mathcal{H}_2$	7793.210	83.821	7647.361	7997.023	65.140
10^4	$\mathcal{H}_3$	7891.596	84.879	7763.210	8073.036	57.312
N	$\mathcal{F}$	48126.982	95.419	45706.950	50530.164	958.234
N	$\mathcal{NF}$	50295.169	99.718	49877.013	50867.788	195.987
N	$\mathcal{RW}$	50409.017	99.944	50077.339	50897.000	164.873
N	$\mathcal{H}_1$	50408.670	99.943	50076.913	50896.481	164.818
N	$\mathcal{H}_2$	50408.912	99.944	50077.948	50895.982	164.386
N	$\mathcal{H}_3$	50437.379	100.000	50114.404	50917.094	161.747

TABLE VII
EXPERIMENTAL RESULTS FOR POWER-LAW GRAPHS WITH $\lambda = 2.7$ AND AVERAGE DEGREE 5

A	B	Mean	Perc. of best	Minimum	Maximum	σ
5^5	$\mathcal{F}$	3011.315	99.738	3005.965	3022.554	3.308
5^5	$\mathcal{NF}$	3019.211	100.000	3008.989	3026.991	3.596
5^5	$\mathcal{RW}$	2980.109	98.705	2969.928	2987.853	3.372
5^5	$\mathcal{H}_1$	2980.191	98.708	2969.948	2987.954	3.377
5^5	$\mathcal{H}_2$	2982.111	98.771	2972.243	2989.603	3.276
5^5	$\mathcal{H}_3$	3003.378	99.476	2997.927	3008.582	2.309
5^6	$\mathcal{F}$	12793.147	93.138	12475.974	13190.481	153.404
5^6	$\mathcal{NF}$	13735.759	100.000	13679.750	13781.684	20.349
5^6	$\mathcal{RW}$	13534.701	98.536	13478.217	13580.439	20.018
5^6	$\mathcal{H}_1$	13534.736	98.537	13478.076	13580.664	20.042
5^6	$\mathcal{H}_2$	13538.208	98.562	13482.125	13583.686	19.938
5^6	$\mathcal{H}_3$	13567.524	98.775	13518.301	13609.670	18.139
N	$\mathcal{F}$	46328.091	81.009	44077.817	49146.477	1150.080
N	$\mathcal{NF}$	57188.476	100.000	57081.934	57304.162	45.304
N	$\mathcal{RW}$	56648.461	99.056	56526.905	56767.528	51.876
N	$\mathcal{H}_1$	56648.330	99.055	56527.328	56767.581	51.869
N	$\mathcal{H}_2$	56650.912	99.060	56530.570	56770.165	51.712
N	$\mathcal{H}_3$	56678.879	99.109	56563.200	56795.721	50.489

TABLE VIII
EXPERIMENTAL RESULTS FOR POWER-LAW GRAPHS WITH $\lambda = 2.7$ AND AVERAGE DEGREE 10

A	B	Mean	Perc. of best	Minimum	Maximum	σ
10^4	$\mathcal{F}$	9124.481	99.661	9068.833	9168.929	21.113
10^4	$\mathcal{NF}$	9128.352	99.704	9093.659	9156.773	12.420
10^4	$\mathcal{RW}$	9125.727	99.675	9091.333	9153.804	12.458
10^4	$\mathcal{H}_1$	9125.678	99.675	9091.241	9153.888	12.460
10^4	$\mathcal{H}_2$	9127.180	99.691	9092.927	9155.113	12.398
10^4	$\mathcal{H}_3$	9155.475	100.000	9125.139	9180.442	10.894
N	$\mathcal{F}$	50858.047	87.630	49452.064	52578.884	758.606
N	$\mathcal{NF}$	58020.466	99.971	57883.655	58121.120	49.104
N	$\mathcal{RW}$	58005.910	99.946	57870.928	58105.437	49.140
N	$\mathcal{H}_1$	58005.785	99.946	57870.597	58104.735	49.115
N	$\mathcal{H}_2$	58006.970	99.948	57872.266	58106.013	49.120
N	$\mathcal{H}_3$	58037.065	100.000	57904.662	58134.364	48.071

TABLE IX
EXPERIMENTAL RESULTS FOR CLUSTERED GRAPHS THAT HAVE CLIQUES AS THEIR CLUSTERS

A	B	Mean	Perc. of best	Minimum	Maximum	σ
5^5	$\mathcal{F}$	100.030	56.911	100.030	100.030	0.000
5^5	$\mathcal{NF}$	102.935	58.563	102.870	103.034	0.030
5^5	$\mathcal{RW}$	175.767	100.000	175.311	176.331	0.211
5^5	$\mathcal{H}_1$	157.599	89.663	157.261	158.017	0.174
5^5	$\mathcal{H}_2$	112.824	64.190	112.701	112.938	0.049
5^5	$\mathcal{H}_3$	101.041	57.486	101.032	101.054	0.004
5^6	$\mathcal{F}$	106.030	25.259	106.030	106.030	0.000
5^6	$\mathcal{NF}$	113.059	26.934	112.899	113.206	0.070
5^6	$\mathcal{RW}$	419.766	100.000	418.520	420.919	0.520
5^6	$\mathcal{H}_1$	366.124	87.221	365.302	367.044	0.353
5^6	$\mathcal{H}_2$	263.615	62.800	263.009	264.106	0.222
5^6	$\mathcal{H}_3$	129.970	30.962	129.859	130.110	0.047
N	$\mathcal{F}$	106.030	6.147	106.030	106.030	0.000
N	$\mathcal{NF}$	165.639	9.603	165.320	165.956	0.136
N	$\mathcal{RW}$	1724.959	100.000	1718.010	1730.641	2.469
N	$\mathcal{H}_1$	1268.211	73.521	1263.732	1271.687	1.674
N	$\mathcal{H}_2$	614.545	35.627	613.530	615.662	0.434
N	$\mathcal{H}_3$	418.431	24.257	418.201	418.762	0.115

TABLE X
EXPERIMENTAL RESULTS FOR CLUSTERED GRAPHS THAT HAVE $G_{100,1/2}$ AS THEIR CLUSTERS

A	B	Mean	Perc. of best	Minimum	Maximum	σ
5^5	$\mathcal{F}$	103.031	43.716	103.004	103.058	0.011
5^5	$\mathcal{NF}$	105.593	44.803	105.485	105.718	0.047
5^5	$\mathcal{RW}$	235.685	100.000	234.990	236.534	0.324
5^5	$\mathcal{H}_1$	203.065	86.159	202.585	203.573	0.221
5^5	$\mathcal{H}_2$	124.482	52.817	124.381	124.638	0.051
5^5	$\mathcal{H}_3$	101.830	43.206	101.819	101.842	0.005
5^6	$\mathcal{F}$	104.499	15.740	104.481	104.517	0.008
5^6	$\mathcal{NF}$	123.664	18.626	123.361	123.955	0.103
5^6	$\mathcal{RW}$	663.923	100.000	662.021	665.353	0.709
5^6	$\mathcal{H}_1$	530.968	79.974	530.004	532.036	0.457
5^6	$\mathcal{H}_2$	349.951	52.710	349.500	350.451	0.191
5^6	$\mathcal{H}_3$	154.479	23.268	154.316	154.690	0.073
N	$\mathcal{F}$	107.441	3.407	107.231	107.754	0.106
N	$\mathcal{NF}$	213.300	6.763	212.950	213.704	0.158
N	$\mathcal{RW}$	3153.820	100.000	3138.785	3166.954	5.562
N	$\mathcal{H}_1$	2349.913	74.510	2337.288	2359.542	4.724
N	$\mathcal{H}_2$	951.127	30.158	948.444	953.171	0.999
N	$\mathcal{H}_3$	497.740	15.782	497.048	498.619	0.318

TABLE XI
EXPERIMENTAL RESULTS FOR CLUSTERED GRAPHS THAT HAVE $G_{100,1/5}$ AS THEIR CLUSTERS

A	B	Mean	Perc. of best	Minimum	Maximum	σ
5^5	$\mathcal{F}$	103.110	27.427	103.082	103.137	0.013
5^5	$\mathcal{NF}$	112.424	29.904	112.263	112.648	0.073
5^5	$\mathcal{RW}$	375.948	100.000	375.029	377.084	0.453
5^5	$\mathcal{H}_1$	303.682	80.778	302.964	304.491	0.282
5^5	$\mathcal{H}_2$	153.582	40.852	153.335	153.787	0.098
5^5	$\mathcal{H}_3$	107.919	28.706	107.820	107.998	0.036
5^6	$\mathcal{F}$	125.272	9.616	124.761	125.789	0.226
5^6	$\mathcal{NF}$	150.315	11.538	149.833	150.693	0.186
5^6	$\mathcal{RW}$	1302.779	100.000	1298.263	1306.771	1.798
5^6	$\mathcal{H}_1$	973.049	74.690	970.787	975.678	1.110
5^6	$\mathcal{H}_2$	506.337	38.866	505.410	507.813	0.446
5^6	$\mathcal{H}_3$	225.869	17.337	225.381	226.277	0.180
N	$\mathcal{F}$	170.164	2.397	169.353	171.078	0.408
N	$\mathcal{NF}$	300.069	4.227	299.385	300.726	0.274
N	$\mathcal{RW}$	7098.130	100.000	7063.073	7121.211	12.755
N	$\mathcal{H}_1$	5816.386	81.943	5773.399	5849.654	16.886
N	$\mathcal{H}_2$	1904.744	26.834	1895.577	1911.731	4.032
N	$\mathcal{H}_3$	809.288	11.401	806.836	812.015	1.328

to discover a node. For power-law graphs and large number of queries flooding does not have good performance and our hybrid algorithm outperforms it. For clustered topologies flooding and normalized flooding are ineffective and random walk has the best performance. The hybrid algorithms' behaviour mediates between flooding and random walk.

TABLE XII
PERCENTAGE OF DISCOVERED NODES BY THE k-WALKERS ALGORITHM TO THE HYBRID ALGORITHM FOR POWER-LAW GRAPHS

Parameters	Mean	Minimum	Maximum	σ
$\lambda = 2.1$, average degree 5	75.469	3.490	100.000	33.948
$\lambda = 2.1$, average degree 10	77.399	3.026	100.000	32.857
$\lambda = 2.7$, average degree 5	71.490	0.146	100.000	38.505
$\lambda = 2.7$, average degree 10	75.945	0.448	100.000	35.159

TABLE XIII
PERCENTAGE OF DISCOVERED NODES BY THE k-WALKERS ALGORITHM TO THE HYBRID ALGORITHM FOR CLUSTERED TOPOLOGIES

Clusters	Mean	Minimum	Maximum	σ
$G_{100,1/5}$	99.828	98.638	100.173	0.407
$G_{100,1/2}$	99.970	99.689	100.199	0.110
Cliques	99.966	99.699	100.198	0.114

D. Hybrid Algorithm vs. k-Walkers

Our proposed hybrid algorithm uses flooding to find k distinct outer nodes and then starts k independent walkers from those nodes. An alternative is to initiate k independent walkers from the origin, i.e., the k-walkers algorithm. Intuitively, the walkers have less overlap in the hybrid algorithm. Therefore the hybrid algorithm increases the dispersion.

We conducted several experiments to compare the performance of these two algorithms on power-law and clustered topologies. We considered different number of walkers and compared the percentage of discovered nodes by the two algorithms with the same number of messages. For power-law graphs, the hybrid algorithm outperforms k-walkers most of the time. We considered 158 different values for k from 1 to 30000. Table XII shows the average, minimum, maximum, and standard deviation of the percentage of nodes discovered by the k-walkers algorithm to the hybrid algorithm for power-law graphs.

According to these results, the k-walkers algorithm never outperforms the hybrid algorithm and on average discovers around 3/4 as many nodes as the hybrid algorithm. The minimum percentage for all these settings is achieved for $k = 25500$. For $k = 1$, the algorithms are equivalent and the percentage is 100. For large number of walkers, the hybrid algorithm is much more effective than the k-walkers algorithm.

For the clustered topologies, they have similar performance and each of them outperforms the other on some settings. We considered 69 distinct values for k from 1 to 1000. The corresponding experimental results are shown in Table XIII. On average, the hybrid algorithm outperforms the k-walkers algorithm, but their performance is very close.

IV. CONCLUSIONS

In this paper we analyzed the performance of several search algorithms on unstructured peer-to-peer networks. We introduced a new hybrid algorithm that combines flooding with random walk. It uses flooding to find sufficient, say k, outer nodes and then starts k independent random walks from those nodes. We compared this algorithm with classic search algorithms such as flooding and random walk, on binomial

random graphs, regular random graphs, power-law graphs, and clustered topologies. These are the common topologies for peer-to-peer networks. Our experiments showed that

- For binomial random graphs and regular random graphs, all algorithms have similar behaviour.
- For power-law graphs, flooding is effective for small number of messages, but for large number of messages our hybrid algorithm outperforms it.
- Flooding is ineffective for clustered topologies in which random walk is the best algorithm. For these topologies, our hybrid algorithm provides a compromise between flooding and random walk.

We also compared the proposed hybrid algorithm with the k-walker algorithm on power-law and clustered topologies. The k-walker algorithm starts k independent walks from the origin. Our experiments showed that while they have similar performance on clustered topologies, the hybrid algorithm has much better performance on power-law graphs.

We did not compute the average time each algorithm takes to discover nodes. This is the main future work of this paper. We expect that our proposed hybrid algorithm has small average discovery time. This is because if the requested node is close to the origin then the initial flooding will discover it. Otherwise, it is far from the origin and one of the random walkers will reach it.

V. Acknowledgments

Part of this work was done while the second author was visiting the Cesarea Rothschild Institute at the University of Haifa, under the auspices of the University of Waterloo-Haifa International Experience Program.

This work was made possible by the facilities of

- the Shared Hierarchical Academic Research Computing Network SHARCNET (www.sharcnet.ca): 8,082 CPUs,
- the Atlantic Computational Excellence Network ACEnet (www.ace-net.ca): 412 CPUs,
- the Department of Combinatorics and Optimization, University of Waterloo (www.math.uwaterloo.ca/CandO_Dept): 80 CPUs.

References

[BKM+00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. In *Proceedings of the 9th International World Wide Web Conference*, pages 309–320, 2000.

[Bon04] A. Bonato. A survey of models of the web graph. In *Proceedings of the 1st Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN '04)*, pages 159–172, 2004.

[CS02] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. *Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '02)*, 32(4):177–190, 2002.

[FFF99] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM '99)*, pages 251–262, 1999.

[GMS04] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, volume 1, pages 120–130, 2004.

[GMS05] C. Gkantsidis, M. Mihail, and A. Saberi. Hybrid search schemes for unstructured peer-to-peer networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, volume 3, pages 1526–1537, 2005.

[JŁR00] S. Janson, T. Łuczak, and A. Ruciński. *Random Graphs*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 2000.

[LCC+02] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured Peer-to-Peer networks. In *Proceedings of the 2002 International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS-02)*, volume 30, 1, pages 258–259, 2002.

[ŁP06] T. Łuczak and P. Prałat. Protean graphs. *Internet Mathematics*, 3(1):21–40, 2006.

[Pra06] P. Prałat. A note on the diameter of protean graphs, 2006. submitted.

[PW] P. Prałat and N. Wormald. Growing protean graphs. *Internet Mathematics*. to appear.

[RW92] A. Ruciński and N. C. Wormald. Random graph processes with degree restrictions. *Combinatorics, Probability & Computing*, 1:169–180, 1992.

[Wor99] N. C. Wormald. Models of random regular graphs. In *Surveys in Combinatorics, 1993, Walker (Ed.), London Mathematical Society Lecture Note Series 187, Cambridge University Press*, volume 276, pages 239–298. 1999.