Preserving Personalized Pagerank in Subgraphs

Andrea Vattani[1] AVATTANI@CS.UCSD.EDU
Deepayan Chakrabarti[2] DEEPAY@YAHOO-INC.COM
Maxim Gurevich[2] MAXIMG@YAHOO-INC.COM
[1]UC San Diego, 9500 Gilman Drive, La Jolla, CA 92093
[2]Yahoo! Research, 701 1st Avenue, Sunnyvale, CA 94089

Abstract

Choosing a subgraph that can concisely represent a large real-world graph is useful in many scenarios. The usual strategy employed is to sample nodes so that the *induced* subgraph matches the original graph's degree distribution, clustering coefficient, etc., but no attempt is made to preserve fine-grained relationships between nodes, which are vital for applications like clustering, classification, and ranking. In this work, we model such relationships via the notion of Personalized PageRank Value (PPV). We show that induced subgraphs output by current sampling methods do not preserve PPVs, and propose algorithms for creating PPV-preserving subgraphs on *any* given subset of graph nodes. Experiments on three large real-world graphs show that the subgraphs created by our method improve upon induced subgraphs in terms of preserving PPVs, clustering accuracy, and maintaining basic graph properties.

1. Introduction

Consider the problem of clustering a large graph or finding communities in a social network. There are many available methods, and their performance varies considerably across graphs, so picking the right method for a given graph is important. Not only that, but each method has its own parameter settings that need to be tuned (e.g., the conductance threshold for (Andersen et al., 2008), or the edge-weight transform function for (van Dongen, 2000)). On a small-sized dataset, the obvious solution is to search the space of parameters iteratively until a good parameter setting is found. However, each iteration requires running the clustering algorithm on the entire graph and observing the quality of the solution. Moreover, this tuning must be performed for every available clustering method to pick the best one. Clearly, this does not scale well to large graphs.

One solution is to find smaller graphs $\{S_1, S_2, \ldots\}$ that mimic the full graph G, and perform tuning and model selection on these S_i. For instance, if we knew the generative model of G, then we could generate S_i from this model. However, real-world graphs rarely match any generative model in all respects, and even then, finding the model parameters might be too expensive.

Alternatively, we could build small graph *samples* from G itself. There has been much recent work on picking "representative" samples of a graph that preserve degree distributions, clustering coefficients, and so on (Leskovec & Faloutsos, 2006; Maiya & Berger-Wolf, 2010). However, these measures might not be relevant for the application (say, clustering), and the measures that are critical to the application (say, pairwise similarities between nodes, conductance between node subsets, etc.) might not be preserved in the sample. Thus, the challenges are twofold: (a) identify a set of measures that are important for a wide range of applications, and (b) design a sampling method that preserves these particular measures, in addition to usual ones like degree distributions.

A Common Measure. Random walk based algorithms have been used in many common graph applications: propagating labels in classification (Szummer & Jaakkola, 2001), finding spam webpages (Gyöngyi et al., 2004), clustering (Andersen et al., 2008), search in peer-to-peer networks (Gkantsidis et al., 2006), collaborative filtering applications (Gori & Pucci, 2007), ranking (Haveliwala, 2003), etc. In addition, ran-

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

dom walks have been shown to be strictly related to other fundamental graph quantities, such as normalized cuts (Meila & Shi, 2000) and the graph spectrum (Orponen et al., 2008). This motivates our goal of creating graph samples that "preserve random walks", that is preserve probabilities p_{ij} of a walk starting from node i hitting node j, for all i, j in the sample graph. These are in fact the Personalized Pagerank Values or PPVs (ignoring the restart probability), and they reflect the pairwise similarities between nodes. Indeed, in all the applications above, random walks are used to compute such similarities (possibly implicitly), which are in turn used for clustering, classification, ranking, etc. Thus, our goal is to build a graph sample where the PPVs between all pairs of nodes in the sample are preserved. Such samples are useful not only for random-walk applications, but also for applications optimizing seemingly unrelated measures — as we show empirically in Section 6.

Sampling Method. Graph sampling typically consists of two parts: (a) picking a subset S of nodes from the full graph G, and (b) selecting the edges between nodes in S. Current sampling algorithms focus almost exclusively on part (a); for part (b), they simply retain those edges in G both of whose endpoints fall in S (call this the "induced" subgraph). Unfortunately, induced subgraphs often fail to preserve PPVs. For instance, two nodes in S that had many common neighbors in G but were not directly linked may become unreachable from each other in the induced subgraph, even if intuitively they are "close" in G and ought to be "close" in the sample as well. Since PPV between these nodes is likely to be high, they would remain "close" in a PPV-preserving subgraph. This motivates us to focus on part (b), and in particular on *creating* a subgraph that preserves PPVs between all pairs of nodes in S, whose edges might, or not, have existed in G. A major advantage of our method is that it applies to *any* subset S, so we can leverage all previous algorithms developed for part (a).

To illustrate the above we show, in Figures 1-3, a graph of about 2,000 nodes, the induced subgraph on a 200-node sample, and the subgraph created by our algorithm on the same sample and having the same average node degree as the full graph. Observe that the induced subgraph only preserves edges between immediate neighbors, leading to many disconnected components. The subgraph generated by our algorithm better preserves graph structure by creating "bridging" edges between components that are most tightly connected (according to PPV) in the original graph.

Results. Our contributions are the following:

(1) We show that induced subgraphs do not preserve PPVs (Section 4), and that even the simpler problem of preserving global PageRank in induced subgraphs is difficult.
(2) We show that, while it might not be possible in general to preserve PPVs for a given subset, it is possible to *exactly* match PPVs by adding just one extra "sink" node to this subset. We present algorithms implementing this approach on an *arbitrary* subset of nodes of a directed graph (Section 5.1). Additionally, we show that adding an extra "source" node allows to exactly maintain global PageRank values as well (Section 5.3).
(3) Perfectly preserving PPVs may require weighted edges between all pairs of nodes in the subset. To reduce the storage requirements, we present rounding methods that remove a large fraction of these edges while still providing provable bounds on the distortion of PPVs (Section 5.2).
(4) Finally, we verify via experiments on three large real-world networks that subgraphs generated by our algorithms outperform existing methods in terms of preserving PPVs, improving clustering accuracy, and preserving basic graph properties. (Section 6).

Due to space constraints, complete proofs of our claims will appear in the full version of the paper.

2. Related Work

Sampling from large graphs while preserving graph properties was introduced by Leskovec and Faloutsos (2006). They proposed several criteria for "good" graph samples and empirically evaluated several node sampling algorithms with respect to these criteria. They observed that different algorithms are better at preserving some graph properties but not others, and that there is no clear winner among the evaluated algorithms. Another line of work considers sampling subgraphs to optimize for one specific goal, such as the preservation of community structures (Maiya & Berger-Wolf, 2010), or the ability to visualize the graph (Jia et al., 2008). Hubler et al. (2008) proposed a generic Metropolis algorithm for sampling nodes so as to minimize distortion of a given graph property. However, the number of steps until convergence is not known in advance (and may be large), and each step requires the evaluation of the distortion function over numerous subgraphs, which may be costly. Our work deviates from all of these in that we are not concerned with sampling of nodes, but rather, given a subset of nodes, we consider the general problem of maintaining multi-scale graph structure by preserving a distance metric (personalized PageRank) between all pairs of

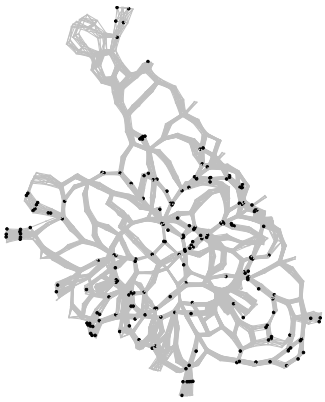

Figure 1. The original graph. The 200 sampled nodes are black.

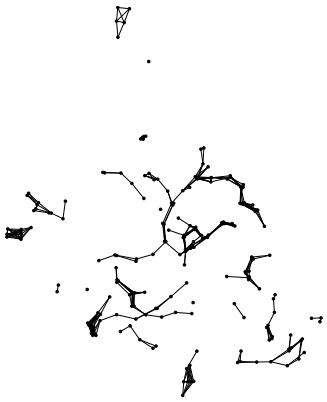

Figure 2. The induced subgraph on the sample.

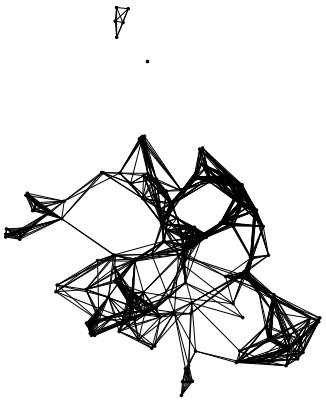

Figure 3. The subgraph produced by our algorithm (UNWEIGHTED).

sampled nodes.

Instead of sampling, Sala et al. (2010) proposed generating synthetic graphs that have similar properties to the original graph. While having advantages like privacy preservation, this approach tries to solve the harder problem of generating a small representative graph using only few parameters, and thus needs to assume a model of the original graph. Our sampling based approach uses more information about the original graph and thus is applicable to graphs that do not accurately follow any known model.

Another related area of research is graph sparsification. Edge sparsification only decreases the number of edges, leaving the set of nodes unchanged. Thus it is not applicable when the goal is to obtain a significantly smaller representative graph. In vertex sparsification, the number of vertices is reduced. Moitra (2009) proposed an algorithm that, given a graph and a subset of nodes, constructs a new graph on these nodes that approximately preserves minimum (unnormalized) cuts. In many applications such as clustering or classification, PPV-related measures like normalized cuts and conductance are believed to be much more effective than unnormalized cuts (Kannan et al., 2004). We thus focus on preserving personalized PageRank which better reflects the relative importances of nodes.

3. Preliminaries

Consider a directed graph $G = (V, E)$ of n nodes, its adjacency matrix A and diagonal out-degree matrix D. The PageRank of G is defined as the vector $\mathbf{p}$ that solves the following system:

$$\mathbf{p} = \alpha\mathbf{r} + (1-\alpha)A^t D^{-1}\mathbf{p}, \tag{1}$$

where $0 < \alpha < 1$ is the restart probability (typically, $\alpha \approx 0.15$) and $\mathbf{r}$ is the personalization vector. Uniform $\mathbf{r}(i) = \frac{1}{n}$ for all $i \in V$ results in $\mathbf{p}$ being the *global* PageRank. If $\mathbf{r}$ contains 1 in the i-th coordinate and is 0 elsewhere, the solution of Eq. 1 gives the i-personalized PageRank $\mathbf{p}_i$ (Haveliwala, 2003). From such $\mathbf{p}_i$'s and the linearity of Eq. (1), the solution $\mathbf{p}$ for arbitrary $\mathbf{r}$ can be easily derived. A $\mathbf{p}_i(j)$ can also be thought of as the probability of an α-discounted random walk from node i ending in node j:

$$\mathbf{p}_i(j) \quad = \quad \sum_{t=0}^{\infty} \alpha(1-\alpha)^t \mathbf{p}_i^{[t]}(j), \tag{2}$$

with $\mathbf{p}_i^{[t]}(j)$ being defined as

$$\mathbf{p}_i^{[t]}(j) \quad = \quad \sum_{k_1=i,k_2,\ldots,k_{t+1}=j} \prod_{l=1}^{t} \frac{1}{d^+(k_\ell)},$$

where $d^+(k_\ell)$ is the out-degree of node k_ℓ, and the summation is over all paths of length t from i to j. An equivalent formulation is given by the decomposition theorem (Jeh & Widom, 2003):

$$\mathbf{p}_i = \alpha\mathbf{e}_i + (1-\alpha) \sum_{j:(i,j)\in E} \frac{\mathbf{p}_j}{d^+(i)}.$$

The decomposition theorem suggests that the "importance" that a node i gives to node $j \neq i$ is proportional to the average importance that i's out-neighbors give to j. All of these can be easily extended to the case of weighted graphs by replacing inverse out-degrees by normalized weights, i.e., $1/d^+(i) = w(i,j)/\sum_j w(i,j)$, where $w(i,j)$ is the weight of edge (i,j).

4. Induced subgraphs

In this section we examine the problem of finding good representative samples using induced subgraphs. The quality of a subgraph is measured by how well the PageRanks of the nodes in the subgraph are preserved.

Consider an undirected graph $G = (V, E)$, where the degree of node i is denoted by $\deg_G(i)$, and restart probability $0 < \alpha < 1$. Note that α induces a global PageRank vector $\mathbf{p}^G$ ($\mathbf{p}^G(i)$ is the PageRank of i). Suppose we are given a sample of size $0 < k < |V|$ and are interested in finding a subset $S \subset V$ of nodes of size k such that the PageRanks in the subgraph $G[S]$ induced by S are as close as possible to $\mathbf{p}^G$ for all nodes in S. By looking at $\mathbf{p}^{G[S]}$ and $\mathbf{p}^G$ as vectors, we want to minimize some notion of distance between them. Two remarks are necessary: first, $\mathbf{p}^G$ is $|V|$-dimensional, so we only consider its restriction to nodes in S; second, since $\sum_{i \in V} \mathbf{p}^G(i) = 1$ and $\sum_{i \in S} \mathbf{p}^{G[S]}(i) = 1$, we rescale $\mathbf{p}^G$; therefore we consider the $|S|$-dimensional scaled vector $\tilde{\mathbf{p}}^G$, where $\tilde{\mathbf{p}}^G(i) := \mathbf{p}^G(i) / \sum_{j \in S} \mathbf{p}^G(j)$ for all $i \in S$. Now our problem is finding an induced subgraph S of G of size k such that $d(\tilde{\mathbf{p}}^G, \mathbf{p}^{G[S]})$ is minimized, where d is some distance function (e.g. a norm).

We show that this problem is NP-hard for any non-trivial distance function d: we only require that $d(\mathbf{a}, \mathbf{a}) = 0$ and that $d(\mathbf{a}, \mathbf{b}) > 0$ if $\mathbf{a} \neq \mathbf{b}$.

Theorem 1. *For any $0 < \alpha < 1$, it is NP-hard to find a subset $S \subset V$, $|S| = k$, that minimizes $d(\tilde{\mathbf{p}}^G, \mathbf{p}^{G[S]})$ for any (non-trivial) distance function d. The NP-hardness holds even when restricted to cubic graphs.*

Next, we show that there exists a family of graphs where a random subset S is likely to contain a node that has higher PageRank relative to many other nodes in S when computed on the induced subgraph, but lower relative PageRank when computed over the entire graph G. Clearly, such induced subgraphs would fail to preserve PPVs as well. The proof of the theorem uses concentration bounds on the degrees of the nodes in a (Erdös–Rényi) random graph $G(n, p)$.

Theorem 2. *Let $0 < p < 1$ and $G = (V, E)$ be sampled from $G(n, p)$. Let S be a random subset of V with $|S| = o(n/\log n)$. Then, with probability at least $1/2 - o(1)$, there exists an $S' \subset S$ of size at least $\Omega(|S|)$, such that a node u^* of maximum degree in $G[S]$ has degree smaller than every node in S' in G. Therefore, for small enough α, every node in S' has PageRank higher than u^* in G, and lower than u^* in $G[S]$.*

5. Preserving PPV

The previous section showed that preserving even *global* graph properties such as PageRank on *induced* subgraphs is difficult. This is expected, since induced subgraphs only preserve links between immediate neighbors, ignoring multi-hop connections. In this section we relax the problem definition by allowing the creation of edges that did not exist in the original graph, and show that this yields a much better approximation of node-personalized PageRank (PPV).

For the rest of this section we assume that all non-existent edges in G have zero weight, and that outgoing degrees are normalized, i.e., $\sum_{j \in V} w_G(i, j) = 1$ for every $i \in V$. This is without loss of generality since PPVs depend only on the normalized weights. We first show that PPVs can be preserved *exactly* using a weighted graph construction that adds a special "sink" node, and present an algorithm to compute edge weights. However, this graph can contain up to $|S|^2$ edges between all pairs of nodes in S. To alleviate this, we present two sparsification algorithms, one of which yields a sparse weighted graph, and the other a sparse unweighted graph. We bound the error in PPVs introduced by this sparsification. Finally, we show that with the addition of another special "source" node, it is possible to preserve global PageRanks in addition to all pairwise PPVs. We stress that our subgraph generation method can be applied to *any* subset of nodes, so popular node-sampling methods in the literature (Leskovec & Faloutsos, 2006) can be used as a first step to pick a node sample on which our algorithms are applied.

5.1. Constructing a PPV-preserving subgraph

Suppose we are given a weighted directed graph $G = (V, w_G)$, where the edge weight function $w_G : V \times V \to \mathcal{R}^+$ defines the edge set, and a subset $S \subset V$ of nodes. We consider the problem of constructing a new weighted graph $H = (S, w_H)$ only on the subset S that preserves PPVs between nodes in S – that is, all values $\mathbf{p}_i^H(j) = \mathbf{p}_i^G(j)$ for $i, j \in S$.

Observation 3. *There exists a graph $G = (V, E)$ and a subset $S \subset V$ such that no weighted graph $H = (S, w_H)$ (with non-negative weights) preserves all PPVs between nodes in S.*

This can be shown via a linear program that reaches an optimum with zero error if it is possible to preserve all PPVs exactly, and a positive error otherwise. Examples of subsets with non-zero error can then easily be found.

Despite the negative result of Obs. 3, adding an extra "sink" node, having only incoming edges from S, allows to preserve PPVs of nodes in H. The main result of this section is the following.

Theorem 4. *Let $G = (V, w_G)$ be a directed weighted graphs. For any $S \subset V$, there exists a weighted directed graph H on $S \cup \{\text{SINK}\}$ where PPVs between*

nodes in S are preserved: for all $i, j \in S$, $\mathbf{p}_i^H(j) = \mathbf{p}_i^G(j)$. Moreover, there exists a polynomial time algorithm that computes such H given G.

The idea of the algorithm, presented in Table. 1, is to start with the original graph G and remove nodes in $V \setminus S$ one by one. For each node z being removed, the algorithm identifies the set of its neighbors, $N(z)$, removes all edges incident to z, and creates edges between all pairs of its neighbors, and from each neighbor to the sink. Specifically, for every $j, k \in N(z)$, directed edges (j, k) and (k, j) are created (parallel edges are merged). The weight of edge (j, k) captures the probability of a random walk starting at j to reach k through z. These new edges, however, do not capture the probability of the walk stopping at z, so an edge to the sink is created from each $j \in N(z)$.

Algorithm 1 NodeRemoval

Input: $G = (V, w_G), S \subseteq V$.
Output: $H = (S \cup \{\text{SINK}\}, w_H)$ s.t. $\mathbf{p}_i^H(j) = \mathbf{p}_i^G(j)$, $i, j \in S$.
Initialize H to be equal to G
Add a node SINK in H: $w_H(i, \text{SINK}) = 0$ for all i, and $w_H(\text{SINK}, \text{SINK}) = 1$
for each $z \in V \setminus S$ **do**
  Remove z from H (and all edges incident to it)
  for each node $x \neq z$ such that $w_G(x, z) > 0$ **do**
    for each node $y \neq z$ such that $w_G(z, y) > 0$ **do**
      Add a new edge $e^*(x, y)$ in H, and set
      $w_H(e^*(x, y)) \mathrel{+}= (1-\alpha) w_G(x, z) \cdot w_G(z, y) \cdot \sum_{t=0}^{\infty} [(1-\alpha) w_G(z, z)]^t$
    end for
    Add a new edge $e^*(x, \text{SINK})$ in H, and set
    $w_H(e^*(x, \text{SINK})) \mathrel{+}= w_G(x, z) - \sum_{y \neq z : w_G(z,y) > 0} w_H(e^*(x, y))$
  end for
  Set G (and its weights) to be H
end for

Thm. 4 follows from the next two lemmas. Lemma 5 follows from simple algebraic manipulations.

Lemma 5. *In each iteration of* NodeRemoval*, (a) all added edges have non-negative weights, and (b) the (weighted) out-degree of each node sums to* 1.

Lemma 6. *An iteration of* NodeRemoval *preserves PPVs between all but the removed node.*

Proof Sketch. The idea of the proof is to provide a mapping between the contributions to PPVs by walks in G and the contributions to the same PPVs by walks in H. The proof shows that a single walk through a newly added edge in H "summarizes" contributions of countably infinitely many walks that pass through the removed node z in G.

Preserving PPVs on Small Subsets. While the NodeRemoval algorithm can be applied to all graphs, it can be costly for small subsets, for which many nodes need to be removed. For this case, we present an alternative algorithm below.

Recall that $\mathbf{p}_i^H(j)$ is the desired PPV value from node i to j, where $i, j \in S$, and we want this value to be equal to $\mathbf{p}_i^G(j)$. Let P^H be the matrix where $\mathbf{p}_i^H(\cdot)$ is row i. Let W_H be the edge weight matrix in H, i.e., $w_H(i, \cdot)$ is row i in W_H. P^H is known, and we want to find W_H (or equivalently, its transpose W_H^t).

Lemma 7. *We have the following relation:*

$$W_H^t = \frac{1}{1-\alpha} \left(I - \alpha (P^H)^{-1}\right)$$

This follows from manipulations of equation (1). The existence of a solution is guaranteed by Thm. 4. This also implies the uniqueness of the solution.

The initial computation of pairwise PPVs takes $O(E + |S|^2)$ time (using (Sarlós et al., 2006)). This lets us compute W_H via inversion of a matrix of size $(|S|+1)$ at a complexity of $O(|S|^{2.376})$. For small subsets, this can be significantly cheaper than NodeRemoval.

5.2. Rounding the Weighted Matrix

The weighted graph H obtained via NodeRemoval or Lemma 7 could be a dense graph with up to $(k+1)^2$ edges. We may want a much sparser subgraph to reduce storage requirements. Next, we present two algorithms for sparsifying the graph, one generating weighted, and the other unweighted graphs. Still, both maintain weighted edges to the sink (which can be thought of as storing one extra number per node in S); as we will later show in Section 6, these edges are important for maintaining PPVs.

Both algorithms are simple instances of rounding: we set a threshold τ and for all edges (u, v) such that $w_H(u, v) < \tau$, we increase $w_H(u, \text{SINK})$ by $w_H(u, v)$ and then set $w_H(u, v) = 0$. Edges with zero weight can obviously be removed, thus sparsifying the graph. Note that this leaves the weights of the remaining edges unchanged, and all node out-degrees still sum to one. We call this algorithm Weighted.

The second algorithm (called Unweighted) takes the output of Weighted and makes all edges unweighted (except for the edges to the sink). Thus, Weighted and Unweighted give sparse weighted

and unweighted graphs respectively (apart from the weights on edges to sink).

Next, we show that the PPV matrix (as defined before Lemma 7) for a graph rounded by WEIGHTED is close to the PPV matrix for the unrounded graph (where the original PPVs are preserved exactly). Let H denote the subgraph obtained via NODEREMOVAL, and R the subgraph after applying WEIGHTED on H. Also, for any node j in S, let $P^H(j)$ and $P^R(j)$ be the j-th columns of P^H and P^R respectively.

Theorem 8. *For all $j \in S$,*

$$\frac{\|P^H(j) - P^R(j)\|_1}{|S|+1} \leq 2\tau \frac{1-\alpha}{\alpha}.$$

The proof follows from a coupling between the Markov chains representing the full PPV matrix and the rounded one. Thus, the rounded graph created by WEIGHTED provides a good approximation to the PPVs of the original graph, provided τ is not too high.

We remark here that in the unlikely boundary case of a node with extremely high out-degree, each of its outgoing edges might get a small weight and be rounded, leading to a seeming paradox of a high-degree node being rounded to a node of zero degree. However, this implies a large $|S|$ and so the bound will be weak unless τ is correspondingly small as well. Also, the edges that reflect the importance of a node are its incoming edges, and these will still be preserved. In our experiments, we pick τ so as to ensure that the rounded subgraph has as many edges as the induced subgraph; this caused no extreme roundings on our graphs. However, if tighter bounds are desired, a smaller τ must be chosen (and hence, more edges retained).

5.3. Preserving Global PageRanks

In this section we show that it is possible to preserve the global PageRank of each node in S by adding to H (computed in Section 5, without rounding) a single "source" node and edges from it to all nodes in S, or equivalently, storing one extra number per node in S.

The following corollary shows that as long as the nodes removed by NODEREMOVAL have same "weight" in the personalization vector $\mathbf{r}$, which is the case for global PageRank, a single value per each node in S can summarize the contributions of PPVs from $V \setminus S$ to the node in S. Let $\mathbf{v}$ be an $|S|$-dimensional vector containing these contributions: $\mathbf{v}(i) = \sum_{j \in V \setminus S} \mathbf{p}_j^G(i)$, $i \in S$. From the linearity of PageRank (Equation (1)) we get:

Corollary 9. *Suppose we are given H, constructed from G by algorithm* NODEREMOVAL*, and the vector $\mathbf{v}$. Then, for every personalization vector $\mathbf{r}$ such that $\mathbf{r}(j) = \mathbf{r}(j')$ for $j, j' \in V \setminus S$, $\mathbf{r}$-personalized PageRank $\mathbf{p}_\mathbf{r}^G(i)$ can be computed for all $i \in S$.*

The information contained in $\mathbf{v}$ can be naturally integrated into the subgraph by adding a single node we call SOURCE. Let H' be the graph obtained from H by adding a node SOURCE, and weighted edges from SOURCE to other nodes in H (including SINK). Then, standard techniques on H' can be used to compute the global PageRank.

To compute the weights of the new edges from SOURCE, we use an auxiliary graph G' capturing the partial PPV contributions of random walks in G that start outside S and end in S, without ever hitting S in the middle. Intuitively, this accounts for the PPV "mass" that flows into S; the distribution of this mass within S and the loss of mass from S to $V \setminus S$ is already handled by our subgraph on $S \cup \{\text{SINK}\}$. In particular, G' is obtained from G by (a) removing all edges from S to $V \setminus S$, (b) collapsing all the nodes in S into a single node, (c) summing up weights of parallel edges, and (d) adding a self-loop of weight 1 to the collapsed node. Then, for $\ell \in S$, edge weights are set to:

$$w_{H'}(\text{SOURCE}, \ell) = \frac{1}{|V \setminus S|} \frac{1}{\alpha} \sum_{j,k \in V \setminus S} \mathbf{p}_j^{G'}(k) \cdot w_G(k, \ell).$$

It can be shown that the weighted out-degree of SOURCE sums to one or less. To normalize it we add an edge to SINK: $w_{H'}(\text{SOURCE}, \text{SINK}) = 1 - \sum_{i \in S} w_{H'}(\text{SOURCE}, i)$. Now, in a manner similar to Thm. 4, we may prove:

Theorem 10. *H' preserves all PPVs between nodes in S. Moreover, for any $0 < \rho < 1$ and any personalization vector $\mathbf{r}$ such that $\mathbf{r}(j) = \rho$ for all $j \in V \setminus S$, $\mathbf{p}_\mathbf{r}^G(i) = \mathbf{p}_{\mathbf{r}'}^{H'}(i)$, for all $i \in S$, where*

$$\mathbf{r}'(k) = \begin{cases} \mathbf{r}(k), & k \in S \\ \rho|V \setminus S|, & k = \text{SOURCE} \\ 0, & k = \text{SINK} \end{cases}$$

6. Experiments

Given a graph G and a subset S of its nodes we compare the subgraphs WEIGHTED and UNWEIGHTED to the baseline INDUCED. As described in Section 5.2, WEIGHTED is obtained by retaining a subset of weighted edges from our PPV-preserving subgraph, and UNWEIGHTED removes the weights from the edges in WEIGHTED though edges to sink are still weighted. (Our rounding process ensures that WEIGHTED and UNWEIGHTED have the same number of edges as INDUCED.) We empirically determine how well each of

these subgraphs preserves PPVs, and the usefulness of our subgraphs for applications like graph clustering.

DATA. We present results on three large real-world social networks. EPINIONS is a trust-based social network between the users of `epinions.com` website, with $75K$ nodes and $508K$ edges. SLASHDOT is a February 2009 snapshot of the friendship network between users of `slashdot.org` website, with $82K$ nodes and $948K$ edges. FACEBOOK is a snapshot, as of September 2005, of Facebook social network of the University of Oklahoma, with $17K$ nodes and $1.78M$ edges.

SAMPLING SUBSETS. Even though our algorithms can work on any subset of nodes, we show results for subsets generated according to prior work on graph sampling. We generated node samples using five techniques defined in (Leskovec & Faloutsos, 2006): (1) RN: select nodes uniformly at random; (2) RNN: select a node uniformly at random together with all its outneighbors; (3) RNE: select a node uniformly at random together with its random outneighbor; (4) RW: select a node uniformly at random and simulate a random walk with restart to the initial node (with probability 0.15), pick all traversed nodes; (5) RJ: identical to RW but with restart to a random node. For each generator, we experimented with subsets of size 100, 1000, and 10,000, though we only present results for subsets of size 1000 for lack of space. Results with other sizes are similar.

PRESERVING PPV. We compute the PPV from node i to node j for all $i, j \in S$, once using the entire graph and once restricted to just the subgraph under consideration. Figure 4 shows the Frobenius norm of the differences between these values, normalized by the Frobenius norm of the original PPV matrix. Observe that for all datasets and subset sampling methods, WEIGHTED improves over UNWEIGHTED by a factor of 3 on average, and both significantly outperform INDUCED. This agrees with our intuition: WEIGHTED is a closer approximation of our subgraphs which perfectly preserve PPVs; hence it performs better. UNWEIGHTED induces additional error since it loses the information encoded in the weights. The poor performance of INDUCED is in part due to the lack of a sink node. When most of a node's neighbors in G are not in S, the sink node allows us to better preserve PPV by absorbing the probability mass that was going to these neighbors. In fact, INDUCED and UNWEIGHTED share only 50% of their edges on average, showing that significant changes must be made to INDUCED to properly preserve PPVs.

CLUSTERING ON SUBGRAPHS. We evaluate three hierarchical clustering algorithms on the INDUCED and

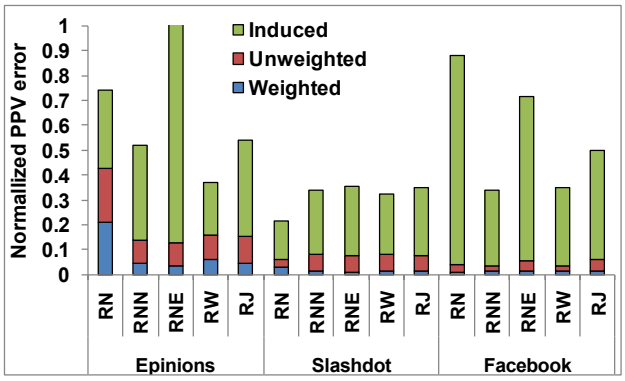

Figure 4. Preserving PPV: WEIGHTED is better than UNWEIGHTED by a factor of 3, and UNWEIGHTED in turn is better than INDUCED by a factor of 10 on average.

UNWEIGHTED subgraphs: *Metis* (Karypis & Kumar, 1999), *Planted* (Condon & Karp, 1999), and *Local* (Andersen et al., 2008). The first two perform recursive graph bisection, while the third performs local clustering based on PageRank vectors. Note that the first two have no obvious relationship with random walks. For a sample S and a seed node $u \in S$, we order all the other nodes in S by their "clustering distance" from u, i.e., the length of the path in the hierarchy tree between the nodes. Intuitively, if the ordering in a subgraph is similar to that in G, then the clustering algorithm gets equivalent results when run on the subgraph or on G, and so any parameter tuning or model selection performed on the subgraph will also hold for G. To measure the similarity of two orderings we consider their prefixes of length $k = 10$ (that is the ten nodes closest to the seed) and compute the Kendall's tau correlation coefficient. The experiment was run with each node in S as the seed node, and the results averaged. Fig. 5 shows, for each dataset and sampling method, the relative improvement of UNWEIGHTED over INDUCED (edges to sink were discarded, so UNWEIGHTED requires the same space as INDUCED here). We see that UNWEIGHTED yields a 50% improvement over INDUCED on average, demonstrating the usefulness of preserving PPVs, even approximately. More improvement is observed with *Metis* w.r.t. *Planted* because the latter uses *unnormalized* cuts; but also w.r.t. *Local* which uses PPVs. A possible explanation is that *Metis* uses a BFS and (implicitly) shortest path distances, which are likely to be composed of a few important edges, whereas a PPV is an average over all paths between two nodes. Since rounding retains the most important edges, it is likely to preserve shortest paths (and hence, *Metis*) better than PPVs (and hence, *Local*). Finally, the RN subsets consistently yield the least improvement. This is

because the induced subgraph for RN is typically extremely sparse (since nodes are picked randomly), and since UNWEIGHTED is allowed no more edges than INDUCED, there is little room for improvement.

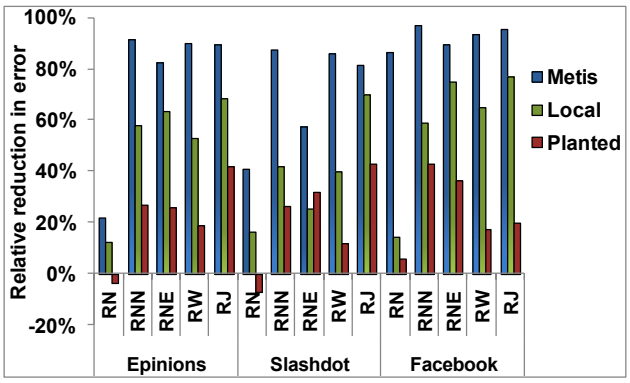

Figure 5. Reduction of error in preserving the clustering hierarchy, for UNWEIGHTED relative to INDUCED.

Conclusions

The PPVs of a graph are important for many applications such as clustering, classification, etc. Methods for representative subgraph construction typically consider only induced subgraphs and fail to preserve PPVs. We show how to build PPV-preserving subgraphs and how to sparsify them while still provably approximating the original PPVs. Experiments on three large real-world graphs show significant improvements of our method in maintaining PPVs, clustering accuracy, as well as in matching degree distributions and clustering coefficients (not shown for lack of space).

Acknowledgments. This work was supported in part by NSF Grant #0905645. We would also like to thank Purnamrita Sarkar for helpful discussions, and the reviewers for their insightful comments.

References

Andersen, R., Chung, F., and Lang, K. Local partitioning for directed graphs using pagerank. *Internet Math.*, 5(1-2):3–22, 2008.

Condon, Anne and Karp, Richard M. Algorithms for graph partitioning on the planted partition model. *Random Structures & Algorithms*, 18:116–140, 1999.

Gkantsidis, C., Mihail, M., and Saberi, A. Random walks in peer-to-peer networks: Algorithms and evaluation. *P2P Computing Systems*, 63(3), 2006.

Gori, M. and Pucci, A. Itemrank: A random-walk based scoring algorithm for recommender engines. In *IJCAI*, pp. 2766–2771, 2007.

Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J. Combating web spam with trustrank. *VLDB*, 2004.

Haveliwala, Taher H. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowl. Data Eng.*, 15(4), 2003.

Hübler, C., Kriegel, H.-P., Borgwardt, K., and Ghahramani, Z. Metropolis algorithms for representative subgraph sampling. In *ICDM*, 2008.

Jeh, Glen and Widom, Jennifer. Scaling personalized web search. In *WWW*, pp. 271–279, 2003.

Jia, Yuntao, Hoberock, Jared, Garland, Michael, and Hart, John. On the visualization of social and other scale-free networks. *IEEE Trans. on Visualization and Computer Graphics*, pp. 1285–1292, 2008.

Kannan, Ravi, Vempala, Santosh, and Vetta, Adrian. On clusterings: Good, bad and spectral. *Journal of ACM*, 51, 2004.

Karypis, George and Kumar, Vipin. A fast and highly quality multilevel scheme for partitioning irregular graphs. *Journal on Scientific Computing*, pp. 359–392, 1999.

Leskovec, Jure and Faloutsos, Christos. Sampling from large graphs. In *KDD*, pp. 631–636, 2006.

Maiya, Arun S. and Berger-Wolf, Tanya Y. Sampling community structure. In *WWW*, pp. 701–710, 2010.

Meila, Marina and Shi, Jianbo. Learning segmentation by random walks. In *NIPS*, pp. 873–879, 2000.

Moitra, Ankur. Approximation algorithms for multicommodity type problems with guarantees independent of the graph size. In *FOCS*, pp. 3–12, 2009.

Orponen, P., Schaeffer, S., and Gaytán, V. Locally computable approximations for spectral clustering and absorption times of random walks. *CoRR*, abs/0810.4061, 2008.

Sala, A., Cao, L., Wilson, C., Zablit, R., Zheng, H., and Zhao, B. Measurement-calibrated graph models for social network experiments. In *WWW*, 2010.

Sarlós, T., Benczúr, A., Csalogány, K., Fogaras, D., and Rácz, B. To randomize or not to randomize: space optimal summaries for hyperlink analysis. In *WWW*, pp. 297–306, 2006.

Szummer, M. and Jaakkola, T. Partially labeled classification with markov random walks. In *NIPS*, 2001.

van Dongen, Stijn Marinus. *Graph clustering by flow simulation*. PhD thesis, Univesity of Utrecht, 2000.