Search in power-law networks

Lada A. Adamic,[1,*] Rajan M. Lukose,[1,†] Amit R. Puniyani,[2,‡] and Bernardo A. Huberman[1,§]
[1]*HP Labs, Palo Alto, California 94304*
[2]*Department of Physics, Stanford University, 382 Via Pueblo Mall, Stanford, California 94305*
(Received 29 April 2001; revised manuscript received 22 June 2001; published 26 September 2001)

Many communication and social networks have power-law link distributions, containing a few nodes that have a very high degree and many with low degree. The high connectivity nodes play the important role of hubs in communication and networking, a fact that can be exploited when designing efficient search algorithms. We introduce a number of local search strategies that utilize high degree nodes in power-law graphs and that have costs scaling sublinearly with the size of the graph. We also demonstrate the utility of these strategies on the GNUTELLA peer-to-peer network.

DOI: 10.1103/PhysRevE.64.046135 PACS number(s): 89.75.Fb, 05.50.+q, 05.40.−a, 89.75.Da

I. INTRODUCTION

A number of large distributed systems, ranging from social [1] to communication [2] to biological networks [3] display a power-law distribution in their node degree. This distribution reflects the existence of a few nodes with very high degree and many with low degree, a feature not found in standard random graphs [4]. A large-scale illustration of such a network is given by the AT&T call graph. A call graph is a graph representation of telephone traffic on a given day in which nodes represent people and links the phone calls among them. As shown by [1], the out-link degree distribution for a massive graph of telephone calls between individuals has a clean power-law form with an exponent of approximately 2.1. The same distribution is obtained for the case of in links. This power law in the link distribution reflects the presence of central individuals who interact with many others on a daily basis and play a key role in relaying information.

While recent work has concentrated on the properties of these power-law networks and how they are dynamically generated [5–7], there remains the interesting problem of finding efficient algorithms for searching within these particular kinds of graphs. Recently, Kleinberg [8] studied search algorithms in a graph where nodes are placed on a two-dimensional (2D) lattice and each node has a fixed number of links whose placement is correlated with lattice distance to the other nodes. Under a specific form of the correlation, an algorithm with knowledge of the target's location can find the target in polylogarithmic time.

In the most general distributed search context however, one may have very little information about the location of the target. Increasingly a number of pervasive electronic networks, both wired and wireless, make geographic location less relevant. A particularly interesting example is provided by the recent emergence of peer-to-peer networks, which have gained enormous popularity with users wanting to share their computer files. In such networks, the name of the target file may be known, but due to the network's *ad hoc* nature, the node holding the file is not known until a real-time search is performed. In contrast to the scenario considered by Kleinberg, there is no global information about the position of the target, and hence it is not possible to determine whether a step is a move towards or away from the target. One simple way to locate files, implemented by NAPSTER, is to use a central server that contains an index of all the files every node is sharing as they join the network. This is the equivalent of having a giant white pages for the entire United States. Such directories now exist online, and have in a sense reduced the need to find people by passing messages. But for various reasons, including privacy and copyright issues, in a peer-to-peer network it is not always desirable to have a central server.

File-sharing systems that do not have a central server include GNUTELLA and FREENET. Files are found by forwarding queries to one's neighbors until the target is found. Recent measurements of GNUTELLA networks [9] and simulated FREENET networks [10] show that they have power-law degree distributions. In this paper, we propose a number of message-passing algorithms that can be efficiently used to search through power-law networks such as GNUTELLA. Like the networks that they are designed for, these algorithms are completely decentralized and exploit the power-law link distribution in the node degree. The algorithms use local information such as the identities and connectedness of a node's neighbors, and its neighbors' neighbors, but not the target's global position. We demonstrate that our search algorithms work well on real GNUTELLA networks, scale sublinearly with the number of nodes, and may help reduce the network search traffic that tends to cripple such networks.

The paper is organized as follows. In Sec. II, we present analytical results on message passing in power-law graphs, followed by simulation results in Sec. III. Section IV compares the results with Poisson random graphs. In Sec. V we consider the application of our algorithms to GNUTELLA, and Sec. VI concludes.

*Email address: ladamic@hpl.hp.com
†Email address: lukose@hpl.hp.com
‡Email address: amit8@stanford.edu
§Email address: huberman@hpl.hp.com

II. SEARCH IN POWER-LAW RANDOM GRAPHS

In this section we use the generating function formalism introduced by Newman [7] for graphs with arbitrary degree distributions to analytically characterize search-cost scaling in power-law graphs.

A. Random Walk Search

Let $G_0(x)$ be the generating function for the distribution of the vertex degrees k. Then

$$G_0(x)=\sum_0^\infty p_k x^k, \tag{1}$$

where p_k is the probability that a randomly chosen vertex on the graph has degree k.

For a graph with a power-law distribution with exponent τ, minimum degree $k=1$ and an abrupt cutoff at $m=k_{max}$, the generating function is given by

$$G_0(x)=c\sum_1^m k^{-\tau}x^k \tag{2}$$

with c a normalization constant that depends on m and τ to satisfy the normalization requirement

$$G_0(1)=c\sum_1^m k^{-\tau}=1. \tag{3}$$

The average degree of a randomly chosen vertex is given by

$$\langle k\rangle=\sum_1^m kp_k=G_0'(1). \tag{4}$$

Note that the average degree of a vertex chosen at random and one arrived at by following a random edge are different. A random edge arrives at a vertex with probability proportional to the degree of the vertex, i.e., $p'(k)\sim kp_k$. The correctly normalized distribution is given by

$$\frac{\sum_k kp_kx^k}{\sum_k kp_k}=x\frac{G_0'(x)}{G_0'(1)}. \tag{5}$$

If we want to consider the number of outgoing edges from the vertex we arrived at, but not include the edge we just came on, we need to divide by one power of x. Hence the number of new neighbors encountered on each step of a random walk is given by the generating function

$$G_1(x)=\frac{G_0'(x)}{G_0'(1)}, \tag{6}$$

where $G_0'(1)$ is the average degree of a randomly chosen vertex as mentioned previously.

In real social networks, it is reasonable that one one would have at least some knowledge of one's friends' friends. Hence, we now compute the distribution of second neighbors. The probability that any of the second neighbors connect to any of the first neighbors or to one another goes as N^{-1} and can be ignored in the limit of large N. Therefore, the distribution of the second neighbors of the original randomly chosen vertex is determined by

$$\sum_k p_k[G_1(x)]^k=G_0(G_1(x)). \tag{7}$$

It follows that the average number of second neighbors is given by

$$z_{2A}=\left[\frac{\partial}{\partial x}G_0(G_1(x))\right]_{x=1}=G_0'(1)G_1'(1). \tag{8}$$

Similarly, if the original vertex was not chosen at random, but arrived at by following a random edge, then the number of second neighbors would be given by

$$z_{2B}=\left[\frac{\partial}{\partial x}G_1(G_1(x))\right]_{x=1}=[G_1'(1)]^2. \tag{9}$$

In both Eqs. (8) and (9) the fact that $G_1(1)=1$ was used.

Both these expressions depend on the values $G_0'(1)$ and $G_1'(1)$ so we calculate those for given τ and m. For simplicity and relevance to most real-world networks of interest we assume $2<\tau<3$,

$$G_0'(1)=\sum_1^m ck^{1-\tau}\sim\int_1^m x^{\tau-1}dx=\frac{1}{\tau-2}(1-m^{2-\tau}), \tag{10}$$

$$G_1'(1)=\frac{1}{G_0'(1)}\frac{\partial}{\partial x}\sum_1^m ck^{1-\tau}x^{k-1} \tag{11}$$

$$=\frac{1}{G_0'(1)}\sum_2^m ck^{1-\tau}(k-1)x^{k-2} \tag{12}$$

$$\sim\frac{1}{G_0'(1)}\frac{m^{3-\tau}(\tau-2)-2^{2-\tau}(\tau-1)+m^{2-\tau}(3-\tau)}{(\tau-2)(3-\tau)} \tag{13}$$

for large cutoff values m. Now we impose the cutoff of Aiello *et al.* [1] at $m\sim N^{1/\tau}$. Since m scales with the size of the graph N and for $2<\tau<3$ the exponent $2-\tau$ is negative, we can neglect terms constant in m. This leaves

$$G_1'(1)=\frac{1}{G_0'(1)}\frac{m^{3-\tau}}{(3-\tau)}. \tag{14}$$

Substituting into Eq. (8) (the starting node is chosen at random) we obtain

$$z_{2A}=G_0'(1)G_1'(1)\sim m^{3-\tau}. \tag{15}$$

We can also derive z_{2B}, the number of second neighbors encountered as one is doing a random walk on the graph,

$$z_{2B}=[G_1'(1)]^2=\left[\frac{\tau-2}{1-m^{2-\tau}}\frac{m^{3-\tau}}{3-\tau}\right]^2. \tag{16}$$

Letting $m\sim N^{1/\tau}$ as above, we obtain

$$z_{2B}\sim N^{2(3/\tau-1)}. \tag{17}$$

Thus, as the random walk along edges proceeds node to node, each node reveals more of the graph since it has information not only about itself, but also of its neighborhood. The search cost s is defined as the number of steps until approximately the whole graph is revealed so that $s\sim N/z_{2B}$, or

$$s\sim N^{3(1-2/\tau)}. \tag{18}$$

In the limit $\tau\to 2$, Eq. (16) becomes

$$z_{2B}\sim\frac{N}{\ln^2(N)} \tag{19}$$

and the scaling of the number of steps required is

$$s\sim\ln^2(N). \tag{20}$$

B. Search utilizing high degree nodes

Random walks in power-law networks naturally gravitate towards the high degree nodes, but an even better scaling is achieved by intentionally choosing high degree nodes. For τ sufficiently close to 2 one can walk down the degree sequence, visiting the node with the highest degree, followed by a node of the next highest degree, etc. Let $m-a$ be the degree of the last node we need to visit in order to scan a certain fraction of the graph. We make the self-consistent assumption that $a\ll m$, i.e., the degree of the node has not dropped too much by the time we have scanned a fraction of the graph. Then the number of first neighbors scanned is given by

$$z_{1D}=\int_{m-a}^{m}Nk^{1-\tau}dk\sim Nam^{1-\tau}. \tag{21}$$

The number of nodes having degree between $m-a$ and m, or equivalently, the number of steps taken is given by $\int_{m-a}^{m}k^{-\tau}\sim a$. The number of second neighbors when one follows the degree sequence is given by

$$z_{1D}*G_1'(1)\sim Nam^{2(2-\tau)}, \tag{22}$$

which gives the number of steps required as

$$s\sim m^{2(\tau-2)}\sim N^{2-4/\tau}. \tag{23}$$

We now consider when and why it is possible to go down the degree sequence. We start with the fact that the original degree distribution is a power law

$$p(x)=\left(\sum_1^m x^{-\tau}\right)^{-1}x^{-\tau}, \tag{24}$$

where $m=N^{1/\tau}$ is the maximum degree. A node chosen by following a random link in the graph will have its remaining outgoing edges distributed according to

$$p'(x)=\left[\sum_0^{m-1}(x+1)^{(1-\tau)}\right]^{-1}(x+1)^{(1-\tau)}. \tag{25}$$

At each step one can choose the highest degree node among the n neighbors. The expected number of the outgoing edges of that node can be computed as follows. In general, the cumulative distribution (CDF) $P_{max}(x,n)$ of the maximum of n random variables can be expressed in terms of the CDF $P(x)=\int_0^x p(x')dx'$ of those random variables: $P_{max}(x,n)=P(x)^n$. This yields

$$p'_{max}(x,n)=n(1+x)^{1-\tau}(\tau-2)[1-(x+1)^{2-\tau}]^{n-1}\times(1-N^{2/\tau-1})^{-n} \tag{26}$$

for the distribution of the number of links the richest neighbor among n neighbors has.

Finally, the expected degree of the richest node among n is given by

$$E[x_{max}(n)]=\sum_0^{m-1}xp'_{max}(x,n). \tag{27}$$

We numerically integrated the above equation to derive the ratio between the degree of a node and the expected degree of its richest neighbor. The ratio is plotted in Fig. 1. For a range of exponents and node degrees, the expected degree of the richest neighbor is higher than the degree of the node itself. However, eventually (the precise point depends strongly on the power-law exponent), the probability of finding an even higher degree node in a neighborhood of a very high degree node starts falling.

What this means is that one can approximately follow the degree sequence across the entire graph for a sufficiently small graph or one with a power-law exponent close to 2 $(2.0<\tau<2.3)$. At each step one chooses a node with a degree higher than the current node, quickly finding the highest degree node. Once the highest degree node has been visited, it will be avoided, and a node of approximately second highest degree will be chosen. Effectively, after a short initial climb, one goes down the degree sequence. This is the most efficient way to do this kind of sequential search, visiting highest degree nodes in sequence.

III. SIMULATIONS

We used simulations of a random network with a power-law link distribution of $\tau=2.1$ to validate our analytical results. As in the analysis above, a simple cutoff at $m\sim N^{1/\tau}$ was imposed. The expected number of nodes among N having exactly the cutoff degree is 1. No nodes of degree higher than the cutoff are added to the graph. In real-world graphs

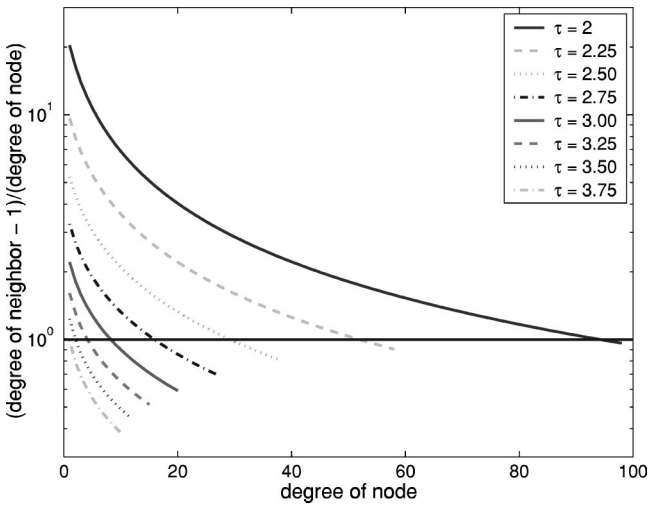

FIG. 1. Ratio r (the expected degree of the richest neighbor of a node whose degree is n divided by n) vs n for τ (top to bottom) $=2.0$, 2.25, 2.5, 2.75, 3.00, 3.25, 3.50, and 3.75. Each curve extends to the cutoff imposed for a 10 000 node graph with the particular exponent.

one, of course, does observe nodes of degree higher than this imposed cutoff, so that our simulations become a worse case scenario. Once the graph is generated, the largest connected component (LCC) is extracted, that is the largest subset of nodes such that any node can be reached from any other node. For $2<\tau<3.48$ a giant connected component exists [1], and all our measurements are performed on the LCC. We observe that the LCC contains the majority of the nodes of the original graph and most of the links as well. The link distribution of the LCC is nearly identical to that of the original graph with a slightly smaller number of 1 and 2 degree nodes.

Next we apply our message-passing algorithm to the network. Two nodes, the source and the target, are selected at random. At each time step the node that has the message passes it on to one of its neighbors. The process ends when the message is passed on to a neighbor of the target. Since each node knows the identity of all of its neighbors, it can pass the message directly to the target if the target happens to be one of it's neighbors. The process is analogous to performing a random walk on a graph, where each node is ''visited'' as it receives the message.

There are several variants of the algorithm, depending on the strategy and the amount of local information available.

(1) The node can pass the message onto one of its neighbors at random or it can avoid passing it on to a node that has already seen the message.

(2) If the node knows the degrees of its neighbors, it can choose to pass the message onto the neighbor with the most neighbors.

(3) The node may know only its neighbors or it may know who its neighbors' neighbors are. In the latter case it would pass the message onto a neighbor of the target.

In order to avoid passing the message to a node that has already seen the message, the message itself must be signed by the nodes as they receive the message. Further, if a node has passed the message and finds that all of its neighbors are already on the list, it puts a special mark next to its name, which means that it is unable to pass the message onto any new node. This is equivalent to marking nodes as follows.

White. Node has not been visited.

Gray. Node has been visited, but all its neighbors have not been visited.

Black. Node and all its neighbors have been visited already.

Here we compare two strategies. The first performs a random walk, where only retracing the last step is disallowed. In the message passing scenario, this means that if Bob just received a message from Jane, he would not return the message to Jane if he could pass it to someone else. The second strategy is a self-avoiding walk that prefers high degree nodes to low degree ones. In each case both the first and second neighbors are scanned at each step.

Figure 2(a) shows the scaling of the average search time with the size of the graph for the two strategies. The scaling (exponent 0.79 for the random walk and 0.70 for the high degree strategy) is not as favorable as in the analytic results derived above (0.14 for the random walk and 0.1 for the high degree strategy when $\tau=2.1$) .

Consider, on the other hand, the number of steps it takes to cover half the graph. For this measure we observe a scaling that is much closer to the ideal. As shown in Figure 2(b), the cover time scales as $N^{0.37}$ for the random walk strategy vs $N^{0.15}$ from Eq. (18). Similarly, the high degree strategy cover time scales as $N^{0.24}$ vs $N^{0.1}$ in Eq. (23).

The difference in the value of the scaling exponents of the cover time and average search time implies that a majority of nodes can be found very efficiently, but others demand high search costs. As Figure 2(c) shows, a large portion of the 10 000 node graph is covered within the first few steps, but some nodes take as many steps or more to find as there are nodes in total. For example, the high degree seeking strategy finds about 50% of the nodes within the first 10 steps (meaning that it would take about $10+2=12$ hops to reach 50% of the graph). However, the skewness of the search time distribution brings the average number of steps needed to 217.

Some nodes take a long time to find because the random walk, after a brief initial period of exploring fresh nodes, tends to revisit nodes. It is a well-known result that the stationary distribution of a random walk on an undirected graph is simply proportional to the distribution of links emanating from a node. Thus, nodes with high degree are often revisited in a walk.

A high degree seeking random walk is an improvement over the random walk, but still cannot avoid retracing its steps. Figure 2(d) shows the color of nodes visited on such a walk for a $N=1000$ node power-law graph with exponent 2.1 and an abrupt cutoff at $N^{1/2.1}$. The number of nodes of each color encountered in 50-step segments is recorded in the bar for that time period. We observe that the self-avoiding strategy is somewhat effective, with the total number of steps needed to cover the graph about 13 times smaller than the pure random walk case, and the fraction of visits to gray and black nodes is significantly reduced.

Although the revisiting of nodes modifies the scaling be-

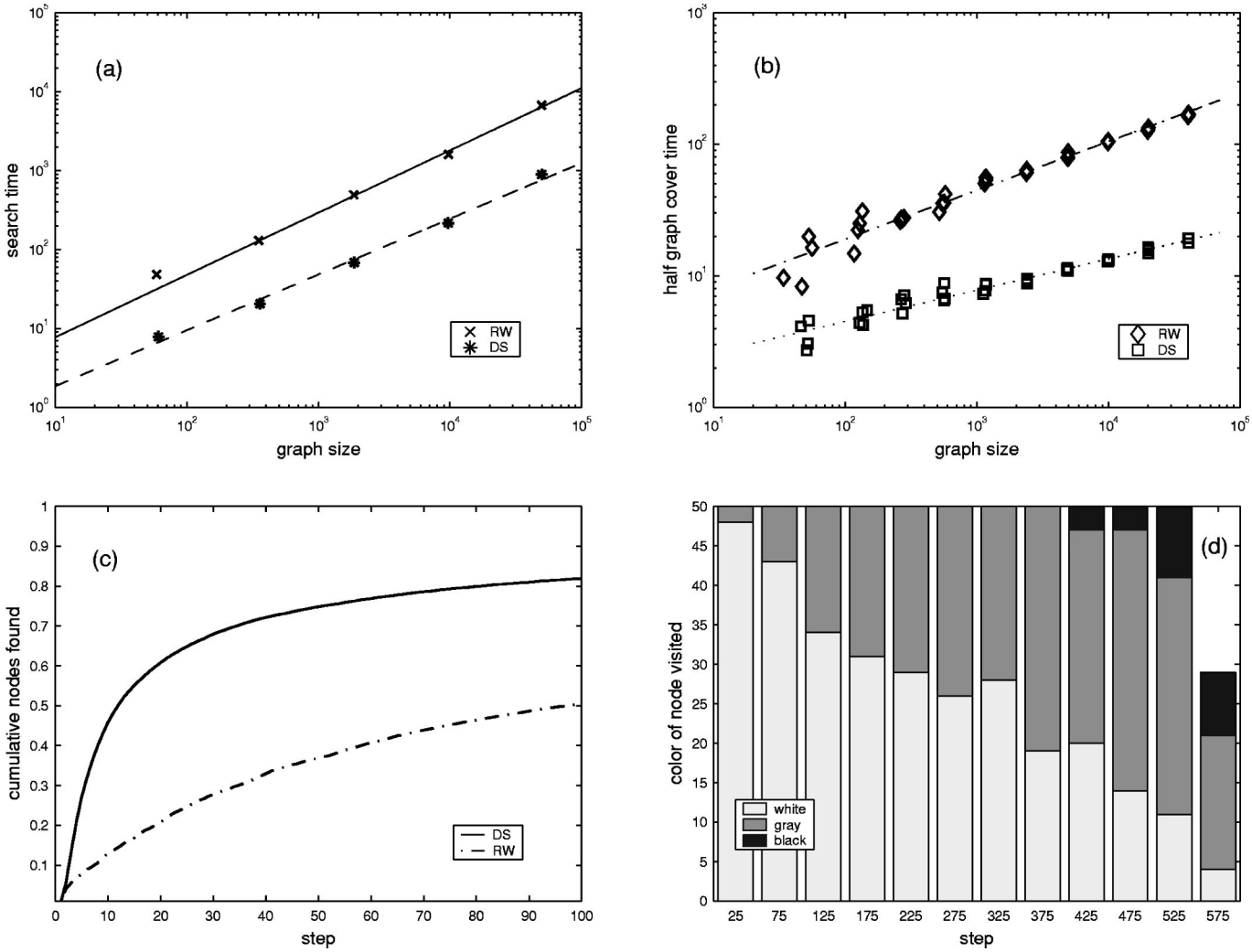

FIG. 2. (a) Scaling of the average node-to-node search cost in a random power-law graph with exponent 2.1, for random walk (RW) and high-degree seeking (DS) strategies. The solid line is a fitted scaling exponent of 0.79 for the RW strategy and the dashed is an exponent of 0.70 for the DS strategy. (b) The observed and fitted scaling for half graph cover times for the RW and DS strategies. The fits are to scaling exponents of 0.37 and 0.24, respectively. (c) Cumulative distribution of nodes seen vs the number of steps taken for the RW and DS strategies on a 10 000 node graph. (d) Bar graph of the color of nodes visited in DS search of a random 1000 node power-law graph with exponent 2.1. White represents a fresh node, gray represents a previously visited node that has some unvisited neighbors, and black represents nodes for which all neighbors have been previously visited.

havior, it is the form of the link distribution that is responsible for changes in the scaling. If nodes were uniformly linked, at every step the number of new nodes seen would be proportional to the number of unexplored nodes in the graph. The factor by which the search is slowed down through revisits would be independent of the size of the graph. Hence, revisiting alone does not account for the difference in scaling.

The reason why the simulated scaling exponents for these search algorithms do not follow the ideal is the same reason why power-law graphs are so well suited to search: the link distribution is extremely uneven. A large number of links point to only a small subset of high degree nodes. When a new node is visited, its links do not let us uniformly sample the graph, they preferentially lead to high degree nodes, which have likely been seen or visited in a previous step. This would not be true of a Poisson graph, where all the links are randomly distributed and hence all nodes have approximately the same degree. We will explore and contrast the search algorithm on a Poisson graph in the following section.

IV. COMPARISON WITH POISSON DISTRIBUTED GRAPHS

In a Poisson random graph with N nodes and z edges, the probability $p=z/N$ of an edge between any two nodes is the same for all nodes. The generating function $G_0(x)$ is given by [7]

$$G_0(x)=e^{z(x-1)}. \tag{28}$$

In this special case $G_0(x)=G_1(x)$, so that the distribution of outgoing edges of a node is the same whether one arrives at the vertex by following a link or picks the node at random.

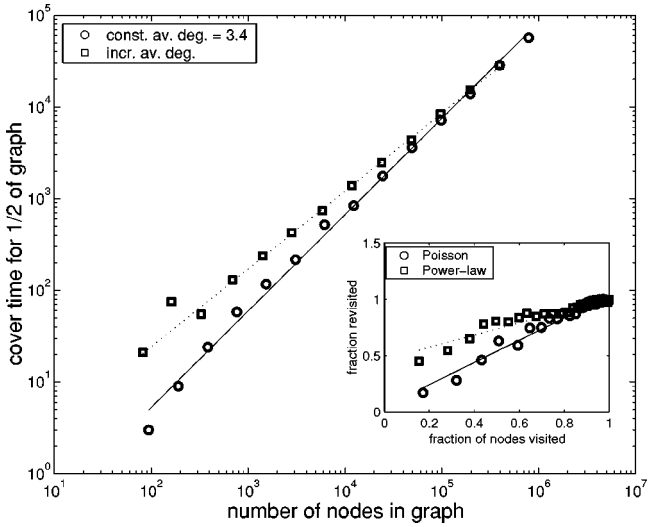

FIG. 3. Squares are scaling of cover time for 1/2 of the graph for a Poisson graph with a constant average degree/node (with fit to a scaling exponent of 1.0). Circles are the scaling for Poisson graphs with the same average degree/node as a power-law graph with exponent 2.1 (with fit to a scaling exponent of 0.85). The inset compares revisitation between search on Poisson vs power-law graphs, as discussed in the text.

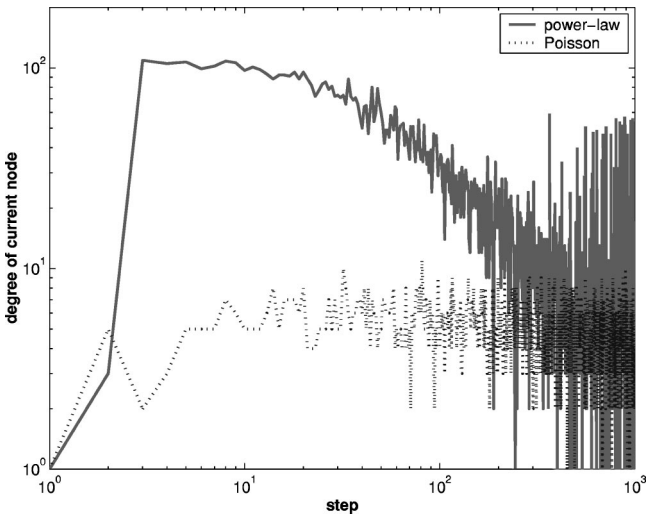

FIG. 4. Degrees of nodes visited in a single search for power law and Poisson graphs of 10 000 nodes.

This makes analysis search in a Poisson random graph particularly simple. The expected number of new links encountered at each step is a constant p. So that the number of steps needed to cover a fraction c of the graph is $s=cN/p$. If p remains constant as the size of the graph increases, the cover time scales linearly with the size of the graph. This has been verified via simulation of the random walk search as shown in Fig. 3.

In our simulations the probability p grows slowly towards its asymptotic value as the size of the graph is increased because of the particular choice of cutoff at $m \sim N^{(1/\tau)}$ for the power-law link distribution. We generated Poisson graphs with the same number of nodes and links for comparison. Within this range of graph sizes, growth in the average number of links per node appears as $N^{0.6}$, making the average number of second neighbors scale as $N^{0.15}$. This means that the scaling of the cover time scales as $N^{0.85}$, as shown in Fig. 3. Note how well the simulation results match the analytical expression. This is because nodes can be approximately sampled in an even fashion by following links.

The reason why the cover time for the Poisson graph matches the analytical prediction and the power-law graph does not is illustrated in Fig. 3 (inset). If links were approximately evenly distributed among the nodes, then if at one point in the search 50% of the graph has already been visited, one would expect to revisit previously seen nodes about 50% of the time. This is indeed the case for the Poisson graph. However, for the power-law graph, when 50% of the graph has been visited, nodes are revisited about 80% of the time, which implies that the same high degree nodes are being revisited before new low degree ones. It is this bias that accounts for the discrepancy between the analytic scaling and the simulated results in the power-law case.

However, even the simulated $N^{0.35}$ scaling for a random, minimally self-avoiding strategy on the power-law graph out performs the ideal $N^{0.85}$ scaling for the Poisson graph. It is also important to note that the the high degree node seeking strategy has a much greater success in the power-law graph because it relies heavily on the fact that the number of links per node varies considerably from node to node. To illustrate this point, we executed the high degree seeking strategy on two graphs, Poisson and power law, with the same number of nodes, and the same exponent $\tau=2$. In the Poisson graph, the variance in the number of links was much smaller, making the high degree node seeking strategy comparatively ineffective as shown in Fig. 4.

In the power-law graph we can start from a randomly chosen node. In this case the starting node has only one link, but two steps later we find ourselves at a node with the highest degree. From there, one approximately follows the degree sequence, that is, the node richest in links, followed by the second richest node, etc. The strategy has allowed us to scan the maximum number of nodes in the minimum number of steps. In comparison, the maximum degree node of the exponential graph is 11, and it is reached only on the 81st step. Even though the two graphs have a comparable number of nodes and edges, the exponential graph does not lend itself to quick search.

V. GNUTELLA

GNUTELLA is a peer-to-peer file-sharing system that treats all client nodes as functionally equivalent and lacks a central server that can store file location information. This is advantageous because it presents no central point of failure. The obvious disadvantage is that the location of files is unknown. When a user wants to download a file, she sends a query to all the nodes within a neighborhood of size ttl, the time to live assigned to the query. Every node passes on the query to all of its neighbors and decrements the ttl by one. In this way, all nodes within a given radius of the requesting node will be queried for the file, and those who have matching files will send back positive answers.

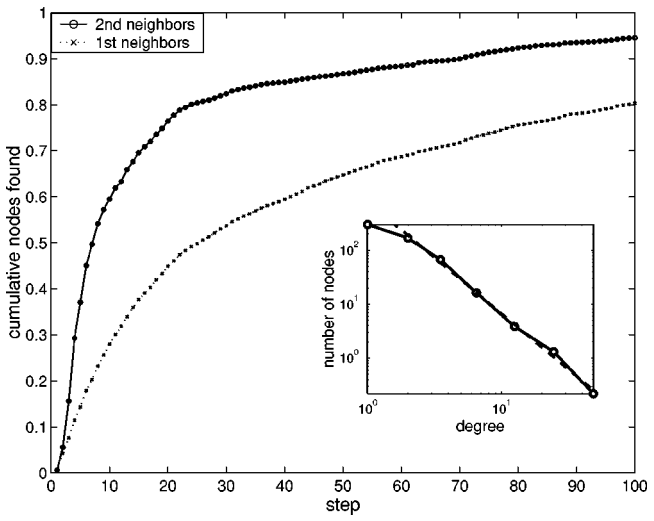

FIG. 5. Cumulative number of nodes found at each step in the GNUTELLA network. The inset shows the measured link distribution of the real GNUTELLA network used in the search simulations and a fit to a power-law link distribution with exponent 2.

This broadcast method will find the target file quickly, given that it is located within a radius of ttl. However, broadcasting is extremely costly in terms of bandwidth. Every node must process queries of all the nodes within a given ttl radius. In essence, if one wants to query a constant fraction of the network, say 50%, as the network grows, each node and network edge will be handling query traffic that is proportional to the total number of nodes in the network.

Such a search strategy does not scale well. As query traffic increases linearly with the size of GNUTELLA graph, nodes become overloaded as was shown in a recent study by the Clip2 company [9]. 56k modems are unable to handle more than 20 queries a second, a threshold easily exceeded by a network of about 1000 nodes. With the 56k nodes failing, the network becomes fragmented, allowing users to query only small section of the network.

The search algorithms described in the previous sections may help ameliorate this problem. Instead of broadcasting a query to a large fraction of the network, a query is only passed onto one node at each step. The search algorithms are likely to be effective because the GNUTELLA network has a power-law connectivity distribution as shown in Fig. 5 (inset).

Typically, a GNUTELLA client wishing to join the network must find the IP address of an initial node to connect to. Currently, *ad hoc* lists of ''good'' GNUTELLA clients exist [9]. It is reasonable to suppose that this *ad hoc* method of growth would bias new nodes to connect preferentially to nodes that are already fairly well connected, since these nodes are more likely to be ''well known.'' Based on models of graph growth [5,6] where the ''rich get richer,'' the power-law connectivity of *ad hoc* peer-to-peer networks may be a fairly general topological feature.

By passing the query to every single node in the network, the GNUTELLA algorithm fails to take advantage of the connectivity distribution. To implement our algorithm the GNUTELLA clients must be modified to keep lists of the files stored by their first and second neighbors have.[1] This information must be passed at least once when a new node joins the network, and it may be necessary to periodically update the information depending on the typical lifetime of nodes in the network. Instead of passing the query to every node, queries are only passed along to the highest degree nodes. The IP numbers of the nodes already queried are appended to the query, and they are avoided.

The modified algorithm places an additional cost on every node, that of keeping track of the filenames of its neighbors' files. Since network connections saturated by query traffic are a major weakness in GNUTELLA, and since computational and storage resources are likely to remain much less expensive than bandwidth, such a tradeoff is readily made. However, now instead of every node having to handle every query, queries are routed only through high connectivity nodes. Since nodes can select the number of connections that they allow, high degree nodes are presumably high bandwidth nodes that can handle the query traffic. The network has in effect created local directories valid within a two link radius. It is resilient to attack because of the lack of a central server. As for power-law networks in general [11], the network is more resilient than Poisson graphs to random node failure, but less resilient to attacks on the high degree nodes.

Figure 5 shows the success of the high degree seeking algorithm on the GNUTELLA network. We simulated the search algorithm on a crawl by Clip2 company of the actual GNUTELLA network of approximately 700 nodes. Assuming that every file is stored on only one node, 50% of the files can be found in eight steps or less. Furthermore, if the file one is seeking is present on multiple nodes, the search will be even faster.

To summarize, the power-law nature of the GNUTELLA graph means that these search algorithms can be effective. As the number of nodes increases, the (already small) number of nodes that will need to be queried increases sublinearly. As long as the high degree nodes are able to carry the traffic, the GNUTELLA network's performance and scalability may improve by using these search strategies.

We also note that even if a network of clients was not power law, a search strategy that possesses knowledge of its neighbors of a network radius greater than two could still improve search. For example, in the Poisson case, the algorithm could attempt to hold more than the contents of a node's first and second neighbors. How efficient this algorithm is on arbitrary network topologies is the subject of future work. Here we have analyzed the naturally occurring power-law topology.

VI. CONCLUSION

In this paper we have shown that local search strategies in power-law graphs have search costs that scale sublinearly

[1]This idea has already been implemented by Clip2 company in a limited way. 56k modem nodes attach to a high bandwidth Reflector node that stores the filenames of the 56k nodes and handles queries on their behalf.

with the size of the graph, a fact that makes them very appealing when dealing with large networks. The most favorable scaling was obtained by using strategies that preferentially utilize the high connectivity nodes in these power-law networks. We also established the utility of these strategies for searching on the GNUTELLA peer-to-peer network.

It may not be coincidental that several large networks are structured in a way that naturally facilitates search. Rather, we find it likely that these networks could have evolved to facilitate search and information distribution. Networks where locating and distributing information, without perfect global information, plays a vital role tend to be power law with exponents favorable to local search.

For example, large social networks, such as the AT&T call graph and the collaboration graph of film actors, have exponents in the range ($\tau = 2.1 - 2.3$) that according to our analysis makes them especially suitable for searching using our simple, local algorithms. Being able to reach remote nodes by following intermediate links allows communication systems and people to get to the resources they need and distribute information within these informal networks. At the social level, our analysis supports the hypothesis that highly connected individuals do a great deal to improve the effectiveness of social networks in terms of access to relevant resources [12].

Furthermore, it has been shown that the Internet backbone has a power-law distribution with exponent values between 2.15 and 2.2 [2], and web page hyperlinks have an exponent of 2.1 [5]. While in the Internet there are other strategies for finding nodes, such as routing tables and search engines, one observes that our proposed strategy is partially used in these systems as well. Packets are routed through highly connected nodes, and users searching for information on the Web turn to highly connected nodes, such as directories and search engines, which can bring them to their desired destinations.

On the other hand, a system such as the power grid of the western United States, which does not serve as a message passing network, has an exponent $\tau \sim 4$ [5]. It would be fairly difficult to pass messages in such a network without a relatively large amount of global information.

ACKNOWLEDGMENT

We would like to thank the Clip2 company for the use of their GNUTELLA crawl data.

[1] W. Aiello, F. Chung, and L. Lu, in *STOC '00, Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, edited by F. Yao (ACM, New York, 2000), pp. 171–180.

[2] M. Faloutsos, P. Faloutsos, and C. Faloutsos, in *SIGCOMM '99, Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, edited by L. Chapin (ACM, New York, 1999), pp. 251–262.

[3] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.-L. Barabasi, Nature (London) **407**, 651 (2000).

[4] P. Erdös and A. Rényi, Publ. Math. Inst. Hung. Acad. Sci., **5**, 17 (1960).

[5] A.-L. Barabasi and R. Albert, Science **286**, 509 (1999).

[6] B.A. Huberman and L.A. Adamic, Nature (London) **401**, 131 (1999).

[7] M.E.J. Newman, S.H. Strogatz, and D.J. Watts, Phys. Rev. E **64**, 026118 (2001).

[8] J.M. Kleinberg, Nature (London) **406**, 845 (2000).

[9] Clip2 Company, Gnutella. http://www.clip2.com/gnutella.html

[10] T. Hong, in *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*, edited by Andy Oram (O'Reilley, Sebastopol, CA, 2001), Chap. 14, pp. 203–241.

[11] R. Albert, H. Jeong, and A.-L. Barabasi, Nature (London) **406**, 378 (2000).

[12] M. Gladwell, *The Tipping Point: How Little Things Can Make a Big Difference* (Little Brown, New York, 2000).