The Power of Local Information in Social Networks

Christian Borgs[1] and Michael Brautbar[2] and Jennifer Chayes[1] and Sanjeev Khanna[2] and Brendan Lucier[1]

[1] Microsoft Research New England {borgs,jchayes,brlucier}@microsoft.com
[2] Computer and Information Science, University of Pennsylvania {brautbar,sanjeev}@cis.upenn.edu

Abstract. We study the power of *local information algorithms* for optimization problems on social and technological networks. We focus on sequential algorithms where the network topology is initially unknown and is revealed only within a local neighborhood of vertices that have been irrevocably added to the output set. This framework models the behavior of an external agent that does not have direct access to the network data, such as a user interacting with an online social network.
We study a range of problems under this model of algorithms with local information. When the underlying graph is a preferential attachment network, we show that one can find the root (i.e. initial node) in a polylogarithmic number of steps, using a local algorithm that repeatedly queries the visible node of maximum degree. This addresses an open question of Bollobás and Riordan. This result is motivated by its implications: we obtain polylogarithmic approximations to problems such as finding the smallest subgraph that connects a subset of nodes, finding the highest-degree nodes, and finding a subgraph that maximizes vertex coverage per subgraph size.
Motivated by problems faced by recruiters in online networks, we also consider network coverage problems on arbitrary graphs. We demonstrate a sharp threshold on the level of visibility required: at a certain visibility level it is possible to design algorithms that nearly match the best approximation possible even with full access to the graph structure, but with any less information it is impossible to achieve a non-trivial approximation. We conclude that a network provider's decision of how much structure to make visible to its users can have a significant effect on a user's ability to interact strategically with the network.

1 Introduction

In the past decade there has been a surge of interest in the nature of complex networks that arise in social and technological contexts; see [9] for a recent survey of the topic. In the computer science community, this attention has been directed largely towards algorithmic issues, such as the extent to which network structure can be leveraged into efficient methods for solving complex tasks. Common problems include finding influential individuals, detecting communities, constructing subgraphs with desirable connectivity properties, and so on.

The standard paradigm in these settings is that an algorithm has full access to the network graph structure. More recently there has been growing interest in *local* algorithms, in which decisions are based upon local rather than global network structure. This locality of computation has been motivated by applications to distributed algorithms [17, 11], improved runtime efficiency [10, 20], and property testing [15, 18]. In this work we consider a different motivation: in some circumstances, an optimization is performed by an external user who has inherently restricted visibility of the network topology. For such a user, the graph structure is revealed incrementally within a local neighborhood of nodes for which a connection cost has been paid. The use of local algorithms in this setting is necessitated by constraints on network visibility, rather than being a means toward an end goal of efficiency or parallelizability.

As a motivating example, consider an agent in a social network who wishes to find (and link to) a highly connected individual. For instance, this agent may be a newcomer to a community (such as an online gaming or niche-based community) wanting to interact with influential or popular individuals, or a recruiter attempting to form strategic connections in a social network application. Finding a high-degree node is a straightforward algorithmic problem without information constraints, but many online and real-world social networks reveal graph structure only within one or two hops from a user's existing connections.

Is it possible for an agent to solve such a problem using only the local information available on an online networking site? This question is relevant not only for individual users, but also to the designer of a social networking service who must decide how much information to reveal. For example, LinkedIn allows each user to see the degree of nodes two hops away in the network, whereas Facebook does not reveal this information by default. We ask: what impact do such design decisions have on an individual's ability to interact with the network?

More generally, we consider graph algorithms in a setting of restricted network visibility. We focus on optimization problems for which the goal is to return a subset of the nodes in the network; this includes coverage, connectivity, and search problems. An algorithm in our framework proceeds by incrementally and adaptively building an output set of nodes, corresponding to those vertices of the graph that have been queried (or connected to) so far. When the algorithm has queried a set S of nodes, the structure of the graph within a small radius of S is revealed, guiding future queries. The principle challenge in designing such an algorithm is that decisions must be based solely on local information, whereas the problem to be solved may depend on the global structure of the graph. In addition to these restrictions, we ask for algorithms that run in polynomial time.

For many problems we derive strong lower bounds on the performance of local algorithms in general networks. We therefore turn to the class of preferential attachment (PA) graphs, which model properties of many real-world social and technological networks. For PA networks, we prove that local information algorithms do well at many optimization problems, including shortest path routing and finding the k vertices of highest degree (up to polylogarithmic factors).

We also consider node coverage problems on general graphs, where the goal is to find a small set of nodes whose neighborhood covers all (or much) of the network. Such coverage problems are especially motivated in our context by applications to employment-focused social networking platforms such as LinkedIn, where there is benefit in having as many nodes as possible within a few hops of one's direct connections[3]. We design local information algorithms whose performances approximately match the best possible even when information about network structure is unrestricted. We also demonstrate that the amount of local information available is of critical importance: strong positive results are possible at a certain range of visibility (made explicit below), but non-trivial algorithms become impossible when less information is made available. This observation has implications for the design of online networks, such as the amount of information to provide a user about the local topology: seemingly arbitrary design decisions may have a significant impact on a user's ability to interact with the network.

Results and Techniques Our first set of results concerns algorithms for preferential attachment (PA) networks. Such networks are defined by a process by which nodes are added sequentially and form random connections to existing nodes, where the probability of connecting to a node is proportional to its degree.

We first consider the problem of finding the root (first) node in a PA network. A random walk would encounter the root in $\tilde{O}(\sqrt{n})$ steps (where n is the number of nodes in the network). The question of whether a better local information algorithm exists for this problem was posed by Bollobas and Riordan [5]. They conjecture that such short paths can be found locally in $\Theta(\log n)$ steps. We make the first progress towards this conjecture by showing that polylogarithmic time is sufficient: there is an algorithm that finds the root of a PA network in $O(\log^4(n))$ time, with high probability. We show how to use this algorithm to obtain polylogarithmic approximations for finding the smallest subgraph that connects a subset of nodes (including shortest path), finding the highest-degree nodes, and finding a subgraph that maximizes vertex coverage per subgraph size.

The local information algorithm we propose uses a natural greedy approach: at each step, it queries the visible node with highest degree. Demonstrating that such an algorithm reaches the root in $O(\log^4(n))$ steps requires a probabilistic analysis of the PA process. A natural intuition is that the greedy algorithm will find nodes of ever higher degrees over time. However, such progress is impeded by the presence of high-degree nodes with only low-degree neighbors. We show that these bottlenecks are infrequent enough that they do not significantly hamper the algorithm's progress. To this end, we derive a connection between node degree correlations and supercritical branching processes to prove that a path of high-degree vertices leading to the root is always available to the algorithm.

We then consider general graphs, where we explore local information algorithms for dominating set and coverage problems. A dominating set is a set S

[3] For example, LinkedIn allows recruiters to execute searches for potential job candidates among all nodes within distance 3 from the recruiter, additionally displaying resume information for those within distance 2.

such that each node in the network is either in S or the neighborhood of S. We design a randomized local information algorithm for the minimum dominating set problem that achieves an approximation ratio that nearly matches the lower bound on polytime algorithms with no information restriction. As has been noted in [14], the greedy algorithm that repeatedly selects the visible node that maximizes the size of the dominated set can achieve a very bad approximation factor. We consider a modification of the greedy algorithm: after each greedy addition of a new node v, the algorithm will also add a random neighbor of v. We show that this randomized algorithm obtains an approximation factor that matches the known lower bound of $\Omega(\log \Delta)$ (where Δ is the maximum degree in the network) up to a constant factor. We also show that having enough local information to choose the node that maximizes the incremental benefit to the dominating set size is crucial: no algorithm that can see only the degrees of the neighbors of S can achieve a sublinear approximation factor.

Finally, we extend these results to related coverage problems. For the partial dominating set problem (where the goal is to cover a given constant fraction of the network with as few nodes as possible) we give an impossibility result: no local information algorithm can obtain an approximation better than $O(\sqrt{n})$ on networks with n nodes. However, a slight modification to the local information algorithm for minimum dominating set yields a bicriteria result (in which we compare performance against an adversary who must cover an additional ϵ fraction of the network). We also consider the "neighbor-collecting" problem, in which the goal is to minimize $c|S|$ plus the number of nodes left undominated by S, for a given parameter c. For this problem we show that the minimum dominating set algorithm yields an $O(c \log \Delta)$ approximation (where Δ is the maximum degree in the network), and that the dependence on c is unavoidable.

Related Work Over the last decade there has been a substantial body of work on understanding the power of sublinear-time approximations. In the context of graphs, the goal is to understand how well one can approximate graph properties in a sublinear number of queries. See [18] and [13] for recent surveys. Motivated by distributed computation, a notion of local computation was formalized by [19] and further developed in [1]. They define a local computation algorithm as computing only certain specified bits of a global solution. In contrast, our notion of locality is motivated by information constraints imposed upon a sequential algorithm. h Local algorithms motivated by efficient computation, rather than informational constraints, were explored by [2, 20]. These works explore local approximation of graph partitions to efficiently find a global solution.

Preferential attachment (PA) networks were suggested by [3] as a model for large social networks. There has been much work studying the properties of such networks, such as degree distribution [6] and diameter [5]; see [4] for a short survey. The problem of finding high degree nodes, using only uniform sampling and local neighbor queries, is explored in [7]. The low diameter of PA graphs can be used to implement distributed algorithms in which nodes repeatedly broadcast information to their neighbors [11, 8]. A recent work [8] showed that such algorithms can be used for fast rumor spreading. Our results on the ability to find short paths in such graphs differs in that our algorithms are sequential, with a small number of queries, rather than applying broadcast techniques.

The ability to quickly find short paths in social networks has been the focus of much study, especially in the context of small-world graphs [16, 12]. It is known that local routing using short paths is possible in such models, given some awareness of global network structure (such as coordinates in an underlying grid). In contrast, our shortest-path algorithm for PA graphs does not require an individual know the graph structure beyond the degrees of his neighbors. However, our result requires that routing can be done from both endpoints; in other words, both nodes are trying to find each other.

For the minimum dominating set problem, Guha and Khuller [14] designed a local $O(\log \Delta)$ approximation algorithm. As a local information algorithm, their method requires that the network structure is revealed up to distance two from the current dominating set. By contrast, our local information algorithm requires less information to be revealed on each step. Our focus, and the motivation behind this distinction, is to determine sharp bounds on the amount of local information required to approximate this problem (and others) effectively.

2 Model and Preliminaries

Graph Notation We write $G = (V, E)$ for an undirected graph with node and edge sets V and E, respectively. We write n_G for the number of nodes in G, $d_G(v)$ for the degree of a vertex v in G, and $N_G(v)$ for the set of neighbors of v. Given a subset of vertices $S \subseteq V$, $N_G(S)$ is the set of nodes adjacent to at least one node in S. We also write $D_G(S)$ for the set of nodes *dominated* by S: $D_G(S) = N_G(S) \cup S$. We say S is a *dominating set* if $D_G(S) = V$. Given nodes u and v, the distance between u and v is the number of edges in the shortest path between u and v. The distance between vertex sets S and T is the minimum distance between a node in S and a node in T. Given a subset S of nodes of G, the subgraph induced by S is the subgraph consisting of S and every edge with both endpoints in S. Finally, Δ_G is the maximum degree in G. In all of the above notation we often suppress the dependency on G when clear from context.

Algorithmic Framework We focus on graph optimization problems in which the goal is to return a minimal-cost[4] set of vertices S satisfying a feasibility constraint. We will consider a class of algorithms that build S incrementally under local information constraints. We begin with a definition of local neighborhoods.

Definition 1 (Local Neighborhood). *Given a set of nodes S in the graph G, the r-*closed neighborhood* around S is the induced subgraph of G containing all nodes at distance less than or equal to r from S, plus the degree of each node at distance r from S. the r-*open neighborhood* around S is the r-closed neighborhood around S, after the removal of all edges between nodes at distance exactly r from S.*

[4] In most of the problems we consider, the cost of set S will simply be $|S|$.

Definition 2 (Local Information Algorithm). *Let G be an undirected graph unknown to the algorithm, where each vertex is assigned a unique identifier. For integer $r \geq 1$, a (possibly randomized) algorithm is an r^+-local algorithm if:*

1. *The algorithm proceeds sequentially, growing step-by-step a set S of nodes, where S is initialized either to $\emptyset$ or to some seed node.*
2. *Given that the algorithm has queried a set S of nodes so far, it can only observe the r-closed neighborhood around S.*
3. *On each step, the algorithm can add a node to S either by selecting a specified vertex from the r-closed neighborhood around S (a* crawl*) or by selecting a vertex chosen uniformly at random from all graph nodes (a* jump*).*
4. *In its last step the algorithm returns the set S as its output.*

Similarly, for $r \geq 1$, we call an algorithm a *r-local algorithm* if its local information (i.e. in item 2) is made from the r-open neighborhood around S.

We focus on computationally efficient (i.e. polytime) local algorithms. Our framework applies most naturally to coverage, search, and connectivity problems, where the family of valid solutions is upward-closed. More generally, it is suitable for measuring the complexity, using only local information, for finding a subset of nodes having a desirable property. In this case the size of S measures the number of queries made by the algorithm; we think of the graph structure revealed to the algorithm as having been paid for by the cost of S.

For our lower bound results, we will sometimes compare the performance of an r-local algorithm with that of a (possibly randomized) algorithm that is also limited to using Jump and Crawl queries, but may use full knowledge of the network topology to guide its query decisions. The purpose of such comparisons is to emphasize instances where it is the lack of information about the network structure, rather than the necessity of building the output in a local manner, that impedes an algorithm's ability to perform an optimization task.

3 Preferential Attachment Graphs

We consider graphs generated by the preferential attachment (PA) process, conceived by Barabási and Albert [3]. The process is defined sequentially with nodes added one by one. When a node is added it sends m links to previously created nodes, connecting to a node with probability proportional to its current degree.

We will use the following, now standard, formal definition of the process, due to [5]. Given $m \geq 1$, we inductively define random graphs G_m^t, $1 \leq t \leq n$. The vertex set for G_m^t is $[t]$. G_m^1 is the graph with node 1 and m self-loops. Given $G_m^{(t-1)}$, form G_m^t by adding node t and then forming m edges from t to nodes in $[t]$, say $p_1(t), \ldots, p_m(t)$. The nodes $p_k(t)$ are referred to as the *parents* of t. The edges are formed sequentially. For each k, node s is chosen with probability $\deg(s)/z$ if $s < t$, or $(\deg(s)+1)/z$ if $s = t$, where z is a normalization factor. Note that $\deg(s)$ denotes degree in G_m^{t-1}, counting previously-placed edges.

We first present a 1-local approximation algorithm for the following simple problem on PA graphs: given an arbitrary node u, return a minimal connected subgraph containing nodes u and 1 (i.e. the root of G_m^n).

Algorithm 1 TraverseToTheRoot

```
1: Initialize a list $L$ to contain an arbitrary node $\{u\}$ in the graph.
2: while $L$ does not contain node 1 do
3:     Add a node of maximum degree in $N(L)\backslash L$ to $L$.
4: return $L$.
```

Algorithm 2 s-t-Connect

```
1: $P_1 \leftarrow$ TraverseToTheRoot$(G, s)$
2: $P_2 \leftarrow$ TraverseToTheRoot$(G, t)$
3: Return $P_1 \cup P_2$
```

Our algorithm, TraverseToTheRoot, is listed as Algorithm 1. The algorithm grows a set S of nodes by starting with $S = \{u\}$ and then repeatedly adding the node in $N(S)\backslash S$ with highest degree. We will show that, with high probability, this algorithm traverses the root node within $O(\log^4(n))$ steps.

Theorem 1. *With probability $1 - o(1)$ over the preferential attachment process on n nodes, TraverseToTheRoot returns a set of size $O(\log^4(n))$.*

Remark: For convenience, we have defined TraverseToTheRoot assuming that the algorithm can determine when it has successfully traversed the root. This is not necessary in general; our algorithm will have the guarantee that, after $O(\log^4(n))$ steps, it has traversed node 1 with high probability.

Before proving Theorem 1, we discuss its algorithmic implications below.

3.1 Applications of Fast Traversal to the Root

We now describe how to use TraverseToTheRoot to implement local algorithms for other problems on PA networks. Proofs are omitted due to space constraints.

s-t connectivity. The *s-t* connectivity (shortest path) problem is to find a small connected subgraph containing two given nodes s and t in an undirected graph.

Corollary 1. *Let G be a PA graph on n nodes. Then, with probability $1 - o(1)$ over the PA process, Algorithm 2 (listed above), a 1-local algorithm, returns a connected subgraph of size $O(\log^4(n))$ containing vertices s and t.*

This result implies that a subset of k nodes can be connected by a local algorithm in $O(k \log^4(n))$ steps, using a subset of size $O(k \log^4(n))$. Also, in the full version of the paper we show that Corollary 1 does not extend to general graphs: local algorithms cannot achieve sublinear approximations.

Finding high degree nodes. A natural problem on graphs is to find a node with maximal degree. The algorithm TraverseToTheRoot gives a polylogarithmic approximation to this problem with high probability. This follows because, with high probability, the root of a PA network has approximately maximal degree.

Corollary 2. *Let G be a preferential attachment graph on n nodes. Then, with probability $1 - o(1)$, algorithm TraverseToTheRoot will return a node of degree at least $\frac{1}{\log^2(n)}$ of the maximum degree in the graph, in time $O(\log^4(n))$.*

In the full version we show that Corollary 2 does not extend to general graphs.

Maximizing coverage versus cost. In the full version of the paper we consider the optimization problem of finding set S such that $|D(S)|/|S|$ is maximized. For this problem the TraverseToTheRoot algorithm obtains a polylogarithmic approximation in $O(\log^4(n))$ queries, and we prove no such result is possible for general graphs.

3.2 Analysis of TraverseToTheRoot

We now turn to the proof of Theorem 1. Let us provide some intuition. We would like to show that TraverseToTheRoot queries nodes of progressively higher degrees over time. However, if we query a node i of degree d, there is no guarantee that subsequent nodes will have degree greater than d; the algorithm may encounter local maxima. Suppose, however, that there were a path from i to the root consisting entirely of nodes with degree at least d. In this case, the algorithm will only ever traverse nodes of degree at least d from that point onward. One might therefore hope that the algorithm finds nodes that lie on such "good" paths for ever higher values of d, representing progress toward the root.

Motivated by this intuition, we will study the probability that any given node i lies on a path to the root consisting of only high-degree nodes (i.e. not much less than the degree of i). We will argue that many nodes in the network lie on such paths. We prove this in two steps. First, we show that for any given node i and parent $p_k(i)$, $p_k(i)$ will have high degree relative to i with probability greater than 1/2 (Lemma 2). Second, since each node i has at least two parents, we use the theory of supercritical branching processes to argue that, with constant probability for each node i, there exists a path to a node close to the root following links to such "good" parents (Lemma 3).

This approach is complicated by the fact that existence of such good paths is highly correlated between nodes; this makes it difficult to argue that such paths occur "often" in the network. To address this issue, we show that good paths are likely to exist even after a large set of nodes (Γ in our argument below) is adversarially removed from the network. We can then argue that each node is likely to have a good path independently of many other nodes, as we can remove all nodes from one path before testing the presence of another.

We now provide an outline of the proof. The proofs of technical lemmas appear in the full version. Set $s_0 = 160\log(n)(\log\log(n))^2$ and $s_1 = \frac{n}{2^{25}\log^2 n}$. We think of vertices in $[1, s_0]$ as close to the root, and vertices in $[s_1, n]$ as very far from the root. Let $I_t = [2^t + 1, 2^{t+1}]$ be a partition of $[n]$ into intervals. Define constants $\beta = 1/4$ and $\zeta = 30$.

Definition 3 (Typical node). *A node i has* typical degree *if either* $\deg(i) \geq \frac{m}{2\zeta}\sqrt{\frac{n}{i}}$ *or $i \leq s_0$.*

Lemma 1. *The following are true with probability* $1 - o(1)$*:*

- $\forall i \geq s_0 : \ \deg(i) \leq 6m\log(n)\sqrt{\frac{n}{i}}$.
- $\forall i \leq s_0 : \ \deg(i) \geq \frac{m\sqrt{n}}{5\log^2(n)}$.
- $\forall i \geq s_0 : \ \mathbb{P}[i \textit{ is connected to } 1] \geq \frac{3.9}{\log(n)\sqrt{i}}$.
- $\forall j \geq i \geq s_0, k \leq m : \ \mathbb{P}[p_k(i) < j] \geq \frac{0.9\sqrt{i}}{\sqrt{j}}$.

Our next lemma states that, for any set Γ that contains sufficiently few nodes from each interval I_t, and any given parent of a node i, with probability greater than 1/2 the parent will be typical, not in Γ, and not in the same interval as i.

Definition 4 (Sparse set). *A subset of nodes* $\Gamma \subseteq [n]$ *is* sparse *if* $|\Gamma \cap I_t| \leq |I_t|/\log\log(n)$ *for all* $\log s_0 \leq t \leq \log s_1$.

Lemma 2. *Fix sparse set* Γ*. Then for each* $i \in [s_0, s_1]$ *and* $k \in [m]$*, the following are true with probability* $\geq 8/15$ *:* $p_k(i) \notin \Gamma$*,* $p_k(i) \leq i/2$*, and* $p_k(i)$ *is typical.*

We now claim that, for any given node i and sparse set Γ, there is likely a short path from i to vertex 1 consisting entirely of typical nodes that do not lie in Γ. Our argument is via a coupling with a supercritical branching process. Consider growing a subtree, starting at node i, by adding to the subtree any parent of i that satisfies the conditions of Lemma 2, and then recursively growing the tree in the same way from any parents that were added. Since each node has $m \geq 2$ parents, and each satisfies the conditions of Lemma 2 with probability $> 1/2$, this growth process is supercritical and should survive with constant probability (within the range of nodes $[s_0, s_1]$). We should therefore expect that, with constant probability, such a subtree would contain some node $j < s_0$.

The argument above leads to the following lemma, which we will use in our analysis of the algorithm TraverseToTheRoot. First a definition.

Definition 5 (Good paths). *For any* $i \in [s_0, s_1]$*, we say* i has a good path *if there is a path from* i *to a node* $j \leq s_0$ *consisting of nodes with typical degree.*

Lemma 3. *Choose any set* T *of at most* $16\log n$ *nodes from* $[s_0, s_1]$*. Then each* $i \in T$ *has a good path with probability at least* $1/5$*, independently for each* i*.*

We will apply Lemma 3 to the set of nodes queried by TraverseToTheRoot to argue that progress toward the root is made after every sequence of polylogarithmically many steps. We can now complete the proof of Theorem 1, which we sketch below; a full proof appears in the full version of the paper.

Our analysis of Algorithm 1 consists of three steps, corresponding to three phases of the algorithm. The first phase consists of all steps until the first time we traverse a node $i < s_1$ with a good path. The second phase then lasts until the algorithm queries a node $i < s_0$. The third phase ends when the algorithm traverses node 1. We will show that each phase lasts at most $O(\log^4(n))$ steps.

We will make use of Lemma 3 in our analysis whenever we consider whether a node has a good path. We will check at most $16\log n$ nodes in this manner, and hence the conditions of Lemma 3 will be satisfied throughout the analysis.

Analysis of phase 1 Phase 1 ends when the algorithm traverses a node $i < s_1$ with a good path. The value of s_1 is set large enough so that every node queried by the algorithm has index $\leq s_1$ with probability at least $\frac{1}{O(\log n)}$, regardless of previous nodes traversed. By Lemma 3, each such node has a good path with probability at least 1/5. Multiplicative Chernoff bounds therefore imply that the phase will end after at most $O(\log^2(n))$ steps, with high probability.

Analysis of phase 2 We split phase 2 into a number of epochs. For each $t \in [\log s_0, \log s_1]$, epoch t begins when some node $i \in I_t$ with a good path has been traversed (and ends when epoch $t+1$ begins). Define random variable Y_t to be the length of epoch t. The total number of steps in phase 2 is $\sum_{t=\log s_0}^{\log s_1} Y_t$.

Suppose the algorithm is in epoch t, having traversed node $i \in I_t$ with a good path. Then i has a parent $j \in I_u$ with $deg(j) \geq \frac{m}{2\zeta}\sqrt{\frac{n}{i}}$ and $u < t$. This node j could be traversed by the algorithm, so any node queried before j must have at least this degree. Moreover, traversing node j would end epoch t, so every step in epoch t traverses a node with degree at least $\frac{m}{2\zeta}\sqrt{\frac{n}{i}}$. By Lemma 1, any such node ℓ satisfies $\ell < zi\log^2(n)$ where z is a constant. But now, by Lemma 1, each node ℓ traversed in epoch t has a parent $r < i/\log^2(n)$ with probability at least $\frac{1}{O(\log^2(n))}$. Any such node r has degree greater than any node in I_t, again by Lemma 1, so if a queried node had such a parent then the subsequent step must query a node of index at most 2^t. Any such node is on a good path with probability at least 1/5, by Lemma 3, in which case epoch t would end.

To summarize, each step of epoch t causes an end to the epoch with probability at least $\frac{1}{O(\log^2(n))}$. We conclude that $\sum_{t=\log s_0}^{\log s_1} Y_t$ is dominated by the sum of at most $\log n$ geometric random variables, each with mean $O(\log^2(n))$. Concentration bounds for geometric random variables then imply that, with high probability, epoch 2 ends in $O(\log^3(n))$ steps.

Analysis of phase 3 We first note that the induced graph on the first s_0 nodes is connected with high probability (see [8], corollary 5.15). By Lemma 1 every node $i \leq s_0$ has degree at least $d = \frac{m\sqrt{n}}{5\log^{1.9}(n)}$, so the algorithm will only traverse nodes of degree at least d in phase 3. By Lemma 1, any node j with degree at least d must satisfy $j < (60\zeta)^2\log^{5.8}(n)$. Also by Lemma 1, for each such j, the probability that j is adjacent to the root is at least $\frac{1}{2^{11}\log^{3.9}(n)}$. Chernoff bounds then imply that such an event will occur with high probability after at most $O(\log^4(n))$ steps. Thus, with high probability, phase 3 will end after at most $s_0 + O(\log^4(n)) = O(\log^4(n))$ steps. This completes the proof of Theorem 1.

4 Minimum Dominating Set on Arbitrary Networks

We now consider the problem of finding a dominating set S of minimal size for an arbitrary graph G. Even with full (non-local) access to the network structure, it is known to be hard to approximate the Minimum Dominating Set Problem to within a factor of $H(\Delta)$ in polynomial time, where $H(n) \approx \ln(n)$ is the nth

Algorithm 3 AlternateRandom

```
1: Select an arbitrary node $u$ from the graph and initialize $S = \{u\}$.
2: while $D(S) \neq V$ do
3:     Choose $x \in \arg\max_{v \in N(S)}\{|N(v) \backslash D(S)|\}$ and add $x$ to $S$.
4:     if $N(x) \backslash S \neq \emptyset$ then
5:         Choose $y \in N(x) \backslash S$ uniformly at random and add $y$ to $S$.
6: return $S$.
```

harmonic number. In this section we explore how much local network structure must be made visible in order for it to be possible to match this lower bound.

Guha and Khuller [14] design an $O(H(\Delta))$-approximate algorithm for the minimum dominating set problem, which can be interpreted in our framework as a 2^+-local algorithm. As we show, the ability to observe network structure up to distance 2 is unnecessary if we allow the use of randomness: we will construct a randomized $O(H(\Delta))$ approximation algorithm that is 1^+-local. We then show that this level of local information is crucial: no algorithm with less local information can return a non-trivial approximation. Proofs in this section are omitted due to space constraints, but appear in the full version of the paper.

4.1 A 1^+-local Algorithm

We now present a 1^+-local randomized $O(H(\Delta))$-approximation algorithm for the min dominating set problem. Our algorithm obtains this approximation factor both in expectation and with high probability in the optimal solution size[5].

Roughly speaking, our approach is to greedily grow a subtree of the network, repeatedly adding vertices that maximize the number of dominated nodes. Such a greedy algorithm is 1^+-local, as this is the amount of visibility required to determine how much a given node will add to the number of dominated vertices. Unfortunately, this greedy approach does not yield a good approximation; it is possible for the algorithm to waste significant effort covering a large set of nodes that are all connected to a single vertex just beyond the algorithm's visibility. To address this issue, we introduce randomness into the algorithm: after each greedy addition of a node x, we will also query a random neighbor of x. The algorithm is listed above as Algorithm 3 (AlternateRandom).

We now show that AlternateRandom obtains an $O(H(\Delta))$ approximation, both in expectation and with high probability. In what follows, $\mathcal{OPT}$ will denote the size of the optimal dominating set in an inplicit input graph. The proof follows by bounding, for each node v in the optimal solution, the expected number of neighbors of v that are queried before v is queried.

Theorem 2. *AlternateRandom is 1^+-local and returns a dominating set S where $\mathbb{E}[|S|] \leq 2(1 + H(\Delta))\mathcal{OPT} + 1$ and $\mathbb{P}[|S| > 2(2 + H(\Delta))\mathcal{OPT}] < e^{-\mathcal{OPT}}$.*

[5] Our algorithm actually generates a connected dominating set, so it can also be seen as an $O(H(\Delta))$ approximation to the connected dominating set problem.

We end this section by showing that 1^+-locality is necessary for constructing good local approximation algorithms. The example we consider is a clique with one edge (u, v) removed, plus two additional nodes u' and v' that are adjacent to nodes u and v respectively.

Theorem 3. *For any randomized* 1*-local algorithm A for the min dominating set problem, there exists an input instance G for which $\mathbb{E}[|S|] = \Omega(n)\mathcal{OPT}$, where S denotes the output generated by A on input G.*

4.2 Partial Coverage Problems

We next study problems in which the goal is not necessarily to cover all nodes in the network, but rather dominate only sections of the network that can be covered efficiently. We consider two central problems in this domain: the partial dominating set problem and the neighbor collecting problem.

Partial Dominating Set In the partial dominating set problem we are given a parameter $\rho \in (0, 1]$. The goal is to find the smallest set S such that $|D(S)| \geq \rho n$.

We begin with a negative result: for any constant k and any k-local algorithm, there are graphs for which the optimal solution has constant size, but with high probability $\Omega(\sqrt{n})$ queries are required to find any ρ-partial dominating set. Our example will apply to $\rho = 1/2$, but can be extended to any constant $\rho \in (0, 1)$. The example is a graph with two embedded stars, one with $n/2 - \sqrt{n}$ leaves and one with only $\sqrt{n}$ leaves; the optimal solution contains the center of each star, but the smaller star requires many queries to locate.

Theorem 4. *For any randomized k-local algorithm A for the partial dominating set problem with $\rho = 1/2$, there exists an input G with optimal partial dominating set $\mathcal{OPT}$ for which $\mathbb{E}[|S|] = \Omega(\sqrt{n}) \cdot |\mathcal{OPT}|$, where S denotes the output generated by A on input G.*

Motivated by this lower bound, we consider a bicriterion result: given $\epsilon > 0$, we compare the performance of an algorithm that covers ρn nodes with the optimal solution that covers $\rho(1 + \epsilon)n$ nodes (assuming $\rho(1 + \epsilon) \leq 1$). We show that a modification to Algorithm 3, in which jumps to uniformly random nodes are interspersed with greedy selections, yields an $O((\rho\epsilon)^{-1}H(\Delta))$ approximation. The proof is similar in spirit to Theorem 2.

Theorem 5. *Given any $\rho \in (0, 1)$, $\epsilon \in (0, \rho^{-1} - 1)$, and set of nodes $\mathcal{OPT}$ with $|D(\mathcal{OPT})| \geq \rho(1 + \epsilon)n$, Algorithm 4 (AlternateRandomAndJump) returns a set S of nodes with $|D(S)| \geq \rho n$ and $\mathbb{E}[|S|] \leq 3|\mathcal{OPT}|(\rho\epsilon)^{-1}H(\Delta)$.*

The Neighbor Collecting Problem We next consider the objective of minimizing the total cost of the selected nodes plus the number of nodes left uncovered: choose a set S of G that minimizes $f(S) = c|S| + |V \backslash D(S)|$ for a given parameter $c > 0$. This problem is motivated by the Prize-Collecting Steiner Tree problem. The proof is similar in spirit to Theorem 2, noting that the optimal dominating set is no worse than a c-approximation to the optimal solution.

Algorithm 4 AlternateRandomAndJump

```
1: Initialize $S = \emptyset$.
2: while $|D(S)| < \rho n$ do
3:     Choose a node $u$ uniformly at random from the graph and add $u$ to $S$.
4:     Choose $x \in \arg\max_{v \in N(S)}\{|N(v) \backslash D(S)|\}$ and add $x$ to $S$.
5:     if $N(x) \backslash S \neq \emptyset$ then
6:         Choose $y \in N(x) \backslash S$ uniformly at random and add $y$ to $S$.
7: return $S$.
```

Theorem 6. *For any $c \geq 1$ and set $\mathcal{OPT}$ minimizing $f(\mathcal{OPT})$, algorithm AlternateRandom returns a set S for which $\mathbb{E}[f(S)] \leq 2c(1 + H(\Delta))f(\mathcal{OPT})$.*

We show in the full version that the dependency on c is unavoidable and that Theorem 6 cannot be extended to 1-local algorithms without significant loss.

5 Conclusions

We presented a model of computation in which algorithms are constrained in the information they have about the input structure, which is revealed over time as expensive exploration decisions are made. Our motivation lies in determining whether and how an external user in a network, who cannot make arbitrary queries of the graph structure, can efficiently solve optimization problems in a local manner. Our results suggest that inherent structural properties of social networks may be crucial in obtaining strong performance bounds.

Another implication is that the designer of a network interface, such as an online social network platform, may gain from considering the power and limitations that come with the design choice of how much network topology to reveal to individual users. On one hand, revealing too little information may restrict natural social processes that users expect to be able to perform, such as searching for potential new connections. On the other hand, revealing too much information may raise privacy concerns, or enable unwanted behavior such as automated advertising systems searching to target certain individuals. Our results suggest that even minor changes to the structural information made available to a user may have a large impact on the class of optimization problems that can be reasonably solved by the user.

Acknowledgments

The author Sanjeev Khanna was supported in part by NSF awards CCF-1116961 and IIS-0904314, and by ONR MURI grant N00014-08-1-0747.

References

1. Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *SODA*, pages 1132–1139, 2012.

2. Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Local graph partitioning using pagerank vectors. In *FOCS*, pages 475–486, 2006.
3. A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
4. Béla Bollobás. Mathematical results on scale-free random graphs. *in Handbook of Graphs and Networks: From the Genome to the Internet*, 2003.
5. Béla Bollobás and Oliver Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004.
6. Béla Bollobás, Oliver Riordan, Joel Spencer, and Gábor E. Tusnády. The degree sequence of a scale-free random graph process. *Random Struct. Algorithms*, 18(3):279–290, 2001.
7. Mickey Brautbar and Michael Kearns. Local algorithms for finding interesting individuals in large networks. In *Innovations in Theoretical Computer Science (ITCS)*, pages 188–199, 2010.
8. Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. Social networks spread rumors in sublogarithmic time. In *STOC*, pages 21–30, 2011.
9. D. Easley and J. Kleinberg. *Networks, Crowds, and Markets, reasoning about a Highly Connected World.* Cambridge University Press, 2010.
10. Christos Faloutsos, Kevin S. McCurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *KDD*, pages 118–127, 2004.
11. George Giakkoupis and Thomas Sauerwald. Rumor spreading and vertex expansion. In *SODA*, pages 1623–1641, 2012.
12. George Giakkoupis and Nicolas Schabanel. Optimal path search in small worlds: dimension matters. In *STOC*, pages 393–402, 2011.
13. Oded Goldreich. Introduction to testing graph properties. In *Property Testing*, pages 105–141, 2010.
14. Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.
15. Avinatan Hassidim, Jonathan A. Kelner, Huy N. Nguyen, and Krzysztof Onak. Local graph partitions for approximation and testing. In *FOCS*, pages 22–31, 2009.
16. Jon M. Kleinberg. The small-world phenomenon: an algorithm perspective. In *STOC*, pages 163–170, 2000.
17. Moni Naor and Larry Stockmeyer. What can be computed locally? In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 184–193, New York, NY, USA, 1993. ACM.
18. Ronitt Rubinfeld and Asaf Shapira. Sublinear time algorithms. *SIAM Journal on Discrete Math*, 25:1562–1588, 2011.
19. Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *ITCS*, pages 223–238, 2011.
20. Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *CoRR*, abs/0809.3232, 2008.