Communities from Seed Sets

Reid Andersen
University of California, San Diego
9500 Gilman Dr., Department 0112
La Jolla, CA 92093
randerse@math.ucsd.edu

Kevin J. Lang
Yahoo Research
3333 Empire Avenue
Burbank, CA 91504
langk@yahoo-inc.com

ABSTRACT

Expanding a seed set into a larger community is a common procedure in link-based analysis. We show how to adapt recent results from theoretical computer science to expand a seed set into a community with small conductance and a strong relationship to the seed, while examining only a small neighborhood of the entire graph. We extend existing results to give theoretical guarantees that apply to a variety of seed sets from specified communities. We also describe simple and flexible heuristics for applying these methods in practice, and present early experiments showing that these methods compare favorably with existing approaches.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory—*Network Problems*; H.3.3 [**Information Systems**]: Information Search and Retrieval—*clustering*

General Terms

Algorithms, Theory, Experimentation

Keywords

community finding, link analysis, graph conductance, random walks, seed sets

1. INTRODUCTION

In this paper, we present a detailed study of the following problem: given a small but cohesive "seed set" of web pages, expand this set to generate the enclosing community (or communities). This seed expansion problem has been addressed by numerous researchers as an intermediate step of various graph-based analyses on the web. To our knowledge, however, it has never been called out as an interesting primitive in its own right. As we will show, the mathematical structure underlying the problem is rich and fruitful, and allows us to develop algorithms that perform dramatically better than naive approaches.

The seed expansion problem first came into prominence in 2000, when Jon Kleinberg introduced the HITS algorithm [7]. That algorithm used a search engine to generate a seed set, and then performed a fixed-depth neighborhood expansion in order to generate a larger set of pages upon which the HITS algorithm was employed. This general recipe has since seen broad adoption, and is now a common technique for local link-based analysis. Variants of this technique have been employed in community finding [8], in finding similar pages [1], in variants of HITS, PageRank [13], and TrustRank [5], and in classification [2]. More sophisticated expansions have been applied by Flake et al. [4] in the context of community discovery.

Alongside this body of work in web analysis, the theoretical computer science community has developed algorithms that find cuts of *provably small conductance* within a carefully expanded neighborhood of a vertex [12]. Intuitively, these methods are efficient because they make non-uniform expansion decisions based on the structure revealed during exploration of the neighborhood surrounding the vertex. Our results are based on these techniques, and we present modified versions of these methods and theorems that apply to the seed set expansion problem.

The actual computation is done by simulating a "truncated" random walk for a small number of steps, starting from a distribution concentrated on the seed set. In each step of the truncated walk, probability is spread as usual to neighboring vertices, but is then removed from any vertex with probability below a certain threshold. This bounds the number of vertices with nonzero probability, and implicitly determines which vertices are examined. After each step in the expansion process, we examine a small number of sets determined by the current random walk distribution, and eventually choose one of these sets to be the community for the seed.

1.1 Discussion of seed set expansion

The graphs we consider in our experiments have small diameter and a small average distance between nodes, so the number of nodes within a fixed distance of the seed set grows quickly. Nonetheless, these graphs contain distinct communities with relatively small cutsizes. We assume that the seed set is largely contained in such a community, which we refer to as a target community. In many of the graphs we consider, the seed is contained in a nested sequence of target communities. The results in section 2.5 show that if a good target community exists for the seed, then the random walk method produces some community that is largely contained within this target.

When expanding the seed with a truncated walk, a target community serves as a bottleneck, containing much of the probability for a significant number of steps. Since probability is removed from low-probability vertices, this prevents the support of the walk from expanding quickly beyond the bottleneck. In contrast, expanding the seed set using a fixed-depth expansion entirely ignores the bottleneck defining the community. Some branch of the BFS tree is likely to cross the bottleneck and rapidly expand in the main graph before a large fraction of the nodes in the community have been reached. Thus a fixed depth expansion will be a bad approximation to the community, and might also be an impractically large "candidate set" for further processing.

The goal of examining a small number of nodes is clear enough, but we also need to pin down what we mean by a "good" community containing a given seed set. We will do this by describing the limitations of several possible ways to define good communities.

Several papers including [4] have defined a community to be a subgraph bounded by a small cut, which can be obtained by first growing a candidate set using BFS, and then pruning it back using ST max-flow. Unfortunately, a small seed set will often be bounded by a cut that is smaller than the cut bounding the target community, so the mininum cut criterion will not want to grow the seed set. Flake et al. address this problem by performing this process several times, adding nodes from the candidate set at each step to ensure expansion.

Another idea for ensuring a reasonable expansion of the seed set that is less *ad hoc* than most is to optimize for conductance instead of cutsize alone. Graph conductance (*aka* the normalized cut metric) is a quotient-style metric that provides an incentive for growing the seed set. Unfortunately, the conductance score can be improved by adding barely related nodes (or even a disconnected component) to the seed set. We need to ensure that only *nearby* nodes are added to the seed set.

This raises a subtle but important point. While stronger parametric flow methods do exist for finding low-conductance cuts within an expanded neighborhood of the seed set, our walk-based method's weaker spectral-style guarantee on conductance is counterbalanced by a valuable "locality" property which ensures that we output a community consisting of nodes that are closely related to the seed set. In practice we try to get the best of both worlds by cleaning up the walk-based cuts with a conservative use of flow that does not disturb this locality property very much. Experiments show that the results compare well with stronger optimization techniques.

The random walks techniques are remarkable because they produce communities with conductance guarantees, yet can be computed locally rather than on the whole graph, often while touching fewer nodes than BFS would. The best low-conductance cuts are more likely than minimum cuts to non-trivially expand the seed set, but still ensure a small boundary defining a natural community. Finally, in a certain sense that we will discuss in section 2.5, the added nodes are close to the seed set.

2. RANDOM WALK METHODS

As part of their work on graph partitioning and graph sparsification [12], Spielman and Teng present a method for finding cuts based on the mixing of a random walk starting from a single vertex. In that paper, the method is used as a subroutine to produce balanced separators and multiway partitions.

In this section, we describe how to use the techniques developed by Spieman-Teng to find communities from a seed set, and describe how the size and quality of the seed set affects the results. In sections 2.1-2.3 we present background on random walks and sweeps. In section 2.4 we show that any seed set that makes up a significant fraction of a target community will produce good results with our seed expansion method, and that most seed sets that are chosen randomly from within a target community will also produce good results. In section 2.5 we present a stripped-down version of the local partitioning techniques developed by Spielman-Teng, and describe quantitatively how the size and quality of the seed set affect the running time and guarantees. In section 2.6 we describe the truncation method used by Spielman-Teng, and describe the heuristics and practical modifications we have employed to keep the running time and total number of vertices examined small.

2.1 Notation

The results in this section apply to graphs that are unweighted and undirected. Let A denote the adjacency matrix of the graph under consideration, and let D be the diagonal matrix where $D_{i,i} = d(v_i)$, the degree of the ith vertex. The volume of a set of vertices is

$$\mathrm{Vol}(S) = \sum_{u \in S} d(u).$$

The edge border is denoted

$$\partial(S) = \{\, \{u,v\} \mid \{u,v\} \in E, u \in S, v \notin S \,\},$$

the cutsize is denoted $|\partial(S)|$, and the conductance of a set of vertices is

$$\phi(S) = \frac{|\partial(S)|}{\min\left(\mathrm{Vol}(S), \mathrm{Vol}(\bar{S})\right)}.$$

This definition of conductance should not be confused with the conductance associated with a particular random walk on the graph. In particular, the conductance associated with a lazy random walk is a factor of 2 smaller.

2.2 Lazy random walks and sweeps

Given a seed set S for which we hope to find a community, we begin with the probability distribution $p_0 = \psi_S$, where

$$\psi_S = \begin{cases} d(x)/\mathrm{Vol}(S) & \text{if } x \in S, \\ 0 & \text{otherwise.} \end{cases}$$

We then simulate several steps of a lazy random walk, computing the probability distributions p_t, where

$$p_t = M^t p_0,$$

and where M is the lazy random walk transition matrix

$$M = \frac{1}{2}(I + AD^{-1}).$$

If the graph is connected, then p_t converges to the stationary distribution ψ_V. We are not interested in this limiting distribution, but rather in the distributions obtained after a small number of walk steps. We compute p_t for all t up to some specified time T. After each step, we sort the vertices in descending order according the degree-normalized probabilities

$$r_t(v) = p_t(v)/d(v),$$

letting v_i^t be the ith vertex in this order, so that

$$r(v_i^t) \geq r(v_{i+1}^t).$$

This ordering defines a collection of sets $S_0^t, \dots, S_J^t$, where $S_j^t = \{v_i^t \mid 1 \le i \le j\}$, and J is the number of vertices with nonzero values of $p(u)/d(u)$. The cutsizes, volumes, and conductances for every set $S_1^t, \dots, S_J^t$ can be computed in time proportional to $\mathrm{Vol}(S_J^t)$, by determining the change to S_i^t due to the addition of vertex v_{i+1}^t. This process is referred to as a sweep. We will show that under certain conditions one of the sweep sets S_j^t will be a good community for the seed S.

To make the degree-normalized distribution $p_t(u)/d(u)$ more intuitive, one can view each vertex as consisting of $d(u)$ minivertices, two minivertices $x_{(u,v)}$ and $x_{(v,u)}$ for each undirected edge $\{u, v\}$. The vector r_t can be interpreted as a probability distribution q_t on minivertices by letting $q_t(x_{(u,v)}) = r_t(u)$. During the t-th step of the lazy walk, $\frac{1}{2}q_t(x_{(u,v)})$ is the amount of probability sent from u to v.

To bound the mixing of the lazy random walk, we consider an ordering of the minivertices so that $q_t(x_i^t) \ge q_t(x_{i+1}^t)$. Using the shorthand notation $q_t(i) = q_t(x_i^t)$, we then define

$$P_t(k) = \sum_{i=1}^{k} q_t(i),$$

and the related quantity

$$H_t(k) = \left[P_t(k) - \frac{k}{2m}\right].$$

The functions $P_t(k)$ and $H_t(k)$ provide a strong way to bound how well the walk has mixed; $P_t(k)$ is the maximum amount of probability on any set of k minivertices at time t, and thus the amount of probability that flows over any set of k edges at time t is at most $\frac{1}{2}P_t(k)$. The maximum amount of probability on a set of vertices A is at most

$$P_t(\mathrm{Vol}(A)) = H_t(\mathrm{Vol}(A)) + \frac{\mathrm{Vol}(A)}{\mathrm{Vol}(G)},$$

and the L_1-distance between p_t and the stationary distribution ψ_V can be written

$$|p_t - \psi_V| = 2\max_{A\subseteq V}\langle p_t - \psi_V, 1_A\rangle = 2\max_k\left[H_t(k)\right].$$

We will need the following monotonicity lemma, which is a corollary of the result of Lovász and Simonovits [10].

Lemma 1. *$P_{t+1}(k) \le P_t(k)$, for any $k \in [0, 2m]$.*

2.3 Mixing of lazy random walks

For large seed sets, the initial distribution p_0 may already be moderately well mixed, which will lead to better bounds on $P_t(k)$. In particular, we have the following lemma.

Lemma 2. *If $p_0 = \psi_S$, then*

$$H_0(k) \le c\min(\sqrt{k}, \sqrt{m-k}),$$

where $c = \sqrt{\frac{1}{\mathrm{Vol}(S)}}$.

Proof. This follows easily from the definition of ψ_S. □

The following lemma due to Spielman and Teng is a strengthening of a result of Lovász and Simonovits [10] [11]. It shows that the lazy random walk mixes well unless one of the sweeps sets S_j^t has both small conductance and large drop in degree-normalized probability between the vertices inside and outside of S_j^t.

Lemma 3. *Let $p_0 = \psi_S$ and let ϕ and α be some fixed constants. Either*

$$H_T(k) \le H_0(k)\left(1 - \frac{\phi^2}{8}\right)^T + \alpha T, \qquad (1)$$

or there exists a sweep cut S_j^t with $t \le T$ such that

1. *$\phi(S_j^t) \le \phi$, and*
2. *$q_t(k_0 - \phi\bar{k}_0) - q_t(k_0 + \phi\bar{k}_0) \ge \frac{2\alpha}{\phi\bar{k}_0}$,*
   where $k_0 = \mathrm{Vol}(S_j^t)$ and $\bar{k}_0 = \min(k_0, 2m - k_0)$.

Proof. This is Lemma 3.7 of [12] □

2.4 Good seed sets

Lemma 3 shows that in the absence of a good sweep cut, the walk mixes rapidly. If we can show that the walk does not mix as rapidly as the lemma should imply, then one of the sweep cuts must be good. One way to do this is to present a target community C that has small volume, but contains a large amount of probability from p_T, and use this to place a lower bound on $H_T(\mathrm{Vol}(C))$.

The amount of probability that has escaped from C after T steps, which we will write $\langle M^T\psi_S, \bar{C}\rangle$, depends on both the target community C and the seed set S. For every set C with small conductance, there are a variety of seed sets for which $\langle M^T\psi_S, \bar{C}\rangle$ is not much larger than $\phi(C)\cdot T$. To motivate this, consider the amount of escaping probability when the seed set is the entire target set C. The following lemma is also due to Spielman-Teng.

Lemma 4. *For any step t, $\langle M^t\psi_C, \bar{C}\rangle \le \frac{1}{2}\phi(C)\cdot t$.*

Proof. Since our starting distribution is $p_0 = \psi_C$, each minivertex in C initially has probability $\frac{1}{\mathrm{Vol}(C)}$. Therefore

$$P_0(|\partial(C)|) = |\partial(C)| \cdot \frac{1}{\mathrm{Vol}(C)} = \phi(C).$$

By the monotonicity lemma, $P_i(|\partial(C)|) \le \phi(C)$ for all i. The amount of probability leaving C during the ith walk step is at most $\frac{1}{2}P_i(|\partial(C)|)$, so the amount of of probability on $\bar{C}$ after t steps is at most $\frac{1}{2}\phi(C)\cdot t$. □

Any set that is fairly large and nearly contained in the target community is also a good seed set. The bounds in the lemma below are weak for smaller seed sets, but are sufficient for seed sets that make up a significant fraction of the target community.

Lemma 5 (Large seed sets). *Let S be a seed set such that $\mathrm{Vol}(C) \le \beta\mathrm{Vol}(S)$, and $\mathrm{Vol}(S \cap C) \ge (1-\delta)\mathrm{Vol}(S)$. Then, for any step t,*

$$\langle M^t\psi_S, \bar{C}\rangle \le (1-\delta)\frac{1}{2}\beta\phi(C)\cdot t + \delta.$$

Proof. Assume for now that $S \subseteq C$. The maximum amount of probability on any minivertex from $p_0 = \psi_S$ is $\frac{1}{\mathrm{Vol}(S)}$, and so

$$P_0(|\partial(C)|) = |\partial(C)| \cdot \frac{1}{\mathrm{Vol}(S)} \le |\partial(C)|\frac{\beta}{\mathrm{Vol}(C)} = \beta\phi(C).$$

By the monotonicity lemma, $P_i(|\partial(C)|) \le \beta\phi(C)$ for all i. The amount of probability leaving C during the ith walk step is at most $\frac{1}{2}P_i(|\partial(C)|)$, and so $\langle M^t\psi_S, \bar{C}\rangle \le \frac{1}{2}\beta\phi(C)\cdot t$.

For the general case where $\text{Vol}(S \cap \bar{C}) = \delta$, ψ_S can be decomposed as

$$\psi_S = (1-\delta)\psi_{S\cap C} + \delta\psi_{S\cap\bar{C}}.$$

Therefore,

$$\begin{aligned}\langle M^t\psi_S, \bar{C}\rangle &\leq (1-\delta)\langle M^t\psi_{S\cap C}, \bar{C}\rangle + \delta\langle M^t\psi_{S\cap\bar{C}}, \bar{C}\rangle \\ &\leq (1-\delta)\frac{1}{2}\beta\phi(C)\cdot t + \delta.\end{aligned}$$

□

Seed sets chosen randomly from within a target community are also likely to be good seed sets for that community. The following result is stated without proof, since it follows by applying a Chernoff-type bound to a weighted sum of independent random variables in the usual way.

LEMMA 6 (RANDOM SEED SETS). *Let S be a set where each vertex from C is included in S independently with probability ϵ. Let $\Delta = \max_{v\in C} d(v)$. For any single specific time t, the bound*

$$\langle M^t\psi_S, \bar{C}\rangle \leq \phi(C)t$$

holds with probability at least

$$1 - \left(e^{\frac{-\epsilon t\phi(C)\text{Vol}(C)}{40\Delta}} + e^{\frac{-\epsilon\text{Vol}(C)}{32\Delta}}\right).$$

2.5 Communities from a seed set

The following is a stripped-down version of a theorem of Spielman-Teng, extended for a seed set instead of a single starting vertex.

THEOREM 1. *Given a seed set S, choose two parameters ϕ and β, let $T = \frac{4}{\phi^2}\ln(16\beta)$ and compute $p_1, \ldots, p_T$ starting from $p_0 = \psi_S$. Assume that there exists a set of vertices C such that $\text{Vol}(C) \leq \frac{1}{2}\text{Vol}(G)$, and such that S is a fairly good seed set for C in that $\langle M^T\psi_S, \bar{C}\rangle \leq a\phi(C)\cdot T$. If the parameters ϕ and β have been chosen so that*

1. $\frac{\text{Vol}(C)}{\text{Vol}(S)} \leq \beta$, *and*

2. $\phi(C) \leq \frac{\phi^2}{32a\ln(16\beta)}$,

then there exists a sweep cut S_j^t with $t \leq T$ such that

1. $\phi(S_j^t) \leq \phi$, *and*

2. $q_t(k_0 - \phi\bar{k}_0) - q_t(k_0 + \phi\bar{k}_0) \geq \frac{\phi}{4\ln(16\beta)k_0}$,
where $k_0 = \text{Vol}(S_j^t)$ and $\bar{k}_0 = \min(k_0, 2m - k_0)$.

PROOF. If equation (1) from Lemma 3 were to hold with parameters ϕ (as chosen in the statement of the theorem) and α (to be set later), then this would imply

$$\begin{aligned}1 - a\phi(C)T &\leq \langle M^T\psi_S, C\rangle \\ &\leq \frac{\text{Vol}(C)}{\text{Vol}(G)} + \sqrt{\frac{\text{Vol}(C)}{\text{Vol}(S)}}\left(1-\frac{\phi^2}{8}\right)^T + \alpha T.\end{aligned}$$

We wish to set T so that we obtain a contradiction. If we assert that $a\phi(C)T \leq \frac{1}{8}$ and $\alpha T \leq \frac{1}{8}$, it suffices to choose T such that

$$\frac{1}{4} \leq \sqrt{\frac{\text{Vol}(C)}{\text{Vol}(S)}}\left(1-\frac{\phi^2}{8}\right)^T \leq \sqrt{\beta}e^{-\frac{\phi^2}{8}T}, \qquad (2)$$

so it suffices to take $T = \frac{8}{\phi^2}\ln(4\sqrt{\beta}) = \frac{4}{\phi^2}\ln(16\beta)$. With this T, the requirement that $a\phi(C)T \leq \frac{1}{8}$ is satisfied if $\phi(C) \leq \frac{1}{8aT} = \frac{\phi^2}{32a\ln(16\beta)}$. We can take $\alpha = \frac{\phi^2}{32\ln(16\beta)}$ and still satisfy the requirement that $\alpha T \leq \frac{1}{8}$. Since equation (1) does not hold with this choice of ϕ,α, and T, Lemma 3 implies that one of the sweep cuts has the desired properties.

□

The sweep cut S_j^t can be viewed as a community of the seed S. In addition to having small conductance, there is a significant drop in probability outside S_j^t by Theorem 1. Figure 3 shows an example of this simultaneous small conductance cut and probability drop.

While these guarantees on S_j^t are fairly clear, the way that the community is related to the seed set is more subtle. The most obvious thing we can say about this relationship is probably the most important: the community is obtained by taking all vertices where the value of $r_t(v) = p_t(v)/d(v)$ is above a certain threshold. If we view $r_t(v)$ as a measure of closeness to the seed set, this is a strong guarantee.

Unfortunately, $r_t(v)$ is not a good measure of closeness to the seed set when t becomes large. As t tends to infinity, $r_t(v)$ tends to a constant, and the ordering of the vertices determined by $r_t(v)$ tends to the same ordering determined by the second-largest eigenvector of $I + D^{-1}A$, which is independent of the seed set if the graph is connected. This means that the vertices from the seed set are not guaranteed to have the largest values of $r_t(v)$, and so the seed set is not necessarily contained in the resulting community. However, we have an upper bound on the number of walk steps taken, and we claim that $r_t(v)$ is a good measure of relationship to the seed set for small values of t. In practice, we have observed that requiring the community to be determined by a threshold on the values of $r_t(v)$ rules out more degenerate behavior than requiring only that the community contains the seed.

We cannot hope that the resulting community S_j^t will closely match the target community C, without placing additional assumptions on the communities inside C. Instead, the target community is used to ensure that some community for S will be found, and this community is likely to be mostly contained within the target community. This can be made precise if a stronger restriction is placed on $\phi(C)$. Spielman-Teng showed that if $\phi(C)$ is roughly ϕ^3, then the gap in probability described in Theorem 1 can be used to show that the majority of vertices in S_j^t are contained in C.

2.6 Truncation

The actual subroutine created by Spielman-Teng, called `Nibble`, has a stronger guarantee than the method we have presented here. One aspect of `Nibble` that we would like to reproduce is the ability to find a small community near the seed set while keeping the number of vertices examined small. This is accomplished by using truncated walk distributions in place of exact walk distributions.

In the truncated walk, the probability on any vertex where $r_t(v) \leq \epsilon$ is set to 0 after each step. This ensures that the support of the truncated walk $Supp_t$ is small, which gives a bound on the running time of the algorithm, since the time required to perform a step of the random walk and a sweep depends on $\text{Vol}(Supp_t)$. For the theoretical guarantees to apply, the truncation parameter ϵ must be

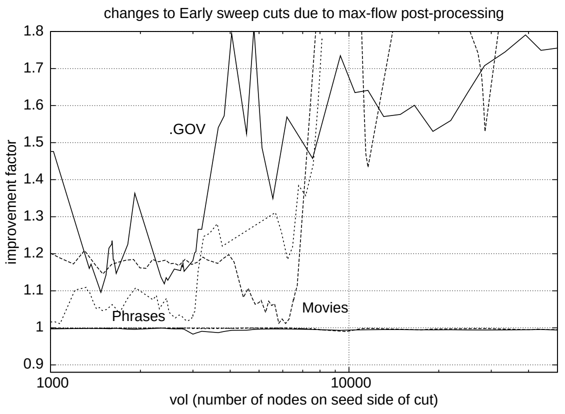


Figure 1: A plot of the factors by which cutsize gets better and enclosed probability gets worse as a result of sliding window ST max flow post-processing. The top 3 lines show that for all three tasks the cutsize is often improved by 10 to 50 percent. The bottom 3 lines (hard to see, but just below y=1.0) show how much enclosed probability is lost due to the rearrangement of nodes caused by the post-processing. Usually less than 1 percent is lost.

small enough so that ϵT is smaller than the probability gap $q(k_0 - \phi\bar{k}_0) - q(k_0 + \phi\bar{k}_0)$ implied by Theorem 1.

In our experiments we perform a more severe kind of truncation, at a stronger level than is supported by these guarantees. We specify a target volume k_0, sort the vertices in terms of $p_t(v)/d(v)$ for the current truncated probability vector p_t, and let i_0 be the last index such that $\mathrm{Vol}(\{v_1^t, \ldots, v_{i_0}^t\}) \le k_0$. We then remove the probability from all vertices v_j^t where $j > i_0$. In section 3.6 we will experimentally show that a target cluster can be accurately found by setting k_0 somewhat larger than the cluster size.

3. EXPERIMENTS ON THREE DATASETS

3.1 From theory to experiment

The previous sections contained theorems proving that given a sufficiently distinct target set (a subgraph with small enough conductance) and a sufficiently "good" seed set lying mostly within it, a lazy local random walk will grow the seed set and return a low-conductance community for the seed set. We also proved that good seed sets are not rare, and that a big enough or random subset of the target set should be good.

Due to constant factors, it is often hard to tell whether a theoretical method will work in practice. Therefore in the following subsections we will describe sanity check experiments on three datasets in which we used a sparse matrix viewer to choose target sets and seed sets that appeared likely to satisfy the preconditions of the theorems, so that we could check whether the method actually recovered the target sets with reasonable accuracy.

3.2 Cut improvement with flow

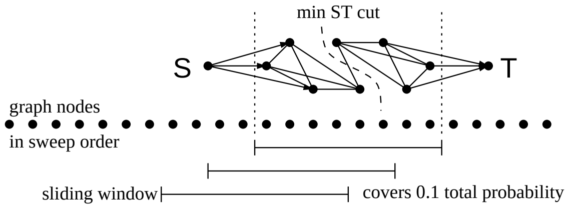
There are several reasons for using flow to clean up the sweep cuts. First, it is generally a good idea to do this for any spectral-type graph partitioning method [9]. Second,

count	recall	URLs starting with:
1381	0.672	www-pao.ksc.nasa.gov/kscpao/
829	0.997	science.ksc.nasa.gov/shuttle/
203	0.975	www.jsc.nasa.gov/Bios/
162	1.000	neurolab.jsc.nasa.gov/
104	1.000	science.ksc.nasa.gov/htbin/
42	1.000	shuttle.ksc.nasa.gov/shuttle/
25	0.925	science.ksc.nasa.gov/persons/
21	0.875	liftoff.msfc.nasa.gov/shuttle/
20	1.000	science.ksc.nasa.gov/docs/
18	0.750	www.ksc.nasa.gov/shuttle/
14	1.000	science.ksc.nasa.gov/facilities/
11	0.916	www.ksc.nasa.gov/persons/
76		other mostly nasa related

Table 1: Contents of cluster grown from 25 percent of the `science.ksc.nasa.gov/shuttle` URLs.

aggressive probability truncation can produce cuts that are even noisier than usual. Third, in the following sections we will see empirical evidence that flow-based improvement makes the sweep cut method less sensitive to the number of walk steps that are taken.

However, this post-processing needs to be done with care; in some preliminary experiments in which we grew well past the size of the target set and then used parametric flow to shrink back to the cut with optimal conductance of any cut containing the seed set, we sometimes got surprising solutions containing distant, essentially unrelated nodes. Those results were a clear demonstration that conductance and containment of the seed set do not entirely capture our goals in seed set expansion. We actually want to find a low-conductance set that is obtained by adding *nearby* nodes to the seed set. Back in section 2.5 we argued that if a set of nodes contains a lot of probability from a random walk that hasn't run for too many steps, then it has a version of the desired locality property.



To improve the sweep cuts without sacrificing the locality property, we did some conservative post-processing with repeated ST max flow calculations across a sliding window covering 10 percent of the total probability.[1] These calculations were free to rearrange the lower probability nodes near a bottleneck, but not the higher probability nodes near the seed set, thus allowing us to reduce the cutsize without risking the loss of too much enclosed probability. The plots in Figure 1 show that our actual losses of enclosed probability were typically at least an order of magnitude smaller than the cutsize improvement factors.

[1]A different rule of thumb that is more motivated by the theory is that the sweep cut S_j^t can be safely improved using a window in which the number of "active" vertices to the left and right of j be such that the amount of active volume on each side is $|\partial(S_j^t)|$.

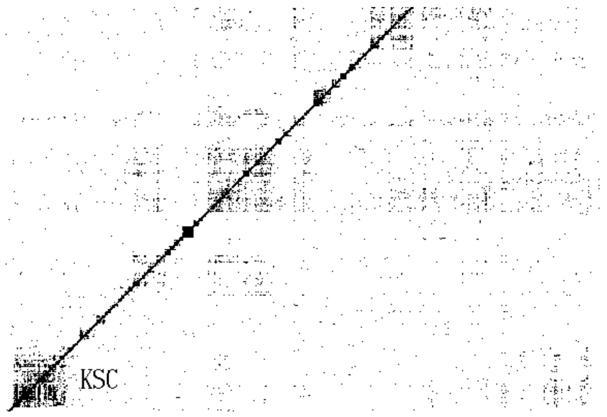


Figure 2: A low-resolution view of a small part of the .GOV adjacency matrix showing the block associated with the Kennedy Space Center.

3.3 .GOV Graph, KSC Seedset

The starting point for this graph was TREC's well-known .GOV web dataset. To make the graph's block structure more obvious (so that we could visually choose a seed set that probably satisfies the requirements of the theory), we repeatedly deleted all nodes with in- or out-degree above 300 or equal to 1 or 2. Then we symmetrized the graph and extracted the largest connected component, yielding an undirected graph with 536560 nodes and 4412473 edges, and node degrees ranging from 3 to 394.

After inspecting this graph's adjacency matrix (see Figure 2), we chose for our target cluster a distinct matrix block containing roughly 3000 URL's associated with the Kennedy Space Center. Among these URL's were 830 beginning with `science.ksc.nasa.gov/shuttle`. We randomly selected 230 of these nodes as a seed set. Jumping ahead to our results, Table 1 shows that by growing this seed set we can recover nearly all of the `science.ksc.nasa.gov/shuttle` URL's, plus large fractions of several other families of URL's that relate to the Kennedy Space Center.

Figure 3-top shows the probability distribution after 60 steps of a lazy random walk starting with uniform probability on the seed set and zero probability elsewhere. Along the x axis nodes are sorted in order of decreasing degree-normalized probability, which we call "sweep order". On the left are the high probability nodes that are close to the seed set. Lower probability nodes appear towards the right of the plot and beyond. The sharp decrease in probability around 2500-3000 nodes is the signature of the low-conductance bottleneck that we are interpreting as a cluster boundary.

Each curve in Figure 3-bottom shows a "cutsize sweep" in which we evaluate all cuts between adjacent nodes in the sweep order defined by the probability distribution at some step in the local random walk.[2] The two curves here are for 30 and 90 steps. Notice that both sweeps show a dip in cutsize around 2500-3000 nodes, which again reveals the bottleneck at the cluster boundary.

[2]It might seem confusing that our sweep plots show cutsize while the random walks algorithm optimizes conductance. However, conductance is a particular way of combining our twin goals of 1) growing the seed set and 2) finding a small boundary. These plots of cutsize as a function of node count display a range of possible tradeoffs between these two goals.

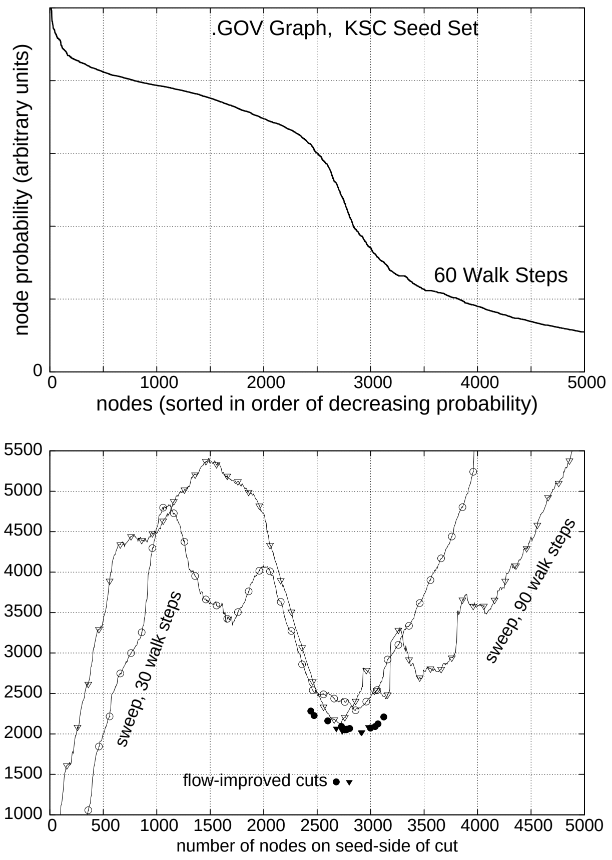


Figure 3: The top plot shows the sharp decrease in degree-normalized walk probability caused by the bottleneck which defines the boundary of the KSC cluster in the .GOV graph. In the bottom plot this bottleneck shows up as the dip in cutsize near 2750 nodes. We note that these two plots are consistent with the two conclusions of theorem 1.

Comparing the 30- and 90-step sweeps, we see that the 30-step sweep looks better towards the left, while the 90-step sweep looks better towards the right. This is because the spreading probability is revealing information about larger sets later. We also see that the flow-improved solutions for 30 and 90 steps are nearly the same; apparently the flow-based post-processing helps us get good results for a wider range of stopping times.

3.4 Sponsored Search Graph, Betting Seedset

This task illustrates what can happen when cluster nesting causes there to be more than one reasonable community for the seed set. The data originated in Yahoo's "sponsored search" business (Google's "ad words" is roughly similar). In this business advertisers make bids on (bidded phrase, advertiser URL) pairs, which can be encoded in a bipartite graph with bidded phrases on one side and advertiser URL's on the other, with each bid represented by an edge. Co-clustering this graph reveals block structure showing that the overall advertising market breaks down into numerous submarkets associated with flowers, travel, financial services, etc. For our experiment we used a 2.4 million

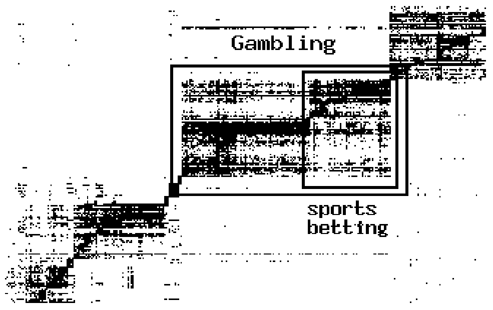


Figure 4: A low-resolution view of part of a very small version of the Yahoo sponsored search bipartite incidence matrix. The gambling co-cluster contains a sports betting subcluster.

node, 6 million edge bipartite graph built from part of an old version of the sponsored search bid database. Node degrees ranged from 1 to about 30 thousand.

Based on earlier experiences we expected that this dataset would contain a distinct gambling cluster which would contain a subcluster focused on sports betting (see the matrix picture in Figure 4). For a seed set we used grep to find all bidded phrases and advertiser URL's containing the substring "betting". This resulted in a 594-node seed set which includes nodes on both sides of the bipartite graph, and is probably mostly contained within the sports betting subcluster.

Our seed set lies within the sports betting cluster, and this in turn lies within the gambling cluster. To which of these clusters should the seed set be expanded? Perhaps the algorithm should tell us about both of the possible answers. In fact, when there are nested clusters surrounding a seed set, over time the local random walk method often does tell us about more than one possible answer as the probability distribution spreads out.

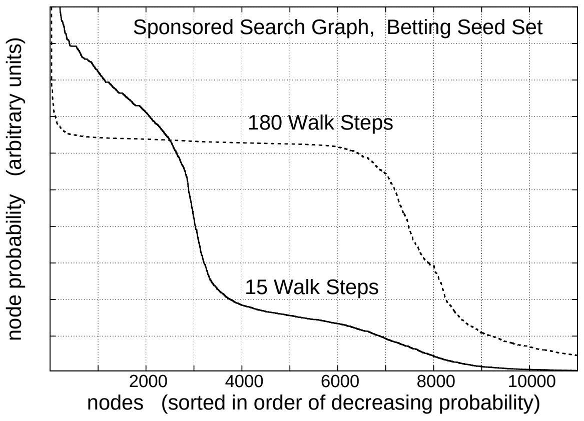
This behavior can be very clearly seen in the probability distributions plotted in Figure 5-top. At 15 steps there is a sharp decrease in probability marking the boundary of the 3000-node betting subcluster. At 180 steps there is a different sharp decrease in probability marking the boundary of the enclosing 8000-node gambling cluster. Notice that within this larger cluster the probabilities are becoming quite uniform at 180 steps, so it is no longer easy to see the boundary of the inner cluster.

The cutsize sweeps of Figure 5-bottom show what all this means in terms of cuts. At 15 steps the boundary of the inner cluster is sharply defined by a dip in cutsize near 3000 nodes. Also at this point we are starting to get a rough preview of the boundary of the enclosing cluster. This preview is much improved by the flow-based post-processing.

At 180 steps, the inner dip in cutsize delimiting the betting subcluster has been washed out (but can be somewhat recovered with flow) but now we have a good dip near 8000 nodes which is the boundary of the outer gambling cluster.

3.5 Movie/Actress Graph, Spain Seedset

This task has even more cluster nesting than the previous one. The graph is bipartite and was built from the IMDB file relating movies and actresses. It has a clear block structure



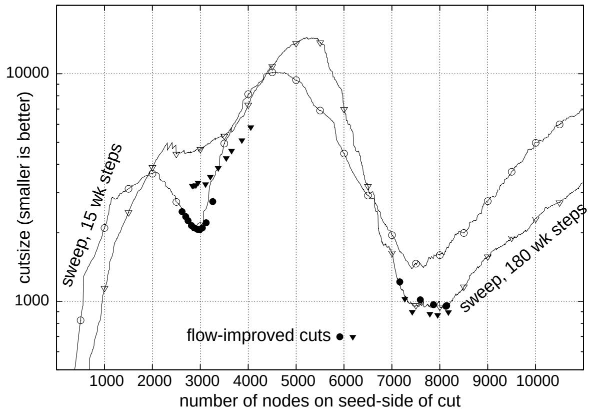


Figure 5: These plots are for the sponsored search task where the seed set lies within two nested clusters. The top plot shows that at different stages of the walk we see two different sharp decreases in node probability that mark the boundaries of these two clusters. In the bottom plot the two boundaries show up as dips in cutsize near 3000 and 8000 nodes. Flow-based post-processing improves the cuts and reduces sensitivity to stopping time.

that is strongly correlated with countries.[3] Therefore we use country-of-production labels for the movies as cluster membership labels for purposes of choosing seed sets and measuring precision and recall.

We cleaned up the problem a bit by deleting all multi-country movies and many non-movie items such as television shows and videos. We combined several nearly synonymous country labels (e.g. USSR and Russia) and then deleted all but the top 30 countries. Finally we deleted all degree-1 nodes and extracted the largest connected component. We ended up with a bipartite graph with 77287 actress nodes, 121143 movie nodes, and 566756 edges. The minimum degree was 2, and the maximum degree was 690.

For a target cluster we chose Spain, whose matrix block can be seen in Figure 10. It appears to part of a super-cluster containing other Spanish- and Portuguese-language countries. One can also see many cross edges denoting ap-

[3]Also many countries contain sub-blocks delimited by disruptive events like the silent / talkie transition, or WWII.

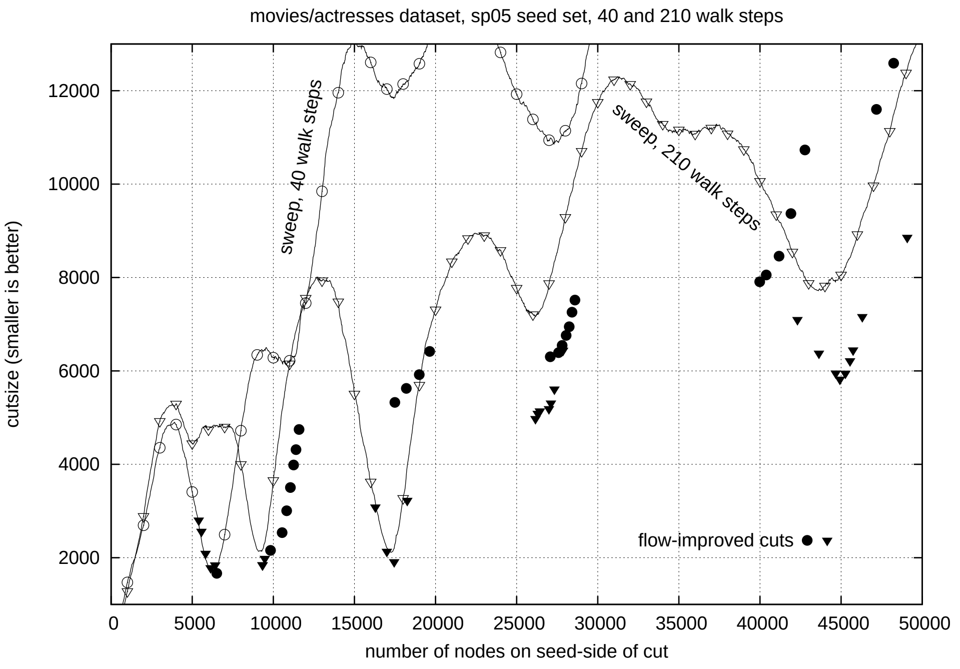


Figure 6: Cutsize sweeps for the movies / actresses task. The dip near 6500 nodes marks the boundary of the Spain cluster which immediately contains the 179-node seed set. The dip near 17500 nodes is the boundary of a super-cluster containing Spanish-language countries plus Portugal. The dip near 45000 nodes is the boundary of a super-cluster containing Romance-language countries. As usual, earlier in the walk we get the inner cluster boundaries, while later in the walk we get the outer cluster boundaries. Flow-based post-processing sharpens the boundaries, especially those that are not the current focus of the walk.

pearances of Spanish actresses in French and Italian movies, and vice versa, so a Romance-language supercluster would not be surprising. For a seed set we randomly selected 5 percent of the movies produced in Spain. This seed set contained 179 movie nodes, and 0 actress nodes.

The curves in Figure 6 show cutsize sweeps based on the probability distribution after 40 and 210 walk steps. The 40-step sweep (open circles) contains a sharp dip at 6500 nodes which marks the boundary of the Spain cluster with high accuracy. Beyond this bottleneck the 40-step sweep is pretty sketchy, but the flow-based cuts derived from it (filled circles) are already surprisingly good.

The open triangle curve in Figure 6 shows that at 210 steps, the random walk has already washed out the boundary of the Spain cluster, but is now giving a better view of the enclosing hierarchy of four superclusters, especially those containing 9000 and 17500 nodes. The filled triangles show that flow-based post-processing recovers the washed-out boundary of the Spain cluster, and greatly improves the boundaries of the 27000 and 45000 node superclusters.

Because we have country labels for the movies we can interpret these clusters, and also measure the accuracy with which they are being recovered. The following table contains precision and recall measurements for the movies[4] in the sets of nodes delimited by the (flow-improved) leftmost, middle, and rightmost dips in Figure 6. These precision and recall measurements show that the 6500-node set is a good approximation of our target cluster of Spain. The 17500-node set (which has the smallest conductance) is a very good approximation of a mostly Spanish-language supercluster containing Spain, Mexico, and Argentina, plus Portugal. The 45000-node set is a good approximation of a Romance-language supercluster containing the previous 4 countries plus Brazil, Italy, and France.

quotient = cutsize / nodes	preci.	recall	cluster contents
0.255835 = 1666 / 6512	0.956	0.979	Spain
0.109531 = 1910 / 17438	0.979	0.995	Spanish+Portugal
0.129574 = 5821 / 44924	0.958	0.988	Romance language

3.6 Sweeps from Truncated Walks

In the experiments up to now we have done full walks with probability spreading to the whole graph. The computation can be made more local by doing a truncated walk, as discussed in section 2.6. Here we briefly examine the qualititative effect of truncating to a fixed number of nodes after each step. Figure 7 compares cut sweeps over node orderings derived from truncated and untruncated walks on the sponsored search and movies/actresses graphs. In the first two plots we truncated to the top 10000 nodes after each step. Up through the bottlenecks near 8000 and 6500 nodes, respectively, these new sweeps look similar to the earlier sweeps over full walks. In the final plot we truncated to 6000 nodes on the movies/actresses task, which is a bit too small for recovering the 6500-node Spain cluster. In this case the results are degraded, but somewhat recoverable with flow-based post-processing.

[4]but not the actresses, for which we lack country labels.

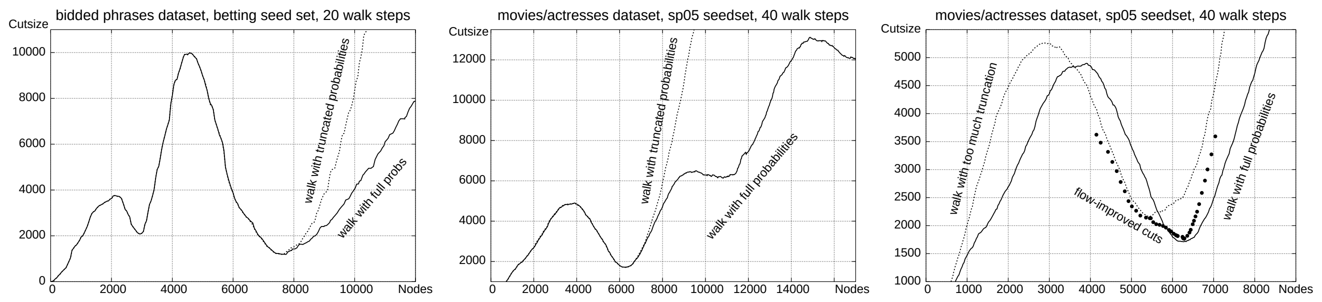


Figure 7: The first two plots show how a truncated walk that only keeps the top 10000 probabilities after each step can still find 8000- and 6000-node clusters in the sponsored search and movies/actresses datasets. In the final plot we truncated to 6000 nodes, leading to degraded results that were improvable with flow.

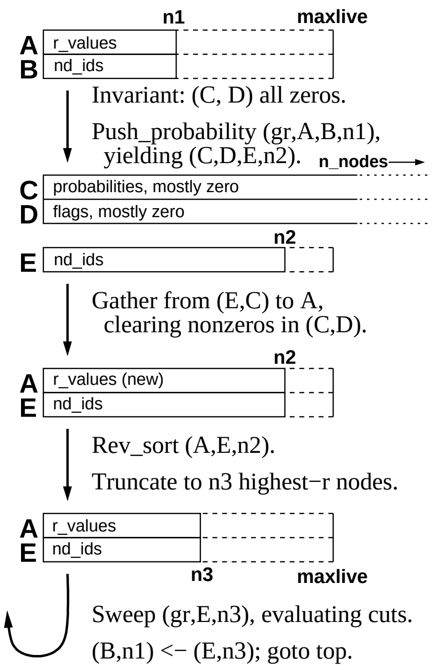

n1
maxlive
A
r_values
B
nd_ids
Invariant: (C, D) all zeros.
Push_probability (gr,A,B,n1),
yielding (C,D,E,n2).
n_nodes
C
probabilities, mostly zero
D
flags, mostly zero
n2
E
nd_ids
Gather from (E,C) to A,
clearing nonzeros in (C,D).
n2
A
r_values (new)
E
nd_ids
Rev_sort (A,E,n2).
Truncate to n3 highest−r nodes.
A
r_values
E
nd_ids
n3
maxlive
Sweep (gr,E,n3), evaluating cuts.
(B,n1) <− (E,n3); goto top.

Figure 8: Sketch of one iteration in code whose run time is (nearly) independent of the graph size.

	IM_eighth	IM_full
n_nodes	22 M	191 M
n_edges	79 M	788 M
max_deg	12883	403161
avg_deg	7.20	8.25
graph space	724 M	7069 M
C+D space	199 M	1720 M
A+B+E space	0.64 M	0.64 M
avg n1	3399	3040
avg n2	14210	14010
bogus msec / iter	159.7	1234.3
actual msec / iter	9.2	13.5

Figure 9: Stats for runs in which we expanded 100 different seed sets for 100 iterations each in two very large graphs. Iterations on IM_full only took 1.5 time longer, even though the graph is 8 times bigger.

4. FAST IMPLEMENTATION

One of the main features of the truncated random walks partitioning method is that its time and space requirements can be independent of the graph size. We will describe some code that basically achieves this time independence. [However, it still uses $O(n)$ working space.] Our data structures, and the algorithm for one iteration are sketched in Figure 8. The graph contains n nodes and m edges, and is representable in CSR format in (n+1)*4 + 2*m*4 bytes.

We use five working arrays. C is a scratch array for probabilities containing n doubles, mostly zero. D is a scratch array containing n flag bytes, mostly zero. Together they occupy $9 * n$ bytes.[5] We use parallel arrays of doubles and node_id's to compactly represent small sets of nonzero values associated with "active" nodes. "Maxlive" is an upper bound on the number of active nodes. B and E are two arrays of node_id's each consuming $4 *$ maxlive bytes. A is an array of r-values (degree-normalized probabilities) consuming $8 *$ maxlive bytes.

The key to speed and scalability is to never do any $O(n)$-time operation, including clearing arrays C or D, except at program initialization time. While expanding a seed set, we only do operations whose time depends on the number of active nodes.

Push_probability(gr, A, B, n_1) works as follows. The input (A, B, n_1) represents the n_1 currently active nodes. We zero the counter n_2. The arrays C and D are already zeroed. For each active node we push half of its probability to itself, while the other half is evenly divided and pushed to its neighbors. To push some probability p to node j, we do: $\{C[j] \leftarrow C[j]{+}p;$ **if** $(0{=}D[j])$ $\{D[j] \leftarrow 1; E[n_2] \leftarrow j; n_2 \leftarrow n_2{+}1\}\}$. We end up with n_2 nonzero probabilities scattered in C. The n_2 entries in E tell where they are, so we can gather them back into A, simultaneously clearing the nonzeros in C and D.

We now describe some timings on two large graphs. The 191-million node graph IM_full is a symmetrized version of the buddy list graph for users of Yahoo Instant Messenger. We used an out-of-core reimplementation of Metis to partition this graph into 8 big pieces, one of which is the 22-million node graph IM_eighth. We also used Metis [6] to break both graphs into numerous tiny pieces of size 150-

[5]Note that we could get below $O(n)$ working space by using hash tables for C and D. An even fancier program could use hash tables to store the relevant part of the graph, with the full graph stored on disk or on a remote server.

200. For each graph, we randomly selected 100 of these tiny pieces to serve as seed sets for 100 runs of our truncated random walk procedure. Each run went for 100 iterations. We used volume-based truncation, on each step cutting back to a volume (sum of degrees) of 20000. On average, the probability pushing step grew the active set from about $n_1 = 3000$ nodes out to about $n_2 = 14000$ nodes, then the truncation step cut it back to about $n_3 = 3000$ nodes.

Figure 9 summarizes these graphs and runs, which were done on a 4-processor 2.4 GHz Opteron with 64 Gbytes of RAM. The average time per iteration was only 9.2 and 13.5 msec for the two graphs. Iterations on IM_full took 1.5 times longer, even though the graph is 8 times bigger. We actually hoped that the iteration time would be the same. A possible explanation for the increase is that there were more cache misses for the bigger graph (the probability pushing step involves a pattern of "random" memory accesses). To put these times into perspective, we also list some "bogus" times where we actually checked our invariant that the C and D arrays are zeroed at the top of the loop. This checking takes $O(n)$ time, resulting in a huge increase in time per step, and a factor of 8 increase from the smaller to larger graph.

4.1 Choosing Parameters

So far we haven't said much about how to choose values for the parameters T (the number of walk steps), and k_0 (the truncation size). It is clear from the previous sections that a seed set can have several possible communities; to choose between them we would need to retain some freedom in choosing parameter values. According to Theorem 1, it suffices to set T to be roughly $\frac{\log(\beta)}{\phi^2}$, where ϕ is a lower bound on the desired conductance and β is an upper bound on the expansion factor $\frac{\mathrm{Vol}(C)}{\mathrm{Vol}(S)}$. [We note that the method often works with much smaller values for T.]

One can obtain an algorithm that depends on a single parameter by choosing a value of ϕ and searching through several values of β, setting $T = \frac{\log(\beta)}{\phi^2}$, and $k_0 = \beta\mathrm{Vol}(S)$. The table below shows the results of such a search on the movies/actresses task, with ϕ set to .1, and with β set to 2^i for each i between 5 and 10. For each value of i, we return the cut with the smallest conductance found by any of the T sweeps. The following table reports the conductance and volume of this cut, and also the step t at which it was found.

i	T	k_0	conductance	volume	t
5	500	27616	.1280	27621	480
6	600	55232	.0452	37499	250
7	700	110463	.0240	103720	700
8	800	220972	.0193	103894	280
9	900	441856	.0162	127374	620
10	1000	883712	.0129	835500	780

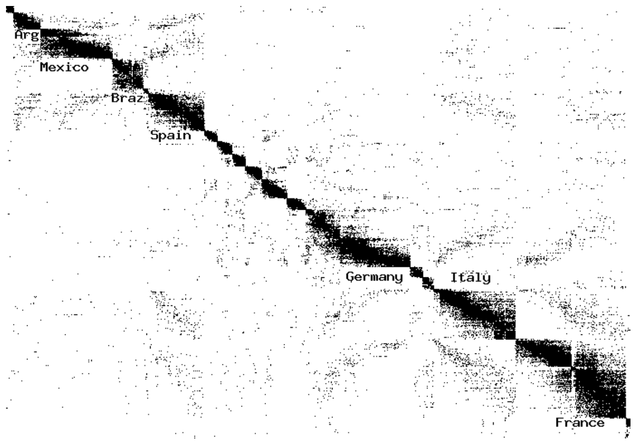


Figure 10: A low-resolution view of part of the movies vs actresses incidence matrix. The Spain co-cluster lies within a supercluster of Spanish- and Portuguese-language countries. Also there are many edges leading from Spain to other Romance-language countries. See section 3.5.

5. REFERENCES

[1] Krishna Bharat and Monika R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *ACM SIGIR-98*, pages 104–111, Melbourne, AU, 1998.

[2] Soumen Chakrabarti, Byron E. Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In Laura M. Haas and Ashutosh Tiwary, editors, *Proceedings of ACM SIGMOD-98*, pages 307–318, Seattle, US, 1998. ACM Press, New York, US.

[3] Fan Chung and Lincoln Lu. Connected components in random graphs with given degree sequences. *Annals of Combinatorics*, 6:125–145, 2002.

[4] Gary Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of web communities. In *Sixth ACM SIGKDD*, pages 150–160, Boston, MA, August 20–23 2000.

[5] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *VLDB*, pages 576–587, 2004.

[6] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359 – 392, 1999.

[7] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[8] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks*, 31(11–16):1481–1493, 1999.

[9] Kevin J Lang. Fixing two weaknesses of the spectral method. In *NIPS*, 2005.

[10] László Lovász and Miklós Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *FOCS*, pages 346–354, 1990.

[11] László Lovász and Miklós Simonovits. Random walks in a convex body and an improved volume algorithm. *Random Struct. Algorithms*, 4(4):359–412, 1993.

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

[13] M. Toyoda and M. Kitsuregawa. Creating a web community chart for navigating related communities, 2001.