Distributed Sparse Cut Approximation*

Fabian Kuhn[1] and Anisur Rahaman Molla[2]

1 **Department of Computer Science, University of Freiburg, Freiburg, Germany**
kuhn@cs.uni-freiburg.de
2 **Department of Computer Science, University of Freiburg, Freiburg, Germany**
armolla@cs.uni-freiburg.de

Abstract

We study the problem of computing a sparse cut in an undirected network graph $G = (V, E)$. We measure the sparsity of a cut $(S, V \setminus S)$ by its conductance $\phi(S)$, i.e., by the ratio of the number of edges crossing the cut and the sum of the degrees on the smaller of the two sides. We present an efficient distributed algorithm to compute a cut of low conductance. Specifically, given two parameters b and ϕ, if there exists a cut of balance at least b and conductance at most ϕ, our algorithm outputs a cut of balance at least $b/2$ and conductance at most $\tilde{O}(\sqrt{\phi})$, where $\tilde{O}(\cdot)$ hides polylogarithmic factors in the number of nodes n. Our distributed algorithm works in the CONGEST model, i.e., it only requires to send messages of size at most $O(\log(n))$ bits. The time complexity of the algorithm is $\tilde{O}(D + 1/b\phi)$, where D is the diameter of G. This is a significant improvement over a result by Das Sarma et al. [ICDCN 2015], where it is shown that a cut of the same quality can be computed in time $\tilde{O}(n + 1/b\phi)$. The improved running time is in particular achieved by devising and applying an efficient distributed algorithm for the all-prefix-sums problem in a distributed search tree. This algorithm, which is based on the classic parallel all-prefix-sums algorithm, might be of independent interest.

1998 ACM Subject Classification F.2.2 [Analysis of Algorithms and Problem Complexity] Nonnumerical Algorithms and Problems, G.2.2 [Discrete Mathematics] Graph Theory

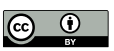
Keywords and phrases sparsest cut, conductance, random walks, all-prefix-sums

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2015.10

1 Introduction and Related Work

The problem of finding sparse cuts in a graph is one of the basic problems in network optimization. In the context of the present paper, the sparsity of a cut is measured by its *conductance*, where the conductance of a cut is defined as the ratio between the number of edges crossing the cut and the sum of the degrees on the smaller side of the cut. In this context, the sum of the degrees of a set of nodes S is also known as the *volume* of S. The conductance of a graph is defined as the smallest conductance of any of its cuts. The conductance determines how well connected a graph is and in particular how well information can be spread within the graph. It is well-known that the conductance of a graph is closely connected to the *mixing time* of a random walk on the graph and consequently also to the *spectral gap* of the graph [14, 20]. Network with high conductance, a large spectral gap and thus a small random walk mixing time for example allows to do fast (almost) uniform random sampling of nodes (see [11] for fast distributed algorithms and applications) or to do low-congestion routing [7, 13]. It is also known that the performance of the random push-pull gossip protocol is very closely related to the conductance of the network [12].

* Research supported by ERC Grant No. 336495 (ACDC).



As the conductance of a graph is tightly connected to the performance of many important random processes and computations in networks, finding cuts of low conductance potentially helps to find lower bounds on the performances of these processes and computations. As it is the sparse cuts which are limiting the speed of many such processes, finding low conductance cuts can help in identifying critical, important edges and bottlenecks in a given network. In a standard centralized setting, the problem of finding sparse cuts and also more generally related graph partitioning problems are well studied with a large body of literature, see e.g., [1, 2, 3, 4, 5, 15, 18, 19, 21, 24]. The first approximation algorithm the conductance of a graph was presented by Leighton and Rao in [18], where an $O(\log n)$-approximation is given. Using semidefinite programming techniques, this was improved to the currently best known approximation ratio of $O(\sqrt{\log n})$ Arora, Rao, and Vazirani [4] (see also [3]).

In a large-scale network, it might not be possible or reasonable to collect the entire structure of the network at a single node and to perform computations in a centralized way. In the present paper, we thus study the problem of finding cuts of low conductance in a distributed manner. Our distributed algorithm is based on random walk techniques that were first developed by Lovász and Simonovits in [19, 20] and later extended by Spielman and Teng in [24, 25]. More directly, our distributed algorithm is based on an algorithm for the same problem in the streaming model [9] and on a simple distributed version of this algorithm which appeared in [10].

The core idea of the algorithm is to test different cuts obtained by the probability distributions of random walks in the graph. More specifically, consider a random walk of some length ℓ starting at a node s in a graph G and order all nodes of G according to a normalized probability of finishing the random walk at the node. Consider all $n-1$ cuts that are defined by all the $n-1$ prefixes of this ordering. Assume that there is a set of nodes S of volume at most half the total volume of G such that the $(S, \bar{S})$ of G has conductance at least ϕ. It was shown in [19, 20, 24, 25] that when doing a random walk of length chosen randomly between 1 and $O(1/\phi)$ starting at a random node $s \in S$, with constant probability one of the induced $n-1$ cuts has conductance at most $\tilde{O}(\sqrt{\phi})$. In [9], it is observed that the technique still works if the random walk probabilities are only approximately computed and the technique is applied to find low conductance cuts in the streaming model. Based on the streaming algorithm of [9], a simple distributed algorithm to solve the same problem in the CONGEST model (i.e., in a message passing model with messages of logarithmic size [23]) has been presented in [10].

The algorithm of [10] is a straightforward implementation of the ideas of [19, 20, 24, 25] in that for each of the random walks it computes, the complete information to compute the sizes of all $n-1$ induced cuts is sent to a global leader (for each node, one needs to know the number of edges to predecessors/successors in the order given by the random walk probabilities). As a result, for each random walk, the algorithm of [10] requires $O(n)$ rounds to compute the sizes of all cuts induced by the computation of one random walk in G. This results in an overall running time of $\tilde{O}\big(\frac{1}{b}\big(\frac{1}{\phi}+n\big)\big)$ to find a cut of conductance at most $\tilde{O}(\sqrt{\phi})$ and balance at least $\Omega(b)$, where the balance $b \leq 1/2$ of a cut is defined as the ratio of the volume of the smaller side and the total volume of the graph. In this paper we improve on this in the following way. Because it is sufficient to have approximate random walk probabilities, we can round the computed probabilities so that we can partition them $\tilde{O}(1/\phi)$ classes of equal normalized probabilities. Within each class, the nodes can then be ordered arbitrarily. Using a given spanning tree of the network to define the order within each class, we then show that the sizes of all cuts can be computed by doing two appropriate all-prefix-sums computations for each class. We show that this can be done efficiently by developing a

distributed variant of the classic parallel all-prefix-sums algorithm [16, 17, 22, 26]. As a result, we obtain an improved running time of $\tilde{O}(D + 1/b\phi)$ for computing a cut of balance $\Omega(b)$ and conductance $\tilde{O}(\sqrt{\phi})$, where D is the diameter of G.

The remainder of the paper is organized as follows. In Section 2, we formally define the communication model and the problem statement. We then formally state the contributions of the paper in Section 3. Sections 4 and 5 are devoted to our technical results, where in Section 4, we first describe the distributed all-prefix-sums algorithm which will then be applied in Section 5, where our main result, an algorithm to compute low conductance cuts will be presented.

2 Model and Definitions

Distributed Computing Model. The network is modeled as an undirected n-node graph $G = (V, E)$. For simplicity, we assume that the graph G is unweighted. We however note that it is not hard to generalize the presented sparse cut approximation algorithm to weighted graphs.[1] We model communication by using the standard CONGEST model: Communication happens in synchronized rounds; in every round, every node is allowed to send a (possibly different) message of at most $O(\log(n))$ bits to each of its neighbors [23]. The *time complexity* of an algorithm is defined as the total number of rounds needed until all nodes terminate. Note that we use the common assumption that local computations at the nodes are for free. We however point out that our algorithms only require very simple, efficient local computations. We further assume that each node is equipped with a unique identifier (ID). Initially, each node knows its own ID as well as the IDs of all its neighbors. Wherever convenient, we slightly abuse notation and identify a node v with its ID.

Random Walks. Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = m$. We use $p_\ell(s, t)$ to denote the probability that a uniform lazy random walk of length ℓ starting at node $s \in V$ ends at node $t \in V$.[2] Hence, $\{p_\ell(s, t) : t \in V\}$ is the probability distribution on nodes induced by a random walk of length ℓ starting from the source node s. For convenience, we abbreviate $p_\ell(s, t)$ by $p(t)$ when source node and length are clear from the text. Let $d(v)$ denote the degree of a node $v \in V$. Given a probability distribution $p(\cdot)$ on the nodes and a node $v \in V$, we further define $\rho_p(v) := p(v)/d(v)$ as the normalized probability of v. Note that the stationary $\pi(\cdot)$ distribution of the uniform lazy random walk is given by $\pi(v) = d(v)/m$ for all $v \in V$.

Cuts and Conductance. Let S be a subset of the nodes V and let $\bar{S} := V \setminus S$. The bipartition $(S, \bar{S})$ of the nodes is called the *cut* induced by S (or also by $\bar{S}$). We use $E(S, \bar{S})$ to denote the set of edges across the cut $(S, \bar{S})$ and $e(S, \bar{S}) := |E(S, \bar{S})|$ for the size of the cut. We measure the *sparsity* of a cut by its *conductance*, where the conductance is defined as follows.

[1] In a weighted graph, we need to substitute the random walk transition matrix with the weighted transition matrix. Also volume and conductance we have to be defined in the natural way for weighted graphs.

[2] In a uniform lazy random walk, in each step, the walk stays at the current node with probability 1/2 and otherwise it moves to a uniformly random neighbor.

▶ **Definition 1** (Conductance)**.** Given a graph $G = (V, E)$, the conductance $\phi(S)$ of the cut $(S, \bar{S})$ induced by a set $S \subseteq V$ is defined as

$$\phi(S) := \frac{e(S, \bar{S})}{\min\{\text{VOL}(S), \text{VOL}(\bar{S})\}},$$

where $\text{VOL}(S) := \sum_{v \in S} d(v)$ is the *volume* of the node set S. Note that clearly $\phi(\bar{S}) = \phi(S)$. The conductance of the graph G is defined as

$$\phi(G) := \min_{S \subseteq V} \phi(S).$$

A *sparsest cut* of G is a cut $(S, \bar{S})$ with conductance $\phi(S) = \phi(G)$. The performance of our algorithm also depends on the *balance* of the cut it computes.

▶ **Definition 2** (Balance)**.** The balance of a cut $(S, \bar{S})$ is denoted by $b(S) = b(\bar{S})$ and it is defined as

$$b(S) := \frac{\min\{\text{VOL}(S), \text{VOL}(\bar{S})\}}{\text{VOL}(V)} = \frac{\min\{\text{VOL}(S), \text{VOL}(\bar{S})\}}{2m}.$$

3 Contributions

We develop approximation algorithm for computing sparse cuts in distributed networks. Given two constants b and ϕ, our algorithm outputs a cut of balance at least $b/2$ and conductance at most $\tilde{O}(\sqrt{\phi})$, provided that there is a set $C \subseteq V$ such that $\phi(C) \leq \phi$ and $b(C) \geq b$. Formally, we prove the following main theorem.

▶ **Theorem 3.** *Given a network graph $G = (V, E)$ and two parameters $b \leq 1/2$ and $\phi < 1$ such that there exits a set $C \subseteq V$ with $b \cdot 2|E| \leq \text{VOL}(C) \leq |E|$ and $\phi(C) \leq \phi$. Then there is a distributed algorithm that finds a cut $(S, \bar{S})$ which satisfies $b|E| \leq \text{VOL}(S) \leq |E|$ and $\phi(S) = O\big(\sqrt{\phi \log n}\big)$ with high probability and finishes in $O\big(D + \frac{\log^2 n}{b\phi}\big)$ rounds in the CONGEST model, where D is the diameter of G.*

Often, we are most interested in computing an approximation of the *sparsest cut* of the graph G. Assume that there is a sparsest cut (i.e., a cut with conductance $\phi(G)$) with balance b. If $\phi(G)$ and b are known, the above theorem then guarantees to find a cut with conductance $O\big(\sqrt{\phi(G)\log(n)}\big)$ and balance at least $b/2$ in time $O\big(\log^2(n)/(b\phi(G))\big)$ (note that we always have $D = O\big(\log(n)/\phi(G)\big)$). Note that the diameter D clearly is a lower bound on computing any approximation of the sparsest cut. Further, there are graphs with diameter $D = \Theta(\log(n)/\phi(G))$ [8]. Hence, if the sparsest cut has constant balance, this above result is optimal up to a factor $O(\log(n))$ in some graphs. We further mention that the lower bound $\tilde{\Omega}(\sqrt{n} + D)$ for computing sparsest cut shown in [10], only works for weighted graphs. Moreover, the graph they considered to claim the lower bound has very small conductance.

Our algorithm can also be extended to compute the sparsest cut without knowing the conductance value $\phi(G)$. The running time then increases to $\tilde{O}\big(\frac{\tau}{b}\big)$, where τ is the mixing time of the lazy random walk on G. This follows because one can easily estimate τ of G in time $\tilde{O}(\tau)$ and thanks to the relation $\Theta\big(\frac{1}{\phi(G)}\big) \leq \tau \leq \Theta\big(\frac{\log(n)}{\phi(G)^2}\big)$ [14].

In order to obtain the time complexity proven in Theorem 3, we develop a result on the distributed computation of all-prefix-sums which might be of independent interest. Essentially, we show that if the ordering of the nodes can be chosen based on the topology of the network an all-prefix-sums instance where each node has some input can be computed in time $O(D)$. Further, we also show that K independent such all-prefix-sums instances can be evaluated in time $O(D + K)$. For a formal problem statement and the formal results, we refer to Section 4.

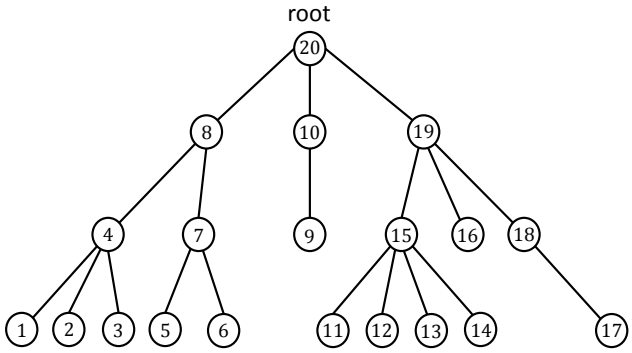


Figure 1 A rooted search tree where the position (in the total order) of each node is greater than the positions of all children and thus the positions of all nodes in its subtrees. The sub-trees are drawn such that they are ordered from left to right.

4 Distributed Prefix Sums Computation

The all-prefix-sums problem takes an ordered tuple of n elements or values $(a_1, a_2, \ldots, a_n)$ and outputs the sums of all prefixes $(a_1, a_1 \oplus a_2, \ldots, a_1 \oplus a_2 \oplus \ldots \oplus a_n)$ with respect to some binary associative (addition) operation $\oplus$. While computing all prefix sums in (optimal) linear time is a trivial task for ordinary sequential algorithms, the problem is more interesting in a parallel or distributed setting. It is a well-known result that the problem can be solved using logarithmic depth and linear work parallel algorithm in all standard parallel computing models, e.g., on an EREW PRAM [16, 17, 22, 26]. As a result the all-prefix-sums computation is used as a basic building block for many classic parallel algorithms [6].

In the present paper, we adapt the classic parallel all-prefix-sums algorithm to a distributed algorithm in the CONGEST model. As the communication network, we assume that we are given an arbitrary N-node rooted search tree $T = (V_T, E_T)$ with root node $v_r \in V_T$ and radius R (w.r.t. v_r). The search order is defined as follows. At each node $v \in V_T$ with children $u_1, \ldots, u_c$, we are locally given a total order $\prec_v$ on the nodes $\{v, u_1, \ldots, u_c\}$. The overall total order $\prec$ on V_T is then given by combining the local orders $\prec_v$ and by extending the resulting partial order as follows. Given a node u and its parent p, if $u \prec_p p$ (and thus $u \prec p$), we have $w \prec p$ for all nodes w in the subtree of u. Similarly, if $p \prec_p u$ (and thus $p \prec u$), we have $p \prec w$ for all nodes w in the subtree of u. For the remainder of the section, for convenience, we name the nodes in V_T by $v_1, \ldots, v_N$ such that $v_1 \prec v_2 \prec \cdots \prec v_N$. We assume that each node $v \in V_T$ initially knows its parent, all its children, as well as the local order $\prec_v$. For an example of a search tree, see Figure 1.

Each node $v \in V_T$ is given an input value a_v, where a_v is from the domain on which the prefix-sums operation $\oplus$ is defined. We assume that each of the values a_v and also each sum (w.r.t. operation $\oplus$) of a subset of the N input values can be represented using $O(\log N)$ bits. For every $k \in \{1, \ldots, N\}$ and thus for every $v_k \in V_T$, the corresponding k^{th} prefix sum s_{v_k} is defined by $s_{v_1} := a_{v_1}$ and $s_{v_k} := s_{v_{k-1}} \oplus a_{v_k}$ for $k > 1$. In a distributed all-prefix-sums algorithm, each node $v_i \in V_T$ needs to compute the corresponding prefix sum s_{v_i}.

We next present a distributed algorithm to solve the all-prefix-sums problem in T in time $O(R)$ (where R is the radius or depth of T) in the CONGEST model. Note that this time complexity is clearly asymptotically optimal since for every pair of nodes $u, v \in V_T$ with $u \prec v$, the prefix sum of v depends on the value of u and thus the diameter of T is a trivial lower bound on the time complexity to compute all prefix sums.

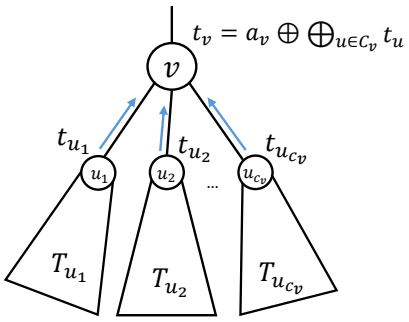


Figure 2 Bottom-up computation by each node v.

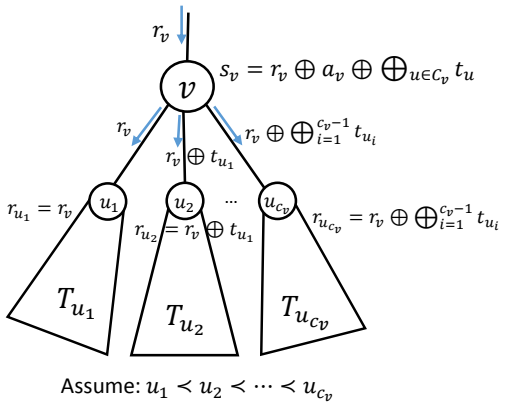


Figure 3 Top-down computation by each node v.

The distributed all-prefix-sums algorithm is an adaptation of the classic parallel algorithm [16, 17, 22, 26] to arbitrary search trees (the classic algorithm builds up a binary search tree). The algorithm consists of two phases. First, there is a bottom-up (convergecast) phase starting from the leaves to compute the sum of the values in each subtree of T. The second phase is a top-down phase in which the prefix sums at all nodes are computed level by level.

Bottom-up Phase (Figure 2). For a node $v \in T$, let T_v be the subtree of T rooted at node v and let $V(T_v)$ be the set of nodes of T_v (including v). Further, for a node v, let C_v be the set of its children and let $c_v := |C_v|$ be the number of its children. We define $t_v := \bigoplus_{u \in T_v} a_u$ to be the sum of all the values in T_v. In the bottom-up phase, each node v recursively computes the value t_v in the obvious way using a convergecast from the leaves to the root, i.e., $t_v = a_v \oplus \bigoplus_{u \in C_v} t_u$.

Top-down Phase (Figure 3). Once the root node v_r has computed t_{v_r}, it initiates the top-down phase. In the top-down phase, each node v computes a value r_v which is defined as follows

$$r_v := \bigoplus_{u \in V_T \setminus V(T_v) : u \prec v} a_u,$$

i.e., r_v is the sum of all input values smaller than v which are not in the subtree of v. After the bottom-up phase, each node $v \in V_T$ knows the value t_v, as well as the values t_u for all children u of v. First note that once a node v also knows r_v, it is straightforward to compute s_v as

$$s_v = r_v \oplus a_v \oplus \bigoplus_{u \in C_v : u \prec v} t_u$$

We can therefore concentrate on the computation of r_v at each node v. For the root node v_r, we clearly have $r_{v_r} = 0$ (here, 0 is assumed to be the neutral element of the operation $\oplus$). For all other nodes $v \in V_T$, let p_v be the parent node of v. As shown by the following lemma, the value of r_v can be computed from r_{p_v}, a_{p_v}, and the values t_u of the children of p_v and of v.

▶ **Lemma 4.** *Let v be a non-root node of T and let p_v be the parent node of v. It holds that*

$$r_v = \begin{cases} r_{p_v} \oplus \bigoplus_{u \in C_{p_v} : u \prec v} t_u & \text{if } v \prec p_v, \\ r_{p_v} \oplus a_{p_v} \oplus \bigoplus_{u \in C_{p_v} : u \prec v} t_u & \text{if } p_v \prec v. \end{cases} \tag{1}$$

Proof. First consider three nodes $u, v, w \in V_T$ such that v is in the subtree of u, but w is not in the subtree of u. Observe that because T is a search tree, it holds that $v \prec w$ if and only if $u \prec w$. For all nodes w which are not in the subtree of p_v it therefore holds the $w \prec p_v$ if and only if $w \prec v$ (and certainly w is also not in the subtree of v). All input values a_w which contribute to r_{p_v} therefore also contribute to r_v and these are the only input values which contribute to r_v and which are not in the subtree of p_v. The value of r_v can therefore be computed by summing r_{p_v} with all values a_x for which $x \prec v$ and where x is in the subtree of p_v but not in the subtree of v. These are exactly the values of all nodes in the subtrees of children u of p_v for which $u \prec v$ and it also includes the value a_{p_v} of p_v if $p_v \prec v$. This proves the lemma. ◀

The following theorem puts everything together and it also shows that it is possible to efficiently solve several concurrent instances of the all-prefix-sums problem.

▶ **Theorem 5.** *Assume that we are given an N-node search tree T and $K \geq 1$ instances of the all-prefix-sums problem on T. That is, each node v has K inputs $a_{v,1}, \ldots, a_{v,K}$ and it needs to compute k output values $s_{v,1}, \ldots, s_{v,k}$ such that for all $v \in V_T$ and all $i \in \{1, \ldots, K\}$, $s_{v,i} = a_{v,i} \oplus \bigoplus_{u \in V_T : u \prec v} a_{u,i}$. In the CONGEST model, the K concurrent all-prefix-sums instances can be computed in time $O(R + K)$, where R is the radius of T.*

Proof. For $K = 1$, the claimed time complexity follows in a straightforward way from the above algorithm description. Both the bottom-up and the top-down phase can clearly be implemented in R rounds. In the bottom-up convergecast, each node v only needs to report t_v to its parent once it knows t_u of all children u. In the top-down phase, as soon as a node u knows r_u, it sends a_u and $\bigoplus_{w \in C_u : w \prec v} t_w$ to each of its children $v \in C_u$. Note that by the assumptions we made, all the messages have a size of at most $O(\log N)$ bits.

For an integer $i \geq 0$, let L_i be the set of nodes at distance exactly i from the root node v_r in T. We call the nodes in L_i the level-i nodes. Note that both the bottom-up and the top-down phase can be implemented such that in each communication round, only the nodes on one level L_i are active. The K concurrent all-prefix-sums instances can therefore be solved in time $O(R + K)$ by using pipelining. ◀

To conclude the section, we adapt the above result to somewhat more general case that we will use for our sparse cut algorithm. As above, assume that as a network, we are given an N-node rooted search tree $T = (V, E)$. However, we will now assume that for the all-prefix-sums problem, the nodes are only partially ordered according to the global order $\prec$ induced by T. Consider a second global order $\sqsubset$ which is defined as follows. We assume that the nodes V are partitioned into K classes $\mathcal{C}_1, \ldots, \mathcal{C}_K$. The order $\sqsubset$ is then defined as the lexicographic order define by the class number and the search order $\prec$ of T. That is, for any $i, j \in \{1, \ldots, K\}$ and any $u \in \mathcal{C}_i$ and $v \in \mathcal{C}_j$, we have $u \sqsubset v$ if and only if either $i < j$ or $i = j$ and $u \prec v$. Assume that each node $v \in V$ has an input value b_v from the domain for which the associative operator $\oplus$ is defined. We define the prefix sum of node v as $\sigma_v := b_v \oplus \bigoplus_{u \in V : u \sqsubset v} b_u$. The following theorem shows that as long as the number of classes $\mathcal{C}_i$ is not too large, the corresponding all-prefix-sums problem can be computed efficiently in the CONGEST model.

▶ **Theorem 6.** *Let $T = (V, E)$ be a rooted search tree and let $\sqsubset$ be a global order on V defined by a partition $\mathcal{C}_1, \dots, \mathcal{C}_K$ as defined above. Further, assume that every node $v \in V$ has an input value b_v. Then, the prefix sums $\sigma_v = b_v \oplus \bigoplus_{u \sqsubset v} b_u$ for all nodes $v \in V$ can be computed in time $O(R + K)$ in the CONGEST model, where R is the radius of T.*

Proof. For each of the node classes $\mathcal{C}_i$, we first define $S_i := \bigoplus_{v \in \mathcal{C}_i} b_v$ to be the sum of all inputs of nodes in $\mathcal{C}_i$. By doing a standard convergecast on T, for each i, S_i can be computed in R rounds. Also, by using pipelining, all K values $S_1, \dots, S_K$ can be computed in time $R + K$. Using another $R + K$ rounds, we can also make sure that all nodes know all the values $S_1, \dots, S_K$.

Let us now concentrate on a single node class $\mathcal{C}_i$. For each node $v \in \mathcal{C}_i$, we define the local prefix sum $\bar{\sigma}_v$ as $\bar{\sigma_v} := b_v \oplus \bigoplus_{u \in \mathcal{C}_i : u \sqsubset v} b_u$. Note that within a single node class, the two global orders $\sqsubset$ and $\prec$ are identical. Hence, by defining the input to be 0 for all nodes outside $\mathcal{C}_i$, the local prefix sums $\bar{\sigma}_v$ for all nodes $v \in \mathcal{C}_i$ can be computed in time $O(R)$ by using Theorem 5. Also note that computing the local prefix sums for the K different node classes corresponds to K independent all-prefix-sums computations on T and by Theorem 5, it can therefore be done in time $O(R + K)$. Once every node know all values $S_1, \dots, S_K$, as well as its local prefix sum $\bar{\sigma}_v$, it can locally compute its prefix sum σ_v as follows:

$$\forall i \in \{1, \dots, K\} : \forall v \in \mathcal{C}_i : \sigma_v = \bar{\sigma}_v \oplus \bigoplus_{j<i} S_j.$$

This concludes the proof. ◀

5 Algorithm for Sparse Cut

In this section, we present our main result, a distributed algorithm to compute a cut of low conductance. More specifically, we are given an undirected network graph G, a target conductance ϕ, and a balance b as inputs. If there exists a cut $(S, \bar{S})$ with balance at least b and conductance at most ϕ, our distributed algorithm finds a cut $(S', \bar{S}')$ with conductance at most $\tilde{O}\left(\sqrt{\phi}\right)$ and balance at least $b/2$. At the end of the algorithm, every node in G knows whether it is in S' or in $\bar{S}'$. Note that throughout the section, we assume that the number of nodes n is sufficiently large. We can do this w.l.o.g., as if n is a constant, we can always collect the whole graph and compute all cuts in constant time. Throughout the section, we also assume that we have a BFS rooted tree T of the network graph G available. Note that such a tree has depth at most D (where D is the diameter of G) and it can be computed in $O(D)$ rounds in the CONGEST model. We further assume that v_r is the root node of T.

Our algorithm is based on computing probabilities of random walks of multiple lengths ℓ and for multiple sources s. The probabilities of each such random walk define a global order on the nodes V. For the following discussion, we specify a global order on V by a bijection $\pi : V \to \mathbb{N}$ between V and $\{1, \dots, n\}$. That is, a node u appears before v according to the global order π if and only if $\pi(u) < \pi(v)$. Given a global order π on V and an integer $i \in \{1, \dots, n-1\}$ we define the node set $S_\pi(i) := \{v \in V : \pi(v) \leq i\}$. Further, as defined in Section 2, let $p_\ell(s, v)$ be the probability to reach $v \in V$ after exactly ℓ steps of a lazy random walk started at node $s \in V$. Further, recall that for a probability distribution $p(v)$ on the nodes $v \in V$, we use $\rho_p(v) := p(v)/d(v)$ to denote the normalized probability of node v. Our distributed low conductance cut algorithm uses conductance approximation techniques developed by Lovász and Simonovits [19, 20] and by Spielman and Teng [24, 25]. Formally, we apply the following lemma which is a relatively simple application of the results of [19, 20, 24, 25] and which was formally proven in [9].

▶ **Lemma 7** ([9]). *Let $G = (V, E)$ be a graph and let $(S, \bar{S})$ be a cut of G of conductance at most ϕ such that $\text{VOL}(S) \leq \text{VOL}(V)/2$. Further, let $s \in S$ be a node sampled randomly from the degree distribution in S and let ℓ be an integer chosen uniformly at random from $\{1, \ldots, 1/8\phi\}$. We define $p(v) := p_\ell(s, v)$ and we assume that for all $v \in V$, $\tilde{p}(v)$ is an estimate for the probability $p(v)$ such that $|\tilde{p}(v) - p(v)| \leq \frac{\epsilon}{2}(p(v) + 1/n)$, where $\epsilon < \phi$. Let $\pi : V \to \mathbb{N}$ be any global order on V such that $\pi(u) < \pi(v)$ whenever $\rho_{\tilde{p}}(u) > \rho_{\tilde{p}}(v)$. Then with constant probability for some set $S_\pi(i)$ for $i \in \{1, \ldots, n-1\}$, we have $\phi(S_\pi(i)) \leq 8\sqrt{\phi \log(n)}$ and $b(S_\pi(i)) \geq b(S)/2$.*

Based on Lemma 7, the strategy for computing a cut of low conductance is as follows. Assume that we are given a network graph $G = (V, E)$ and two parameters $2/n^2 \leq \phi < 1$ and $b \leq 1/2$. For a sufficiently large constant[3] $c > 0$, we define a parameter $Q = \frac{c \cdot \ln n}{b}$. We randomly (independently) select Q nodes $s_1, \ldots, s_Q$ and Q lengths $\ell_1, \ldots, \ell_Q$, where each node s_i is chosen according to the degree distribution of G and each length ℓ_i is chosen uniformly from the range $\{1, \ldots, 1/8\phi\}$. For each $i \in \{1, \ldots, Q\}$, the approximate random walk probabilities $\tilde{p}(v)$ for a walk of length ℓ_i starting at s_i are computed. It then follows directly from Lemma 7 that if c is chosen sufficiently large and if the graph G has a cut $(S, \bar{S})$ with $\phi(S) \leq \phi$ and $b(S) \geq b$, with high probability, for at least one of the Q random walks, one of the computed $n - 1$ cuts has the desired balance and conductance.

The core of our distributed sparse cut algorithm therefore is to compute approximate random walk probabilities for a given starting node s and a given length ℓ and to compute the conductances and balances of the $n - 1$ cuts induced by these approximate random walk probabilities. Theorem 3 will then follow by repeating this $O(\log(n)/b)$ times. In the following, we therefore assume that we have a fixed start node $s \in V$ and a fixed random walk length $\ell \leq 1/8\phi$. We will first show how to compute approximate probabilities $\tilde{p}(v) \approx p_\ell(s, v)$. As a second step, we will show that the properties of these probabilities $\tilde{p}(v)$ allow to use the all-prefix-sums result of Section 4 to quickly compute the balances and conductances of the induced cuts.

5.1 Computing the Random Walk Probabilities

We estimate the probability distribution of a random walk starting from a starting node $s \in V$. Recall that we perform a lazy random walk, i.e., in each step, the walk stays at the current node with probability 1/2. The probability of $p_t(s, v)$ of being at node v after t steps of the random walk can be stated recursively as follows. For $t = 0$, $p_0(s, s) = 1$ and $p_0(s, v) = 0$ for any $v \neq s$. For $t > 0$, we have

$$p_t(s, v) = \frac{1}{2} \cdot p_{t-1}(s, v) + \sum_{u \in N(v)} \frac{p_{t-1}(s, u)}{2d(u)}. \tag{2}$$

Hence, given $p_{t-1}(s, v)$ for all nodes v, in principle, it is possible to exactly compute $p_t(s, v)$ for all nodes v in a single communication round. Note however that the probabilities $p_t(s, v)$ are real values and since in the CONGEST model we are restricted to using at most $O(\log(n))$ bits per message, we need to be a little bit more careful. In the following, assume that for each time t, each node v maintains an approximation $\beta_t(v)$ of $p_t(s, v)$ and let $\boldsymbol{\beta_t}$ be the n-vector of all these approximations. We define $\delta_t(v) := \beta_t(v) - p_t(s, v)$ to be the error of v's approximation

[3] The constant only helps to measure the high probability bound of the result; larger the constant value means higher the probability guarantee.

of $p_t(s,v)$ and we use $\boldsymbol{\delta_t}$ to denote the vector of all errors after step t of the random walk. We assume that the approximations $\beta_t(v)$ are computed as follows. Node v collects $\beta_{t-1}(u)$ from all neighbors u, it evaluates $\beta'_t(v) := \beta_{t-1}(v)/2 + \sum_{u\in N(v)} \beta_{t-1}(u)/2d(u)$, and it then computes $\beta_t(v)$ as $\beta'_t(v)$ rounded to the closest integer multiple of n^8. Note that because $\beta_t(v)$ is always between 0 and 1, there are at most n^8+1 different values for $\beta_t(v)$ and therefore all messages can clearly be encoded using $O(\log(n))$ bits. The following lemma shows that also after ℓ steps, the absolute error $|\delta_\ell(v)|$ of all nodes v is still small.

▶ **Lemma 8.** *For all $v \in V$ and $t \geq 0$, we have $|\delta_t(v)| = |\beta_t(v) - p_t(s,v)| \leq t \cdot n^{-8}$.*

Proof. We prove the lemma by induction on t. As $\beta_0(v) = p_0(s,v)$, the lemma is true for $t = 0$. Let us therefore consider the induction step. Let T be the transition matrix of the considered lazy random walk on G. The recursion 2 can then be expressed as $\boldsymbol{p_t} = T \cdot \boldsymbol{p_{t-1}}$, where $\boldsymbol{p_t}$ is the n-vector defined by the probabilities $p_t(s,v)$ for each node v. Similarly, the vector $\boldsymbol{\beta'_t}$ is computed as

$$\boldsymbol{\beta'_t} = T \cdot \boldsymbol{\beta_{t-1}} = T \cdot (\boldsymbol{p_{t-1}} + \boldsymbol{\delta_{t-1}}) = \boldsymbol{p_t} + T \cdot \boldsymbol{\delta_{t-1}}.$$

Note that because T is a stochastic matrix, $T \cdot \boldsymbol{\delta_{t-1}}$ is a convex combination of the values $\delta_{t-1}(u)$ for $u \in \{v\} \cup N(v)$. The absolute value of $T \cdot \boldsymbol{\delta_{t-1}}$ can therefore be upper bounded by the largest absolute value of $\delta_{t-1}(u)$ for any $u \in V$. By induction, we therefore have $|\beta'_t(v) - p_t(s,v)| \leq (t-1)n^{-8}$. The lemma now follows because $|\beta_t(v) - \beta'_t(v)| \leq n^{-8}$. ◀

We next define how the estimates $\tilde{p}(v) \approx p_\ell(s,v)$ are computed. For sufficiently large constant c, the estimates $\beta_\ell(v)$ are accurate enough to be used in Lemma 7. However, in order to efficiently compute the conductances and balances of all cuts induced by the global order given by the probabilities $\tilde{p}(v)$, we will apply the Theorem 6 (on computing all-prefix-sums). In Theorem 6, we would like the number of node classes to be as small as possible. We therefore define $\delta := \phi/10$ and $\tilde{p}(v)$ as follows:

$$\forall v \in V : \tilde{p}(v) := \begin{cases} 0 & \text{if } \beta_\ell(v) \leq n^{-6}, \\ d(v) \cdot (1+\delta)^{\left\lfloor \log_{1+\delta}\left(\frac{\beta_\ell(v)}{d(v)}\right)\right\rfloor} & \text{otherwise.} \end{cases} \tag{3}$$

That is, $\tilde{p}(v)$ is either 0 or we round down $\beta_\ell(v)/d(v)$ to the next smaller power of $1+\delta$ and we multiply the resulting value by $d(v)$. This guarantees that the value of $\rho_{\tilde{p}}(v) = \tilde{p}(v)/d(v)$ is equal to $\beta_\ell(v)/d(v)$ rounded to the next smaller power of $1+\delta$.

▶ **Lemma 9.** *For all $v \in V$, it holds that $|\tilde{p}(v) - p_\ell(s,v)| \leq \delta\left(n^{-3} + p_\ell(s,v)\right)$. Further, the value $\rho_{\tilde{p}}(v) = \tilde{p}(v)/d(v)$ can only have $O(\log(n)/\phi)$ different values.*

Proof. The second claim follows because $\rho_{\tilde{p}}(v)$ is always a value between 0 and 1 and because it either is 0 or it is of size at least $\Omega(n^{-5})$ and it is an integer power of $1+\delta = 1+\phi/10$.

For the first claim, first observe that because $\phi \geq 2/n^2$, we always have $\ell \leq n^2/16$. Lemma 8 therefore implies that $|p_\ell(s,v) - \beta_\ell(v)| \leq n^{-6}/16$.

Let us first consider the case $\tilde{p}(v) = 0$. In this case, from Equation (3) and Lemma 8, we then get that $p_\ell(s,v) < 2n^{-6}$ and for sufficiently large n, the lemma follows because $\delta = \phi/10 \geq 5/n^2$.

Let us therefore assume that $\tilde{p}(v) > 0$. By the definition of $\tilde{p}(v)$, we then have $1 \leq \beta_\ell(v)/\tilde{p}(v) \leq 1+\delta$. We again use that $|\beta_\ell(v) - p_\ell(s,v)| \leq n^{-6}/16$. Using $\tilde{p}(v) \leq \beta_\ell(v)$, we get that $\tilde{p}(v) \leq p_\ell(s,v) + n^{-6}/16$. Further, by using that $\beta_\ell(v) \leq (1+\delta)\tilde{p}(v) \leq \tilde{p}(v) + \delta\beta_\ell(v)$, we have $(1-\delta)(p_\ell(s,v) - n^{-6}/16) \leq (1-\delta)\beta_\ell(v) \leq \tilde{p}(v)$ and therefore $\tilde{p}(v) \leq p_\ell(s,v) + \delta(p_\ell(s,v) + n^{-6}/16)$. ◀

▶ **Lemma 10.** *Assume that we compute the probability estimates $\tilde{p}(v)$ for Q different random walks where each random walk is started at a random node s chosen according to the degree distribution of G and the length of each random walk is chosen uniformly at random from $\{1, \dots, 1/8\phi\}$. The probability estimates $\tilde{p}(v)$ for all Q random walks can be computed in $O(D + Q/\phi)$ rounds in the CONGEST model, where D is the diameter of the G.*

Proof. Let us first consider the computation of a single random walk. As a first step, we need to randomly choose the starting node s and the length ℓ of the random walk. We can use the BFS tree T to do this. The length ℓ of the random walk can be determined by the root node v_r of T and it can be sent to all nodes in at most D rounds. Assume that each node in T knows the sum of the degrees of all nodes in its subtree. This information can be computed by a simple convergecast in D rounds. Further, based on this knowledge, we can now choose the starting node of the random walk by randomly walking down the tree starting at v_r (at each node we stop or go to a subtree with probability proportional to the corresponding degree sum). The node on which the random walk stops will be the sampled node according to the degree distribution of G. Also note that by using pipelining, the starting nodes and lengths of all the Q random walks can be computed in time $O(D + Q)$.

To compute the probability estimates $\tilde{p}(v)$ of a single random walk, it directly follows from the above discussion that $\beta_\ell(v)$ can be computed in ℓ rounds. Given $\beta_\ell(v)$, node v can compute $\tilde{p}(v)$ locally without any further communication. The lemma therefore follows because $\ell = O(1/\phi)$. ◀

5.2 Evaluating the Induced Cuts

Consider the probability estimates $\tilde{p}(v)$ for some random walk of length ℓ with starting node $s \in V$. We assume that the estimate $\tilde{p}(v)$ are computed as described above. By Lemma 9, the estimates are accurate enough to be used in Lemma 7. In order to evaluate the conductances and balances of the cuts induced by the probability estimates $\tilde{p}(v)$, we intend to use the all-prefix-sums techniques developed in Section 4. In order to apply these techniques, we need a distributed search tree. For this purpose, we can again use the computed BFS tree T of G. In order to get a search tree from T, every node just needs to arbitrarily order its children. Note that the radius of T is upper bounded by the diameter D of G. Let $\prec$ be the search order (on V) defined by the search tree T.

Given the probability estimates $\tilde{p}(v)$, we define $\rho_{\tilde{p}} := \tilde{p}(v)/d(v)$ as before. In order to apply Lemma 7, we need to define a global order $\pi : V \to \mathbb{N}$ on V such that $\pi(u) < \pi(v)$ whenever $\rho_{\tilde{p}}(u) > \rho_{\tilde{p}}(v)$. We define this global order π such that $\pi(u) < \pi(v)$ if and only if either $\rho_{\tilde{p}}(u) > \rho_{\tilde{p}}(v)$ or $\rho_{\tilde{p}}(u) = \rho_{\tilde{p}}(v)$ and $u \prec v$. We first show that an all-prefix-sums problem w.r.t. this global order can be computed efficiently in the CONGEST model.

▶ **Lemma 11.** *Assume that each node $v \in V$ has an integer value b_v (of size at most polynomial in n). Further assume that each node $v \in V$ needs to compute $s_v := \sum_{u \in V : \pi(u) \leq \pi(v)} b_u$. The values s_v for all $v \in V$ can be computed in $O(D + \log(n)/\phi)$ rounds in the CONGEST model. Further, the results of K independent such all-prefix-sums problems (possibly for different random walk probabilities) can be computed in time $O(D + K \log(n)/\phi)$ in the CONGEST model.*

Proof. We first prove the lemma for a single instance of the described all-prefix-sums instance. Note that the global order defined by π has the structure of the order $\sqsubset$ used in Theorem 6. All nodes v with equal value $\rho_{\tilde{p}}(v)$ form a single node class. Two nodes u and v of different classes are then order by the values of $\rho_{\tilde{p}}(u)$ and $\rho_{\tilde{p}}(v)$. Two nodes u and v in the same

class are order by the search tree order $\prec$. We can therefore directly apply Theorem 6 to compute the values s_v for all nodes v. The time complexity of doing this is $O(n+k)$, where k is the number of different node classes. It follows from the second claim of Lemma 9 that the number of classes is at most $O(\log(n)/\phi)$ and thus the lemma follows for $K=1$.

For $K>1$, note that we can use pipelining as described in Theorem 5. For each instance of the all-prefix-sums problem, we run $O(\log(n)/\phi)$ independent all-prefix-sums instances w.r.t. the search tree order $\prec$. In total, we therefore run $O(K\log(n)/\phi)$ independent simple all-prefix-sums instances and the claim of the lemma thus follows. ◀

It remains to show that the problem of evaluating the cuts for a given total order $\pi : V \to \mathbb{N}$ on V can be reduced to computing a few all-prefix-sums computations. Let us therefore consider a global order π on V. We need to compute the conductances and balances of all the cuts defined by the sets $S_\pi(i)$ for $i \in \{1,\dots,n-1\}$. For a node $v \in V$, we define $d^+(v) := |\{u \in N(v) : \pi(u) > \pi(v)\}|$ and let $d^-(v) := |\{u \in N(v) : \pi(u) < \pi(v)\}|$. Note that $d^+(v) + d^-(v) = d(v)$. Note also that every node $v \in V$ can compute $d^+(v)$ and $d^-(v)$ using a single communication round (based on the relative ordering w.r.t. its neighbors). For a node set $S \subset V$, we further let $e(S)$ be the number of edges crossing the cut $(S, \bar{S})$. Recall that $\phi(S) = e(S)/(2mb(S))$ and $b(S) = \min\left\{\text{VOL}(S), \text{VOL}(\bar{S})\right\}/2m$. The following lemma shows that the conductances and balances of all cuts $(S_\pi(i), \bar{S}_\pi(i))$ can be reduced to two all-prefix-sums computations w.r.t. the order π.

▶ **Lemma 12.** *We have $e(S_\pi(1)) = \text{VOL}(S_\pi(1)) = d(v_1)$. For $i > 1$, it further holds that*

$$\begin{aligned} e(S_\pi(i)) &= e(S_\pi(i-1)) + d^+(v_i) - d^-(v_i), \\ \text{VOL}(S_\pi(i)) &= \text{VOL}(S_\pi(i-1)) + d(v_i). \end{aligned}$$

Proof. We first consider the first recursion specifying $e(S_\pi(i))$ The set of edges connecting nodes in $S_\pi(i)$ with nodes in $V \setminus S_\pi(i)$ consists of all edges connecting nodes in $S_\pi(i-1)$ with nodes in $V \setminus S_\pi(i)$ and of all edges connecting v_i with nodes in $V \setminus S_\pi(i)$. The number of edges connecting nodes in $S_\pi(i-1)$ with nodes in $V \setminus S_\pi(i)$ is $e(S_\pi(i-1)) - d^-(v_i)$ and the number of edges connecting v_i with nodes in $V \setminus S_\pi(i)$ is $d^+(v_i)$. The first recursion therefore follows. The second recursion follows immediately by the definition of the volume $\text{VOL}(S)$ of a node set S. ◀

We now have everything we need to prove the main theorem.

▶ **Theorem 3 (restated).** *Given a network graph $G = (V, E)$ and two parameters $b \leq 1/2$ and $\phi < 1$ such that there exits a set $C \subseteq V$ with $b \cdot 2|E| \leq \text{VOL}(C) \leq |E|$ and $\phi(C) \leq \phi$. Then there is a distributed algorithm that finds a cut $(S, \bar{S})$ which satisfies $b|E| \leq \text{VOL}(S) \leq |E|$ and $\phi(S) = O\big(\sqrt{\phi \log n}\big)$ with high probability and finishes in $O\big(D + \frac{\log^2 n}{b\phi}\big)$ rounds in the CONGEST model, where D is the diameter of G.*

Proof. We have already seen that Lemma 7 implies the quality of the returned cut with high probability if we consider all the cuts induced by $O(\log(n)/b)$ random walks (where starting node and length of each random walk are chosen randomly as specified by Lemma 7 and the computed probability estimates satisfy the accuracy demanded by Lemma 7). By Lemma 10, the probability estimates $\tilde{p}(v)$ for $O(\log(n)/b)$ such random walks can be computed in $O(D + \log(n)/(b\phi))$ rounds. Further by Lemma 9, the accuracy of the probability estimates $\tilde{p}(v)$ is good enough to be used in Lemma 7.

In order to prove the theorem, it therefore remains to show that for the $O(\log(n)/b)$ random walks, the conductances and balances of all induced cuts can be computed in

$O\big(D + \frac{\log^2 n}{b\phi}\big)$ rounds. By Lemma 12, for a given global order π on the nodes V, the conductances and balances of all cuts $S_\pi(i)$ for $i \in \{1, \ldots, n-1\}$ can be computed by using 2 all-prefix-sums computations (one for $e(S_\pi(i))$ and one for $\text{VOL}(S_\pi(i))$). The conductances and balances of the cuts of all $O(\log(n)/b)$ random walks can therefore be computed by carrying out $O(\log(n)/b)$ independent all-prefix-sums computations. By Lemma 11, we can therefore compute the conductances and balances of all cuts induced by all random walks in time $O\big(D + \log^2(n)/b\phi\big)$. Note that when doing this, for each cut, possibly only one node in G knows the results. However, we can do a convergecast on the BFS tree T to find the best of all the computed cuts in D additional rounds. This completes the proof of the main theorem. ◀

References

1. Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Using pagerank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007.
2. Reid Andersen and Yuval Peres. Finding sparse cuts locally using evolving sets. In *Proc. of 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 235–244, 2009.
3. Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proc. of 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 227–236, 2007.
4. Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.
5. Sandeep N. Bhatt and Frank T. Leighton. A framework for solving VLSI graph layout problems. *J. Comput. Syst. Sci.*, 28(2):300–343, 1984.
6. Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, 1990.
7. Keren Censor-Hillel and Hadas Shachnai. Fast information spreading in graphs with large weak conductance. *SIAM J. Comput.*, 41(6):1451–1465, 2012.
8. Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Almost tight bounds for rumour spreading with conductance. In *Proc. of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 399–408, 2010.
9. Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. Sparse cut projections in graph streams. In *Proc. of 17th Annual European Symposium (ESA)*, pages 480–491, 2009.
10. Atish Das Sarma, Anisur R. Molla, and Gopal Pandurangan. Distributed computation of sparse cuts via random walks. In *Proc. of 16th International Conference on Distributed Computing and Networking (ICDCN)*, pages 6:1–6:10, 2015.
11. Atish Das Sarma, Danupon Nanongkai, Gopal Pandurangan, and Prasad Tetali. Distributed random walks. *J. ACM*, 60(1):2, 2013.
12. George Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *Proc. Int. Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 57–68, 2011.
13. Christos Gkantsidis, Milena Mihail, and Amin Saberi. Conductance and congestion in power-law graphs. In *Proc. of International Conference on Measurements and Modeling of Computer Systems (SIGMETRICS)*, pages 148–159, 2003.
14. Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989.
15. David R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, 2000.
16. Peter M. Kogge and Harold S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, C-22(8):786–793, 1973.

17 Richard E. Ladner and Michael J. Fischer. Parallel prefix computation. *J. ACM*, 27(4):831–838, 1980.

18 Frank T. Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

19 László Lovász and Miklós Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *Proc. of 31st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 346–354, 1990.

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

21 David W. Matula and Farhad Shahrokhi. Sparsest cuts and bottlenecks in graphs. *Discrete Applied Mathematics*, 27(1-2):113–123, 1990.

22 Yu Ofman. On the algorithmic complexity of discrete functions. *Soviet Physics Doklady*, 7(7):589–591, 1963.

23 David Peleg. *Distributed computing: a locality-sensitive approach.* SIAM, Philadelphia, PA, USA, 2000.

24 Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proc. of 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2004.

25 Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *CoRR*, arXiv: abs/0809.3232v1, 2008.

26 Harold S. Stone. Parallel tridiagonal equation solvers. *ACM Transactions on Mathe- matical Software*, 1(4):289–307, 1975.