Community membership identification from small seed sets

Isabel M. Kloumann
Cornell University
657 Rhodes Hall
Ithaca, NY, 14853
imk36@cornell.edu

Jon M. Kleinberg
Cornell University
318 Gates Hall
Ithaca, NY, 14853
kleinber@cs.cornell.edu

ABSTRACT

In many applications we have a social network of people and would like to identify the members of an interesting but unlabeled group or community. We start with a small number of exemplar group members – they may be followers of a political ideology or fans of a music genre – and need to use those examples to discover the additional members. This problem gives rise to the seed expansion problem in community detection: given example community members, how can the social graph be used to predict the identities of remaining, hidden community members? In contrast with global community detection (graph partitioning or covering), seed expansion is best suited for identifying communities locally concentrated around nodes of interest. A growing body of work has used seed expansion as a scalable means of detecting overlapping communities. Yet despite growing interest in seed expansion, there are divergent approaches in the literature and there still isn't a systematic understanding of which approaches work best in different domains.

Here we evaluate several variants and uncover subtle trade-offs between different approaches. We explore which properties of the seed set can improve performance, focusing on heuristics that one can control in practice. As a consequence of this systematic understanding we have found several opportunities for performance gains. We also consider an adaptive version in which requests are made for additional membership labels of particular nodes, such as one finds in field studies of social communities. This leads to interesting connections and contrasts with active learning and the trade-offs of exploration and exploitation. Finally, we explore topological properties of communities and seed sets that correlate with algorithm performance, and explain these empirical observations with theoretical ones.

We evaluate our methods across multiple domains, using publicly available datasets with labeled, ground-truth communities.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database applications—*Data mining*

Keywords: Seed set expansion, Ground-truth communities

KDD'14

1. INTRODUCTION

There are many settings in which we are interested in accessing or studying a group of people in a social network, but instead of the full membership of the group we know only a few examples. A natural goal in this case is to expand these examples into a larger set that approximates the full extent of the group, and this goal has been the focus of recent work on *seed set expansion* in networks. Phrasing the problem slightly informally for purposes of this discussion, we are given a graph G that contains a group of nodes C whose identities we'd like to uncover, and we are told the identities of a small subset S of C. Given a budget k, can we find k additional nodes such that as many of them as possible come from C?

Examples of the seed set expansion problem are numerous. For example, recent work studying political activism has started from a small set of representative of each competing ideology, collected through detailed field work, and then attempted to expand these representatives into the larger groups that they come from [18]. Recommendation tools for forming on-line groups have the potential to collect a few initial suggestions from a user and then produce a longer list of recommended group members. Similarly, a marketer may want to expand a set of a few interested consumers of a product into a longer list of people who might also be interested in the product. Seed set expansion has also been used to infer missing attributes in user profile data [14] and to detect e-mail addresses of spammers [20]. Nor are the applications limited to social networks; as we will see below, we could ask similar questions in which we start with a few items such as products for sale, and we then attempt to use a co-purchase network to expand these items into a product category that contains all of them.

It is useful to note a few properties of the seed set expansion problem, consistent with these sources of motivation. First, we focus on cases in which the expansion is guided by an underlying graph structure — the basic premise is that if a person or item v is "tightly linked" in the graph to many members of the group C, then this provides evidence that v may also be a member of the group. Second, the goal in the seed set expansion problem as it has been studied in prior work — and the goal we pursue here — is neither to find the full extent of the group C nor to sample uniformly from it, but instead to "collect" a fixed number of members from it with as little error as possible.

The present work: Principles for seed set expansion. The number of approaches to seed set expansion has proliferated rapidly, but there is still very little understanding of the principles through which we can reason about trade-offs between different approaches or the types of instances on which we can expect good performances. In this work we seek to begin developing some of the principles underlying the seed set expansion problem. Among the

questions that guide our study, we consider the following. Do certain approaches to seed set expansion produce consistently better results than others, across a range of domains? Can characteristics of the initial seed set S help us understand when seed set expansion will be effective? And how do structural characteristics of the group C affect the quality of the solution?

We can go further in our analysis by taking into account the following issue. If we think about many of the applications that motivate the seed set expansion problem, there is a potentially rich interaction available between the expansion algorithm and the "expert" who can recognize members of the group C. Consider for example the problem of identifying members of political movements noted earlier [18]. Here there is a domain expert who has provided the initial representatives of a group, and if we are trying to expand these representative members into a larger set, we may well have the ability to adaptively query the expert — a few nodes at a time — and make future choices based on the result of this feedback. There is thus an opportunity to incorporate such interaction between the algorithm and the domain expert into the formalism of seed set expansion. Such interaction clearly has a structure similar to work in active learning, although we should emphasize that unlike traditional work in that domain, we are not seeking a classification of the full underlying graph, nor do we have a subset of the data available for training; rather, we want to collect a set of nodes from C based only on the initial examples S.

The present work: Overview of results. We first consider a wide range of techniques that have been used in prior work for seed set expansion, applying them to three main datasets: two social networks (a co-authorship network among researchers and the YouTube social network among its users), and a product co-purchase network in which the groups are product categories. We find that measures based on PageRank are by far the most effective. Moreover, almost all of the performance gains from PageRank come from running just two or three iterations of the PageRank update rule — a finding that is novel to the best of our knowledge, and consistent with our analysis of where PageRank is achieving most of its relative performance gains over more local neighbor-based methods, on nodes that are outside the immediate vicinity of the seed set S.

We then consider how properties of the seed set affect performance. In thinking about trade-offs here, it is useful to consider the interaction between the expansion algorithm and the domain expert discussed above: in that context, our motivation is to identify how a domain expert should best use their knowledge to compile a seed set of members. In practice we are highly constrained by those community members of which a domain expert has knowledge. One natural question is to ask whether performance is better when S consists of the highest-degree nodes in C or a uniformly random subset of C. In choosing a large degree seed set, we model a domain expert who returns a list of the most popular or most famous nodes in the network. In choosing a random seed set, we model a domain expert who returns a seed set that is representative of the community, to wit, one potentially consisting of high and low degree nodes. We also consider the effect of seed set size, exploring a basic trade-off: if the seed set is too large a fraction of the group, it can be hard to find the remaining members, but if it is too small, then it is not providing a sufficiently useful set of examples for the full extent of the group.

We similarly look at trade-offs in the structural properties of the group C, finding that denser groups — those with a higher ratio of edges to nodes — tend to result in better performance for seed set expansion.

Finally, we look at different ways of managing the interaction between the algorithm and the domain expert. We find contexts in which regularly interspersing expansion steps with queries to the expert can outperform approaches in which the queries are batched in larger blocks. We also find that for our objective function of collecting members of C as quickly as possible, asking the expert about nodes on the "margin" of C can be effective in finding the boundaries of the group, but this benefit is more than offset by the downside of querying the expert about a greater number of nodes that turn out not to be in C.

Dataset	Nodes	Edges	Communities
DBLP	317080, authors	1049866, co-authorship	13477, conferences
Amazon	334863, products	925872, co-purchased	151037, product categories
YouTube	1134890, users	2987624, friendship	8385, user-defined groups

Table 1: The number and substantive interpretation of nodes, edges, and communities in each network. All sourced from [21].

2. SETUP: DATA, PERFORMANCE, ALGORITHMS

Data. We use network data with ground-truth community membership from the Stanford Network Analysis Project (snap.stanford.edu). Table 1 gives a summary of the datasets used in this paper; see Yang & Leskovec [21] for additional background on these datasets.

Seed Sets and Performance. We are given a graph G that contains a collection of potentially overlapping communities $\mathcal{C}$, and we have an interest in a particular community $C \in \mathcal{C}$. We are given a set of labeled community members $S \subset C$. Thus $C - S$ consists of the unlabeled, not-yet-discovered community members. We have a budget to make a prediction of size k, and we will call the prediction P. We wish to maximize the recall, $|P \cap C|/|C - S|$, i.e. the fraction of the unlabeled community recovered by the algorithm.

Unless otherwise specified we choose S to be a random subset of C of size $|C|/10$.

Communities. From Table 1 we see that there are roughly 10^4-10^5 labeled, ground-truth communities in each dataset. These communities vary in size from 6 to roughly 10^4. In this paper we focus on the 600 communities closest in size to $m^{3/4}$, where m is the size of the largest community; let us call this set $\mathcal{C}^{600}_{3/4}$.[1]

Stopping criteria. Here we model the scenario in which a researcher knows that there are approximately k community members, and so they select the top k results from their choice algorithm, A, as the predicted community. That is, we choose a simple

[1]Given the same number of guesses, it would be unfair to compare the recall of an algorithm on a community of size 10000 with one of size 100. We find that there are 600 communities centered about this log-space third-quartile all close enough in size that such biases do not taint our results. Rather, thanks to the large number of moderately sized communities, we are able to estimate performances with good standard error estimates. In point of fact, good statistical convergence that distinguishes the various algorithms can be achieved with only 20 communities, and so our consideration of all 600 provides a large extra margin.

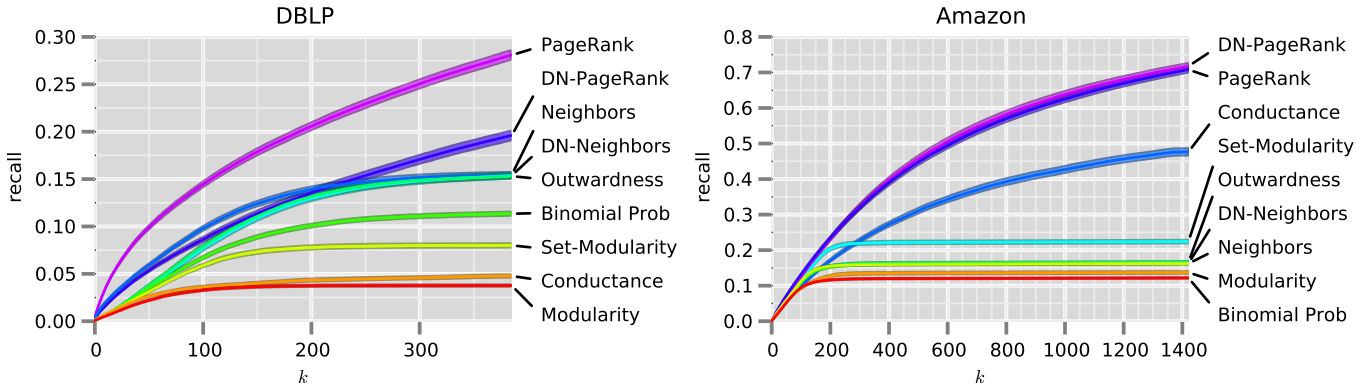

Figure 1: Recall averaged over $\mathcal{C}^{600}_{3/4}$. Rankings for YouTube are the same as for DBLP. The envelopes represent two standard errors centered about the mean.

stopping criterion of a fixed number of guesses equal to the size of the central value; for example on $C^{600}_{3/4}$ we fix $k = m^{3/4}$ and set our prediction P to be the k top nodes according to A's ranking. As we will see in Figures 1, 3, and 5, the relative performance rankings are not very sensitive to the choice of k.

This stopping rule has the advantage that it is not sensitive to the topology of the prediction P, which would be undesirable given that the algorithms we compare produce communities with a variety of typical topologies (this is discussed in detail in [1]). We discuss alternate stopping criteria in §7.

3. RESULTS: PREDICTION ALGORITHMS

3.1 PageRank's success

Figure 1 shows the recall values for a wide range of algorithms detailed in §7 and the appendix. Variants of PageRank — in which we rank by the stationary probability of a random walk with restarts originating at the seed set — are the clear winners. This is consistent with PageRank's success in other applications, but it is nonetheless perhaps surprising that it is so much more powerful than other methods that have been used for this problem. We also note that in two of our three domains, pure unnormalized PageRank significantly outperforms variants such as degree-normalized PageRank (DN-PageRank); this poses an interesting contrast to the fact that DN-PageRank rather than pure PageRank has typically been the preferred method for seed set expansion.

We now consider three questions that suggest insights into the structure of the problem and how to the use these approaches in practice.

1. Why does PageRank outperform other methods such as neighbor counting?
2. In computing PageRank with the power method, how many iterations (random walk steps) does it need to take to achieve this high performance?
3. Could variations beyond DN-PageRank and PageRank achieve even better performance?

3.2 Whom does PageRank find?

In addition to the success of PageRank and its variants in Figure 1, it is also striking to see how PageRank climbs smoothly with k in contrast with neighbor-counting methods that flatten abruptly as we increase.

Looking into this behavior helps us understand where PageRank gets some of its power. In particular, we ask which true positives

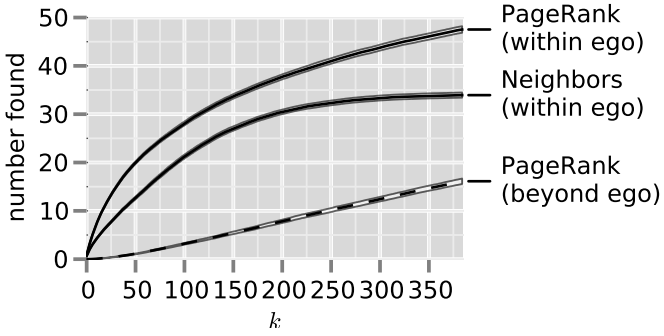

Figure 2: Number of community members found within $\mathrm{ego}(S)$, the seed set's ego network, and outside of it for positive PageRank and positive neighbor counting. Results are averaged over $\mathcal{C}^{600}_{3/4}$. Envelopes represent two standard errors centered about the mean.

are found by PageRank compared to Neighbors, categorizing based on whether they belong to $\mathrm{ego}(S)$, the set of nodes directly adjacent to the seed set S. Neighbor-counting methods cannot effectively use information about nodes outside $\mathrm{ego}(S)$. Is this hindrance the sole factor underlying PageRank's advantage over Neighbors?

Figure 2 untangles this issue: first, we see that PageRank's rate of discovery of members within $\mathrm{ego}(S)$ is significantly higher than that of Neighbors; second, we see that it finds true positives outside the ego at a constant linear rate. So in addition to having this broad reach beyond the ego, which we expected, PageRank is even better at identifying which members of the seed set's immediate neighbors are true positives. PageRank's success over neighbor-counting is thus both inside *and* outside $\mathrm{ego}(S)$.

3.3 How many steps does it take to get to the community?

Next we consider a question regarding PageRank itself: in computing PageRank with the power method, how many random walk steps are needed for PageRank to realize its maximum performance? This is a basic question about PageRank's iterative nature, and the concrete performance measures underlying our problem formulation make it natural to evaluate the question in this context.

The results in Figure 3 indicate that after only three random walk steps PageRank's performance has converged to its upper limit, and it is already close to this limit after two steps. It is striking that

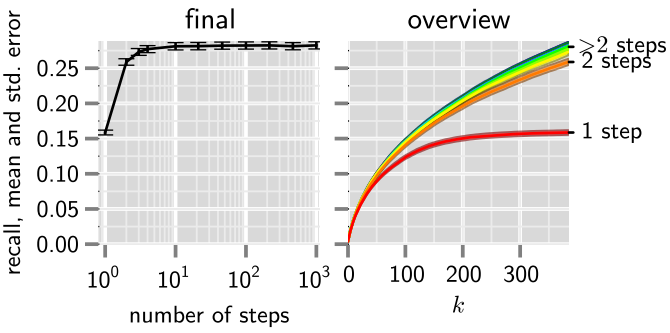

Figure 3: Comparison of PageRank performance for a variety of walk lengths. For each community the same random seed set was used as the walk length was varied. Results are averaged over $\mathcal{C}^{600}_{3/4}$ and the envelopes represent two standard errors centered about the mean.

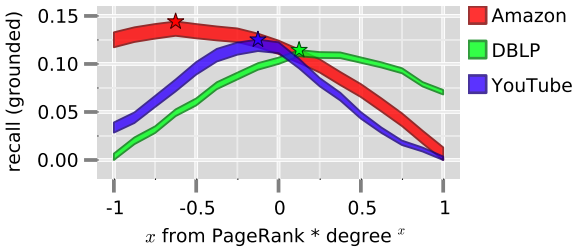

Figure 4: Mean performance as a function of x, where x is a variable exponent in the sorting heuristic PageRank*degree^x. The performances have been shifted vertically such that the lowest performance is grounded at 0, resulting in shifts for Amazon, DBLP, and YouTube of 0.52, 0.18, and 0.049, respectively. The star symbol indicates a curve's maximum. Results are averaged over $C^{600}_{3/4}$ and the envelopes represent two standard errors centered about the mean.

most of PageRank's power on these networks comes from just its first few iterations. To appreciate this we consider PageRank's interpretation at each step and the corresponding performance. 0-step PageRank represents random guessing. 1-step PageRank is closely related to DN-Neighbors[2] — and indeed the performance curve of 1-step PageRank has the same "flattening out" that stood out in the performance curve for neighbor-counting, as well as a comparable final performance value. 2-step PageRank reaches one step beyond $\mathrm{ego}(S)$ and in Figure 3 we see that in the transition from 1 to 2 steps PageRank's performance exhibits a dramatic increase, nearly reaching its full potential. This indicates that most of the members found by PageRank are within 2-steps of the seed set. Finally, 3-step PageRank yields PageRank's full potential, and t-step PageRank continues at this level as $t \to \infty$.

3.4 Variations on PageRank

To normalize or not to normalize. Not to normalize. As mentioned, essentially the only PageRank-derivative used in the literature for community detection by seed set expansion has been DN-PageRank [3, 16, 19, 21]. Yet PageRank yields much higher performance than degree-normalized PageRank in DBLP and Youtube and they reach a tie on the Amazon network.

In Figure 1 we find that in DBLP and Youtube (not pictured) unnormalized PageRank, or simply PageRank, find true community members with greater accuracy than degree-normalized PageRank does. This performance increase is robust after controlling for and considering all community sizes, and it is true in both "easy to detect" and "hard to detect" communities. Indeed, PageRank is best or tied for best on roughly 80% of the communities, on an instance-by-instance basis.

On the Amazon product network, in contrast, PageRank and DN-PageRank reach a statistical tie. It would be interesting to understand the differences in domain that lead to this, including the natural contrast that Amazon is a network on items for purchase, rather than a social network on people as in both DBLP and YouTube.

A continuum: degree-normalization to amplification. Finally we note that DN-PageRank and PageRank are two special cases of using the sorting metric $\rho \cdot d^x$ with $x = -1$ for normalization and $x = 0$ for pure PageRank. It's therefore natural to consider the performance for all values of x, and we show this for all three datasets in Figure 4.

As we see there, the optimal exponents for DBLP and YouTube are both close to 0, indicating the power of unnormalized PageRank on these two social networks, whereas in Amazon the results were statistically indistinguishable for exponents $x \in [-1, 0]$. It is interesting to note that the optimal exponent x in DBLP is in fact slightly positive — in other words, rather than normalizing PageRank, the optimal strategy is to inflate it slightly. This can perhaps be motivated by the fact that many of the false positives being recovered by the algorithm are low degree nodes.[3]

3.5 Combining Multiple Measures

A natural extension of the current framework is to treat each of these network measures as a feature, and to choose nodes for C by training a classifier on the labeled examples and classifying the remaining nodes each according to their corresponding feature vector. We tried this using a support vector machine (SVM) on a feature in which node v's features are the values of the network measures (PageRank, Neighbors, etc.) evaluated on v. For example, for a simple classifier that combines Neighbors and PageRank, the feature vector for a node u was $[ego(u), \rho(u)]$ and the feature matrix used to train the classifier was $(|S| + |N|) \times 2$. The result is interesting in the negative direction: we were not able to realize any performance gains by combining multiple measures. Rather, the higher-dimensional classifiers performed only as well as its best-performing single-dimensional classifier submember. For example,

[2]But instead of normalizing by the target's user degree, the normalization happens with respect to the outgoing nodes

[3]Who are the false positives?
Note that because the communities in these datasets are overlapping, the nodes recovered by the algorithm should really be classified as being one of three types: true positives, false positives, and *neutral positives*. Neutral positives are nodes that are in some community together with the seed node, simply not in the community of interest C. In that sense, when the algorithm recovers a neutral positive it is accurately discovering information about the graph's community structure. If we relabel the original group of false positives into neutral positives or false positives, we find that in DBLP the 'real' false positives have a much lower average degree and very low variance in degree compared to the neutral or true positives. This distinction is not evident if one only considers the original binary labeling of being in the target community or not. That PageRank is making most of its big mistakes on low-degree nodes motivates the slight degree-amplification that we see is optimal for DBLP in Figure 4.

the 2D SVM consisting of PageRank and Neighbors performed as well as PageRank, and the 2D SVM consisting of Neighbors and BinomProb performed as well as Neighbors.

To construct the classifier we build a feature vector for each of the PageRank-based and Neighbor-based methods, e.g. all the algorithms except Conductance and Modularity. (The latter were excluded because they do not assign attributes for every node in the graph – only for ones local to the seed set.) We build the feature vectors by seeding each of the algorithms with 25% of the input nodes; we reserve such a large fraction so as to emphasize teaching the algorithm about the attributes of 'hidden' positive members, rather than seeded ones (which will typically have much larger values of, for example, PageRank). To choose the C value for the SVM classifier we perform 3-fold cross validation with a 75/25 train/test split. We consider linear and radial basis function kernels and normalize all features to have unit $||L_2||$ norm before training.

Negative examples and information. Note that to train the SVM we require both positive and negative examples, and so for the learning framework we introduce the notion of a negative seed set, T. Much like S, the seed set of known community members, T consists of nodes that are known from the outset (e.g. thanks to a domain expert's knowledge) to be non-members. To choose the negative seed set we tested the same heuristics as we did for the positive seed set, namely random nodes as in §2 and higher degree nodes as in see §4.1.

The introduction of the negative examples lead us to consider the possibility that the information about their non-membership could help improve classification. For example, just as we expect nodes tightly knit with the positive seed set S to more likely be members themselves, we expect nodes tightly knit to the negative seed set T to be less likely to be members.

We used SVMs to empirically verify both of these intuitions. That is, we seed PageRank with the negative seed set T and call the resulting metric on the nodes Negative-PageRank. For the purposes of this discussion, we call the original PageRank seeded with S Positive-PageRank. We then train an SVM using these two attributes as node features, and find that the SVM's weight vector has a positive coefficient for the Positive-PageRank feature and a negative coefficient for the Negative-PageRank feature, as expected.

However, the introduction of Negative-PageRank ultimately had no significant effect, neither improving nor hurting the performance of the classifier. The same is true regarding analogous versions of Negative-Neighbors and Negative-BinomProb).

4. RESULTS: SEED SETS

Having looked at the relative performance of different algorithms for seed set expansion, we now consider the effect that different structural properties of the seed set itself can have on performance.

4.1 Heuristics for seed set selection

We begin by considering the effect of the node degrees in the seed set. In Figure 5 we see that for seeding PageRank it is highly advantageous to use a random positive seed set compared with one consisting of high-degree nodes. Though we have not pictured it here, this result holds for all domains, community sizes, and high-performing algorithms. It is true for the neighbor counting metrics as well (with the exception of binomial probability), however for the neighbor counting metrics the improvement is not as striking.

In many settings we should expect to have relatively little control over which members are in the seed set: the community is hidden to us and the seed set consists of those members for whom we happen to have labels. However, the particular contrast we analyze here,

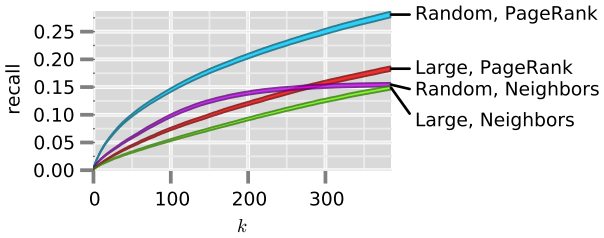

Figure 5: Comparison of algorithm performance for positive seed sets composed of random versus high-degree nodes, using PageRank and Neighbors. Results are averaged over $\mathcal{C}_{3/4}^{600}$ from DBLP and the envelopes represent two standard errors centered about the mean.

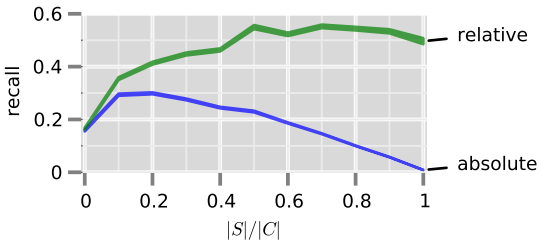

Figure 6: Performance as a function of the fraction of the community used to seed PageRank for all DBLP communities. For each community C ten seed sizes were tested with uniform spacing between 1 and $\lfloor 0.99|C| \rfloor$. We distinguish between two types of recall: relative recall is $|P \cap C|/|C - S|$, i.e. the fraction of unlabeled nodes that were discovered; whereas absolute recall is $|P \cap C|/|C|$, the fraction of the total community recovered during the evaluation stage. Envelopes represent two standard errors centered about the mean.

between random and high-degree nodes, corresponds naturally to two distinct scenarios for interacting with a domain expert: if the expert knows the most popular or most famous nodes in the community, this would lead to a high-degree seed set, while if the expert returns a more representative seed set, this would be modeled by a set consisting of random nodes. Some experience indicates that experts will often have a tendency to identify high-degree members, which, we see here, is not in fact the most effective way to gather a seed set for further expansion.

This lesson is an important heuristic to consider. Given that our recommendation is to use PageRank over more local methods when possible, it would also be advantageous for domain experts to heuristically search for nodes with a more diverse degree distribution, rather than searching for and validating the membership of those with highest degree. Note that even when this is not possible, and PageRank is seeded with a large degree seed set, it still outperforms Neighbors as a method.

4.2 Seed set size and performance

In Figure 6 we examine the algorithm's performance as a function of the fraction of the community C used in the seed set S, $|S|/|C|$. We evaluate performance using two measures of recall: the *relative recall* $|P \cap C|/|C - S|$, and the *absolute recall* $|P \cap C|/|C|$. We find that the relative recall eventually plateaus as $|S|/|C|$

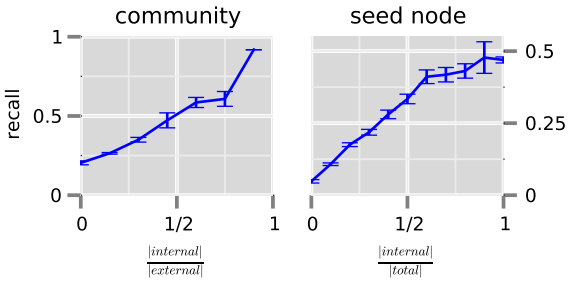

Figure 7: Left: Performance as a function of the target community's ratio of internal to external edge density. Right: Performance as a function of the fraction of edges the seed has within the community. Both plots use PageRank as the sorting metric and test on $C_{3/4}^{600}$.

is increased whereas the absolute recall has an interior maximum.

Intuitively we expect this interior maximum in absolute performance: by starting off with very little of the community C we will be lacking sufficient information about C and will find it difficult to accurately characterize and identify additional members. In the other extreme, if we begin with all but a few of the members, we are inherently limited in the number of additional members we can discover. Thus we expect there to be some internal maximum in absolute performance, as we see at $|S|/|C| = 0.1$ in Figure 6.

For relative recall, in contrast, we find that performance simply plateaus after a certain point, meaning that the additional information is neither helping nor hindering our relative rate of discovery.

4.3 Internal seed set structure

Finally, we consider how well the seed set is connected both internally and to the (unobserved) remainder of the community. Although we have seen that seed nodes with overall high degree point to reduced performance, we find here that performance is significantly higher when a large fraction of the seed nodes' edges are to nodes that lie in the community. Figure 7 (right panel) shows the performance in terms of this fraction; the left panel of the figure establishes a related point, that performance is better when the community C has a high ratio of internal edge density to external edge density.

These findings highlight several points. First, it suggests that in practice one can form an *a priori* estimate of the success of seed set expansion on a per-instance basis, based on structural properties of the seed set and/or the community, if one has estimates about these edge density parameters. Second, the strong relation to performance forms an interesting connection to mathematical work on community detection. The literature has emphasized that a good mathematical definition of a community is a set of nodes whose internal edge density is higher than its external one. It is interesting then that in these communities which are defined only by a shared qualitative property of member nodes that this canonical metric emerges as being correlated with high performance.

Finally, as with some of our earlier findings about seed set structure, we cannot necessarily control the seed set properties with which we make our prediction. But it does suggest a heuristic in which one can try to elicit from the domain expert a set of seed nodes that have good internal connectivity into the rest of the community, relative to their connectivity to the rest of the graph. The community results are a reminder that when searching for nodes in a community we should only expect high prediction accuracy if we can also expect that the members for whom we are searching form a densely connected subset of the graph.

5. RICHER INTERACTIONS FOR PRODUCING LABELS

We have been thinking about seed set expansion in terms of interaction with a domain expert who is able to provide us with an initial set of examples. In this section we enrich this interaction by asking the expert to initially label some number of selected nodes beyond the seed set before we make further guesses about nodes likely to belong to the community. Thus, there are three types of nodes here, in order: (i) the initial seed set; (ii) the nodes explored in the *interactive* phase, when the expert is being actively queried; and (iii) the nodes explored in the *non-interactive* phase, when a set of nodes is guessed and only evaluated afterward for membership in C.

This second, additional round of interaction introduces several further questions. (1) How many nodes should be labeled in the interactive phase? We'll call this quantity the query budget. (2) Which nodes should be labeled in the interactive phase? We'll call this function the strategy. (3) Should the nodes in the interactive phase be labeled all at once? Or is there a performance gain to be had by introducing a feedback loop, in which at every iteration b nodes (strictly fewer than the query budget) are chosen according to the strategy, labeled, and used to refine the strategy input on the next round?

Finally, there are two ways of evaluating the algorithm, and we will consider both. One approach, in keeping with the initial motivation for seed set expansion, is to say that all nodes after the initial seed set count toward the performance of the algorithm, including those in the interactive phase. The other approach, more akin to a training/test split, is to say that the nodes in the interactive phase are purely for calibration, and the performance of the algorithm is only evaluated by its success in the non-interactive phase.

The former case is natural for settings where the goal is simply to collect members of the community — for example, in the case of a marketer who simply wants as large a set of likely purchasers as possible. In this case, there is an interesting trade-off between collecting nodes likely to belong to C in the interactive phase versus asking the expert about nodes that are less likely to belong to C, but which will help refine the boundary of C. This will be one of the main trade-offs we explore in this section.

Computational experiments. We fix the size of the query budget to be that of the size of the seed set, that is, 10% the size of the community. We now focus on questions (2) and (3) above.

We consider four heuristics for choosing which nodes to query based on the values of one of our classifiers for membership in C: (1) nodes on the boundary of the decision function; (2) nodes most likely to be positive, as predicted by the classifier; (3) nodes most likely to be negative, as predicted by the classifier; (4) random nodes. We model this selection process by viewing (1)-(4) as having the associated probabilities $(p_0, p_+, p_-, p_*) = p$, and in each step of the interactive phase selecting, for example, $b_+ = p_+ b$ nodes predicted to be positive. We measure performance throughout this 3-dimensional parameter space ($p_0 + p_+ + p_- + p_* = 1$). We start with two observations that allow us to simplify our discussion and exploration of this parameter space:

1. It is never advantageous to have $p_* > 0$, that is, there is no benefit to querying random nodes instead of putting that probability mass on some other dimension. This is not surprising and in fact this parameter was introduced as a baseline.

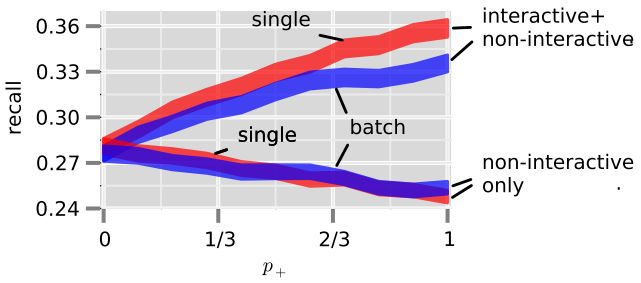

Figure 8: Performance of PageRank with two querying strategies and two evaluation metrics. In *batch* mode, we use the query budget in one sweeping request. *Single* mode is the other extreme: we query the expert for one node's label, reseed PageRank with the updated label, and repeat for as many steps as the query budget allows. Here we fix the query budget to be 10% the expected size of the community. The lower *non-interactive only* curves indicate the final performance of the classifier, not including any positive examples recovered during the interactive phase. The upper *interactive + non-interactive* curves include true positives recovered during both the interactive and non-interactive phases.

2. It is never advantageous to have $p_- > 0$, that is, there is no benefit to querying nodes that are most certainly not in the community. While this is not very surprising we did find it worth considering the possibility that having very clear examples of what community members do not look like could improve the classifier.

These observations imply that, for the scenarios we have defined, $p_- > 0$ and $p_* > 0$ are both dominated by strategies with $p_- = p_* = 0$. Thus we are left with a one-dimensional parameter space, $1 = p_+ + p_0$. In Figure 8 we explore how the performance is affected by labeling boundary nodes (via p_0), positive nodes (via p_+), or some combination thereof. We find that the optimal strategy depends on how performance is being measured:

1. If nodes found in both the interactive and non-interactive phases are counted towards performance, then $p_+ = 1$ is optimal.

2. If only nodes from the non-interactive phase count towards performance , then querying all nodes on the classifier boundary is optimal ($p_+ = 0$).

In the next section, we will see how the trade-off between these two results is reflected in the contrast between two different ways of formulating the problem of identifying nodes in a community.

Note, however, that when the interactive phase does not count towards performance, only a small performance gain is had by $p_0 = 1$ compared to $p_0 = 0$, though the gain is statistically significant (compare the performance of the lower curve on the left and right extremes in Figure 8). In contrast, when the goal is to maximize the number of community members collected (the upper curves in Figure 8), $p_+ = 1$ is a clearly dominant strategy.

Finally we address question (3), whether there is a performance gain to be had by introducing a feedback loop in the labeling stage. We find that it is advantageous to use smaller block sizes when nodes found in the interactive phase count toward performance and the strategy $p_+ = 1$ is used (which is the optimal one for this case). In all other cases we find that the performances are statistically indistinguishable. The performance gain in the first case is all had in the interactive phase; that is, smaller block sizes do not improve the final classifier, but they do yield improved likelihood of finding positive nodes in the interactive stage. For the other scenarios there are either few positive nodes to be found by that strategy (i.e. querying on the boundary) or the nodes found in this stage do not count towards performance and so are irrelevant for evaluation purposes. This is also true in both cases that we discussed above: querying the most positive nodes ($p_+ = 1$) and querying on the boundary ($p_0 = 1$).

6. SEED SET EXPANSION: THEORETICAL RESULTS

If we abstract beyond the specific methods used for seed set expansion, our discussion thus far has highlighted a number of themes implicit purely in the formulation of the problem itself — the difference between collecting a fixed number of nodes from a group C and finding the full set C; the trade-off, as in the previous section, between exploring in the vicinity of nodes known to be in C versus exploring near the estimated boundary of C; and the role of negative information, about non-membership in C.

We now consider a theoretical framework that seeks to highlight how these trade-offs and contrasts work across the different problem formulations. At the top level, it will be based on the distinction between the following two problems: *enumeration*, in which we want to find the full set C; and *seed set expansion*, in which we want to collect "many" members of C but not the full set.

Basic Set-up. To set up these problems, let us assume we are given an undirected n-node graph $G = (V, E)$, and a subset $C \subseteq V$ is specified by a *membership oracle* that takes a node $v \in V$ and reports whether or not $v \in C$. We are also given a seed set $S \subseteq C$ of nodes that we know at the outset to belong to C. Finally, we will make the assumption that C is a connected set of nodes in G.[4]

In these terms, *enumeration* is now the problem of finding all the nodes of C using as few queries as possible to the membership oracle. *Seed set expansion*, in contrast is the problem in which, given a "budget" k, we want to find as many nodes of $C - S$ as possible using at most k queries to the membership oracle.

A Motivating Example. To get an initial picture of the contrasts between these two problem formulations, let's consider them on an extremely simple graph, the n-node cycle, which just consists of nodes $v_0, v_1, \ldots, v_{n-1}$ such that v_i is connected to v_{i-1} and v_{i+1} (addition modulo n). The group C we are trying to discover is a connected subset of the cycle (and hence a contiguous interval of nodes on it).

Suppose we are given a single seed node $v_j \in C$. Then the optimal algorithm for the seed set expansion problem it to begin querying nodes for membership in C starting at v_j and moving in a clockwise direction. The first time we come to a node $v_\ell \notin C$, we know we have fallen off one end of the interval defined by C. We then go back to v_j and do the same thing in the counter-clockwise

[4]We view this as a reasonable approximation to the real problem in practice, since many of the groups C we are interested in studying will have a giant component $\tilde{C} \subseteq C$. Unless we have seed notes in the smaller components, it is not reasonable in any case to try discovering them; thus, we can view our problem as operating on this giant component. Indeed, some formulations of the seed set expansion problem have explicitly described it as searching for a specific component of the group.

direction. In this way, either we discover all of C (if our budget k is large enough), or else we collect nodes at almost full efficiency; aside from the node v_ℓ, every node we query is in C, and so we collect at least $k-1$ nodes with our budget of k.

An efficient algorithm for the enumeration problem has a quite different structure. First, for the enumeration problem it is natural to make an extra assumption that we didn't need for the seed set expansion problem — that we also know a node $z \notin C$. (Otherwise, we would have to begin with an essentially brute-force search for a non-member of C.) Given $s \in C$ and $z \notin C$, there are two paths of C that run between them: one clockwise from s to z, and the other one counter-clockwise from s to z. We perform binary search on the first of these paths to find a pair of adjacent nodes (v, w) such that $v \in C$ and $w \notin C$. We do the same on the other path, and thus find the two endpoints of the interval defining C in $O(\log n)$ queries to the membership oracle.

Contrasting algorithms. Let us now contrast the approaches to these two problems. When $k \ll \log n$, seed set expansion collects nodes of C with almost no waste (i.e. almost no querying of non-members of C), while the efficient algorithm for enumeration could spend its first $\Omega(\log n)$ without ever identifying another member of C. On the other hand, when $k \gg \log n$, seed set expansion is collecting nodes of C one-by-one, whereas after the initial investment of $O(\log n)$ probes, the algorithm for enumeration implicitly knows all of C even though it hasn't visited all of its nodes explicitly.

These two contrasting strategies also relate to some other themes from earlier sections. As in the previous section, the seed set expansion algorithm does well by focusing attention near the nodes of C that it already knows about, whereas the enumeration algorithm focuses attention on farther-away nodes as it attempts to find the boundary of C. And negative information is not particularly relevant for the seed set expansion algorithm, whereas it is crucial for the efficiency of the enumeration algorithm — a reflection of the role that negative information played in our empirical results as well.

Thus far, however, these insights are all based on an extremely simple instance of the problem — finding a contiguous interval on the cycle. Do the same contrasts apply in other graphs? We now prove two theorems establishing that in fact they do, in a very strong sense: for *every* graph G, one can obtain contrasting and asymptotically optimal bounds for seed set expansion and enumeration that naturally extend the results we obtained for the simple case of the cycle.

We stress that in this theoretical analysis, we are focusing on comparing the consequences of the two problem formulations, seeking algorithms for each that are asymptotically optimal in the worst case, rather than trade-offs among specific heuristics for the problems.

General theorems. We continue with the set-up defined at the outset, with an arbitrary graph $G = (V, E)$, a connected set of nodes $C \subseteq V$ that we want to discover, and a given seed set $S \subseteq C$. A key structure for analyzing our algorithms will be the set of edges $\delta(C)$, consisting of all edges that have one end in C and the other end not in C; a central quantity for parametrizing the performance of the algorithms will be $c = |\delta(C)|$.

We begin with the generalization for the seed set expansion problem, essentially showing that there is an algorithm that can collect at least $k - c$ nodes of C. This is the same sense in which the algorithm on the cycle was collecting nodes a perfect efficiency except for the queries in which it fell off the end of C.

THEOREM 1. *Given a budget of k queries to the membership oracle, there is an algorithm that finds at least $\min(k-c, |C-S|)$ nodes in $C - S$. (In other words, it either finds at least $k - c$ nodes of $C - S$, or else it finds all of $C - S$.) This is asymptotically tight in the worst case.*

PROOF. We show that the guarantee in the statement of the theorem will be achieved by any algorithm with the following structure: for k iterations, look for a node $v \in C$ connected by an edge to a node w whose membership in C is not yet known, and query w. If at any point before the k iterations are over, there are no such pairs (v, w), then the algorithm can stop and declare that it has found all of C.

Let us first verify why the algorithm is correct when it concludes it has found all of C. Since $G[C]$ is connected, as long as some nodes of C have not yet been found, there must exist an edge (v, w) such that $v, w \in C$, with node v already known to belong to C and node w not yet known to belong to C. Hence as long as all of C has not yet been found, the algorithm can execute another iteration.

Now, in each of the k iterations for which the algorithm does not discover a new node in $C - S$, it instead finds an edge $(v, w) \in E$ for which $v \in C$ and $w \notin C$. Thus $(v, w) \in \delta(C)$. Since there are only c edges in $\delta(C)$, there can be at most c iterations in which the algorithm does not find a new node in $C - S$; in the remaining iterations, at least $k - c$ in total, it must discover a new node in $C - S$, and this establishes the performance guarantee of the algorithm.

To see why the bound of $k - c$ is tight in the worst case, consider the star graph, equal to a tree with a central node v connected to $n - 1$ other nodes $w_1, w_2, \ldots, w_{n-1}$. If $S = \{v\}$ and C consists of v plus all but c of the leaf nodes, then in the worst case the algorithm will discover all c nodes not in C before moving on to any nodes in $C - S$.

We now give the contrasting generalization for the enumeration problem — that all of C can be found with $O(c \log n)$ queries. For this algorithm, as in the case of the cycle, we need to assume the presence of negative information; in particular, we assume there is a set of nodes $Z \subseteq V - C$ that is rich enough in its coverage that Z contains at least one node from each component of $G - C$.

THEOREM 2. *Given seed set $S \subseteq V$ and negative set $Z \subseteq V$ satisfying the assumptions above, there is an algorithm to find all the nodes of C using at most $O(c \log n)$ queries to the membership oracle. This is asymptotically tight in the worst case.*

PROOF. The algorithm works as follows. Let s be any node in the given set $S \subseteq C$; we say that an *s-Z path* is any path whose first node is s and whose last node belongs to Z. Edges will get *marked* during the execution of the algorithm; initially all edges start out unmarked. While there is an s-Z path P in G consisting entirely of unmarked edges, we perform binary search over the ordered sequence of nodes on P to find an edge (v, w) on P for which $v \in C$ and $w \notin C$. We then mark this edge (v, w). This is one iteration of the algorithm, and it uses $O(\log n)$ queries to the membership oracle in order to perform the binary search.

How many iterations can there be? Each iteration marks an edge in $\delta(C)$ that was previously unmarked; since $|\delta(C)| = c$, there can be at most c iterations. It follows that in total the algorithm performs at most $O(c \log n)$ queries.

Let F be the set of marked edges when the algorithm terminates, and let $U \subseteq V$ be the connected component of $G - F$ that contains the node s. We claim that $U = C$. Indeed, suppose there were a node $u \in C$ such that $u \notin U$. Then since $G[C]$ is connected, there is an s-u path consisting entirely of nodes in C, and hence using

no marked edges. This contradicts the assumption that $u \notin U$. Conversely, suppose there were a node $u \in U$ such that $u \notin C$. Let z be a node of Z that belongs to same component of $G - C$ that u does. There is a u-z path Q such that all nodes belong to $V - C$; since each marked edge contains a node of C, there are no marked edges on Q. Concatenating Q with an s-u path using no marked edges, we get an s-z path using no marked edges, contradicting the termination of the algorithm.

These arguments show that $C \subseteq U$ and $U \subseteq C$, so $U = C$ and hence the algorithm produces the correct output set C.

Finally, we argue that there exist instances with a graph $G = (V, E)$ and a set $C \subseteq V$ for which $\Omega(c \log n)$ queries are required. One such graph is a collection of c parallel paths each of length n/c, that each run from s to z, but which otherwise have no nodes or edges in common. Any set C obtained by choosing a prefix of each s-z path, and taking the union of these c prefixes, will have $s \in C$ and $z \notin C$, with $\delta(C) = c$. There are $\Omega(n^c)$ such sets C, and hence any algorithm that uniquely identifies one of them through a sequence of yes/no questions to an oracle must make at least $\Omega(\log n^c) = \Omega(c \log n)$ queries in the worst case.

7. RELATED WORK

The seed set expansion problem has its roots in a number of overlapping areas, including the problem of identifying central nodes in social networks [6, 9] and finding related and/or important Web pages from an initial set of query results [10, 15].

In particular, the PageRank algorithm broadened from its initial focus on Web search [15] to also include methods for finding nodes "similar" to an initial root, by starting short random walks from the root and seeing which other nodes were likely to be reached [7]. Spielman & Teng developed methods that started with a seed node and sorted all other nodes by their degree-normalized PageRank with respect to this seed [17]; they also introduced ideas based on truncation of small values, leading to a method known as PageRank-Nibble. Anderson & Lang and Andersen *et al.* built on these methods to formulate an algorithm for detecting overlapping communities in networks [2, 3]; in our evaluation, their method serves as our version of DN-PageRank, short for degree-normalized, personalized PageRank. DN-PageRank was adopted by Leskovec *et al.* [11] and Yang & Leskovec for global and local community detection. In a large comparison study they established DN-PageRank as competitive with METIS [8], a sophisticated and popular graph partitioning algorithm. Finally, Abrahao *et al.* observe that from among approximately ten popular community detection algorithms, ground-truth communities are structurally most similar to the communities discovered by random walk methods [1].

In parallel with the development of PageRank-based methods, another line of work explored methods for seed set expansion by adding nodes to a growing community (or removing them) if a target measure such as conductance or modularity is improved by doing so. Clauset [5] used this idea by adding single nodes to increase modularity; Luo *et al.* [12] allowed for addition and deletion of larger sets; and Mislove *et al.* [14] used greedy node addition to reduce conductance.

Finally, a number of approaches evaluated nodes based on the number of neighbors they had in and out of the community, adding nodes to the community when they optimized a function of these two quantities. Bagrow [4] did this for a measure called *outwardness*, defined as the degree-normalized difference between neighbors inside and outside the community. Mehler & Skiena [13] used several variations of neighbor counting methods for seeded community detection, the main ones being pure neighbor count, neighbor ratio, and binomial probability of neighbor distribution. More recently in 2013 Weber *et al.* used another variation of a neighbor-counting metric to infer the political ideology of Twitter users, based on which community a user retweeted more frequently.

In an analysis of the effect of seed-set structure, Whang, Gleich, & Dillon [19] systematically compared several sophisticated approaches for choosing the seed sets with which to seed PageRank-based measures for community detection. Their methods outperform existing sophisticated methods, but do not significantly outperform the use of random nodes.

Finally, essentially all seed set expansion algorithms need to make a decision about the choice of *stopping criterion* — when does one stop expanding the set? Such a criterion can be treated relatively independently from the choice of expansion rule. Andersen & Lang [3] and Yang & Leskovec [11] choose the first nodes that represent a set with a locally minimal conductance (given that additions happen in the order induced by sorted, DN-PageRank). Mehler & Skiena continue until the mining rate of reserved labeled nodes passes below a certain threshold; Bagrow [4] looks for transitions and cusps in the modularity that one expects at a community border. Others such as Mislove *et al.* [14], Luo *et al.* [12], and Clauset [5] greedily add and subtract nodes from the predicted community until a local maximum is reached. In 2012 Yang & Leskovec used PageRank-type measures to empirically compare different topological parameters to identify community boundaries in real-world data sets. They found that the result was somewhat domain dependent, but that either the set's conductance or its triad participation ratio were, most reliably, high accuracy stopping rules.

8. CONCLUSION

The seed set expansion problem has been gaining visibility as a general-purpose framework for identifying members of a networked community from a small set of initial examples. But subtle trade-offs in the formulation and underlying methods can have a significant impact on the way this process works, and in this paper we have identified several such principles about the relative power of different expansion heuristics, and the structural properties of the initial seed set. Our investigations have involved analyses of datasets across diverse domains as well as theoretical trade-offs between different problem formulations.

There are a number of interesting directions for further work. In particular, the power of PageRank-based methods raises the question of whether these are indeed the "right" algorithms for seed set expansion, or whether they should be viewed as proxies for a richer set of probabilistic approaches that could yield strong performance. Second, the contrast between seed sets consisting of random nodes versus those consisting of high-degree nodes suggests that deeper structural contrasts may be present as well; a richer understanding of the seed sets that lead to the most effective expansions to a larger community could provide useful insights for the application of these methods. And finally, as noted earlier in the paper, nodes in a network tend to belong to multiple communities simultaneously, and a robust way of expanding several overlapping communities together is a natural question for further study.

9. ACKNOWLEDGMENTS

We thank Michael Macy, Silvana Toska, and Shaomei Wu for useful discussions. Research supported in part by an NSF Graduate Research Fellowship, a Simons Investigator Award, a Google Research Grant, an ARO MURI grant, and NSF grant IIS-0910664.

10. REFERENCES

[1] Bruno Abrahao, Sucheta Soundarajan, John Hopcroft, and Robert Kleinberg. On the separability of structural classes of communities. In *In KDD '12*, pages 624–632. ACM, 2012.

[2] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.

[3] Reid Andersen and Kevin J Lang. Communities from seed sets. In *In WWW '06*.

[4] James P Bagrow. Evaluating local community methods in networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(05):P05001, 2008.

[5] Aaron Clauset. Finding local community structure in networks. *Physical review E*, 72(2):026132, 2005.

[6] Charles H Hubbell. An input-output approach to clique identification. *Sociometry*, 1965.

[7] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *In WWW '03*, pages 271–279. ACM, 2003.

[8] George Karypis and Vipin Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

[9] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[10] Jon M Kleinberg. Authoritative sources in a hyperlink environment. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[11] Jure Leskovec, Kevin J Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *In WWW '10*, pages 631–640. ACM, 2010. data source.

[12] Feng Luo, James Z Wang, and Eric Promislow. Exploring local community structures in large networks. *Web Intelligence and Agent Systems*, 6(4):387–400, 2006.

[13] Andrew Mehler and Steven Skiena. Expanding network communities from representative examples. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(2):7, 2009.

[14] Alan Mislove, Bimal Viswanath, Krishna P Gummadi, and Peter Druschel. You are who you know: inferring user profiles in online social networks. In *In WSDM '10*, pages 251–260. ACM, 2010.

[15] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[16] Jason Riedy, David A Bader, Karl Jiang, Pushkar Pande, and Richa Sharma. Detecting communities from given seeds in social networks. 2011.

[17] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *In STOC '04*, pages 81–90. ACM, 2004.

[18] Ingmar Weber, Venkata R Kiran Garimella, and Alaa Batayneh. Secular vs. islamist polarization in egypt on twitter. In *In ASONAM '13*, pages 290–297. ACM, 2013.

[19] Joyce Jiyoung Whang, David F Gleich, and Inderjit S Dhillon. Overlapping community detection using seed set expansion. In *In CIKM '13*, pages 2099–2108. ACM, 2013.

[20] Baoning Wu and Kumar Chellapilla. Extracting link spam using biased random walks from spam seed sets. In *In AIRWeb '07*, pages 37–44. ACM, 2007.

[21] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *In MDS '12*, page 3. ACM, 2012.

APPENDIX

A. SUMMARY OF ALGORITHMS

Here we provide brief summaries of the algorithms used in the main text; for more details see citation. We distinguish between three types of algorithms:

Neighbor counting.
(a) Outwardness, the degree-normalized difference between the number of edges a node has within and without of the labeled community [4];
(b) Neighbors, the number of neighbors one has in the labeled community [13];
(c) DN-Neighbors, the degree-normalized version of Neighbors [13];
(d) BinomProb, the binomial probability that a node is in the community, given the number of neighbors it has in the labeled community [13].

Greedy structural optimization.
(e) Modularity, greedy algorithm: in each step add the node that yields the highest increase in the set modularity of the predicted community, [5];
(f) SetModularity, greedy algorithm: in each step add the nodes that yield a positive increase in set modularity, then remove the set of all nodes whose removal precipitates an increase, [12];
(g) Conductance, greedy algorithm: in each step add the node that yields the most negative change in conductance, [14].

PageRank.
(h) PageRank, implemented here with personalization and computed using the power method and jumpback probability $\alpha = 0.10$, see [7] or [5] for implementation details. For comparison with [2, 21] we also implemented a version with ϵ truncation (semi-accurate estimate of PageRank), however we found that below a $\epsilon \approx 1/|G|$) there were no significant differences in performance, and past this ϵ, the performance steeply plummets to approach that achieved by random guessing.
(i) DN-PageRank, the degree-normalized version of PageRank, [17], also see footnote [5].

[5] Let ρ^t be the tth random walk vector given that the initialization set S is the set of known community members.
Let $\chi(S)$ be an indicator vector where $\chi_i(S) = 1$ if $i \in S$ and 0 otherwise. Let A be the adjacency matrix, where $A_{ij} = 1$ if j links to i and 0 otherwise. The degree of node j is given by $d_j = \sum_i A_{ij}$. The random walk is initialized with $\rho^0 = \chi(S)/|S|$. In step $t+1$ each node i distributes $\alpha\rho_i^t$ probability mass uniformly over the seed set S and $(1-\alpha)\rho_i^t$ probability mass over its neighbors. The corresponding probability transition matrix $M(S)$ is:

$$M_{ij}(S) = \alpha\frac{\chi_i(S)}{|S|} + (1-\alpha)\frac{A_{ij}}{d_j}.$$