1-2010

Local Algorithms for Finding Interesting Individuals in Large Networks

Michael Brautbar
University of Pennsylvania

Michael J. Kearns
University of Pennsylvania, mkearns@cis.upenn.edu

Recommended Citation

Michael Brautbar and Michael J. Kearns, "Local Algorithms for Finding Interesting Individuals in Large Networks", . January 2010.

Brautbar, M. & Kearns, M., Local Algorithms for Finding Interesting Individuals in Large Networks, *Innovations in Computer Science*, Jan. 2010, http://conference.itcs.tsinghua.edu.cn/ICS2010/content/papers/16.html

Local Algorithms for Finding Interesting Individuals in Large Networks

Abstract

We initiate the study of local, sublinear time algorithms for finding vertices with extreme topological properties — such as high degree or clustering coefficient — in large social or other networks. We introduce a new model, called the Jump and Crawl model, in which algorithms are permitted only two graph operations. The Jump operation returns a randomly chosen vertex, and is meant to model the ability to discover "new" vertices via keyword search in the Web, shared hobbies or interests in social networks such as Facebook, and other mechanisms that may return vertices that are distant from all those currently known. The Crawl operation permits an algorithm to explore the neighbors of any currently known vertex, and has clear analogous in many modern networks. We give both upper and lower bounds in the Jump and Crawl model for the problems of finding vertices of high degree and high clustering coefficient. We consider both arbitrary graphs, and specializations in which some common assumptions are made on the global topology (such as power law degree distributions or generation via preferential attachment). We also examine local algorithms for some related vertex or graph properties, and discuss areas for future investigation.

Disciplines

Computer Sciences

Comments

Brautbar, M. & Kearns, M., Local Algorithms for Finding Interesting Individuals in Large Networks, *Innovations in Computer Science*, Jan. 2010, http://conference.itcs.tsinghua.edu.cn/ICS2010/content/papers/16.html

Local Algorithms for Finding Interesting Individuals in Large Networks

Mickey Brautbar   Michael Kearns
Computer and Information Science, University of Pennsylvania, Philadelphia, 19104, USA
brautbar@cis.upenn.edu   mkearns@cis.upenn.edu

Abstract: We initiate the study of local, sublinear time algorithms for finding vertices with extreme topological properties — such as high degree or clustering coefficient — in large social or other networks. We introduce a new model, called the *Jump and Crawl* model, in which algorithms are permitted only two graph operations. The *Jump* operation returns a randomly chosen vertex, and is meant to model the ability to discover "new" vertices via keyword search in the Web, shared hobbies or interests in social networks such as Facebook, and other mechanisms that may return vertices that are distant from all those currently known. The *Crawl* operation permits an algorithm to explore the neighbors of any currently known vertex, and has clear analogous in many modern networks.

We give both upper and lower bounds in the Jump and Crawl model for the problems of finding vertices of high degree and high clustering coefficient. We consider both arbitrary graphs, and specializations in which some common assumptions are made on the global topology (such as power law degree distributions or generation via preferential attachment). We also examine local algorithms for some related vertex or graph properties, and discuss areas for future investigation.

Keywords: social networks; graph theory; algorithms

1 Introduction

The proliferation of very large social and technological networks over the last decade or so — and the attendant scientific and cultural interest they have attracted — has led to the documentation of certain local topological properties that are now believed to be quite common. Perhaps beginning with earlier sociological interest in *global* structure such as the "six degrees" phenomenon (small diameter) and structural holes, recent research has further identified *local* topological properties, such as individuals with extraordinarily high degree (sometimes dubbed "connectors" or "hubs"), local neighborhoods with a high degree of clustering (fraction of edges present) compared to the overall edge density, vertices of high "centrality" for various definitions of that term, and so on. There is now a compelling dialogue in the literature between empirical works documenting and refining these various notions of extreme individuals and neighborhoods, and theoretical works attempting to explain their persistent emergence via generative models for network formation [11].

Given the presence of such "interesting" individuals in large networks, how would we actually *find* them — especially considering that for many such networks (including the Web, or for non-employee researchers of online social networks such as Facebook), there may not exist an accessible, centralized description of the network? This question is the topic of the current paper, and while there are a few prior works that touch on related topics (see Related Work below), it appears there has been no systematic study of finding extremal vertices from only local operations. In this paper we initiate and partially populate such a study.

We introduce a simple model for local graph exploration that we call the *Jump and Crawl* model. As mentioned in the Abstract and detailed below, this model is meant to capture the two kinds of operations that seem to be commonly available in many modern networks:

- *Crawling.* In many networks, once we are aware of the existence or identity of a vertex, we are also provided with links that allow us to examine any or all of its neighbors. For instance, in the Web we have text hyperlinks allowing us to crawl to neighboring pages. In Facebook (ignoring privacy settings) and other social networks, knowing one user's profile lets us visit those of all their friends.

- *Jumping.* Many modern networks also provide some sort of global search mechanism that permits the discovery of "new" vertices that may be quite distant from all those previously known. Web search lets us enter text phrases and see relevant pages; Facebook's "Friend Finder" and other mechanisms lets one similarly "jump" to new profiles. Obviously in such cases there is clear structure or bias to the vertices returned in response to a query (since they are relevant to the query itself); for simplicity we assume the Jump operation produces a vertex uniformly chosen at random from the entire network. Obviously other distributional assumptions or other Jump mechanisms should be considered in future work.

As the Web, Facebook and other networks are massive and growing, we would like to examine algorithms in the Jump and Crawl model whose running time scales slowly (certainly sublinearly) with the global network size. Within this framework, we examine the problems of finding vertices of high degree, high clustering coefficient, and a number of related properties.

1.1 Summary of Results

In Section 2 we provide nearly tight upper and lower bounds for the problem of approximating the maximum degree vertex in arbitrary graphs; these bounds show a general trade-off between increased Jump and Crawl operations and improved approximation.

Still considering high degree vertices, we then proceed to show that considerably improved upper bounds can be obtained for more specific classes of graphs. For graphs with a power law degree distribution, we prove that a sampling-based algorithm enjoys an improved bound due to the assumed degree structure. For networks generated according to preferential attachment — which have a power law degree distribution in expectation, but also obey additional structural restrictions — we exploit rapid mixing results to obtain further improvements in approximation.

In Section 3 we turn our attention to finding vertices of both high degree and high clustering coefficient (densely connected neighborhood); retaining the high-degree condition prevents trivial solutions such as a triangle of vertices. We provide a general impossibility result and a general approximation algorithm, as well as an improved approximation algorithm for power law networks with some structural assumptions. We conclude by discussing future research directions.

We now more formally define the Jump and Crawl model, mention some related work, and proceed with the technical development.

1.2 The *Jump and Crawl* Model

We assume a graph G of n vertices, with the value of n given [1]. We assume that each vertex is identified by a unique and arbitrary label, with no structure or relationship assumed between the vertex labels and graph connectivity.

Upon visiting a vertex, we learn its label and also the labels of all its neighbors (these are the hyperlinks of the Web or the friends' profiles of Facebook), and nothing more. In a *Jump* operation, we visit a vertex uniformly chosen at random from the n vertices. In a *Crawl* operation, we must first know the label of a given vertex and one of its neighbors in G, upon which we can then visit that neighbor via crawling. Our goal is to study algorithms finding "interesting" individuals in G, as discussed in the Introduction, using only Jump and Crawl operations on G. The total number of such operations is our complexity measure of interest.

A little thought (and our subsequent results) will reveal that in general it is too much to expect algorithms can find truly extremal vertices in sublinear time. For this reason we introduce a natural notion of approximation.

Definition 1 *Given a numerical property P of vertices (such as degree or clustering coefficient), let u^* be the vertex with the maximum value for P. Then if vertex v is such that $P(u^*) \leq k \cdot P(v)$, we say that v is a k-approximation to u^*. A similar definition holds for the minimum.*

[1] In Appendix B we prove that knowledge of n is necessary in general, in that it cannot be approximated sublinearly in $\sqrt{n}$ in the Jump and Crawl model.

1.3 Related Work

While we are not aware of any systematic prior studies of approximating extreme vertex properties from only local operations, there are a number of related works that we now briefly survey. Schank et al. [16] are interested in estimating the average clustering coefficient across the *entire* network (in contrast to our interest in finding *individual* vertices with high clustering, which is not implied by a global approximation). The authors assume a model where Jump queries are allowed and there is a constant time oracle that checks whether two vertices are connected by an edge. Under this model the authors provide a constant time randomized algorithm which computes an unbiased estimator for the average clustering coefficient in the network. The estimator is based towards counting triangles in the network, and thus not appropriate for finding extreme individual vertices.

Eubank et al. [8] are interested in computing statistical properties of social networks, under a model similar to the Jump and Crawl model. The authors provide a general method for estimating the number of pairs of vertices that are at distance i from each other, given that the number of such pairs is at least a constant fraction of all possible pairs. In general their method runs in linear time in the network size, in contrast to our interest in sublinear algorithms. The authors also provide a good estimation of the (again global) average clustering coefficient using a logarithmic query size, based on random sampling of vertices.

Also somewhat related is the large literature on efficient search or message routing in social networks from local information [1, 2, 12], where messages are passed between network vertices in search of a target or destination individual; but again there is no direct interest in explicitly identifying extremal vertices. Similarly, the literature on property testing in large graphs often considers local operations that are somewhat different than Jump and Crawl (such as testing for the presence of an edge between any pair of vertices) but again focuses on global properties such as connectivity rather than extreme vertices.

2 Finding a High Degree Vertex

Given a network on n vertices, denote the maximum degree in the network by d^*. Then given $0 < \beta < 1$, consider the goal of finding a vertex v such that $d^* \leq \text{degree}(v) \cdot n^{1-\beta}$. In this section we provide upper and lower bounds for this problem under a variety of assumptions on the network structure, beginning with no assumptions.

2.1 Arbitrary Networks

Let us first assume we know the value d^* in advance, an assumption we eventually remove. One possible strategy proceeds as follows: If the maximum degree is smaller than $n^{1-\beta}$, then any vertex would provide the necessary approximation. If not, we notice that the expected size of a random sample one has to take in order to see a neighbor of the maximum degree vertex is at most $\frac{n}{d^*}$. We therefore sample about $\frac{n}{d^*}$ random vertices. If one of them has a degree more than $\frac{d^*}{n^{1-\beta}}$ we stop and return its degree. Otherwise, one of these vertices is a neighbor of the maximum degree vertex. This strategy, which we call *FindHighDegreeVertex*, is formalized below. Finally, to remove the assumption that d^* is known, we are only left with simulating the possible values of d^*. This is done with logarithmic overhead via a simple doubling trick.

Theorem 1 (Upper Bound) *For any given $0 < \beta < 1$, algorithm FindHighDegreeVertex uses $n^\beta \log n$ Jump and Crawl queries and approximates the maximum degree to an expected multiplicative factor of $O(n^{1-\beta})$.*

Proof: Without loss of generality one may assume the network size to be a power of two. Next, we may assume that d^*, the highest degree in the network, is known since we may use a simple doubling trick where we may simulate the possible values of d^* in multiplicative intervals of 2, from 1 to n. Algorithm *FindHighDegreeVertex*, given below, finds a $O(n^{1-\beta})$ approximation, with $n^\beta \log n$ queries.

The outer loop of the algorithm (line 5) costs at most $\frac{n}{d^*} \log n$ and the inner loop (line 10) at most $\frac{d^*}{n^{1-\beta}}$. Therefore the total cost is bounded by their product $O(n^\beta \log n)$.

The maximum degree vertex has d^* neighbors. Therefore, the probability of hitting such a neighbor by making a Jump query is $\frac{d^*}{n}$. Therefore, by making $\frac{n}{d^*} \log n$ Jump queries, we hit such a vertex with probability at least $1 - (1 - \frac{d^*}{n})^{\frac{n}{d^*} \log n} \geq 1 - O(\frac{1}{n})$.

A close inspection of the algorithm reveals that by feeding it with a value of d that differ by a factor of two, the approximation value the algorithm returns would change by at most a factor of at two.

As mentioned before, we end by using a simple doubling trick where we simulate all the possible values of d^* as $1, 2, 4, \ldots, \frac{n}{2}, n$. This simulation adds only a multiplicative factor of $\log n$ to the query complexity. By the previous lemma, for one of the simulated values, we will find a vertex of degree at least $d^* \cdot n^{1-\beta}$. ■

We next show that *FindHighDegreeVertex* is optimal (up to logarithmic factors).

Algorithm 1 FindHighDegreeVertex

```
Require: Network $G$, the maximum degree value $d^*$, parameter $0 < \beta < 1$.
Initialize a pointer p to point to an arbitrary vertex.
if $d^* < n^{1-\beta}$ then
    Stop and return the vertex found with one Jump query.
else
    for $\frac{n}{d^*} \log n$ times do
        Make a Jump query. Let $v$ be the vertex found.
        if degree$(v) \geq \frac{d^*}{n^{1-\beta}}$ then
            Stop and return $v$.
        else
            Make degree(v) Crawl queries from $v$ to all of $v$'s neighbors to find the maximum degree neighbor of $v$, call it $u$.
            if degree$(p) >$ degree$(u)$ then
                Set $p = u$
            end if
        end if
    end for
    Output $p$
end if
```

Theorem 2 (Lower Bound) *Let A be an algorithm for approximating the maximum degree property in the Jump and Crawl model. Let $0 < \beta < 1$. Then if A uses at most n^β queries, A approximates the maximum degree to an expected multiplicative approximation of $\Omega\big(n^{1-\beta}\big)$.*

Proof: We shall build, for any given values of n and β, a network $G(n, \beta)$. First set $m = n^{(1-\beta)}$, $k = n - n^{(1-\beta)}$. Denote the set of vertices in the network by $V = \{v_1.v_2, \ldots, v_n\}$. The network $G(n, \beta)$ may be thought of as a concatenation of a line subgraph with a star subgraph; see Figure 1. The line subgraph is made up of the vertices $v_1, v_2, \ldots, v_k$, where two consecutive vertices are connected by an edge. The star subgraph is made of a vertex v_{k+1} connected to m leaf vertices (degree one vertices) $v_{k+2}, v_{k+3}, \ldots, v_n$. The final network $G(n, \beta)$ is created by connecting v_k (the rightmost vertex of the line subgraph) with the hub of the star subgraph, vertex v_{k+1}.

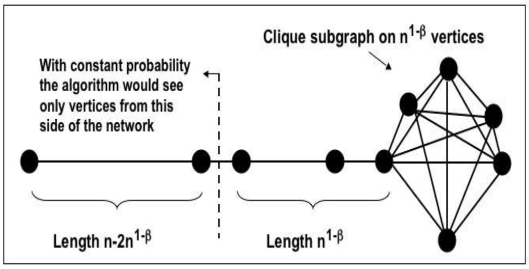
Clique subgraph on n^{1-β} vertices
With constant probability the algorithm would see only vertices from this side of the network
Length n-2n^{1-β}
Length n^{1-β}

Figure 1: The network $G(n, \beta)$

Now set

$$S = \{v_{k-m}, v_{k-m+1}, v_{k-m+2}, \ldots, v_k, \ldots, v_n\}.$$

Clearly, $|S| = 2n^{1-\beta}$. Therefore, using n^β Jump queries algorithm A would fail to sample a vertex from S with probability

$$\left(1 - 2\frac{n^{1-\beta}}{n}\right)^{n^\beta} \approx \frac{1}{e^2}.$$

Therefore, with constant probability, any Jump query will return a degree 2 vertex from $V - S$ (namely, from the "left side" of the line subgraph). Now in order for A to discover the hub vertex it must cross all the "right side" of the line subgraph, namely the vertices in S, which is impossible since the number of queries needed for doing so is more than n^β. Therefore, with constant probability, A would see only degree 2 vertices, while the highest degree is $n^{1-\beta}$ and we are done. We remark that a similar construction to $G(n, \beta)$ would give an analogous lower bound for densely connected graphs. ■

2.2 Power Law Networks

Over the past decade researchers have discovered that the degree distribution of many natural networks resembles a power law. By this it is usually meant that for some constant γ, the fraction of degree d vertices is "close" to $\frac{1}{d^\gamma}$, if d is "large enough". Both "close" and "large enough" are often left unspecified in the literature, but for rigorous statements must be quantified. We thus suggest a simple, rigorous definition of power law networks. Our definition formalizes the above intuition and has the advantage that it treats all degree values in a unifying way.

2.3 Rigorous Definition

We first define a finite power law distribution.

Definition 2 (Finite Power Law Distribution) *Let $m < n$ be positive integers. Let $\gamma > 1$. We say the P is a finite power law distribution, denoted PL(m, n, γ), if:*

I The support of P is the integers between m and n.

II $P(d) = (1/Z)\frac{1}{d^\gamma}$ for $m \leq d \leq m$, where $Z = Z(m, n, \gamma) = \sum_{d=m}^{d=n} \frac{1}{d^\gamma}$.

Definition 3 (Power Law Network) *Let G be a network on n vertices. Let Q be its empirical degree distribution, namely, $Q(d) = \frac{1}{n} \cdot |v \in G : degree(v) = d|$. Let P be a finite power law distribution $P =$ PL(m, n, γ). We say that G is a power law network, denoted $G \in$ Network(m, n, γ), if:*

1. *The support of Q is on the integers between m and n.*
2. *For $m \leq d \leq n$, $|Q(d) - P(d)| \leq \frac{1}{d^{\gamma+1}}$.*

It can be easily shown that many such networks exist.

Two useful properties of power law networks are given below.

Lemma 1 *Let $G \in Network(m, n, \gamma)$ be a power law network. Then the highest degree in the network d^* is upper bounded by $(\frac{n}{Z}(1 + Zo(1)))^{\frac{1}{\gamma}}$.*

Lemma 2 *Let $G \in Network(m, n, \gamma)$ be a power law network with $\gamma > 2$ and let $d \leq \frac{d^*}{2}$, where d^* is the maximum degree in G. Then the fraction of vertices with degree of at least d is $\theta(\frac{1}{Zd^{\gamma-1}})$*

The proof of the lemmas is given in appendix A.

2.4 A Faster Algorithm for Power Laws

We now show that faster algorithms for the maximum degree property exist when the network is a power law network with exponent γ.

Theorem 3 (Upper Bound) *Let $0 < \beta < \frac{\gamma-1}{\gamma}$. Assume $\gamma > 2$. Then algorithm FindHighDegreeVertexOnPowerLaws (see pseudocode) uses $O(n^\beta \log n)$ Jump and Crawl queries and approximates the maximum degree to an expected multiplicative factor of $O\big(n^{\frac{1}{\gamma} - \frac{\beta}{(\gamma-1)}}\big)$.*

Proof: Since $\gamma > 2$ it follows that $1 \leq Z \leq \frac{\pi^2}{6}$ so we may regard Z as a constant. The strategy behind the algorithm is to randomly sample the expected number of vertices needed in order to see a vertex of degree at least $d = n^{\frac{\beta}{(\gamma-1)}}$. The algorithm makes $\Theta(d^{\gamma-1} \log n)$ Jump queries. By Lemma 2 the inverse probability of sampling a vertex of degree at least d is $\Theta(d^{\gamma-1})$. Therefore, by a standard amplification argument, one indeed find such a vertex, with $\Theta(d^{\gamma-1} \log n)$ Jump queries, with probability $1 - O(\frac{1}{n})$. Since the maximum degree in the network is at most $(\frac{n}{Z}(1 + Zo(1))^{\frac{1}{\gamma}}$, the approximation guarantee is $O(\frac{(\frac{n}{Z})^{\frac{1}{\gamma}}}{d}) = O\big(n^{\frac{1}{\gamma} - \frac{\beta}{(\gamma-1)}}\big)$ ■

2.5 Preferential Attachment Networks

In this section we make the further assumption that the unknown network was created by the *preferential attachment* process of Barabasi [3]. In this model, one first fixes an integer parameter $m \geq 1$. Then on each round, a new vertex is added and is connected to m existing (previously added) vertices; the probability the new vertex is connected to existing vertex v is proportional to the (current) degree of v. As was shown by Bollobás et al. [4], asymptotically, the *expected* degree distribution of the network is a power law with exponent

Algorithm 2 FindHighDegreeVertexOnPowerLaws

```
Require: Power law network $G$, the power law network's exponent $\gamma$, parameter $0 < \beta < 1$.
Initialize a pointer p to point to an arbitrary vertex.
for $n^\beta \log n$ times do
    Make a Jump query. Let $v$ be the vertex found.
    if degree($p$) $>$ degree($v$) then
        Let $p = v$
    end if
end for
Output $p$
```

$\gamma = 3$; but it is also known that the actually realized degree sequence may be far from its expectation. However, for small degree values, the degree distribution is close to its expectation[6]. In this sense a preferential attachment network may be seen as a special family of power law networks. Moreover, the highest degree in a typical preferential attachment network is $\sqrt{n}$ [9]. Therefore in order to find a vertex with high degree one can apply the techniques of the upper bound given in the previous section and get a $n^{\frac{1}{2}-\frac{\beta}{2}}$ approximation using a query size of $O(n^\beta \log n)$.

However, due to additional network structure inherent in the preferential attachment process we can do even better, based on the following two facts. Fact 1: a Lazy Random Walk on the preferential attachment network is rapidly mixing (in polylog time in n) to the degree distribution; Fact 2: When sampling the degree distribution, the expected time one has to wait in order to see a vertex of degree at least d is d. These intuitions are formalized in *FindHighDegreeVertexOnPA* below.

Theorem 4 (Upper Bound) *Let $0 < \beta < \frac{1}{11}$. Then algorithm FindHighDegreeVertexOnPA uses $O(n^\beta \log n)$ Jump and Crawl queries and approximates the maximum degree to an expected multiplicative ratio of $O(n^{\frac{1}{2}-\beta})$.*

The first proof ingredient is to show that the preferential attachment network mixes, w.h.p., in poly logarithmic time. We start by defining the lazy random walk on G and then proceed to show it is rapidly mixing.

Algorithm 3 FindHighDegreeVertexOnPA

```
Require: Preferential attachment network $G$, parameter $0 < \beta < \frac{1}{11}$.
Initialize a pointer p to point to an arbitrary vertex.
for $n^\beta \log n$ times do
    Run a lazy random walk from any arbitrary vertex for $2\log^2 n$ steps via Crawl queries.
    Take the vertex $v$ found at the end of the walk.
    if degree($p$) $>$ degree($v$) then
        Let $p = v$
    end if
end for
Output $p$
```

Definition 4 *A* lazy random walk *on a connected network G (LRW for short) stays at the current vertex with probability $\frac{1}{2}$, and with probability $\frac{1}{2}$ moves to a uniformly chosen random neighbor. This random walk forms an ergodic Markov chain. We denote this chain by K. We denote by K^t its t-th power, the stationary distribution of K by π, and the spectral gap of K as $g = max\{\lambda_2, |\lambda_n|\}$.*

Note that the LRW requires only Jump and Crawl queries in its operation. Algebraically, the LRW is more appealing than its random walk counterpart since all its eigenvalues are nonnegative. We next state a few known facts about LRWs on connected networks (the proofs may be found in [7], pages 153-167):

1. The unique stationary distribution of K is $\pi(v) = \frac{degree(v)}{2mn}$.
2. The spectral gap equals λ_2 since all the eigenvalues of the LRW are nonnegative.
3. $max_{i,j}\{|\frac{K^t(i,j)}{\pi(j)} - 1|\} \leq \frac{1}{\pi_{min}} g^t$.
4. $1 - g \geq \frac{h^2}{2}$.
5. The conductance
$$h = min_{\pi(S) \leq \frac{1}{2}} \frac{\sum_{i \in S, j \in S^c} \pi(i) K(i,j)}{\pi(S)}$$
is constant for preferential attachment networks.

Corollary 1 *The mixing time for the lazy random walk on a typical preferential attachment graph is* $t = \theta(\log n)$*. That is,* $max_{i,j}\{|\frac{K^t(i,j)}{\pi(j)} - 1|\} \leq O(\frac{1}{n})$

Proof: By the previous lemma we get that the spectral gap is less than a constant smaller than 1. Next, $\pi_{min} = \Omega(\frac{1}{n})$, and the corollary follows immediately. ■

The second proof ingredient is a theorem due to Chung and Lu:

Theorem 5 (Adapted from [6], page 70) *Let* G *be a network of* n *vertices created using the preferential attachment process with parameter* m*. Let* $m_{k,0}$ *the number of vertices with degree* k *in the initial network. Then the number of vertices* m_n *with degree* d *is* $nM_k + m_{k,0} + O(\sqrt{(k+m-1)^3 n \log n})$*, where* $M_m = \frac{2}{3}$ *and* $M_k = O(k^{-3})$*, for* $k \geq m+1$*.*

Next, we provide the proof sketch for Theorem 4. Line 4 in the algorithm returns the last node visited in an LRW walk of length $\log^2 n$ queries. By returning this node we are in fact sampling a node from the degree distribution of the network, as shown in Corollary 1. Consider small degree values of d, for which we know by the Chung and Lu theorem that the degree distribution is very close to its expected values. Choose the maximum k such that $\frac{1}{n}\sqrt{(k+m-1)^3 n \log n} \leq \frac{1}{k^{3+1}}$. The solution is $k = o(n^{\frac{1}{11}})$. Under such a k we conclude, using Theorem 5, that the probability of seeing a vertex with degree exactly d, under the degree distribution, is $n \cdot c\frac{1}{d^3} \cdot \frac{d}{2mn} \sim \frac{1}{d^2}$. By Lemma 2, the probability of sampling a vertex of degree d or more is $O(\sum_{v:degree(v)\geq d} \frac{1}{d^2}) = O(\frac{1}{d})$ (this is true since sampling a smaller than d vertex is 1 minus the value given by the lemma). Next, notice that the maximum degree of a preferential attachment network is about $\sqrt{n}$ [9]. Now define the degree d to be the solution to $d = n^\beta$ (any $0 < \beta < \frac{1}{11}$ is valid). Then *FindHighDegreeVertexOnPA*, constructs, with $n^\beta \log n$ queries, an expected multiplicative approximation of $\left(n^{\frac{1}{2}-\beta}\right)$.

Discussion: Random walks allow effective sampling from the degree distribution of the preferential attachment network. In fact, for sparse graphs the probability of sampling a vertex of degree d or more, from any degree distribution is always bigger then sampling such a vertex from the uniform distribution. Thus, for any network where the LRW mixes in polylogarithmic time we may devise an analog algorithm to FindHighDegreeVertexOnPA. This algorithm would give better query results then an algorithm that samples directly from the uniform distribution. ■

2.6 Comparing the Rates

It is interesting to compare the approximation rate achievable for arbitrary connected networks with those possible for power law and preferential attachment networks. The relevant plots are given in Figure 2. The x-axis measures the number of queries used, and units are the log number of queries divided by log network size (thus extracting the exponent or root of n). The y-axis measures the approximation guarantee and is given in units of log of approximation ratio divided by log network size (again extracting the approximation exponent). As we proved, for arbitrary networks the optimal algorithm may achieve with query exponent β an approximation exponent of $1-\beta$. For power law networks with given degree distribution exponent $\gamma > 1$ we could do better and achieve, with query exponent β, an approximation exponent of $\frac{1}{\gamma} - \frac{\beta}{\gamma-1}$. This exponent is always better than the $1-\beta$ exponent achievable for arbitrary graphs; we plot the achievable power law network rates for two values of γ. For preferential attachment networks, with a query exponent of β we can achieve an approximation ratio of $\frac{1}{2} - \beta$, but could only prove so for $\beta < 1/11$, and thus this tradeoff is represented as a line fragment rather than a full line. We note that while power law and preferential graphs are of course subsets of the class of all graphs, they are not directly comparable to each other — as discussed above, preferential attachment graphs obey our definition of power law degree distributions only in expectation, and are known with high probability to violate this expectation at large degrees.

3 High Clustering Coefficient

The clustering coefficient (CC for short) of a given vertex measures how densely connected its neighbors are.

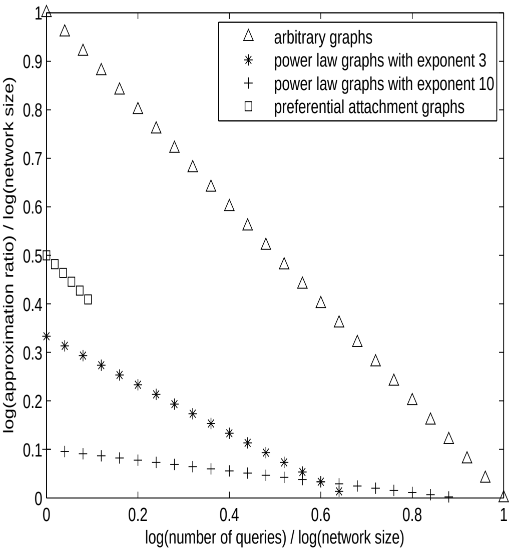

Figure 2: Summary of achievable trade-offs between Jump and Crawl query complexity and maximum degree approximation for various assumptions on the network; see text for details and discussion.

Definition 5 (Clustering Coefficient) *Given a vertex v with degree d, the clustering coefficient (CC) of v is defined as*

$$CC(v) = \frac{\textit{number of triangles containing } v}{\binom{d}{2}}.$$

If $d = 0$ we define $CC(v) = 0$.

This definition is equivalent to the edge density (fraction of possible edges present) among the neighbors of v (excluding v itself).

Many empirical papers have shown natural networks often have vertices of high clustering coefficient (as well as high degree); see for instance [11] Chapter 2 for a detailed survey. In this section we examine the problem of finding such vertices in the Jump and Crawl model. Eubank et al. [8] showed that the global *average* of the CC can be estimated quickly in a Jump and Crawl model using standard Chernoff bounds. This immediately provides a strategy to find a vertex with more than the average CC value (using Markov's inequality). However, finding a vertex with a high CC may not be that illuminating: it may be the case that the vertex with the highest CC has only very few neighbors. Take the extreme case of a vertex with two neighbors that are also connected to each other (a triangle). In this case the CC of v would be the highest possible of 1. This motivates us to ask how hard is it to find a vertex with *simultaneously* high CC and high degree. We shall phrase this approximation problem as follows: given a degree lower bound d as input, find a vertex of degree not much smaller than d whose CC approximates the maximum CC among all vertices of degree d or larger.

Definition 6 *Given a graph on n vertices, and a degree value d, let v^* be the vertex with the highest CC among vertices of degree d or more. We say that v is a (α, d, ϵ)-approximation to the maximum CC if $degree(v) \geq \alpha \cdot d$ and $CC(v^*) \leq CC(v) + \epsilon$, for $0 < \alpha \leq 1$ and $0 < \epsilon < 1$. If there are no vertices of degree at least $\alpha \cdot d$ in the network we say that every vertex is a $(\alpha, d, 0)$-approximation.*

Let us start by noting that since we are requiring a degree lower bound on the vertices found in addition to high CC, it is natural to begin by attempting to adapt our results for finding high degree vertices to the CC problem. Indeed, a simple adaptation of the lower bound for arbitrary networks given in Theorem 2 already yields similar difficulty for the CC problem. Consider Figures 3 and 4, which are slight variants of the construction in Theorem 2. In each variant there is a single high-degree vertex, but in one variant the CC of that vertex is 0 (the lowest possible) and in the other it is 1 (the highest possible). If an algorithm fails to find this high-degree vertex, it cannot hope to approximate the clustering coefficient by a nontrivial additive amount, thus establishing a lower bound of $n^{1-\beta}$ queries on the $(1, n^\beta, 1/2)$-approximation problem for CC.

On the other hand, it is unfortunately *not* clear how to adapt the *upper bound* for the degree problem on arbitrary graphs given by Theorem 1 to the CC problem. The difficulty is that the algorithm of that upper bound will only produce *some* vertex of high degree — but if there are many such vertices, it provides no guarantee that the one produced will also have high CC. Therefore we next ask whether a (α, d, ϵ) is achievable for some nontrivial $\alpha < 1$. In the following theorem we show that at the expense of a logarithmic factor in the de-

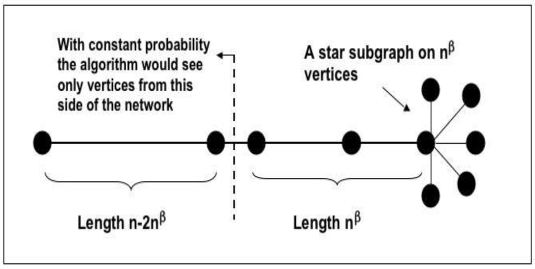
With constant probability the algorithm would see only vertices from this side of the network
A star subgraph on n^β vertices
Length n-2n^β
Length n^β

Figure 3: The line-star network G_1

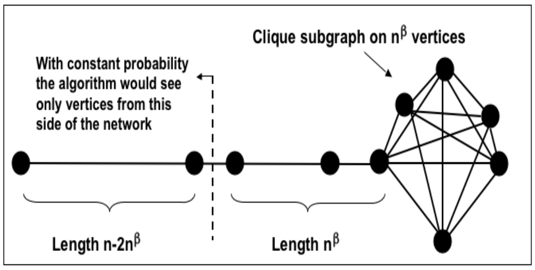
Clique subgraph on n^β vertices
With constant probability the algorithm would see only vertices from this side of the network
Length n-2n^β
Length n^β

Figure 4: The line-clique network G_2

gree, we can obtain arbitrarily small degradation to the maximum clustering coefficient.

Theorem 6 (Upper Bound) *For any given $0 < \beta < 1$, algorithm FindHighCCHighDegreeVertex uses $\tilde{O}(n^{1-\beta})$ Jump and Crawl queries and returns a $(\frac{1}{\log n}, n^{\beta}, \frac{1}{\log n})$ approximation to the maximum clustering coefficient, whp. In other words, if v^* is the vertex with the highest CC value between all vertices with degree at least n^β the algorithm returns a vertex v of degree at least $\frac{n^\beta}{\log n}$ and $cc(v^*) \leq CC(v) + \frac{1}{\log n}$, with probability of $1 - O(\frac{1}{n})$.*

Proof: The algorithm makes all its queries in line 1 and therefore uses at most $\tilde{O}(n^{1-\beta})$ Jump and Crawl queries. Let $v^* = argmax\{CC(v) : degree(v) \geq d\}$. Then probability the v^* is added to T can be easily shown to be at least $1 - n^{-\log n}$, using Hoeffding bound. Next, vertices w with degree at most $\frac{n^\beta}{\log n}$ are excluded from T with high probability. For such vertex w the expected number of times a neighbor of w is sampled is less then $\log^3 n$. Therefore, using Hoeffding bound (see appendix C for restatement of the bound), the probability that at least $\log^4 n$ neighbors of w are

Algorithm 4 FindHighCCHighDegreeVertex

```
Require: Network $G$, parameter $0 < \beta < 1$.
 1: Take $2n^{1-\beta}\log^4 n$ Jump queries and store the
    vertices found (including repetitions) in a multiset $S$.
 2: for each node $v$ in the network do
 3:     Compute $neighbors[v]$ to be the number of
        elements $u$ in $S$ s.t. $u$ is a neighbor of $v$.
 4: end for
 5: Initialize $T = \emptyset$.
 6: for each node $v$ in the network do
 7:     if $neighbors[v] \geq \log^4 n$ then
 8:         Add $v$ to $T$.
 9:     end if
10: end for
11: for each node $v$ in $T$ do
12:     Compute $S_v$ to equal the elements of $S$ (including
        repetitions) that are neighbors of $v$.
13:     Compute $\hat{CC}(v)$=ApproxCCBySample($v$,$S_v$)
14: end for
15: return argmax$_{v \in T}\{\hat{CC}(v)\}$
```

Algorithm 5 ApproxCCBySample

```
Require: Network $G$, a vertex $v$, a multiset $S_v$ of
    elements that are neighbors of $v$.
1: Set $count = 0$.
2: for $i = 1$ to $\log^3 n$ do
3:     Find two elements $u, w$ in $S_v$ that correspond
       to different vertices of $G$ and set
       $S_v = S_v - \{u, w\}$. If there are none return
       FAIL.
4:     if $u$ is a neighbor of $w$ then
5:         $count = count + 1$.
6:     end if
7: end for
8: return $\frac{count}{\log^3 n}$
```

sampled and added to S is smaller then $n^{-2\log n}$. Using the union bound this probability is kept as $n^{-\log n}$ over all such w's. To finish up the argument we need to show that for each vertex in T the algorithm approximates its CC value to an additive value of $\frac{1}{\log n}$. To show this we first notice that the projection of S onto the set of neighbors of v, namely S_v is a random sample of size S_v taken form the neighbors of v. Next, it was shown that when one samples uniformly at random from a set of n elements then with probability of at least $1-n^{-1}$ each element appears at most $\frac{3\log n}{\log\log n}$ times (see for example [13] page 93). Therefore, S_v contains at least $\log^3 n$ distinct pairs and algorithm ApproxCCBySample, with high probability, will not return FAIL. In that case, FindHighCCHighDegreeVertex computes a sum over indicator functions where each indicator function checks if two randomly sampled vertices are connected and form a triangle together with v. By using the additive Hoeffding bound we conclude that the CC of v will be estimated to an additive factor of $\frac{1}{\log n}$ with probability of at least $1-\frac{1}{n^2}$. Using the union bound the clustering coefficient of each vertex in T is indeed estimated to an additive factor of $\frac{1}{\log n}$ with probability of at least $1-\frac{1}{n}$. Thus, algorithm FindHighCCHighDegreeVertex returns a vertex v of degree at least $\frac{n^\beta}{\log n}$ and $CC(v^*) \leq CC(v) + \frac{1}{\log n}$, with probability of $1-O(\frac{1}{n})$. ∎

For power law networks where the lazy random walk converges fast we can do substantially better. Denote the mixing time as usual by τ.

Theorem 7 (Upper Bound) *Assume that the network at hand follows a power law with exponent $\gamma \geq 3$, and that the LRW mixes in time τ over the network. Then for any given $0 < \beta < 1$, algorithm FindHighCCHighDegreeVertexForPowerLaws uses $\tilde{O}(n^{1-2\beta}) \cdot \tau$ Crawl queries and returns a $(1, n^\beta, \frac{1}{\log n})$ approximation to the maximum clustering coefficient property. In other words, if v is the vertex with the highest CC value between all vertices with degree at least n^β the algorithm returns vertex v of degree at least n^β, and $CC(v^*) \leq CC(v) + \frac{1}{\log n}$, with probability of $1-e^{-2}-O(\frac{1}{n})$.*

Proof: First, without loss of generality we assume that the value $CC(v^*)$ is known to the algorithm. We can then simulate the possible values using a standard doubling trick, starting from $c = \frac{1}{\log n}$ up to 1, on the expense of an additional logarithmic factor to the query time. Second, the number of queries the algorithm makes is $\tilde{O}(\frac{n}{d^2})$ since $\gamma \geq 3$ and $CC(v) \geq \frac{1}{\log n}$. Next, we prove the correctness of the algorithm. We use the following observation : if a vertex of degree d has $CC(v) = c$ then at least $\frac{cd}{5}$ of v's neighbors each has degree at least $\frac{cd}{5}$ (otherwise we get a contradiction $CC(v) < c$). Therefore, by sampling $\frac{25kn}{d^2}$ times from the LRW we are effectively sampling that much from its stationary distribution which is the degree distribution. The probability to sample a neighbor of v^* is therefore at least $\frac{cd^2}{25kn}$. Therefore, with $\frac{50kn}{cd^2}$ samples taken from the degree distribution we shall sample v^* with probability $1-e^{-2}$. Using Hoeffding bound we conclude that with probability of at least $1-n^{-\log n}$ we shall not sample neighbors of a vertex with degree less than $\frac{d}{4\log n}$. Last, given a vertex v in a graph of n vertices, *ApproxVertexCCValue* (see pseudocode), with $O(\log^3 n)$ Crawl queries, approximates v's clustering coefficient to an additive error of at most $\frac{1}{\log n}$, with probability $1-O(\frac{1}{n^2})$. Using the union bound the clustering coefficient of each vertex in T is indeed estimated to an additive factor of $\frac{1}{\log n}$ with probability of at least $1-\frac{1}{n}$. Last, by lemma 2 the network has at most $n(1+o(1)(\frac{\log n}{d})^{\gamma-1}$ nodes of degree at least $\frac{d}{\log n}$ so the algorithm, with high probability, won't return FAIL and T will contain that many vertices. Thus, the algorithm returns a vertex v of degree at least $\frac{n^\beta}{\log n}$ and $CC(v^*) \leq CC(v) + \frac{1}{\log n}$, with probability of $1-e^{-2}-O(\frac{1}{n})$. ∎

Using a standard amplification trick the success probability can be amplified to $1-O(\frac{1}{n})$ on the expense of an additional logarithmic factor in the query complexity.

As a sample application of this theorem to another well-studied class of networks, we have:

Corollary 2 *Take $d = n^\beta$ for $0 < \beta < 1$. Then with $\tilde{O}(n^{1-2\beta})$ queries, algorithm FindHighCCHighDegreeVertexForPowerLaws produces a node v of degree at least n^β and $CC(v^*) \leq CC(v) + \frac{1}{\log n}$, in the following cases.*

1. *Take a network created using the preferential attachment process. Then with high probabil-*

ity over the process the LRW over the network is $O(\log n)$ mixing and follows a power law over degrees $n^{\beta} \leq n^{11}$.

2. *Consider the model of producing a random graph with a given degree sequence of Newman et al [14]. It was shown that with high probability a random network would have a $O(\log^3 n)$-mixing time, if all the degrees are bigger than 2 (see [7] page 160 for more details). Take the given graph to be a power law network $G \in Network(3, n, \gamma)$, for $\gamma \geq 3$.*

Thus, we again beat the hardness results for arbitrary networks — by making structural assumptions that allow the additive error to go to 0 with network size.

Algorithm 6 FindHighCCHighDegreeVertexForPowerLaws

```
Require: Power law network $G \in Network(m, n, \gamma)$ with $n$ vertices and
    $kn$ edges, mixing time $\tau$, a degree value $d$, the clustering coefficient value $c$.
$S = \emptyset$.
for for $\frac{50kn}{cd^2}$ times do do
    Run an LRW from an arbitrary vertex for $\tau$ steps via Crawl queries. Add the vertex
    found at the end of the walk to $S$.
end for
Initialize a set $T = \emptyset$.
for each node $v$ in the network do
    if there exists a vertex $u$ in $S$ that is a neighbor of $v$ then
        Add $v$ to $T$ if $v \notin T$.
    end if
end for
if size of $T$ is bigger than $2n(\frac{\log n}{d})^{\gamma-1}$ then
    return FAIL
end if
if $\forall v \in T$ degree$(v) < d$ then
    return FAIL
end if
for each node $v$ in $T$ do
    Approximate $CC[v]$ directly by computing
    $\hat{CC}(v)$=ApproxVertexCCValue$(v)$.
end for
return argmax$_{v \in T}\{\hat{CC}(v)$ : degree$(v) \geq d\}$
```

Algorithm 7 ApproxVertexCCValue

```
Require: Network $G$, a vertex $v$.
Initialize a counter, $cnt = 0$.
for $\log^3 n$ times do
    choose two distinct random neighbors of $v$, call them $u, w$.
    if $u$ appears in the adjacency list of $w$ then
        $cnt = cnt + 1$.
    end if
end for
Output $\frac{cnt}{\log^3 n}$
```

4 Future Research

In this work we initiated a study of local algorithms for finding vertices with extreme topological properties in large social or other networks. The next property we wish to understand is the *betweenness centrality* (and other common centrality measures). Betweenness centrality measures how many shortest paths are passing through a vertex. Vertices with high betweenness centrality value may therefore be susceptible to many kinds of attacks, or play important roles in organizations. It is of interest to understand how quickly one can find such vertices in the Jump and Crawl model.

Acknowledgments

This research was supported by ONR MURI grant number N00014-08-1-0747.

References

[1] L. A. Adamic, R. M. Lukose and B. A. Huberman. Local search in unstructured networks. *Handbook of Graphs and Networks: From the Genome to the Internet*, S. Bornholdt, and H.G. Schuster (eds.), Wiley, 2002.

[2] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power law networks. *Physical Review E*, Volume 64, 2001.

[3] A. Barabsi. Emergence of scaling in random networks. *Science*, **286**, 509-512, 1999.

[4] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády. The degree sequence of a scale-free random graph process. *Random Struct. Algorithms*, **18**, 279, 2001.

[5] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, **23 (4)**, 493-507, 1952.

[6] F. Chung and L. Lu. *Complex Graphs and Networks*. American Mathematical Society, 2006.

[7] R. Durrett. *Random Graph Dynamics*. Cambridge University Press, 2007.

[8] S. Eubank, V.S. Anil Kumar, M.V. Marathe, A. Srinivasan, and N. Wang. Structural and algorithmic aspects of massive social networks. *SODA*, 718-726, 2004.

[9] A. Flaxman , A. Frieze, and T. Fenner. High degree vertices and eignenvalues in the preferential attachment graph. *Internet Mathematics*, **2**, 1-19, 2005.

[10] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statistical Association*, **58**, 13–30, 1963.

[11] M. Jackson. *Social and Economic Networks*. Princeton University Press, 2008.

[12] M. Kleinberg. The small-world phenomenon: an algorithm perspective. *STOC 2000*, 163–170, 2000.

[13] M. Mitzenmacher and E. Upfal. *Probability and Computing*. Cambridge University Press, 2005.

[14] M.E.J. Newman, and S.H. Strogatz, and D.J. Watts. Random graph models of social networks. *PNAS*, **99**, 2566-2572, 2002.

[15] S. Raskhodnikova, D. Ron, A. Shpilka, and A. Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *FOCS*, 559-569, 2007.

[16] T. Schank, and D. Wagner. Approximating clustering coefficient and transitivity. *J. Graph Algorithms Appl.*, **9(2)**, 265-275, 2005.

Appendix A: Omitted Proofs

Proof of lemma 1: Let Q be the empirical degree distribution of G. We need to find a degree d such that $Q(d) = \frac{1}{n}$. Namely, we need to solve $\frac{1}{Zd^\gamma} + \Delta = \frac{1}{n}$, where Δ is the difference between $P(d)$, the power law distribution *PL*(m, n, γ) and the distribution $Q(d)$. Since $\frac{1}{d^{\gamma+1}} = o(\frac{1}{d^\gamma})$, the solution is $(\frac{n}{Z}(1 - Zo(1)))^{\frac{1}{\gamma}} \leq d \leq (\frac{n}{Z}(1 + Zo(1)))^{\frac{1}{\gamma}}$. ■

Proof of lemma 2: As before, let P be a finite power law distribution $PL(m, n, \gamma)$, and denote as usual $Z = \sum_{d=m}^{d=n} \frac{1}{d^\gamma}$. By the previous lemma the probability to uniformly sample a vertex v with a degree at least d is at least $\sum_{i=d}^{i=(\frac{n}{Z}(1-Zo(1))^{\frac{1}{\gamma}}} [\frac{1}{Z(d+i)^\gamma}(1 - o(1))]$. Denote $r1 = \sum_{i=d}^{i=(\frac{n}{Z})^{\frac{1}{\gamma}}} \frac{1}{Z(d+i)^\gamma}$. It suffices to show that $r_1 = \theta(\frac{1}{Zd^{\gamma-1}})$.

Let $d > 1$. Since $f(x) = \frac{1}{x^\gamma}$ is continuous and monotonically decreasing we get,

$$\int_{x=d-1}^{x=(\frac{n}{Z})^{\frac{1}{\gamma}}} \frac{1}{Z(d+x)^\gamma} dx \leq r1$$

and

$$r1 \leq \int_{x=d-1}^{x=(\frac{n}{Z})^{\frac{1}{\gamma}}+1} \frac{1}{Z(d+x)^\gamma} dx.$$

Next,

$$\int_{x=d}^{x=(\frac{n}{Z})^{\frac{1}{\gamma}}} \frac{1}{Z(d+x)^\gamma} dx =$$

$$\frac{1}{Z(\gamma+1)d^{\gamma-1}} - \frac{1}{Z(\gamma+1)(d+(\frac{n}{Z})^{\frac{1}{\gamma}})^{\gamma-1}} =$$

$\theta(\frac{1}{Zd^{\gamma-1}})$, since $d \leq \frac{d^*}{2}$. ■

Appendix B: Network Size

We discuss here a related problem of estimating the network size in the *Jump and Crawl model*. The goal is to compute an estimator $\hat{n}$ which gives a k-approximation to the network size: $\frac{1}{k} \leq \frac{\hat{n}}{n} \leq k$. Clearly, any known algorithm that estimates the uniform distribution support size is a valid algorithm to use here. For the distribution support size problem it is known that any algorithm must use at least $\sqrt{n}$ samples and that $O(\sqrt{n})$ suffices, in order to get a tight approximation (see [15] for a discussion). However, in the *Jump and Crawl model* one is allowed to take Crawl queries in addition to Jump queries. Therefore, given a connected network the *Jump and Crawl model* seems more powerful than just a model with Jump queries. Interestingly, we next show this isn't the case - namely, no algorithm can estimate the network size well using less than $\sqrt{n}$ queries.

Theorem 8 (Lower Bound) *Let G be a 2-vertex connected graph. Let A be an algorithm for estimating the network size, working under the Jump* *and Crawl model. Then if A uses $o(\sqrt{n})$ queries A would fail to approximate the network size to any finite factor.*

Proof: Fix an integer $s > 1$. Take two cycle networks one with n nodes and the other with n^s (a cycle network is another name for a 2 regular graph). We next show that with $o(\sqrt{n})$ queries A will fail to differentiate between the two networks.

First, without loss of generality, we may assume that any given algorithm first make all its Jump queries before making its Crawl queries. We can always simulate the behavior of the original algorithm by first taking all the Jump queries. The cost (number of queries) of the simulation would be at most twice of that the original algorithm. Saying that, we now consider what strategy an algorithm may use on the cycle network (even when knowing in advance it is a cycle network). After the Jump phase, the algorithm is only left with making Crawl queries so it can move left or right from each vertex found, a long the cycle. We next show that not only there are no repetitions in the vertices found in the Jump phase but that these vertices are spread around the cycle, with distance of at least $o(\sqrt{n})$ between any pair of them. Therefore, any algorithm that uses $o(\sqrt{n})$ queries will never see a vertex twice. This behavior would still be true even if we replaced n by n^s. Therefore, the algorithm cannot differentiate between these two cases. To finish up the proof we are left with the calculation of the distance between vertices found in the Jump phase. Let v_i be the vertex found by the i'th Jump query. Let k be the total number of vertices found in the Jump phase. We know that $k = o(\sqrt{n})$. Next we show the probability that the distance between any two such vertices is less than k tends to 1 as n goes to infinity. It is suffice to show that the complement probability goes to zero. Let E_i be the event that the vertex added by the i'th Jump query is closer than k to some of the vertices $v_1, v_2, \ldots, v_n$. In particular we are interested in calculating $Pr(E_i | \neg E_{i-1}, \ldots, \neg E_2, \neg E_1)$. This probability is upper bounded by $\frac{2ik}{n}(1 - \frac{(i-1)k}{n})(1 - \frac{(i-2)k}{n}) \ldots (1 - \frac{k}{n})$. Using the inequality $1 - x \leq \exp(-x)$, we get,

$$Pr(E_i | \neg E_{i-1}, \ldots, \neg E_2, \neg E_1) \leq \frac{2ik}{n} \exp\left(\frac{-ki^2}{2n}\right)$$

$$\leq \frac{2i}{\sqrt{n}} \exp\left(\frac{-ki^2}{2n}\right).$$

By the union bound the probability that some query i is bad, namely, the new vertex is close to some previous vertex, is $o(1)$ and goes to zero as n goes to infinity. ∎

Appendix C: Concentration Bounds

Theorem 9 (Additive Hoeffding Bound) *[10] Let $X_1, X_2, \ldots, X_m$ be a sequence of m independent Bernoulli trials, each with probability of success $E[X_i] = p$. Let, $S = X_1 + X_2 + \ldots + X_m$. Let $0 \leq \gamma \leq 1$. Then,*

$$Pr[\frac{S}{m} > p + \gamma] \leq exp(-2m\gamma^2)$$

and,

$$Pr[\frac{S}{m} < p - \gamma] \leq exp(-2m\gamma^2).$$

Theorem 10 (Multiplicative Chernoff Bound) *[5] Let $X_1, X_2, \ldots, X_m$ be a sequence of m independent Bernoulli trials, each with probability of success $E[X_i] = p$. Let, $S = X_1 + X_2 + \ldots + X_m$. Let $\gamma \geq 0$. Then,*

$$Pr[\frac{S}{m} > (1+\gamma)p] \leq exp(\frac{-mp\gamma^2}{3})$$

and,

$$Pr[\frac{S}{m} < (1-\gamma)p] \leq exp(\frac{-mp\gamma^2}{2}).$$