Sparse Cut Projections in Graph Streams

Atish Das Sarma[1], Sreenivas Gollapudi and Rina Panigrahy[2]

[1] atish@cc.gatech.edu, Georgia Institute of Technology,
[2] {sreenig, rina}@microsoft.com, Microsoft Research, Silicon Valley

Abstract. Finding sparse cuts is an important tool for analyzing large graphs that arise in practice, such as the web graph, online social communities, and VLSI circuits. When dealing with such graphs having billions of nodes, it is often hard to visualize global partitions. While studies on sparse cuts have traditionally looked at cuts with respect to all the nodes in the graph, some recent works analyze graph properties projected onto a small subset of vertices that may be of interest in a given context, e.g., relevant documents to a query in a search engine. In this paper, we study how sparse cuts in a graph partition a certain subset of nodes. We call this partition a *cut projection*. We study the problem of finding cut projections in the streaming model that is appropriate in this context as the input graph is too large to store in main memory. Specifically, for a d-regular graph G on n nodes with a cut of conductance Φ and constant balance, we show how to partition a randomly chosen set of k nodes in $\tilde{O}(\frac{1}{\sqrt{\alpha\Phi}})$ passes over the graph stream and space $\tilde{O}(n\alpha + \frac{n^{3/4}k^{1/4}}{\sqrt{\alpha}\Phi^{19/4}})$, for any choice of $\alpha \leq 1$. The resulting partition is the projection of a cut of conductance of at most $\tilde{O}(\sqrt{\Phi})$. We note that for $k < n\alpha^6\Phi^{O(1)}$, this can be done in $\tilde{O}(1/\sqrt{\alpha\Phi})$ passes and space $\tilde{O}(n\alpha)$ that is sublinear in the number of nodes.

1 Introduction

The problem of finding sparse cuts on graphs has been studied extensively [6, 5, 13, 4, 22, 20]. Sparse cuts form an important tool for analyzing/partitioning real world graphs, such as the web graph, click graphs from search engine query logs and online social communities [8]. While traditionally studies on sparse cuts have looked at cuts with respect to all the nodes in the graph, more recent works [17, 16] study graph properties projected onto a small subset of nodes that may be of interest. For example, while the web graph may consist of several billions of nodes, in a given context, one may only be interested in the most important nodes such as those with high PageRank or those nodes (representing web pages) relevant to a specific search query. Specifically, we may be interested in finding how connected components of the graph partition these nodes, or we may wish to compute the diameter of the graph with respect to these nodes, or perhaps compute the distance between these nodes with respect to the original graph. Such operations not only enable us to understand the structure of a facet of the graph, but also make it feasible to visualize the graph using a much smaller set of nodes. This approach has been taken in several studies including HITS [14], SALSA [15], and web projections [16], where they study the properties of the web graph restricted to a set of documents that match a given query.

A well developed framework for studying large graphs under memory constraints is the streaming model wherein the input graph is assumed to be on disk, and the algorithm is allowed to make few sequential passes over the input while using a small amount of space in main memory. Our approach works on the streaming model with sub-linear space. The space and pass requirements on streaming algorithms can vary significantly depending on the problem. Demetrescu et. al. [10] give an excellent exposition on the space-passes trade off in graph streaming problems. Henzinger et. al. [12] showed linear lower bounds on the "space $\times$ passes" product for several graph problems including connectivity and shortest path problems. In this work, we present a streaming algorithm for finding how a sparse cut partitions a small random set of nodes when the graph is presented as a stream of edges in no particular order. Our streaming algorithm uses space sublinear in the number of nodes. We also provide an algorithm for finding a sparse cut on the entire graph. We now introduce some definitions below.

Definition 1 (Conductance and Sparsity). *The conductance of a graph $G = (V, E)$ of n nodes $1, 2, \ldots, n$ is defined as $\Phi(G) = \min_{S:E(S)\leq E(V)/2} \frac{E(S,V\setminus S)}{E(S)}$ where $E(S, V \setminus S)$ is the number of edges crossing the cut $(S, V \setminus S)$ and $E(S)$ is the number of edges with at least one end point incident on S. For d-regular graphs, $\Phi(G) = \min_{S:|S|\leq|V|/2} \frac{E(S,V\setminus S)}{d|S|}$. Further, this is within a factor two of $\min_S \frac{nE(S,V\setminus S)}{d|S||V\setminus S|}$. We also note that the sparsity of a d-regular graph is related to the conductance by a factor d.*

Definition 2 (Balance). *The balance of a cut $(S, V \setminus S)$ is defined as $\min\{\frac{|V\setminus S|}{|V|}, \frac{|S|}{|V|}\}$.*

Definition 3 (Cut Projections). *Given a cut $(S, V \setminus S)$, we will say that $(S \cap U, V \setminus S \cap U)$ is a projection of the cut $(S, V \setminus S)$ on U. Further, we will say that a cut $(C, U \setminus C)$, where $C \subseteq U$, is a projected cut of conductance Φ if it is a projection of a cut $(S, V \setminus S)$ with conductance Φ.*

1.1 Contributions of this study

Our approach builds on the streaming algorithms presented in [9] for performing a large number of random walks efficiently on a graph stream. These random walks are used to estimate the probability distribution of the random walk that is in turn be used to find a sparse cut by adapting the method of Lovasz and Simonovits [18, 22].

One of the main contributions of this paper is an algorithm to estimate the probability distributions on an arbitrarily chosen subset of k nodes in a d-regular graph. To obtain the probability of reaching destination t from source s after a walk of length l, the algorithm runs multiple walks (starting with length $l/2$) from source-destination pairs, and recursively estimates probability distributions of mid-points, by looking at the "collisions" of these walks. A similar idea has been used in property testing for expander graphs in [11]. However, in their case, they just need to run walks of length $l/2$ and investigate the collisions. Since we need a good estimate of the probability distribution at t, the algorithm needs to run walks recursively of shorter lengths. All our techniques depend on the reversibility of the random walk, and hence only work for d-regular, unweighted graphs. We now describe our results beginning with a definition of some notation.

Definition 4. *Let $P_l[st]$ denote the probability of landing at node t after a random walk of length l starting from s. Further, let $p_l(i) = P_l[si]$. We drop the subscript l when it is clear from context.*

The following theorem, proved in Section 3, shows how to compute the approximate distribution on a arbitrarily chosen subset K of k nodes.

Theorem 1. *Given an arbitrarily chosen subset K of k nodes, one can compute an estimate $\tilde{p}(i)$ for $p(i)$ (the probability distribution after a walk of length l) for all $i \in K$ in $\tilde{O}(\sqrt{\frac{l}{\alpha}})$ passes and $\tilde{O}(n\alpha + \frac{1}{\epsilon}\sqrt{\frac{nkl}{\alpha}})$ space for any choice of $\alpha \leq 1$, such that the error in the estimate $|\tilde{p}(i) - p(i)|$ is at most $\tilde{O}(l\sqrt{\frac{p(i)\epsilon}{n}} + \frac{l\epsilon}{n} + (l\sqrt{\epsilon})p(i))$.*

Our main results for computing projected cuts (described in Section 4) with sparsity at most $\tilde{O}(\sqrt{\Phi})$ are stated below.

Theorem 2. *For any d-regular graph G that has a cut of balance b and conductance at most Φ, given a set K of randomly chosen k nodes, we show that there is an algorithm that achieves the following on a graph stream (for any choice of $\alpha \leq 1$).*
(a) Partitions K into two sets such that the partitioning is a projected cut of conductance at most $\tilde{O}(\sqrt{\Phi})$, in $\tilde{O}(\frac{1}{\sqrt{\alpha\Phi}})$ passes and $\tilde{O}(n\alpha + \frac{n^{3/4}k^{1/4}}{b\sqrt{\alpha}\Phi^{19/4}})$ space.
(b) Outputs k candidate partitions such that at least one of them is a projected cut of conductance at most $\tilde{O}(\sqrt{\Phi})$, in $\tilde{O}(\frac{1}{\sqrt{\alpha\Phi}})$ passes and $\tilde{O}(n\alpha + \frac{\sqrt{nk}}{b\sqrt{\alpha}\Phi^{\frac{9}{2}}})$ space.

Corollary 1. *Given a set of randomly chosen $k \leq n\alpha^6 b^4 \Phi^{O(1)}$ nodes, there is an algorithm that partitions them, in $\tilde{O}(\frac{1}{\sqrt{\Phi\alpha}})$ passes and $\tilde{O}(n\alpha)$ space, into a projected cut of conductance at most $\tilde{O}(\sqrt{\Phi})$ w.h.p.*

Observe that the space required is sublinear in the number of nodes if k satisfies the bound in the above corollary. Our algorithms can also be extended to partition all the n nodes in the graph; that is, find the entire (approximate, sparse) cut. The following theorem shows how find an approximate sparse cut in (possibly) sublinear space. The proof is detailed in the full version of the paper.

Theorem 3. *For any d-regular graph G that has a cut of conductance at most Φ and balance b, there is an algorithm that performs $\tilde{O}(\sqrt{\frac{1}{\Phi\alpha}})$ passes over the graph stream and using space $\tilde{O}(\min\{n\alpha + \frac{1}{b}\left(\frac{n\alpha}{d\Phi^3} + \frac{n}{d\sqrt{\alpha}\Phi^{5/2}}\right), (n\alpha + \frac{1}{b}\frac{n}{d\alpha\Phi^2})\sqrt{\frac{1}{\Phi\alpha}} + \frac{1}{\Phi}\})$, for any choice of $\alpha \leq 1$. and outputs, with high probability, a cut of conductance at most $\tilde{O}(\sqrt{\Phi})$.*

1.2 Related Work

A well-known approach for graph partitioning is to compute the second eigenvector that can be used to compute a sparse cut by ordering the nodes in increasing order of coordinate value in the eigenvector. The second eigenvector technique has been analyzed in a series of results [2, 7, 21] relating the gap between the first and second eigenvalue.

The best known approximation algorithm to compute the sparsest cut in a graph is due to Arora, Rao, and Vazirani [4]. They provide $O(\sqrt{\log n})$-approximation algorithm using semi-definite programming techniques. While their algorithm guarantees good approximation ratios, it is slower than algorithms based on spectral methods and random walks.

Lovasz and Simonovits [18, 19] proposed another approach to finding cuts of small conductance. They showed how random walks can be used to find sparse cuts. Specifically, they show that if you start a random walk from a certain node and order the nodes by the probability of reaching them, then this ordering contains a sparse cut. They prove that if the sparsest cut has conductance ϕ, then their method can be used to find a cut with conductance at most $O(\sqrt{\phi})$.

Spielman and Teng [22] build upon the work of Lovasz and Simonovits and show how it can be implemented more efficiently by sparsifying the graph. They show that for a dense graph, it is possible to look at a near linear number of edges and only compute the sparse cuts on the sampled set of vertices. Given a graph $G = (V, E)$ with a cut $(S, V \setminus S)$ with sparsity ϕ and balance $b(S) = |e(S)|/2|E| \geq 1/2$ where $e(.)$ denotes the set of edges incident on nodes in S, their algorithm finds a cut $(D, V \setminus D)$ with sparsity $O(\phi^{1/3} \log^{O(1)} n)$ and balance of the cut $(D, V \setminus D)$, $b(D) \geq b(S)/2$.

Andersen, Chung, and Lang [3] proposed a local partitioning algorithm to find cuts near a specified vertex and global cuts. The running time of their algorithm was proportional to the size of small side of the cut. Their results improve upon those in [22];

In a more recent work [9], the authors proposed algorithms to perform several random walks efficiently on graphs presented as edge streams using a small number of passes. A recent study [1] shows how to find $1 + \epsilon$-approximate sparse cuts in $\tilde{O}(n)$ space by making use of graph sparsifiers. In contrast, our algorithm requires sublinear space for a certain range of parameters, but provides much a weaker approximation to the sparsest cut.

2 Cuts from approximate probability distributions of random walks

In this section we will show how one can compute candidate sparse cuts from approximate probability distributions of random walks. We start from a random source s from the smaller side of the best cut with conductance Φ and perform a random walk of length about $1/\Phi$. We extend the algorithm of Lovasz and Simonovits [18] to find sparse cuts using approximate distributions. This is similar to the work by Spielman and Teng [22] that also works with estimates of $p(i)$. But the magnitude of error allowed in our work is larger than in theirs. We adapt a set of lemmas from their work to prove Theorem 4 below. The proof is detailed in the full version of this paper.

Definition 5. *For a probability distribution $p(i)$ on nodes, let $\rho_p(i) = p(i)/d(i)$. Let π_p denote the ordering of nodes in decreasing order of ρ_p; that is, $\rho_p(\pi_p(i)) \geq \rho_p(\pi_p(i + 1))$.*

Recall that $p(i)$ denotes the probability of ending at node i after a random walk of length l. The following theorem shows how one can find candidate sparse cuts using

approximate values $\tilde{p}(i)$ of $p(i)$. It looks at the n candidate cuts obtained by ordering the nodes in the order $\pi_{\tilde{p}}$.

Theorem 4. *Let $(U, V \setminus U)$ (with $|U| \leq |V|/2$) be a cut of conductance at most Φ. Let $\tilde{p}(i)$ denote an estimate for the probability $p(i)$ of a random walk of length l from a source s from U. Assume that $|\tilde{p}(i) - p(i)| \leq \epsilon(p(i) + 1/n)$, where $\epsilon \leq o(\Phi)$. Consider the n candidate cuts obtained by ordering the vertices in decreasing order of $\rho_{\tilde{p}}(i)$; each candidate cut $(S, V \setminus S)$ is obtained by setting S equal to a prefix $S_j = \pi_{\tilde{p}}\{1, 2, \ldots, j\}$. If the source node s is chosen randomly from U and the length l is chosen randomly in the range $\{1, \ldots, O(1/\Phi)\}$, then with constant probability, one of these n candidate cuts has conductance $\Phi(S_j) \leq \tilde{O}(\sqrt{\Phi})$*

Note that the source node s needs to be sampled from U, the smaller side of the cut. To obtain such a source, we have to sample several sources from V, since U is not known, and execute the algorithm in parallel (so as not to increases the number of passes required). Given a cut of balance b, this increases the number of walks required by a factor of $\tilde{O}(\frac{1}{b})$, and therefore the space required accordingly; in all our space bounds, the first term of $n\alpha$, however, does not depend on the number of walks performed.

In section 4 we will show how Theorem 4 in conjunction with Theorem 1 is used to prove Theorem 2. The essential idea is to look at the k candidate cuts obtained by arranging the nodes in decreasing order of $\pi_{\tilde{p}}$ and then estimate the conductance across these candidate cuts to pick the best one.

In proving these theorems, we use the techniques presented in [9] for performing a large number of random walks efficiently on a graph stream. They show how to perform $O(n/l)$ independent random walks using $\tilde{O}(n\alpha)$ space and $\tilde{O}(\sqrt{\frac{l}{\alpha}})$ passes over the graph stream. Their main result is stated below.

Theorem 5 ([9]). *One can perform k independent random walks from a given source distribution, on a graph stream, in $\tilde{O}(\sqrt{l/\alpha})$ passes and $\tilde{O}(\min\{n\alpha + kl\alpha + k\sqrt{l/\alpha}, n\alpha\sqrt{l/\alpha} + k\sqrt{l/\alpha} + l\})$ space, for any choice of $\alpha \leq 1$. For $k = \frac{n}{l}$ walk, this requires $\tilde{O}(\sqrt{l/\alpha})$ passes and $\tilde{O}(n\alpha)$ space for $1/l \leq \alpha \leq 1$.*

Next we will show how performing random walks can be used to compute the probability distribution approximately so that we may apply theorem 4. To get good approximations, we need to perform random walks recursively as shown in the next section.

3 Estimating probability distribution p_i on a small set of nodes

In this section we show how to estimate the probability distribution on a small set of k nodes resulting in Theorem 1. The distribution is required for the endpoint of a random walk of length l from a specific source node s (or more generally a source distribution). The naïve approach would be to perform several random walks of length l from s and look at the end points of these walks to see how many times each of the k nodes occurs. This can be inefficient as k may be much smaller than n and most of the random walks may end up at nodes other than the k nodes we are interested. So we seek a more

efficient approach tailored towards estimating the distribution of a specific small set of nodes.

We will begin by stating the following technical lemma. The lemma is later used to approximate distributions. It bounds the error in estimating a_{ij} for a matrix A, where i and j are drawn from two different probability distributions. The guarantee is stated as a trade-off with the number of samples drawn for i and for j.

Lemma 1. *Let $A = \{a_{ij}\}_{m \times n}$ denote a matrix of non-negative entries a_{ij}. Let $\mu_{XY} = E_{i \in X, j \in Y}[a_{ij}]$ denote the expected value of a_{ij} where i and j are drawn independently from probability distributions $X = \{x_i, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ on the rows and columns respectively. Assuming $||A^t x||_\infty \leq \tilde{O}(\frac{1}{\epsilon n_x})$ and $||Ay||_\infty \leq \tilde{O}(\frac{1}{\epsilon n_y})$, one can obtain an estimate $\mu_{\tilde{X}\tilde{Y}}$ for μ_{XY} by drawing $\tilde{O}(n_x)$ samples from X and $\tilde{O}(n_y)$ samples from Y. Here $\tilde{X}$ and $\tilde{Y}$ are the distributions induced by the $\tilde{O}(n_x)$ and $\tilde{O}(n_y)$ samples respectively. The error $|\mu_{\tilde{X}\tilde{Y}} - \mu_{XY}|$ is at most $\tilde{O}(\sqrt{\frac{\mu_{XY}}{\epsilon n_x n_y}} + \frac{1}{\epsilon n_x n_y})$ w.h.p.*

Proof. Let $\mu_D = E_{i \in D}[c_i]$. For a distribution D and a vector c with non-negative entries between $[0, 1]$, $\tilde{O}(n)$ samples are drawn from D to estimate μ_D w.h.p such that $|\mu_{\tilde{D}} - \mu_D| \leq \sqrt{\frac{\mu}{n}} + \frac{1}{n}$ by Chernoff bounds. More generally, $|\mu_{\tilde{D}} - \mu_D| \leq \sqrt{\frac{\mu ||c||_\infty}{n}} + \frac{||c||_\infty}{n}$.

We need to bound $|\mu_{XY} - \mu_{\tilde{X}\tilde{Y}}| \leq |\mu_{XY} - \mu_{X\tilde{Y}}| + |\mu_{\tilde{X}Y} - \mu_{\tilde{X}\tilde{Y}}|$. Set $c_i = E_{j \in Y}[a_{ij}]$, i.e., $c = Ay$. This gives $|\mu_{XY} - \mu_{\tilde{X}Y}| \leq \sqrt{\frac{\mu_{XY} ||Ay||_\infty}{n_x}} + \frac{||Ay||_\infty}{n_x} \leq \sqrt{\frac{\mu_{XY}}{\epsilon n_x n_y}} + \frac{1}{\epsilon n_x n_y}$.

Further, setting $c_j = E_{i \in \tilde{X}}[a_{ij}]$ or $c = A^t \tilde{x}$ gives $|\mu_{\tilde{X}Y} - \mu_{\tilde{X}\tilde{Y}}| \leq \sqrt{\frac{\mu_{\tilde{X}Y} ||A^t \tilde{x}||_\infty}{n_y}} + \frac{||A^t \tilde{x}||_\infty}{n_y}$ w.h.p. But, note that $||A^t \tilde{x}||_\infty \leq ||A^t x||_\infty + \tilde{O}(\sqrt{\frac{A^t ||x||_\infty}{n_x}} + \frac{A^t ||x||_\infty}{n_x}) \leq \tilde{O}(\frac{1}{\epsilon n_x})$ as $||A^t x||_\infty \leq \tilde{O}(\frac{1}{\epsilon n_x})$. And since $\mu_{\tilde{X}Y} \leq \mu_{XY} + \tilde{O}(\sqrt{\frac{\mu_{XY}}{\epsilon n_x n_y}} + \frac{1}{\epsilon n_x n_y}) \leq \tilde{O}(\mu_{XY} + \frac{1}{\epsilon n_x n_y})$, the difference is at most $\tilde{O}(\sqrt{\frac{(\mu_{XY} + \frac{1}{\epsilon n_x n_y}) 1/\epsilon n_x}{n_y}} + \frac{1}{\epsilon n_x n_y}) \leq \tilde{O}(\sqrt{\frac{\mu_{XY}}{\epsilon n_x n_y}} + \frac{1}{\epsilon n_x n_y})$. Combining the two, the lemma follows. ∎

We first describe the main idea in estimating the probabilities after a random walk for a subset of nodes as Algorithm RECURSIVERANDOMWALK. The algorithm uses a parameter m that controls the accuracy of error in estimation. Given a set of nodes K, with $|K| = k$, and a source node s, we wish to estimate $P_l[st]$ for all nodes in $t \in K$ up to an additive accuracy of about $O(\sqrt{P_l[st]/m})$. Rather than performing m walks of length l from s, $\tilde{\Theta}(\sqrt{mk})$ walks are performed from s and $\tilde{\Theta}(\sqrt{\frac{m}{k}})$ walks from each node in K, all of length $l/2$. Note that the product of the number of walks performed is m. We then use collisions in the end points of these walks of length $l/2$ to estimate the probabilities (since it is a reversible random walk). The key insight is that $P_l[st] = \sum_i P_{l/2}[su_i].P_{l/2}[tu_i] = \sum_i x_i y_i$ where $x_i = P_{l/2}[su_i]$ and $y_i = P_{l/2}[tu_i]$.

That is, one can break all paths from s to t at half way, and sum over all $l/2$ length paths from s to u_i and u_i to t. Any node u_i may either have a *small* or a *large* probability of being reached from s after a random walk of length $l/2$; the same observation holds for for t as well. Roughly, a node u has a small probability for source s if $P_{l/2}[su]$ is $o(1/\sqrt{mk})$, and large probability otherwise. In the formal description of the algorithm we denote these sets of nodes by S_a and S_b respectively. Notice that $o(1/\sqrt{mk})$ is less than the reciprocal of the number of walks run from s. We denote the number of walks of length $l/2$ performed from s by n_s. Similarly, a node u has small probability estimate from t if $P_{l/2}[tu]$ is $o(\sqrt{m/k})$, and a large probability otherwise. In the algorithm, these sets of nodes are denoted by T_a and t_b respectively. The total number of walks of length $l/2$ performed from each node t is denoted by n_t. Now, four cases arise for every u_i.

- $x_i y_i$ is $\Omega(1/m)$ and x_i is $\Omega(\frac{1}{\sqrt{mk}})$ and y_i is $\Omega(\sqrt{\frac{k}{m}})$.
- $x_i y_i$ is $o(1/m)$ and x_i is $o(\frac{1}{\sqrt{mk}})$ and y_i is $o(\sqrt{\frac{k}{m}})$.
- $x_i y_i$ is $\Omega(1/m)$ but x_i is $o(\frac{1}{\sqrt{mk}})$ and y_i is $\Omega(\sqrt{\frac{k}{m}})$.
- $x_i y_i$ is $\Omega(1/m)$ but x_i is $\Omega(\frac{1}{\sqrt{mk}})$ and y_i is $o(\sqrt{\frac{k}{m}})$.

The first case is when u_i has large $P_{l/2}[su_i]$ and $P_{l/2}[tu_i]$, and therefore, it will be seen in walks from both ends and gives us a good estimate of u_i's contribution to $\sum_i P_{l/2}[su_i].P_{l/2}[tu_i]$. The second case is when u_i has small probability for both s and t. In this case, w.h.p., u_i will not be seen in either set of walks. Therefore, u_i's contribution to $\sum_i P_{l/2}[su_i].P_{l/2}[tu_i]$ cannot be estimated. However, since $P_{l/2}[su_i].P_{l/2}[tu_i]$ itself is $o(1/m)$, u_i's contribution to the estimate of $P_l[st]$ is *negligible*.

The difficulty arises in estimating the product for u_i in the third and fourth cases. In both these scenarios, just the walks described above aren't sufficient to estimate the product and yet the contribution may be significant. This is because u_i has a large probability from one end, but a small probability from the other end. The small probability cannot be estimated with just these walks, but the product could be significant, in particular the product could be $\Omega(1/m)$. Hence one needs to resort to a recursive estimation algorithm where the small estimate is captured by performing further walks.

For any node u_i with a large value of $P_{l/2}[su_i]$ and a small value of $P_{l/2}[tu_i]$, we adopt a recursive approach between u_i and t by performing random walks of length $l/4$ from all such u_i and from t. Similarly, for all nodes u_i that have a large value of $P_{l/2}[tu_i]$ and a small value of $P_{l/2}[su_i]$, we perform random walks of length $l/4$ from s and all these u_i to get better estimates of $P_{l/2}[su_i]$ and consequently the product $P_l[st]$. These probabilities may themselves be estimated by random walks of length $l/8$, in another depth of the recursion, and so on. Eventually, combining all of these carefully gives us the probability distribution of t from s after a random walk of length l (notice that we use the reversibility of the random walk). The exact details are stated in Algorithm 1.

We now state and prove the guarantee of RECURSIVERANDOMWALK in estimating probabilities by making use of Lemma 1.

Lemma 2. *Algorithm* RECURSIVERANDOMWALK *gives an estimate of* $\mu = P_l[st]$ *within an additive error of* $\tilde{O}(l\sqrt{\epsilon}\mu + l\sqrt{\frac{\mu}{\epsilon m}} + \frac{l}{\epsilon m})$.

Algorithm 1 RECURSIVERANDOMWALK(s, K, l)

1: **Input:** Starting node/distribution s, length l, and chosen set of k nodes K. Set K need not necessarily be chosen at random.
2: **Output:** $p(t) = \tilde{P}_l[st]$ for each $t \in K$, an estimate of $P_l[st]$ with explicit bound on additive error.
3: Perform $n_s = \tilde{\Theta}(\sqrt{mk})$ walks from s and $n_t = \tilde{\Theta}(\sqrt{m/k})$ walks from each $t \in K$, all of length $l/2$.
4: Denote by S_a the set of nodes seen at most $\tilde{O}(\frac{1}{\epsilon})$ times (small number of times) as endpoints of the n_s walks from s and denote the remaining nodes (seen large number of times) by S_b. Similarly, for each t, partition the nodes into t_a(seen small number of times from t) and t_b(seen large number of times from t). Denote by $\tilde{x}$ and $\tilde{y}$ the distributions of nodes in the end points of the walks from s and t respectively. Thus, $\tilde{x}_i$ is the fraction of walks from s that end up at node i.
5: Let $w(S_b) = \sum_{i \in S_b} \tilde{x}_i$ and $w(t_b) = \sum_{i \in t_b} \tilde{y}_i$. Denote by D_{S_b} the distribution of end points of walks over the nodes in S_b, i.e., the probability of i in D_{S_b} is $\tilde{x}_i / w(S_b)$ if $i \in S_b$ and 0 otherwise. Similarly, denote by D_{t_b} the distribution of end points of walks over the nodes in t_b.
6: For each $t \in K$, set $P_l[st] = \sum_{i \in S_a \cap t_a} \tilde{x}_i \tilde{y}_i + w(S_b) P_{l/2}[D_{S_b} t] + w(t_b) P_{l/2}[s D_{t_b}] - \sum_{i \in S_b \cap t_b} \tilde{x}_i \tilde{y}_i$.
7: In the above expression, $P_{l/2}[D_{S_b} t]$ and $P_{l/2}[s D_{t_b}]$ are estimated recursively for all the $t \in K$. Note that if length is 1, $P_1[st]$ can be computed exactly in one pass.

Proof. $P_l[st] = \sum_{i=1}^{n} x_i y_i$. If all x_i are smaller than $\tilde{O}(\frac{1}{\epsilon\sqrt{km}})$ and y_i is smaller than $\tilde{O}(\frac{1}{\epsilon\sqrt{m/k}})$, then by Lemma 1, the algorithm estimates $\mu = P_l[st]$ within error of $\tilde{O}(\sqrt{\frac{\mu}{\epsilon m}} + \frac{1}{\epsilon m})$. More generally, $P_l[st]$ can be computed as a sum over four sets as $P_l[st] = \sum_{i \in S_a \cap t_a} x_i y_i + \sum_{i \in t_b} x_i y_i + \sum_{i \in S_b} x_i y_i - \sum_{i \in S_b \cap t_b} x_i y_i$; here $i \in S_a$ if x_i is $\tilde{O}(\frac{1}{\epsilon\sqrt{km}})$ and $i \in S_b$ otherwise. $-\sum_{i \in S_b \cap t_b} x_i y_i$ is required as it is counted once in each of $\sum_{i \in t_b} x_i y_i$ and $\sum_{i \in S_b} x_i y_i$. Similarly $j \in t_a$ if y_j is $\tilde{O}(\frac{1}{\epsilon\sqrt{m/k}})$, and $j \in t_b$ otherwise. We will argue that step 6 of the algorithm RECURSIVERANDOMWALK is summing the estimates of each term.

Let $\mu_{aa}, \mu_{*b}, \mu_{b*}, \mu_{bb}$ respectively denote $\sum_{i \in S_a \cap t_a} x_i y_i$, $\sum_{i \in t_b} x_i y_i$, $\sum_{i \in S_b} x_i y_i$ and $\sum_{i \in S_b \cap t_b} x_i y_i$. Note that it is not known which of x_i's or y_i's are small or large. The number of walks performed, however, are sufficient to obtain the right classification to small or large, by standard Chernoff bounds.

Let $\tilde{x}$ and $\tilde{y}$ denote the distributions induced by the end points of the n_s walks and n_t walks respectively. Note that by Lemma 1, $\sum_{i \in S_a \cap t_a} x_i y_i$ can be estimated as $\sum_{i \in S_a \cap t_a} \tilde{x}_i \tilde{y}_i$ with error at most $\tilde{O}(\sqrt{\frac{\mu_{aa}}{\epsilon m}} + \frac{1}{\epsilon m})$. Also note that if $i \in S_b$, then $\tilde{x}_i$ is within $(1 \pm \sqrt{\epsilon})\tilde{x}_i$ w.h.p. from Chernoff bounds. Thus, μ_{bb} can be estimated with error at most $\sqrt{\epsilon}\mu_{bb}$. Again, μ_{*b} can be estimated as $\mu_{*\tilde{b}} = \sum_{i \in t_b} x_i \tilde{y}_i$ where the error $|\mu_{*b} - \mu_{*\tilde{b}}| \leq \sqrt{\epsilon}\mu_{*b}$. Similarly, μ_{b*} can be estimated as $\mu_{\tilde{b}*} = \sum_{i \in S_b} \tilde{x}_i y_i$ with additive error at most $\sqrt{\epsilon}\mu_{b*}$.

Observe that the $\mu_{*\tilde{b}} = \sum_{i \in t_b} x_i \tilde{y}_i = w(S_b) P_{l/2}[D_{S_b} t]$ and $\mu_{b\tilde{*}} = \sum_{i \in S_b} \tilde{x}_i y_i = w(t_b) P_{l/2}[s D_{t_b}]$ are estimated recursively by computing $P_{l/2}[D_{S_b} t]$ and $P_{l/2}[s D_{t_b}]$.

Let $\delta_l(P_l[st])$ denote the error in estimating the probability $P_l[st]$. Then $\delta_l(\mu) = \sqrt{\frac{\mu_{aa}}{\epsilon m}} + \frac{1}{\epsilon m} + \sqrt{\epsilon}\mu_{bb} + \sqrt{\epsilon}\mu_{b*} + \sqrt{\epsilon}\mu_{*b} + w(S_b)\delta_{l/2}(P_{l/2}[D_{S_b}t]) + w(t_b)\delta_{l/2}(P_{l/2}[sD_{t_b}]) \leq \sqrt{\frac{\mu}{\epsilon m}} + \frac{1}{\epsilon m} + 3\sqrt{\epsilon}\mu + w(S_b)\delta_{l/2}(\frac{\mu}{w(S_b)}) + w(t_b)\delta_{l/2}(\frac{\mu}{w(t_b)})$.

The branching factor of the recursion is 2 and has depth $\log l$ with l leaves. Also note that at the leaf, $\delta_1(P_1[st]) = 0$. It follows from standard methods for solving recursion that $\delta_l(\mu) = \tilde{O}(l\sqrt{\epsilon}\mu + l\sqrt{\frac{\mu}{\epsilon m}} + \frac{l}{\epsilon m})$. ∎

We now use the approach in [9] to bound the number of passes and space required in performing RECURSIVERANDOMWALK. The analysis is simple and only requires a careful calculation of the number of walks performed for each length $l/2, l/4, l/8, \ldots$.

Lemma 3. *Algorithm* RECURSIVERANDOMWALK *can be implemented on a graph stream in* $\tilde{O}(\sqrt{\frac{l}{\alpha}})$ *passes and* $\tilde{O}(n\alpha + \sqrt{\frac{mkl}{\alpha}})$ *space for any choice of* $\alpha \leq 1$.

Proof. Notice that in the first phase of RECURSIVERANDOMWALK, $O(\sqrt{mk})$ walks are performed from s of length $l/2$ and $O(\sqrt{m/k})$ walks from each t of length $l/2$. Whenever recursively calculating the distribution, $O(\sqrt{mk})$ walks are required for any source distribution to destination distribution pair. Since the length of the walks halve with every recursive depth, the number of levels is $O(\log l)$ before the length of the walks required becomes a constant. However, the pairs for which the distributions need to be estimated keeps doubling. So after, say, t phases, we perform $O(2^t\sqrt{mk})$ walks of length $l/2^t$.

From [9], the space required for k walks is $\tilde{O}(n\alpha + kl\alpha + k\sqrt{l/\alpha})$. Although [9] assumes that the source distribution was the same for all the k walks, it is easy to extend their result to perform a specific number of walks from different source distributions (to a total of k walks), with only a logarithmic factor increase in the space (by Chernoff bounds).

Summing this over t phases, the first term remains $\tilde{O}(n\alpha)$. The second term of $\tilde{O}(kl\alpha)$ is always $\tilde{O}(\sqrt{mkl}\alpha)$, as only a $\log l$ factor increases. The third term of $k\sqrt{l/\alpha}$ is dominated by the last phase (since this term depends linearly in k but only as square-root in the length of the walks), where $l\sqrt{mk}$ walks of length $O(1)$ are performed. The dominating term therefore is $\tilde{O}(\sqrt{mkl}\alpha)$, completing the proof. ∎

Remark 1. If algorithm RECURSIVERANDOMWALK is to be performed from r different sources, this would increase the space requirement to $\tilde{O}(n\alpha + r\sqrt{\frac{mkl}{\alpha}})$. This follows from the fact that the first term of $\tilde{O}(n\alpha)$ in the space requirement does not depend on the number of walks performed.

Notice that combining Lemmas 2, 3, and choosing $m = \frac{n}{\epsilon^2}$ immediately gives Theorem 1. Observe that by setting $\epsilon = o(\Phi^2/l^2)$ we satisfy the conditions of Theorem 4 and can thus compute candidate cuts.

Algorithm 2 CUTPROJECTIONCANDIDATES(G, K, s)

1: **Input:** Graph G, set K of k randomly sampled nodes, and a source node s.
2: **Output:** k partitions on these k nodes.
3: Estimate probabilities on the k nodes using RECURSIVERANDOMWALK for source s and walk of length l, where l is chosen at random between 1 and $O(1/\phi^2)$.
4: Order the k nodes in decreasing order of the probability estimates. Return the k candidate cuts implied by taking prefixes of this ordering.

Algorithm 3 CUTPROJECTION

1: **Input:** Graph G with cut of conductance at most Φ and source s from the smaller side of this cut; set K of k randomly sampled nodes.
2: **Output:** Partition of these k nodes such that this is a projected cut of conductance is at most $\phi = \tilde{O}(\sqrt{\Phi})$ with constant probability.
3: Sample additional nodes randomly and add it to the set K so that the resulting set K' is of size $k' = \sqrt{\frac{kn}{\phi}}$.
4: Call CUTPROJECTIONCANDIDATES with G', K', s.
5: Consider all of the k' cuts returned by CUTPROJECTIONCANDIDATES that have at least $\frac{k'}{k}$ nodes on either side of the cut. Notice that each of these cuts has at least one of the k nodes in K on either side, with constant probability.
6: For each of these cuts, compute the conductance on the induced subgraph over these k' nodes.
7: Output the cut induced on the k nodes by the cut on the k' nodes that has the minimum conductance in the induced subgraph.

4 Finding sparse cut projections on a small set of nodes

In this section, we prove Theorem 2. Approximate values of $p(i)$ are known from Theorem 1. By setting $\epsilon = \tilde{O}(\frac{\Phi^2}{l^2})$ in theorem 4, we find probability estimates $\tilde{p}(i)$ that satisfy the condition required in theorem 1. Notice that part (b) of Theorem 2 follows directly by using walks of length $l = O(\frac{1}{\Phi})$. If we order all the n vertices by the probability estimates at least one cut has conductance at most $\tilde{O}(\sqrt{\Phi})$. Naturally this ordering induces an ordering on the k vertices that results in k candidate cuts. Note that in our algorithm we need to sample $\tilde{\Omega}(1/b)$ sources so that at least one falls on the smaller side of the optimal cut with high probability. The factor $1/b$ is not applied to the $n\alpha$ term as we can perform all the random walks concurrently as stated in remark 1.

We now prove Theorem 2 part (a). We need to estimate the projected cut conductance for each of the k candidate cuts from the ordering of approximate probabilities. This is done by boosting the number of random nodes from k to k'. It turns out that one can estimate the projected cut conductance value of a specific cut on the k nodes by looking at the conductance on the induced subgraph and cuts on the k' nodes (for an appropriate choice of k'), as stated in Lemma 4. The formal description is in algorithm CUTPROJECTIONCANDIDATES and algorithm CUTPROJECTION. Let $\Phi_{K'}(U, K' \setminus U)$ denote the conductance of the cut $(U, K' \setminus U)$ on the induced subgraph on nodes in K'.

Lemma 4. *For a set K' of randomly chosen nodes with $|K'| = k' \geq \sqrt{\frac{kn}{\phi}}$ and $|U| \geq \frac{k'}{k}$ and $|K' \setminus U| \geq \frac{k'}{k}$, and let $(U, K' \setminus U)$ be a projected cut of conductance of be Ψ. Then $\Phi_{K'}(U, K' \setminus U)$ gives a constant factor approximation to Ψ with high probability.*

Proof (sketch). The essential idea is that considering the conductance of the induced cut on $\sqrt{\frac{kn}{\phi}}$ nodes, is identical to estimating $\sum_{(i,j)\in E} n a_{ij} x_i y_j / (|X||Y|)$ by sampling each x_i with probability $k'|X|/n$ and each y_i with probability $k'|Y|/n$ and a_{ij} is $\frac{1}{d}$ for an edge (i, j). The proof is then completed using Lemma 1. A more detailed exposition is presented in the full version of the paper. ∎

This lemma automatically gives a method for estimating the projected cut conductance for a partition of a subset of k nodes. We are now ready to prove the main theorem 2 part (b).

Proof (Proof of Theorem 2(b)). By setting $\epsilon = \tilde{O}(\frac{\Phi^2}{l^2})$ in theorem 4, we find probability estimates $\tilde{p}(i)$ that satisfy the condition required in theorem 1. So if we order all the n vertices by the probability estimates at least one cut has conductance at most $\tilde{O}(\sqrt{\Phi})$. Naturally this ordering induces an ordering on the k' vertices that contain the set K. We are simply estimating the conductance of all the cuts in this ordering that has at least one vertex from K in the smaller side. If none of these cuts give the required conductance of $\tilde{O}(\sqrt{\Phi})$, then all k nodes are put on the same side of the cut and output. Note that in our algorithm we need to sample $\tilde{\Omega}(1/b)$ sources so that at least one falls on the smaller side of the optimal cut with high probability. This gives $\sqrt{\frac{l}{\Phi\alpha}}$ passes and $\tilde{O}(n\alpha + \frac{1}{b}\frac{1}{\epsilon}\sqrt{\frac{nk'}{\phi\alpha}}) = \tilde{O}(n\alpha + \frac{n^{3/4}k^{1/4}}{b\sqrt{\alpha}\Phi^{19/4}})$ space. The factor $1/b$ is not applied to the $n\alpha$ term as the we can perform all the random walks concurrently as stated in remark 1. ∎

5 Conclusions

In this work, we present an approach for finding cuts that approximate the conductance of a graph presented as a stream of edges. In particular, we show that this problem can be solved more efficiently if we are only required to partition a small set of k random nodes with respect to a sparse cut. The streaming algorithms we present require space that is sublinear in the number of nodes for a certain range of parameters.

Acknowledgments: We thank D. Sivakumar, S. Vempala, and K. Munagala for comments on the paper.

References

1. Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In *ICALP*, 2009.
2. Noga Alon. Eigenvalues and expanders. In *Combinatorica*, pages 6(2) 83–96, 1986.

3. Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Local Graph Partitioning using PageRank Vectors. In *FOCS*, pages 475–486, 2006.
4. Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *STOC*, pages 222–231, 2004.
5. András A. Benczúr and David R. Karger. Approximating *s-t* Minimum Cuts in $\tilde{O}(n^2)$ Time. In *STOC*, pages 47–55, 1996.
6. Sandeep N. Bhatt and Frank Thomson Leighton. A Framework for Solving VLSI Graph Layout Problems. *J. Comput. Syst. Sci.*, 28(2):300–343, 1984.
7. Ravi Boppana. Eigenvalues and graph bisection: an average case analysis. In *28th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1987.
8. Christian Borgs, Jennifer T. Chayes, Mohammad Mahdian, and Amin Saberi. Exploring the community structure of newsgroups. In *KDD*, pages 783–787, 2004.
9. Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. Estimating PageRank on graph streams. In *PODS*, pages 69–78, 2008.
10. C. Demetrescu, I. Finocchi, and A. Ribichini. Trading of space for passes in graph streaming problems. In *ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 714–723, 2006.
11. Oded Goldreich and Dana Ron. Property Testing in Bounded Degree Graphs. In *STOC*, pages 406–415, 1997.
12. M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. In *External Memory Algorithms, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 50, pages 107–118, 1999.
13. David R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, 2000.
14. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In *SODA*, pages 668–677, 1998.
15. Ronny Lempel and Shlomo Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, 2001.
16. Jure Leskovec, Susan Dumais, and Eric Horvitz. Web projections: learning from contextual subgraphs of the web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 471–480, New York, NY, USA, 2007. ACM.
17. Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *KDD*, pages 631–636, 2006.
18. László Lovász and Miklós Simonovits. The Mixing Rate of Markov Chains, an Isoperimetric Inequality, and Computing the Volume. In *FOCS*, pages 346–354, 1990.
19. László Lovász and Miklós Simonovits. Random Walks in a Convex Body and an Improved Volume Algorithm. *Random Struct. Algorithms*, 4(4):359–412, 1993.
20. Rajsekar Manokaran, Joseph Naor, Prasad Raghavendra, and Roy Schwartz. SDP gaps and UGC hardness for multiway cut, 0-extension, and metric labeling. In *STOC*, pages 11–20, 2008.
21. Alistair Sinclair and Mark Jerrum. Conductance and the mixing property of markov chains; the approximation of the permanent resolved. In *Proc. of the 20th annual ACM Symposium on Theory of computing*, pages 235–244, 1988.
22. Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90, 2004.