SIAM J. COMPUT.
Vol. 42, No. 1, pp. 1–26

A LOCAL CLUSTERING ALGORITHM FOR MASSIVE GRAPHS AND ITS APPLICATION TO NEARLY LINEAR TIME GRAPH PARTITIONING*

DANIEL A. SPIELMAN† AND SHANG-HUA TENG‡

Abstract. We study the design of *local algorithms* for massive graphs. A local graph algorithm is one that finds a solution containing or near a given vertex without looking at the whole graph. We present a local clustering algorithm. Our algorithm finds a good cluster—a subset of vertices whose internal connections are significantly richer than its external connections—near a given vertex. The running time of our algorithm, when it finds a nonempty local cluster, is nearly linear in the size of the cluster it outputs. The running time of our algorithm also depends polylogarithmically on the size of the graph and polynomially on the conductance of the cluster it produces. Our clustering algorithm could be a useful primitive for handling massive graphs, such as social networks and web-graphs. As an application of this clustering algorithm, we present a partitioning algorithm that finds an approximate sparsest cut with nearly optimal balance. Our algorithm takes time nearly linear in the number edges of the graph. Using the partitioning algorithm of this paper, we have designed a nearly linear time algorithm for constructing spectral sparsifiers of graphs, which we in turn use in a nearly linear time algorithm for solving linear systems in symmetric, diagonally dominant matrices. The linear system solver also leads to a nearly linear time algorithm for approximating the second-smallest eigenvalue and corresponding eigenvector of the Laplacian matrix of a graph. These other results are presented in two companion papers.

Key words. graph clustering, graph partitioning, local algorithms, sparsest cut, approximation algorithms

AMS subject classifications. 68Q25, 05C85, 05C81

DOI. 10.1137/080744888

1. Introduction.

Given a vertex of interest in a massive graph, we would like to find a small cluster around that vertex *in time proportional to the size of the cluster*. The algorithm we introduce will solve this problem while examining only vertices near an initial vertex, under some reasonable notion of nearness. We call such an algorithm a *local* algorithm.

Our local clustering algorithm provides a powerful primitive for the design of fast graph algorithms. In section 3 of this paper, we use it to design the first nearly linear time algorithm for graph partitioning that produces a partition of nearly optimal balance among those partitions approximating a target conductance. In the papers [ST11] and [ST08], we use this graph partitioning algorithm to design nearly

*Received by the editors December 29, 2008; accepted for publication (in revised form) July 22, 2012; published electronically January 10, 2013. This paper is the first in a sequence of three papers expanding on material that appeared first under the title "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems" [ST04]. The second paper, "Spectral sparsification of graphs" [ST11], contains further results on partitioning graphs and applies them to producing spectral sparsifiers of graphs. The third paper, "Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems" [ST08], contains the results on solving linear equations and approximating eigenvalues and eigenvectors. This material is based upon work that was supported by the National Science Foundation under grants 0325630, 0634957, 0635102, 0707522, 0915487, 0964481, and 1111270.

http://www.siam.org/journals/sicomp/42-1/74488.html

†Department of Computer Science, Program in Applied Mathematics, Yale University, New Haven, CT 06520 (spielman@cs.yale.edu).

‡Department of Computer Science, Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90089 (shanghua@usc.edu).

linear time algorithms for sparsifying graphs and for solving symmetric, diagonally dominant linear systems.

1.1. Local clustering. We say that a graph algorithm is a *local algorithm* if it is given a particular vertex as input and at each step after the first examines only vertices connected to those it has seen before. The use of a local algorithm naturally leads to the question of in which order one should explore the vertices of a graph. While it may be natural to explore vertices in order of shortest-path distance from the input vertex, such an ordering is a poor choice in graphs of low diameter, such as social network graphs [LH08]. We suggest first processing the vertices that are most likely to occur in short random walks from the input vertex. That is, we consider a vertex to be near the input vertex if it is likely to appear in a short random walk from the input vertex.

In section 2, we use a local graph exploration process to find a cluster that is near the input vertex. Following Kannan, Vempala, and Vetta [KVV04], we say that a set of vertices is a good cluster if it has low *conductance*, that is, if it has many more internal than external edges. We give an efficient local clustering algorithm, `Nibble`, *whose running time is proportional to the size of the cluster it outputs and depends only mildly on the size of the whole graph.*

Although our algorithm may not find a local cluster for some input vertices, we will show that it is usually successful when started from a random vertex in a good cluster. In particular, we prove the following statement, which provides a statistical performance guarantee for `Nibble`: There exists a constant $\alpha > 0$ such that for any target conductance ϕ and any cluster C_0 of conductance at most $\alpha \cdot \phi^2 / \log^3 n$, when given a random vertex v sampled according to degree inside C_0, `Nibble` will, in time $O(|C| \log^6 n/\phi^4)$, return a cluster C mostly inside C_0 and with conductance at most ϕ, with probability at least $1/2$. The slightly unusual form of this statement is dictated by the requirement that our algorithm be local. We discuss the nature of the guarantees one can hope to achieve in section 2.1.5.

The local clustering algorithm `Nibble` makes a novel use of random walks. For a positive integer t, suppose $p_{t,v}$ is the probability distribution of the t-step random walk starting at v. As the support of $p_{t,v}$—the set of nodes with positive probability—could grow rapidly, `Nibble` maintains a truncated version $q_{t,v}$ of the distribution. At each step of the truncated random walks, `Nibble` looks for a cluster among only nodes with high probability. The truncation is critical to ensure that the running time of the clustering algorithm is output sensitive. It guarantees that the size of the support of the distribution that `Nibble` maintains is not too much larger than the size of the cluster it produces. The cluster that `Nibble` produces is local to the starting vertex v in the sense that it consists of nodes that are among the most favored destinations of random walks starting from v.

By using the personal PageRank vector [PBMW98] to define nearness, Andersen, Chung, and Lang [ACL06] have produced an improved version of our algorithm `Nibble`, which they call `PageRank-Nibble`. Their algorithm is further improved by Andersen and Peres [AP09], who base their algorithm on the volume-biased evolving set process. Following our work, other local algorithms have been designed by Andersen et. al. [ABC$^+$07] for approximately computing personal PageRank vectors, by Andersen [And08] for finding dense subgraphs, and by Andersen, Chung, and Lang [ACL07] for partitioning directed graphs.

1.2. Nearly linear time algorithms. Our local clustering algorithm provides a

powerful tool for designing fast graph algorithms. In this paper and its two companion papers, we show how to use it to design randomized, nearly linear time algorithms for several important graph-theoretic and numerical problems.

The need for algorithms whose running time is linear or nearly linear in their input size has increased as algorithms handle larger inputs. For example, in circuit design and simulation, an Intel Quad-Core Itanium Tukwila processor has more than two billion transistors, which is more than 200 times the number of transistors that the Pentium had in 2000 [Cor, Wik]; in scientific computing, one often solves linear systems that involve trillions of variables [NSW+09]; in modern information infrastructure, the Web has grown into a graph of more than a trillion nodes [Goo08] and Facebook has 30 billion content items shared each month [Fac10]. As a result of this rapid growth in problem size, what used to be considered an efficient algorithm, such as an $O(n^{1.5})$-time algorithm, may no longer be adequate for solving problems of these scales. Space complexity poses an even greater problem.

Many basic graph-theoretic problems such as connectivity and topological sorting can be solved in linear or nearly linear time. The efficient algorithms for these problems are built on linear-time primitives such as Breadth-First-Search (BFS) and Depth-First-Search (DFS). Minimum Spanning Trees (MST) and Shortest-Path Trees are examples of other commonly used nearly linear time primitives. We hope to build up the library of nearly linear time graph algorithms that may be used as primitives. While the analyzable variants of the algorithms we present here, and even their improved versions by Andersen, Chung, and Lang [ACL06] and Andersen and Peres [AP09], may not be immediately useful in practice, we believe practical algorithms may be derived from them by making less conservative choices of parameters [VTX09].

Our local clustering algorithm provides an exciting new primitive for developing nearly linear time graph algorithms. Because its running time is proportional to the size of the cluster it produces and depends only mildly on the size of the whole graph, we can repeatedly apply it to remove many clusters from a graph, all within nearly linear time.

In the second part of this paper, we use `Nibble` as a subroutine to construct a randomized graph partitioning algorithm that runs in nearly linear time. To the best of our knowledge, this is the first nearly linear time partitioning algorithm that finds an approximate sparsest cut with approximately optimal balance. A faster algorithm with better output guarantees has since been developed by Orecchia and Vishnoi [OV11]. In our first companion paper [ST11], we apply this new partitioning algorithm to develop a nearly linear time algorithm for producing spectral sparsifiers of graphs. We begin that paper by extending the partitioning algorithm of this paper to obtain a stronger guarantee on its output: if it outputs a small set, then the complement must be contained in a subgraph whose conductance is higher than the target.

2. Clusters, conductance, and local exploration of graphs. Let $G = (V, E)$ be an undirected graph with $V = \{1, \ldots, n\}$. A *cluster* of G is a subset of V that is richly intraconnected but sparsely connected with the rest of the graph. The quality of a cluster can be measured by its conductance, the ratio of the number of its external connections to the number of its total connections.

We let $d(i)$ denote the degree of vertex i. For $S \subseteq V$, we define $\mu(S) = \sum_{i \in S} d(i)$ (often called the volume of S). So, $\mu(V) = 2|E|$. For $S, T \subseteq V$, let $E(S, T)$ be the set of edges connecting a vertex in S with a vertex in T. We define the *conductance*

of a set of vertices S, written $\Phi(S)$, by

$$\Phi(S) \stackrel{\text{def}}{=} \frac{|E(S, V-S)|}{\min(\mu(S), \mu(V-S))}.$$

The *conductance* of G is then given by

$$\Phi_G \stackrel{\text{def}}{=} \min_{S \subset V} \Phi(S).$$

We sometime refer to a subset S of V as a *cut* of G and refer to $(S, V-S)$ as a *partition* of G. The *balance* of a cut S or a partition $(S, V-S)$ is then equal to

$$\mathbf{bal}(S) = \min(\mu(S), \mu(V-S))/\mu(V).$$

We call S a *sparsest cut* of G if $\Phi(S) = \Phi_G$ and $\mu(S)/\mu(V) \leq 1/2$.

In the construction of a partition of G, we will be concerned with vertex-induced subgraphs of G. However, when measuring the conductance and volumes of vertices in these vertex-induced subgraphs, we will continue to measure the volume according to the degrees of vertices in the original graph. For clarity, we define the conductance of a set S in the subgraph induced by $A \subseteq V$ by

$$\Phi_A^G(S) \stackrel{\text{def}}{=} \frac{|E(S, A-S)|}{\min(\mu(S), \mu(A-S))}$$

and

$$\Phi_A^G \stackrel{\text{def}}{=} \min_{S \subset A} \Phi_A^G(S).$$

For convenience, we define $\Phi_A^G(\emptyset) = 1$ and, for $|A| = 1$, $\Phi_A^G = 1$.

For $A \subseteq V$, we let $G(A)$ denote the subgraph of G induced by the vertices in A. We introduce the notation $G\{A\}$ to denote graph $G(A)$ to which self-loops have been added so that every vertex in $G\{A\}$ has the same degree as in G. Each self-loop adds 1 to the degree. We remark that if $G(A)$ is the subgraph of G induced on the vertices in A, then

$$\Phi_A^G = \Phi_{G\{A\}} \leq \Phi_{G(A)}.$$

So, when we prove lower bounds on Φ_A^G, we obtain lower bounds on $\Phi_{G(A)}$.

The identification of clusters may be viewed as an optimization problem: Given an undirected graph G and a parameter ϕ, find a cluster C such that $\Phi(C) \leq \phi$ or determine that no such cluster exists. The problem is NP-complete (see, for example, [LR99] or [SS06]). But, approximation algorithms exist. Leighton and Rao [LR99] used linear programming to obtain $O(\log n)$-approximations of the sparsest cut. Arora, Rao, and Vazirani [ARV04] improved this to $O(\sqrt{\log n})$ through semidefinite programming. Faster algorithms obtaining similar guarantees have been constructed by Arora, Hazan, and Kale [AHK04], Khandekar, Rao, and Vazirani [KRV06], Arora and Kale [AK07], Orecchia et al. [OSVV08], and Sherman [She09].

2.1. The algorithm `Nibble`. The algorithm `Nibble` works by approximately computing the distribution of a few steps of the random walk starting at a seed vertex v. It is implicit in the analysis of the volume estimation algorithm of Lovász and

Simonovits [LS93] that one can find a cut with small conductance from the distributions of the steps of the random walk starting at any vertex from which the walk does not mix rapidly. We will observe that a random vertex in a set of low conductance is probably such a vertex. We then extend the analysis of Lovász and Simonovits to show that one can find a cut with small conductance from approximations of these distributions and that these approximations can be computed quickly. In particular, we will truncate all small probabilities that appear in the distributions to 0. In this way, we reduce the work required to compute our approximations.

2.1.1. Diffusion of probability mass: Distributions of random walks. For the rest of this section, we will work with a graph $G = (V, E)$ with n vertices and m edges, so that $\mu(V) = 2m$. We will allow some of these edges to be self-loops. Except for the self-loops, which we allow to occur with multiplicities, the graph is assumed to be unweighted. We will let A be the adjacency matrix of this graph. That is,

$$A(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \text{ and } u \neq v, \\ k & \text{if } u = v \text{ and this vertex has } k \text{ self-loops}, \\ 0 & \text{otherwise.} \end{cases}$$

We define the following two vectors supported on a set of vertices S:

$$\chi_S(u) = \begin{cases} 1 & \text{for } u \in S, \\ 0 & \text{otherwise,} \end{cases}$$

$$\psi_S(u) = \begin{cases} d(u)/\mu(S) & \text{for } u \in S, \\ 0 & \text{otherwise.} \end{cases}$$

We will consider the random walk that at each time step stays at the current vertex with probability 1/2 and otherwise moves to the endpoint of a random edge attached to the current vertex. Thus, self-loops increase the chance that the walk stays at the current vertex. For example, if a vertex has four edges, one of which is a self-loop, then when the walk is at this vertex it has a 5/8 chance of staying at that vertex and a 1/8 chance of moving to each of its three neighbors.

In our local clustering algorithm `Nibble`, we care more about the probability distribution induced by a random walk after a certain number of steps than the vertex that the walk reaches. From this perspective, it might be more convenient to think in terms of diffusion of probability mass: We start with some initial distribution of probability mass, and at each step each node keeps half of its mass and equitably distributes the other half of its mass to its neighbors.

The change in probability mass after one step of the random walk is a linear operator that is realized by multiplying the column vector of probabilities by the matrix

$$M \stackrel{\text{def}}{=} (AD^{-1} + I)/2,$$

where $d(i)$ is the degree of node i, and D is the diagonal matrix with diagonal entries $(d(1), \ldots, d(n))$. Typically, a random walk starts at a node v. In this case, the distribution of the random walk at time t evolves according to $p_t = M^t \chi_v$.

This diffusion process always converges on a connected graph. Moreover, upon convergence the amount of mass at each vertex will be proportional to its degree,

which is independent of the starting distribution of the probability mass. Thus, ψ_V is *the steady-state distribution* of any random walk, and ψ_S is the restriction of that walk to the set S.

2.1.2. Vector-based graph partitioning. Like the spectral partitioning algorithm of Hagen and Kahng [HK92] and that implicit in the work of Cheeger [Che70] (see also [Mih89, ST96]), and the probability-based algorithm implicit in the work of Lovász and Simonovits [LS90], `Nibble` extracts a cluster from a vector by the following algorithm:

1. Given an n-dimensional real vector x, with one entry $x(u)$ for each vertex $u \in \{1, \ldots, n\}$, compute the permutation π that orders the entries of x from largest to smallest, i.e., $x(\pi(1)) \geq x(\pi(2)) \geq \cdots \geq x(\pi(n))$.
2. Examine the $n-1$ potential cuts, $S_1, \ldots, S_{n-1}$, where S_i is either $\{\pi(1), \ldots, \pi(i)\}$ or $\{\pi(i+1), \ldots, \pi(n)\}$ (e.g., whichever has the smaller volume), and return the cut of best quality.

For example, in [Mih89, ST96], an approximate Fiedler vector of a graph is used as x and the conductance of S_i is used as the measure of its quality (we recall that the Fiedler vector of a graph G is the eigenvector associated with the second-smallest eigenvalue of the Laplacian matrix of G). Inspired by Lovász and Simonovits [LS90], we will extract clusters from probability distributions of random walks. In the statement of the algorithm and its analysis, we will use the following notation. For a vector p with nonnegative entries, we let $S_j(p)$ be the set of j vertices u maximizing $p(u)/d(u)$, breaking ties lexicographically. That is, $S_j(p) = \{\pi(1), \ldots, \pi(j)\}$, where π is the permutation such that

$$\frac{p(\pi(i))}{d(\pi(i))} \geq \frac{p(\pi(i+1))}{d(\pi(i+1))}$$

for all i, and $\pi(i) < \pi(i+1)$ when these two ratios are equal. That is, $S_j(p)$ is the jth set induced by the *degree-normalized vector* of p:

$$\left(\frac{p(1)}{d(1)}, \ldots, \frac{p(n)}{d(n)}\right).$$

We then set

$$\lambda_j(p) = \mu\left(S_j(p)\right) = \sum_{u \in S_j(p)} d(u).$$

Note that $\lambda_n(p)$ always equals $2m$.

In the stationary distribution of a random walk on a graph, the probability of being at a node is proportional to its degree. So, in the limit $p(i)/d(i)$ will approach $1/n$. While this limiting distribution provides little information, Lovász and Simonovits [LS90] show that the distributions from the early stages of random walks can be used to find good clusters.

2.1.3. Curves as potential function values. We view the analysis of the rate of convergence of random walks performed by Lovász and Simonovits [LS90] as a potential function argument. Unlike traditional potential function arguments, their potential function does not take on real values. Rather, the value of their potential function is a concave function mapping $[0, 2m]$ onto $[0, 1]$. By plotting this function, we obtain a concave curve in $\mathbb{R}^2$ from the point $(0, 0)$ to the point $(2m, 1)$. The curve

corresponding to the stationary distribution is the straight line between these points. The curve corresponding to the distribution at any step of a random walk lies strictly below the curve from the previous step. The amount by which one curve lies beneath the previous depends on a factor related to the conductance of the graph, and one can show that in graphs of high conductance the curve quickly approaches the straight line.

Following Lovász and Simonovits [LS90], we set

$$I(p,x) = \max_{\substack{w\in[0,1]^n \\ \sum w(u)d(u)=x}} \sum_{u\in V} w(u)p(u). \tag{1}$$

This function $I(p,\cdot)$ is essentially the same as the function h defined by Lovász and Simonovits [LS90]—it differs only by a linear transformation. One can easily check that $I(p,0)=0$ and that $I(p,2m)=1$. As the distribution p approaches the stationary distribution, the curve $I(p,\cdot)$ approaches the straight line.

We remark that for $x=\lambda_j(p)$, $I(p,x)=p(S_j(p))$, and that $I(p,x)$ is linear in x between these points. Finally, we let $I_x(p,x)$ denote the partial derivative of $I(p,x)$ with respect to x, with the convention that for $x=\lambda_j(p)$,

$$I_x(p,x) = \lim_{\delta\to 0} I_x(p,x-\delta) = \frac{p(\pi(j))}{d(\pi(j))},$$

where π is the permutation specified above so that $\pi(j)=S_j(p)-S_{j-1}(p)$.

As $p(\pi(i))/d(\pi(i))$ is nonincreasing, $I_x(p,x)$ is a nonincreasing function in x and $I(p,x)$ is a concave function in x. We discuss the function $I(p,x)$ further in section 2.3.

2.1.4. Constants and parameters. During the course of our exposition, we will need to set many constants, which we collect here for convenience. For each, we provide a suitable value and indicate where it is first used in the paper.

constant	value	where first used
c_1	200	(10)
c_2	280	(13)
c_3	1800	(16)
c_4	140	`Nibble`, condition (C.4)
c_5	20	Definition 2.11
c_6	60	Definition 2.11

The following is an exhaustive list of the inequalities we require these constants to satisfy:

$$c_2 \geq 2c_4, \tag{2}$$

$$c_6 \geq 2c_5, \tag{3}$$

$$c_3 \geq 8c_5, \tag{4}$$

$$c_4 \geq 4c_5, \tag{5}$$

$$\frac{1}{2c_6} - \frac{1}{c_3} - \frac{1}{2c_5c_6} \geq \frac{1}{c_4}, \tag{6}$$

$$\frac{1}{2c_5} \geq \frac{6}{5c_6} + \frac{1}{c_1}, \tag{7}$$

$$\frac{1}{5} \geq \frac{1}{c_5} + \frac{4c_6}{3c_3} + \frac{1}{2c_1} + \frac{1}{2c_2}. \tag{8}$$

Given a ϕ, we set constants that will play a prominent role in our analysis:

$$\ell \stackrel{\text{def}}{=} \lceil \log_2 (\mu(V)/2) \rceil, \tag{9}$$

$$t_1 \stackrel{\text{def}}{=} \left\lceil \frac{2}{\phi^2} \ln \left(c_1(\ell+2)\sqrt{\mu(V)/2} \right) \right\rceil, \tag{10}$$

$$t_h \stackrel{\text{def}}{=} ht_1 \quad \text{for } 0 \le h \le \ell+1, \tag{11}$$

$$t_{last} \stackrel{\text{def}}{=} (\ell+1)t_1, \tag{12}$$

$$f_1(\phi) \stackrel{\text{def}}{=} \frac{1}{c_2(\ell+2)t_{last}}. \tag{13}$$

Note that

$$f_1(\phi) \ge \Omega\left(\frac{\phi^2}{\log^3 \mu(V)} \right).$$

2.1.5. `Nibble` and its statistical guarantee. The algorithm `Nibble` approximately computes the distributions of random walks, but restricts the probability mass to a subset of vertices (near the starting vertex) that is not too much larger than the size of cluster we are aiming for. To do this, it computes *truncated* probability distributions in which all small probability values are rounded to zero.

We will use the truncation operation defined by

$$[p]_\epsilon(u) = \begin{cases} p(u) & \text{if } p(u) \ge d(u)\epsilon, \\ 0 & \text{otherwise.} \end{cases}$$

Our algorithm `Nibble` will generate a sequence of vectors starting at χ_v by the rules

$$q_t = \begin{cases} \chi_v & \text{if } t = 0, \\ Mr_{t-1} & \text{otherwise,} \end{cases} \tag{14}$$

$$r_t = [q_t]_\epsilon. \tag{15}$$

That is, at each time step, we will evolve the random walk one step from the current density and then round every $q_t(u)$ that is less than $d(u)\epsilon$ to 0. Note that q_t and r_t are not necessarily probability vectors, as their components may sum to less than 1.

In our analysis, we will extend the theory of Lovász and Simonovits [LS90] from precise diffusions to these truncated diffusions.

$C = \texttt{Nibble}(G, v, \phi, b)$
where v is a vertex
$0 < \phi < 1$
b is a positive integer governing the size of the set returned and the running time.

1. Set

$$\epsilon = 1/(c_3(\ell + 2)t_{last}2^b). \tag{16}$$

2. Set $q_0 = \chi_v$ and $r_0 = [q_0]_\epsilon$.
3. For $t = 1$ to t_{last}
   (a) Set $q_t = Mr_{t-1}$.
   (b) Set $r_t = [q_t]_\epsilon$.
   (c) If there exists a j such that
   (C.1) $\Phi(S_j(q_t)) \leq \phi$,
   (C.2) $\lambda_j(q_t) \leq (5/6)\mu(V)$,
   (C.3) $2^b \leq \lambda_j(q_t)$, and
   (C.4) $I_x(q_t, 2^b) \geq 1/c_4(\ell + 2)2^b$,
   then return $C = S_j(q_t)$ and quit.
4. Return $C = \emptyset$.

Condition (C.1) in the pseudocode for `Nibble` guarantees that the set C has low conductance. Condition (C.2) ensures that it does not contain too much volume, while condition (C.3) ensures that it does not contain too little. Condition (C.4) guarantees that many elements of C have large probability mass. While it would be more natural to define condition (C.4) as a constraint on $I_x(q_t, \lambda_j(q_t))$ instead of $I_x(q_t, 2^b)$, our proof of correctness requires the latter.

In the rest of this section, we will prove the following theorem on the performance of `Nibble`.

THEOREM 2.1 (`Nibble`). `Nibble` *can be implemented so that on all inputs, it runs in time* $O(2^b(\log^6 m)/\phi^4)$. *Moreover,* `Nibble` *satisfies the following properties:*

(N.1) *When* $C = \texttt{Nibble}(G, v, \phi, b)$ *is nonempty,*

$$\Phi(C) \leq \phi \quad \textit{and} \quad \mu(C) \leq (5/6)\mu(V).$$

(N.2) *Each set* S *satisfying*

$$\mu(S) \leq (2/3)\mu(V) \quad \textit{and} \quad \Phi(S) \leq f_1(\phi)$$

has a subset S^g *such that*
(N.2.a) $\mu(S^g) \geq \mu(S)/2$, *and*
(N.2.b) $v \in S^g$ *and* $C = \texttt{Nibble}(G, v, \phi, b) \neq \emptyset$ *imply* $\mu(C \cap S) \geq 2^{b-1}$.

(N.3) *The set* S^g *may be partitioned into subsets* $S^g_0, \ldots, S^g_\ell$ *such that if* $v \in S^g_b$, *then the set* C *output by* $\texttt{Nibble}(G, v, \phi, b)$ *will not be empty.*

The assertions we prove about the algorithm `Nibble` may at first seem unusual. This is mostly due to our requirements that `Nibble` behave locally and that it run in a limited time budget.

We have designed `Nibble` so that it runs in time $O(2^b(\log^6 m)/\phi^4)$ when looking for a cluster of conductance at most ϕ. This is one reason that `Nibble` might fail to return a cluster: if there is no such cluster of size less than this time bound, then `Nibble` can certainly not return one. To ensure that the running time of `Nibble` is

not too much greater than its output size, we require in condition (C.3) that the set it outputs have volume at least 2^b.

If there is a cluster S of conductance less than ϕ, then we must seed `Nibble` with a vertex near this cluster to have any hope of finding it. Some vertices in the cluster may be insufficient if they are more strongly attached to vertices outside the cluster than inside it. This can happen because our definition of a cluster does not require any sort of local optimality. To handle this problem, we define the *diffusion core* of a cluster S to be the set $S^g \subseteq S$ such that random walks started from vertices in S^g probably stay inside S for a long time. Property (N.2.a) of Theorem 2.1 says that the diffusion core is at least half of S, measured by degree. We view this as a statistical guarantee: with probability at least $1/2$ a vertex chosen from S according to degree lies inside the *diffusion core*.

If the set S is big and `Nibble` is started from a random vertex in the diffusion core of S, it is still possible that `Nibble` will succeed for a small value of b. The reason is that the set S might contain small clusters inside it, and `Nibble` could reasonably decide to return one of these. For this reason, we prove just that there is some size parameter b for which `Nibble` succeeds. This provides a powerful guarantee when randomization is used: If we sample the vertices of S according their degree and sample b uniformly at random from $\{1, \ldots, \log m\}$, then $\texttt{Nibble}(G, v, \phi, b)$ will be successful with probability at least $1/(2 \log m)$.

In section 3, we will use this statistical property of `Nibble` to design the first nearly linear time algorithm for computing graph partitions of nearly optimal balance. The guarantee that the partitions are balanced is a result of the guarantee in property (N.2.b) of Theorem 2.1 that $v \in S^g$ and $C = \texttt{Nibble}(G, v, \phi, b) \neq \emptyset$ imply $\mu(C \cap S) \geq 2^{b-1}$.

2.2. Basic inequalities about random walks. We first establish some basic inequalities that will be useful in our analysis. Readers who are eager to see the analysis of `Nibble` can first skip this subsection. Suppose $G = (V, E)$ is an undirected graph. Recall $M = (AD^{-1} + I)/2$, where A is the adjacency matrix of G.

PROPOSITION 2.2 (monotonicity of multiplication by M). *For all nonnegative vectors p,*

$$\left\|D^{-1}(Mp)\right\|_\infty \leq \left\|D^{-1}p\right\|_\infty.$$

Proof. Applying the transformation $z = D^{-1}p$, we see that it is equivalent to show that for all z,

$$\left\|D^{-1}MDz\right\|_\infty \leq \|z\|_\infty.$$

To prove this, we note that $D^{-1}MD = D^{-1}(AD^{-1} + I)D/2 = M^T$ and that the sum of the entries in each row of this matrix is 1. ∎

DEFINITION 2.3. *For a set $S \subseteq V$, we define the matrix D_S to be the diagonal matrix such that $D_S(u, u) = 1$ if $u \in S$ and 0 otherwise.*

PROPOSITION 2.4. *For every $S \subseteq V$, all nonnegative vectors p and q, and every $t \geq 1$,*

$$p^T(D_S M)^t q \leq p^T M^t q.$$

Proof. For $t = 1$, we observe that

$$p^T(M)q = p^T((D_S + D_{\bar{S}})M)q = p^T(D_S M)q + p^T(D_{\bar{S}} M)q \geq p^T(D_S M)q,$$

as p, q, $D_{\bar{S}}$, and M are all nonnegative. The proposition follows by induction. ∎

PROPOSITION 2.5 (escaping mass). *For all $t \geq 0$ and for all $S \subset V$,*

$$\Pr[\textit{t-step walk starting with } \psi_S \textit{ stays entirely in } S] = \mathbf{1}^T (D_S M)^t \psi_S \geq 1 - t\Phi_V(S)/2.$$

Proof. Note that $M\psi_S$ is the distribution after one step of a walk from a random vertex in S and $\mathbf{1}^T D_S (M\psi_S)$ is the probability that this step stays inside S. Thus, $\mathbf{1}^T (D_S M)^t \psi_S$ is the probability that a t-step walk starting from a random vertex in S stays entirely in S.

We first prove by induction that for all $t \geq 0$,

$$\left\| D^{-1} (D_S M)^t \psi_S \right\|_\infty \leq 1/\mu(S). \tag{17}$$

The base case, $t = 0$, follows from the fact that $\left\| D^{-1}\psi_S \right\|_\infty = 1/\mu(S)$. To complete the induction, observe that if x is a nonnegative vector such that $\left\| D^{-1}x \right\|_\infty \leq 1/\mu(S)$, then

$$\left\| D^{-1}(D_S M)x \right\|_\infty = \left\| D_S D^{-1} M x \right\|_\infty \leq \left\| D^{-1} M x \right\|_\infty \leq \left\| D^{-1} x \right\|_\infty \leq 1/\mu(S),$$

where the second-to-last inequality follows from Proposition 2.2.

We will now prove that for all t,

$$\mathbf{1}^T (D_S M)^t \psi_S - \mathbf{1}^T (D_S M)^{t+1} \psi_S \leq \Phi_V(S)/2,$$

from which the proposition follows, as $\mathbf{1}^T \psi_S = 1$.

Observing that $\mathbf{1}^T M = \mathbf{1}^T$, we compute

$$\begin{aligned}
&\mathbf{1}^T (D_S M)^t \psi_S - \mathbf{1}^T (D_S M)^{t+1} \psi_S \\
&= \mathbf{1}^T (I - D_S M)(D_S M)^t \psi_S \\
&= \mathbf{1}^T (M - D_S M)(D_S M)^t \psi_S \\
&= \mathbf{1}^T (I - D_S) M (D_S M)^t \psi_S \\
&= \chi_{\bar{S}}^T M (D_S M)^t \psi_S \\
&= (1/2)\chi_{\bar{S}}^T (I + AD^{-1})(D_S M)^t \psi_S \\
&= (1/2)\chi_{\bar{S}}^T (AD^{-1})(D_S M)^t \psi_S && (\text{as } \chi_{\bar{S}}^T I D_S = \mathbf{0}) \\
&\leq (1/2)\left| E(S, V-S) \right| \left\| D^{-1}(D_S M)^t \psi_S \right\|_\infty \\
&\leq (1/2)\left| E(S, V-S) \right| / \mu(S) && (\text{by inequality (17)}) \\
&\leq \Phi_V(S)/2. \qquad ∎
\end{aligned}$$

2.3. The analysis of `Nibble`. Our analysis of `Nibble` consists of three main steps. First, we define the diffusion core S^g of a cluster S mentioned in Theorem 2.1 and establish property (N.2). We then refine the structure of S^g to define sets S_b^g and prove property (N.3). The sets S^g and S_b^g are defined in terms of the distributions of random walks from a vertex in S, without reference to the truncation we perform in the algorithm. We then analyze the impact of truncation used in `Nibble` and extend the theory of Lovász and Simonovits [LS93] to truncated random walks.

Step 1: The diffusion core of a cluster and its properties.

DEFINITION 2.6 (S^g). *For each set $S \subseteq V$, we define S^g to be the set of nodes v in S such that for all $t \leq t_{last}$,*

$$\chi_{\bar{S}}^T M^t \chi_v \leq t_{last} \Phi(S).$$

Note that $\chi_{\bar{S}}^T M^t \chi_v$ denotes the probability that a t-step random walk starting from v terminates outside S. Roughly speaking, S^g is the set of vertices $v \in S$ such that a random walk from v is reasonably likely to still be in S after t_{last} time steps. We will prove the following bound on the volume of S^g.

LEMMA 2.7 (volume of S^g).

$$\mu(S^g) \geq \mu(S)/2.$$

Proof. Let $S \subseteq V$, and let D_S be the diagonal matrix such that $D_S(u,u) = 1$ if $u \in S$ and 0 otherwise. For $t \geq 0$,

$$\begin{aligned}
\chi_{\bar{S}}^T M^t \chi_v &= (\mathbf{1} - \chi_S)^T M^t \chi_v \\
&= \mathbf{1}^T \chi_v - \chi_S^T M^t \chi_v \quad (\text{by } \mathbf{1}^T M^t = \mathbf{1}^T) \\
&= 1 - \chi_S^T M^t \chi_v \\
&\leq 1 - \mathbf{1}^T (D_S M)^t \chi_v \quad (\text{by Proposition 2.4}) \\
&\leq 1 - \mathbf{1}^T (D_S M)^{t_{last}} \chi_v,
\end{aligned}$$

as $\mathbf{1}^T (D_S M)^t \chi_v$ is a nonincreasing function of t. Define

$$S' = \left\{ v : 1 - \mathbf{1}^T (D_S M)^{t_{last}} \chi_v \leq t_{last} \Phi(S) \right\}.$$

So, $S' \subseteq S^g$, and it suffices to prove that $\mu(S') \geq \mu(S)/2$.

Applying Proposition 2.5, we obtain

$$\begin{aligned}
\frac{t_{last} \Phi(S)}{2} &\geq 1 - \mathbf{1}^T (D_S M)^{t_{last}} \psi_S \\
&= \sum_{v \in S} \frac{d(v)}{\mu(S)} \left(1 - \mathbf{1}^T (D_S M)^{t_{last}} \chi_v\right) \\
&> \sum_{v \in S - S'} \frac{d(v)}{\mu(S)} t_{last} \Phi(S) \quad (\text{by the definition of } S') \\
&= \frac{\mu(S - S')}{\mu(S)} t_{last} \Phi(S).
\end{aligned}$$

So, we may conclude that

$$\frac{\mu(S - S')}{\mu(S)} < \frac{1}{2},$$

from which the lemma follows. ∎

We now prove the following lemma, which says that if `Nibble` is started from any $v \in S^g$ with parameter b and returns a nonempty set C, then $\mu(C \cap S) \geq 2^{b-1}$. This lemma will be used in section 3 to ensure that our graph partitioning algorithm produces a cut of nearly optimal balance.

LEMMA 2.8 (property (N.2) of Theorem 2.1). *Let $S \subseteq V$ be a set of vertices such that $\Phi(S) \leq f_1(\phi)$. If* `Nibble` *is run with parameter b, is started at a $v \in S^g$, and outputs a nonempty set C, then $\mu(C \cap S) \geq 2^{b-1}$.*

Proof. For $v \in S^g$, let q_t be given by (14) and (15). Then, for $t \leq t_{last}$,

$$\chi_{\bar{S}}^T q_t \leq \chi_{\bar{S}}^T M^t \chi_v \leq \Phi(S) t_{last} \leq f_1(\phi) t_{last} \leq \frac{1}{c_2(\ell+2)},$$

where the second inequality follows from the definition of S^g.

Let t be the index of the step at which the set C is generated. Let j' be the least integer such that $\lambda_{j'}(q_t) \geq 2^b$. Condition (C.3) implies $j' \leq j$. As I_x is nonincreasing in its second argument and constant between 2^b and $\lambda_{j'}(q_t)$, condition (C.4) guarantees that for all $u \in S_{j'}(q_t)$,

$$q_t(u)/d(u) \geq 1/c_4(\ell+2)2^b.$$

Thus,

$$\mu\left(S_{j'}(q_t) \cap \bar{S}\right) = \sum_{u \in S_{j'}(q_t) \cap \bar{S}} d(u) \leq \sum_{u \in S_{j'}(q_t) \cap \bar{S}} c_4(\ell+2)2^b q_t(u)$$
$$\leq c_4(\ell+2)2^b(\chi_{\bar{S}}^T q_t) \leq \frac{c_4(\ell+2)2^b}{c_2(\ell+2)} \leq 2^{b-1}$$

by (2). So, $\mu(S_{j'}(q_t) \cap S) \geq 2^{b-1}$, and, as $j' \leq j$,

$$\mu(S_j(q_t) \cap S) \geq \mu(S_{j'}(q_t) \cap S) \geq 2^{b-1}. \qquad \square$$

Step 2: Refining the diffusion core. Before defining the sets S_b^g, we first recall some of the properties of the curves $I(p_t, .)$ established by Lovász and Simonovits [LS90]. These facts will motivate our definitions and analysis.

In the first part of the proof of Lemma 1.3 of [LS90], Lovász and Simonovits prove the following lemma.

LEMMA 2.9. *For every nonnegative vector p and every x,*

$$I(Mp, x) \leq I(p, x). \tag{18}$$

For each p_t, $I(p_t, x)$ is a concave function that starts at $(0, 0)$ and goes to $(\mu(V), 1)$. Lemma 2.9 says that for each t, the curve defined by $I(p_{t+1}, \cdot)$ lies below the curve defined by $I(p_t, \cdot)$. In particular,

$$\forall x, I(p_{t+1}, x) \leq I(p_t, x). \tag{19}$$

If none of the sets $S_j(p_{t+1})$ has conductance less than ϕ, then Lovász and Simonovits [LS90] prove a bound on how far below $I(p_t, \cdot)$ the curve of $I(p_{t+1}, \cdot)$ must lie. The following lemma is a special case of Lemma 1.4 of [LS93], restricted to points x of the form $\lambda_j(Mp)$. Lovász and Simonovits [LS90] claim that the following is true for all x, but they point out in the journal version of their paper [LS93] that this claim was false. Fortunately, we do not need the stronger claim.

LEMMA 2.10 (Lovász–Simonovits: potential drop in diffusion). *For any nonnegative vector p, if $\Phi(S_j(Mp)) \geq \phi$, then for $x = \lambda_j(Mp)$,*

$$I(Mp, x) \leq \frac{1}{2}\left(I\big(p, x - 2\phi\widehat{x}\big) + I\big(p, x + 2\phi\widehat{x}\big)\right),$$

where $\widehat{x}$ *denotes* $\min(x, 2m - x)$.

The mistake in [LS90] is the assertion in the beginning of the proof that the inequality holds for all x if it holds for all x of form $\lambda_j(Mp)$.

When this lemma applies, one may draw a chord across the curve of $I(p_t, \cdot)$ around x of width proportional to ϕ and know that $I(p_{t+1}, x)$ lies below.

Proof overview. Before moving on to the more technical part of this paper, we would like to outline our proof strategy.

Our proof extends the analysis of Lovász and Simonovits [LS90] to handle the impact of truncation in the diffusion. We follow the basic strategy of Lovász and Simonovits [LS90], who showed that if a random walk does not mix quickly, then one of the sets $S_j(p_t)$ in the early steps of the random walk must have small conductance. Their analysis uses proof by contradiction: assuming that none of the sets $S_j(p_t)$ has conductance less than ϕ, Lemma 2.10 gives a lower bound on the rate at which the curve $I(p_t, \cdot)$ approaches a straight line. On the other hand, Proposition 2.5 tells us that some point of $I(p_{t_{last}}, \cdot)$ lies well above this line, establishing the needed contradiction.

We analyze `Nibble` by examining the evolution of the curves $I(q_t, .)$. In Lemma 2.13, we bound the impact of the truncation on the curves $I(q_t, .)$. This bound will be used in Lemma 2.16 to show that if the truncation parameter ϵ is small enough and none of the sets $S_j(q_t)$ has conductance less than ϕ, then the curve corresponding to the truncated diffusion also approaches a straight line at a good speed. On the other hand, in Lemma 2.14 we show that if S satisfies the desired balance and conductance condition, and if ϵ is properly chosen, then some point of the curve $I(q_t, .)$ must lie well above the straight line. So, as long as the ϵ is small enough, the argument of Lovász and Simonovits [LS90] can be extended to truncated diffusions.

However, the running time of `Nibble` increases as we decrease ϵ. Thus, the key is to get ϵ just right. Equation (16) of `Nibble` ϵ is set to be inversely proportional to 2^b, which is approximately the volume of the cluster we hope to return.

Refining S^g. We refine the diffusion core S^g by partitioning it into sets S^g_b such that `Nibble` will return a cluster if it is started from a vertex $v \in S^g_b$ and run with parameter b. We define the sets S^g_b for $b = 1, \ldots, \ell$, simultaneously with two quantities, h_v and x_h, so that `Nibble` will stop between iterations $t_{h_v - 1}$ and t_{h_v} and so that the point $(x_{h_v}, q_{t_{h_v}}(x_{h_v}))$ is far above the straight line from $(0, 0)$ to $(2m, 1)$. The existence of this point far above the straight line guarantees that at least one of the sets considered by `Nibble` has sufficiently low conductance. Note that t_h is defined in (11).

DEFINITION 2.11 (x_h, h_v, and S^g_b). *Given a $v \in S^g$, let $p_t = M^t \chi_v$. For $0 \le h \le \ell + 1$, define $x_h(v)$ to be the real number such that*

$$I(p_{t_h}, x_h(v)) = \frac{h + 1}{c_5(\ell + 2)}.$$

We write x_h instead of $x_h(v)$ when v is clear from the context. Define

$$h_v = \begin{cases} \ell + 1 & \text{if } x_\ell(v) \ge 2m/c_6(\ell + 2), \\ \min\{h : x_h \le 2x_{h-1}\} & \text{otherwise.} \end{cases}$$

We define

$$S^g_0 = \{v : x_{h_v - 1}(v) < 2\},$$

and for $b = 1, \ldots, \ell$, we define

$$S_b^g = \left\{ v : x_{h_v - 1}(v) \in [2^b, 2^{b+1}) \right\}.$$

PROPOSITION 2.12 (basic soundness). *The quantities h_v are well defined, and the sets S_b^g partition S^g. Moreover, $x_{h-1} < x_h$ for all h.*

Proof. It follows from the definition of the curve I that for a probability vector p, the slope of $I(p, \cdot)$ is always less than 1; so, $x_0 \geq 1/c_5(\ell + 2)$. If $x_\ell < \mu(V)/c_6(\ell + 2)$, then

$$\frac{x_\ell}{x_0} < \frac{\mu(V)\, c_5(\ell + 2)}{c_6(\ell + 2)} \leq \frac{\mu(V)}{2} \quad \text{(by inequality (3))},$$

so there is an integer $h \leq \ell$ such that $x_h \leq 2x_{h-1}$, and so the quantities h_v are well defined.

To see that the sets S_b^g partition S^g, it now suffices to observe that $x_{h_v - 1} < \mu(V) \leq 2^{\ell+1}$.

Finally, to show that $x_{h-1} < x_h$, we apply Lemma 2.9 to show that

$$I(p_{t_h}, x_{h-1}) \leq I(p_{t_{h-1}}, x_{h-1}) = \frac{h}{c_5(\ell + 2)}.$$

As $I(p_{t_h}, \cdot)$ is nondecreasing and

$$I(p_{t_h}, x_h) > \frac{h}{c_5(\ell + 2)},$$

we can conclude that $x_h > x_{h-1}$. ∎

Step 3: Clustering with truncated random walks. We now establish that vectors produced by the truncated random walk do not differ too much from those produced by the standard random walk.

LEMMA 2.13 (low-impact truncation). *For all $u \in V$ and t,*

$$p_t(u) \geq q_t(u) \geq r_t(u) \geq p_t(u) - t\epsilon d(u), \tag{20}$$

where q_t and r_t are defined in (14) and (15). For all t and x,

$$I(p_t, x) \geq I(q_t, x) \geq I(r_t, x) \geq I(p_t, x) - \epsilon x t. \tag{21}$$

Proof. The left-hand inequalities of (20) are trivial. To prove the right-hand inequality of (20), we consider $p_t - [p_t]_\epsilon$, observe that by definition

$$\left\| D^{-1} \left(p_t - [p_t]_\epsilon \right) \right\|_\infty \leq \epsilon,$$

and then apply Proposition 2.2. Inequality (21) then follows from (1). ∎

LEMMA 2.14 (lower bound on I). *Let $S \subseteq V$ be a set of vertices such that $\mu(S) \leq (2/3)\mu(V)$ and $\Phi(S) \leq f_1(\phi)$, and let v lie in S_b^g. Define q_t by running* `Nibble` *from v with parameter b.*

1. *If $x_\ell(v) \geq 2m/c_6(\ell + 2)$, then*

$$I(q_{t_{\ell+1}}, (2/3)(2m)) \geq 1 - \frac{1}{c_2(\ell + 2)} - \frac{4c_6}{3c_3}. \tag{22}$$

2. *Otherwise,*

$$I(q_{t_{h_v}}, x_{h_v}) \geq \frac{h_v + 1/2}{c_5(\ell+2)}. \tag{23}$$

Proof. In the case $x_\ell(v) \geq 2m/c_6(\ell+2)$, we compute

$$\begin{aligned} I(p_{t_{\ell+1}}, (2/3)(2m)) &\geq I(p_{t_{\ell+1}}, \mu(S)) \\ &\geq \sum_{u \in S} p_{t_{\ell+1}}(u) && \text{(by (1))} \\ &= \chi_S^T p_{t_{\ell+1}} \\ &\geq 1 - t_{last} f_1(\phi) && \text{(by the definition of } S^g) \\ &\geq 1 - \frac{1}{c_2(\ell+2)} && \text{(by (13)).} \end{aligned}$$

As $2^{b+1} > x_\ell \geq 2m/c_6(\ell+2)$, we may use Lemma 2.13 to show that

$$I(q_{t_{\ell+1}}, (2/3)(2m)) \geq 1 - \frac{1}{c_2(\ell+2)} - \epsilon(4m/3)t_{last} = 1 - \frac{1}{c_2(\ell+2)} - \frac{4c_6}{3c_3}$$

by (16).

If $x_\ell(v) < 2m/c_6(\ell+2)$, we compute

$$\begin{aligned} I(q_{t_{h_v}}, x_{h_v}) &\geq I(p_{t_{h_v}}, x_{h_v}) - \epsilon t_{last} x_{h_v} \quad \text{(by Lemma 2.13)} \\ &= \frac{h_v + 1}{c_5(\ell+2)} - \epsilon t_{last} x_{h_v} \\ &= \frac{h_v + 1}{c_5(\ell+2)} - \frac{x_{h_v}}{c_3(\ell+2)2^b} \quad \text{(by (16))} \\ &\geq \frac{h_v + 1}{c_5(\ell+2)} - \frac{2x_{h_v-1}}{c_3(\ell+2)2^b} \quad (\text{as } x_{h_v} \leq 2x_{h_v-1}) \\ &> \frac{h_v + 1}{c_5(\ell+2)} - \frac{2^{b+2}}{c_3(\ell+2)2^b} \quad (\text{as } x_{h_v-1} < 2^{b+1}) \\ &\geq \frac{h_v + 1/2}{c_5(\ell+2)} \quad \text{(by (4)).} \end{aligned}$$

□

LEMMA 2.15 (condition (C.4)). *Let $S \subseteq V$ be a set of vertices such that $\mu(S) \leq (2/3)\mu(V)$ and $\Phi(S) \leq f_1(\phi)$, and let v lie in S_b^g. If* `Nibble` *is run from v with parameter b, then for all $t \in (t_{h_v-1}, t_{h_v}]$, condition* (C.4) *is satisfied.*

Proof. We first consider the case in which $x_\ell < 2m/c_6(\ell+2)$, which by definition implies $x_{h_v} \leq 2x_{h_v-1}$.

In this case, we have

$$I(q_t, x_{h_v-1}) \leq I(p_t, x_{h_v-1}) \leq I(p_{t_{h_v-1}}, x_{h_v-1}) = h_v/c_5(\ell+2),$$

where the first inequality follows from Lemma 2.13 and the second follows from Lemma 2.9.

As $I_x(q_t, x)$ is nonincreasing in x and $x_{h_v-1} < x_{h_v} \leq 2x_{h_v-1}$, we have

$$\begin{aligned} I_x(q_t, x_{h_v-1}) \geq \frac{I(q_t, x_{h_v}) - I(q_t, x_{h_v-1})}{x_{h_v} - x_{h_v-1}} &\geq \frac{I(q_{t_{h_v}}, x_{h_v}) - I(q_t, x_{h_v-1})}{x_{h_v} - x_{h_v-1}} \\ &\geq \frac{1/2}{c_5(\ell+2)x_{h_v-1}}, \end{aligned}$$

where the second inequality follows from Lemma 2.9 and the definition of q_t, and the last one follows from (23).

If $x_{h_v-1} \geq 2$, then $b \geq 1$ and we have $2^b \leq x_{h_v-1} < 2^{b+1}$, and so

$$I_x(q_t, 2^b) \geq I_x(q_t, x_{h_v-1}) \geq \frac{1}{2c_5(\ell+2)2^{b+1}},$$

and by (5) condition (C.4) is satisfied.

If $x_{h_v-1} < 2$, then $b = 0$, and so $I_x(q_t, 2^b) = I_x(q_t, x)$ for all $x < 1$, which implies

$$I_x(q_t, 2^b) \geq I_x(q_t, x_{h_v-1}) \geq \frac{1}{2c_5(\ell+2)x_{h_v-1}} > \frac{1}{2c_5(\ell+2)2^b},$$

and condition (C.4) is satisfied.

If $x_\ell \geq 2m/c_6(\ell+2)$, in which case $h_v = \ell+1$ and $x_\ell < 2^{b+1}$, we apply Lemma 2.13 to show that for all $t \in (t_\ell, t_{\ell+1}]$,

$$I(q_t, 2m) \geq I(p_t, 2m) - 2m\epsilon t_{last} = 1 - \frac{2m}{c_3(\ell+2)2^b} \geq 1 - \frac{2m}{c_3(\ell+2)x_\ell/2} \geq 1 - \frac{2c_6}{c_3}.$$

On the other hand,

$$I(q_t, x_\ell) \leq I(p_t, x_\ell) \leq I(p_{t_\ell}, x_\ell) < 1/c_5.$$

As $I_x(q_t, \cdot)$ is nondecreasing and $x_\ell \geq 2^b$, we have

$$\begin{aligned} I_x(q_t, 2^b) \geq I_x(q_t, x_\ell) &\geq \frac{I(q_t, 2m) - I(q_t, x_\ell)}{2m - x_\ell} \\ &\geq \frac{1}{2m}\left(1 - \frac{2c_6}{c_3} - \frac{1}{c_5}\right) \geq \frac{1}{c_6(\ell+2)2^{b+1}}\left(1 - \frac{2c_6}{c_3} - \frac{1}{c_5}\right), \end{aligned}$$

as $2^{b+1} > x_\ell \geq 2m/c_6(\ell+2)$, and so by (6), condition (C.4) is satisfied. ∎

It remains to show that conditions (C.1)–(C.3) are met for some $t \in (t_{h_v-1}, t_{h_v}]$. We will do this by showing that if at least one of these conditions fails for every j and every $t \in (t_{h_v-1}, t_{h_v}]$, then the curve $I(q_{t_h}, \cdot)$ will be too low, in violation of Lemma 2.14.

LEMMA 2.16 (curve evolution in truncated diffusion). *If there exist a $\beta > 0$ and an $h \in [1, \ell+1]$ such that for all $t \in (t_{h-1}, t_h]$ and for all j either*

1. $\Phi(S_j(q_t)) \geq \phi$,
2. $\lambda_j(q_t) > (5/6)2m$, *or*
3. $I(q_t, \lambda_j(q_t)) < \beta$,

then, for all x, letting $\widehat{x} = \min(x, 2m - x)$,

$$I(q_{t_h}, x) < \beta + \frac{3x}{5m} + \sqrt{\widehat{x}}\left(1 - \frac{\phi^2}{2}\right)^{t_1}.$$

Proof. We will prove by induction that the conditions of the lemma imply that for all $t \in [t_{h-1}, t_h]$ and for all x,

$$I(q_t, x) < \beta + \frac{3x}{5m} + \sqrt{\widehat{x}}\left(1 - \frac{\phi^2}{2}\right)^{t - t_{h-1}}. \tag{24}$$

The base case is when $t = t_{h-1}$, in which case (24) is satisfied because of the following:

- For $1 \leq x \leq 2m-1$, $I(q_t, x) \leq I(q_t, 2m) \leq 1 \leq \sqrt{\widehat{x}}$.
- For $0 \leq x \leq 1$, we have $I(q_t, x) \leq \sqrt{\widehat{x}}$, as both are 0 at $x = 0$, the right-hand term dominates at $x = 1$, the left-hand term is linear in this region, and the right-hand term is concave.
- For $2m - 1 \leq x \leq 2m$, we note that at $x = 2m$, $I(q_t, x) = 1 < 3x/5m$, and that we already know the right-hand term dominates at $x = 2m - 1$. The inequality then follows from the fact that the left-hand term is linear in this region and the right-hand term is concave.

Let

$$f(x) \stackrel{\text{def}}{=} \sqrt{\widehat{x}}.$$

Lovász and Simonovits [LS90] observe that

$$\frac{1}{2}\left(f(x - 2\phi\widehat{x}) + f(x + 2\phi\widehat{x})\right) \leq f(x)\left(1 - \frac{\phi^2}{2}\right). \tag{25}$$

We now prove that (24) holds for t, assuming it holds for $t-1$, by considering three cases. As the right-hand side is concave and the left-hand side is piecewise-linear between points of the form $\lambda_j(q_t)$, it suffices to prove the inequality at the points $\lambda_j(q_t)$. If $x = \lambda_j(q_t)$ and $I(q_t, x) \leq \beta$, then (24) holds trivially. Similarly, if $x = \lambda_j(q_t) > (5/6)2m$, then (24) holds trivially as well, as the left-hand side is at most 1, and the right-hand side is at least 1. In the other cases, we have $\Phi(S_j(q_t)) \geq \phi$, in which case we may apply Lemma 2.10 to show that for $x = \lambda_j(q_t)$,

$$\begin{aligned}
I(q_t, x) &= 7I(Mr_{t-1}, x) && \text{(by definition)}\\
&\leq \frac{1}{2}\left(I\big(r_{t-1}, x - 2\phi\widehat{x}\big) + I\big(r_{t-1}, x + 2\phi\widehat{x}\big)\right) && \text{(by Lemma 2.10)}\\
&\leq \frac{1}{2}\left(I\big(q_{t-1}, x - 2\phi\widehat{x}\big) + I\big(q_{t-1}, x + 2\phi\widehat{x}\big)\right)\\
&< \frac{1}{2}\left[\beta + \frac{3(x - 2\phi\widehat{x})}{5m} + \sqrt{x - 2\phi\widehat{x}}\left(1 - \frac{\phi^2}{2}\right)^{t-1-t_{h-1}}\right.\\
&\qquad \left. + \beta + \frac{3(x + 2\phi\widehat{x})}{5m} + \sqrt{x + 2\phi\widehat{x}}\left(1 - \frac{\phi^2}{2}\right)^{t-1-t_{h-1}}\right] && \text{(by induction)}\\
&= \beta + \frac{3x}{5m} + \frac{1}{2}\left(\sqrt{x - 2\phi\widehat{x}} + \sqrt{x + 2\phi\widehat{x}}\right)\left(1 - \frac{\phi^2}{2}\right)^{t-1-t_{h-1}}\\
&\leq \beta + \frac{3x}{5m} + \sqrt{\widehat{x}}\left(1 - \frac{\phi^2}{2}\right)^{t-t_{h-1}}
\end{aligned}$$

by (25). □

We now observe that t_1 has been chosen to ensure

$$\sqrt{\widehat{x}}\left(1 - \frac{\phi^2}{2}\right)^{t_1} < \frac{1}{c_1(\ell + 2)}. \tag{26}$$

LEMMA 2.17 (conditions (C.1)–(C.3)). *Let S be a set of vertices such that $\mu(S) \leq (2/3)(2m)$ and $\Phi(S) \leq f_1(\phi)$, and let v lie in S_b^g. If* `Nibble` *is run from v with parameter b, then there exist a $t \in (t_{h_v-1}, t_{h_v}]$ and a j for which conditions* (C.1)–(C.3) *are satisfied.*

Proof. We first show that for $t \in (t_{h_v-1}, t_{h_v}]$, (C.3) is implied by $I(q_t, \lambda_j(q_t)) \geq h_v/c_5(\ell+2)$. To see this, note that for $b > 0$,

$$\begin{aligned} I(q_t, 2^b) &\leq I(q_t, x_{h_v-1}) && (\text{as } x_{h_v-1} \geq 2^b) \\ &\leq I(p_t, x_{h_v-1}) && (\text{by Lemma 2.13}) \\ &\leq I(p_{t_{h_v-1}}, x_{h_v-1}) && (\text{by Lemma 2.9}) \\ &= \frac{h_v}{c_5(\ell+2)}. \end{aligned}$$

So, $I(q_t, \lambda_j(q_t)) \geq h_v/c_5(\ell+2)$ implies $\lambda_j(q_t) \geq 2^b$, and we may prove the lemma by exhibiting a t and j for which (C.1), (C.2), and $I(q_t, \lambda_j(q_t)) \geq h_v/c_5(\ell+2)$ hold. On the other hand, if $b = 0$, then $\lambda_j(q_t) \geq 1 = 2^b$ for all $j \geq 1$, and so $I(q_t, \lambda_j(q_t)) \geq h_v/c_5(\ell+2)$ trivially implies $j \geq 1$ and therefore (C.3).

We will now finish the proof by contradiction: we show that if no such t and j exist, then the curve $I(q_{t_{h_v}}, \cdot)$ would be too low. If for all $t \in (t_{h_v-1}, t_{h_v}]$ and for all j one of (C.1), (C.2), or $I(q_t, \lambda_j(q_t)) \geq h_v/c_5(\ell+2)$ fails, then Lemma 2.16 tells us that for all x,

$$I(q_{t_{h_v}}, x) \leq \frac{h_v}{c_5(\ell+2)} + \frac{3x}{5m} + \sqrt{\widehat{x}}\left(1 - \frac{\phi^2}{2}\right)^{t_1} \leq \frac{h_v}{c_5(\ell+2)} + \frac{3x}{5m} + \frac{1}{c_1(\ell+2)}$$

by inequality (26).

In the case $x_\ell < 2m/c_6(\ell+2)$, we obtain a contradiction by plugging in $x = x_{h_v}$ to find

$$I(q_{t_{h_v}}, x_{h_v}) < \frac{1}{\ell+2}\left(\frac{h_v}{c_5} + \frac{6}{5c_6} + \frac{1}{c_1}\right),$$

which by (7) contradicts (23).

In the case in which $x_\ell \geq 2m/c_6(\ell+2)$, and so $h_v = \ell+1$, we substitute $x = (2/3)2m$ to obtain

$$I(q_{t_{\ell+1}}, (2/3)(2m)) < \frac{4}{5} + \frac{1}{c_5} + \frac{1}{c_1(\ell+2)},$$

which by (8) contradicts (22). ∎

2.4. Proof of Theorem 2.1. Property (N.1) of Theorem 2.1 follows from conditions (C.1) and (C.2) in the algorithm. Given a set S satisfying $\mu(S) \leq (2/3)\mu(V)$, the lower bound on the volume of the set S^g is established in Lemma 2.7. If $\Phi(S) \leq f_1(\phi)$ and $v \in S_b^g$, then Lemmas 2.17 and 2.15 show that the algorithm will output a nonempty set. Finally, Lemma 2.8 tells us that if $\Phi(S) \leq f_1(\phi)$, $v \in S^g$, and the algorithm outputs a nonempty set C, then it satisfies $\mu(C \cap S) \geq 2^{b-1}$.

It remains to bound the running time of `Nibble`. The algorithm will run for t_{last} iterations. We will now show that with the correct implementation, each iteration takes time $O((\log n)/\epsilon)$. Instead of performing a dense vector multiplication in step 3(a), the algorithm should keep track of the set of vertices u at which $r_t(u) > 0$. Call this set V_t. The set V_t can be computed in time $O(|V_t|)$ in step 3(b). Given knowledge of V_{t-1}, the multiplication in step 3(a) can be performed in time proportional to

$$\mu(V_{t-1}) = \sum_{u \in V_{t-1}} d(u) \leq \sum_{u \in V_{t-1}} r_t(u)/\epsilon \leq 1/\epsilon.$$

Finally, the computation in step 3(c) might require sorting the vectors in V_t according to r_t, which could take time at most $O(|V_t| \log n)$. Thus, the runtime of `Nibble` is bounded by

$$O\left(t_{last}\frac{\log n}{\epsilon}\right) = O\left(t_{last}^2 2^b \log^2 m\right) = O\left(\frac{2^b \log^6 m}{\phi^4}\right).$$

3. Nearly linear time graph partitioning. In this section, we apply `Nibble` to design a partitioning algorithm `Partition`. This new algorithm runs in nearly linear time. It computes an approximate sparsest cut with approximately optimal balance. In particular, we prove that there exists a constant $\alpha > 0$ such that for any graph $G = (V, E)$ that has a cut S of sparsity $\alpha \cdot \theta^2 / \log^3 n$ and balance $b \leq 1/2$, with high probability, `Partition` finds a cut D with $\Phi_V(D) \leq \theta$ and $\mathbf{bal}(D) \geq b/2$. Actually, `Partition` satisfies an even stronger guarantee: with high probability either the cut it outputs is well balanced,

$$\frac{1}{4}\mu(V) \leq \mu(D) \leq \frac{3}{4}\mu(V),$$

or touches most of the edges touching S,

$$\mu(D \cap S) \geq \frac{1}{2}\mu(S).$$

The expected running time of `Partition` is $O(m \log^7 n/\phi^4)$. Thus, it can be used to quickly find crude cuts.

`Partition` calls `Nibble` via a routine called `Random Nibble` that calls `Nibble` with carefully chosen random parameters. `Random Nibble` has a very small expected running time and is expected to remove a similarly small fraction of any set with small conductance.

3.1. Procedure `Random Nibble`.

$C =$ `RandomNibble`(G, ϕ)

(1) Choose a vertex v according to ψ_V.

(2) Choose a b in $1, \ldots, \lceil \log m \rceil$ according to

$$\Pr[b = i] = 2^{-i}/(1 - 2^{-\lceil \log m \rceil}).$$

(3) $C =$ `Nibble`(G, v, ϕ, b).

LEMMA 3.1 (`Random Nibble`). *Let m be the number of edges in G. The expected running time of* `Random Nibble` *is $O\left(\log^7 m/\phi^4\right)$. If the set C output by* `Random Nibble` *is nonempty, it satisfies*

(R.1) $\Phi_V(C) \leq \phi$, *and*

(R.2) $\mu(C) \leq (5/6)\mu(V)$.

Moreover, for every set S satisfying

$$\mu(S) \leq (2/3)\mu(V) \quad \text{and} \quad \Phi_V(S) \leq f_1(\phi),$$

(R.3) $\mathbf{E}[\mu(C \cap S)] \geq \mu(S)/4\mu(V)$.

Proof. The expected running time of `Random Nibble` may be upper bounded by

$$O\left(\sum_{i=1}^{\lceil \log m \rceil}\left(2^{-i}/(1-2^{\lceil \log m \rceil})\right)\left(2^i \log^6(m)/\phi^4\right)\right) = O\left(\log^7(m)/\phi^4\right).$$

Properties (R.1) and (R.2) follow directly from property (N.1) of Theorem 2.1. To prove part (R.3), define α_b by

$$\alpha_b = \frac{\mu\left(S_b^g\right)}{\mu\left(S^g\right)}.$$

So, $\sum_b \alpha_b = 1$. For each i, the chance that v lands in S_i^g is $\alpha_i \mu\left(S^g\right)/\mu\left(V\right)$. Moreover, the chance that $b = i$ is at least 2^{-i}. If v lands in S_i^g, then by property (N.3) of Theorem 2.1, C satisfies

$$\mu\left(C \cap S\right) \geq 2^{i-1}.$$

So,

$$\begin{aligned}
\mathbf{E}\left[\mu\left(C \cap S\right)\right] &\geq \sum_i 2^{-i}\alpha_i\left(\mu\left(S^g\right)/\mu\left(V\right)\right)2^{i-1} \\
&= \sum_i (1/2)\alpha_i\left(\mu\left(S^g\right)/\mu\left(V\right)\right) \\
&= \mu\left(S^g\right)/2\mu\left(V\right) \\
&\geq \mu\left(S\right)/4\mu\left(V\right). \quad \text{(by property (N.2.a) of Theorem 2.1).}
\end{aligned}$$

□

3.2. `Partition`**.** Our graph partitioning algorithm will exploit two properties of `Random Nibble` in addition to the fact that it outputs clusters of low conductance. The first is that `Random Nibble` is very fast—its expected running time is polylogarithmic. The second is that the set returned by `Random Nibble` is expected to touch at least a $1/8m$ fraction of every set S of sufficiently low conductance. While this fraction is very small, `Random Nibble` is so fast that we can call it $O(m)$ times, and this is what we do. The algorithm `Partition` simply calls `Random Nibble` $O(m)$ times, each time removing the set returned from the graph. The union of the sets removed forms the output of `Partition`.

We now define `Partition` and analyze its performance. First, define

$$f_2(\theta) \stackrel{\text{def}}{=} f_1(\theta/7)/2, \tag{27}$$

and note that

$$f_2(\theta) \geq \Omega\left(\frac{\theta^2}{\log^3 m}\right).$$

$D =$ `Partition`(G, θ, p), where G is a graph, $\theta, p \in (0,1)$.
(0) Set $W_0 = V$, $j = 0$, and $\phi = \theta/7$.
(1) While $j < 12m\left\lceil \lg(1/p)\right\rceil$ and $\mu\left(W_j\right) \geq (3/4)\mu\left(V\right)$,
  (a) Set $j = j + 1$.
  (b) Set $D_j =$ `RandomNibble`$(G[W_{j-1}], \phi)$.
  (c) Set $W_j = W_{j-1} - D_j$.
(2) Set $D = D_1 \cup \cdots \cup D_j$.

THEOREM 3.2 (`Partition`). *The expected running time of* `Partition` *on a graph with m edges is $O\left(m \lg(1/p) \log^7 m/\theta^4\right)$. Let D be the output of* `Partition`(G, θ, p)*, where G is a graph and $\theta, p \in (0,1)$. Then the following hold:*

(P.1) $\mu(D) \leq (7/8)\mu(V)$.

(P.2) *If $D \neq \emptyset$, then $\Phi_V(D) \leq \theta$.*

(P.3) *If S is any set satisfying*

$$\mu(S) \leq \mu(V)/2 \quad \text{and} \quad \Phi_V(S) \leq f_2(\theta), \tag{28}$$

then with probability at least $1-p$, $\mu(D) \geq \mu(S)/2$.

In particular, with probability at least $1-p$ either

(P.3.a) $\mu(D) \geq (1/4)\mu(V)$, *or*

(P.3.b) $\mu(S \cap D) \geq \mu(S)/2$.

Property (P.3) is a little unusual and deserves some explanation. It says that for every set S of low conductance, with high probability either D is a large fraction of S, or it is a large fraction of the entire graph. While we would like to pick just one of these properties and guarantee that it holds with high probability, this would be unreasonable: on one hand, there might be no big set D of small conductance; and, on the other hand, even if S is small, the algorithm might cut out a large set D that completely avoids S.

Proof of Theorem 3.2. The bound on the expected running time of `Partition` is immediate from the bound on the running time of `Random Nibble`.

Let j_{out} be the iteration at which `Partition` stops, so that $D = D_1 \cup \cdots \cup D_{j_{out}}$. To prove (P.1), note that $\mu(W_{j_{out}-1}) \geq (3/4)\mu(V)$, and so $\mu(D_1 \cup \cdots \cup D_{j_{out}-1}) \leq (1/4)\mu(V)$. By property (R.2) of Lemma 3.1, $\mu(D_{j_{out}}) \leq (5/6)\mu(W_{j_{out}-1})$. So,

$$\begin{aligned}\mu(D_1 \cup \cdots \cup D_{j_{out}}) &\leq \mu(V) - \mu(W_{j_{out}-1}) + \frac{5}{6}\mu(W_{j_{out}-1}) \\ &= \mu(V) - \frac{1}{6}\mu(W_{j_{out}-1}) \leq \frac{7}{8}\mu(V).\end{aligned}$$

To establish (P.2), we first compute

$$\begin{aligned}|E(D, V-D)| &= \sum_{i=1}^{j_{out}} |E(D_i, V-D)| \\ &\leq \sum_{i=1}^{j_{out}} |E(D_i, W_{i-1} - D_i)| \\ &\leq \sum_{i=1}^{j_{out}} \phi\mu(D_i) \quad \text{(by (R.1))} \\ &= \phi\mu(D).\end{aligned}$$

So, if $\mu(D) \leq \mu(V)/2$, then $\Phi_V(D) \leq \phi$. On the other hand, we established above that $\mu(D) \leq (7/8)\mu(V)$, from which it follows that

$$\mu(V-D) \geq (1/8)\mu(V) \geq (8/7)(1/8)\mu(D) = (1/7)\mu(D).$$

So,

$$\Phi_V(D) = \frac{|E(D, V-D)|}{\min(\mu(D), \mu(V-D))} \leq 7\frac{|E(D, V-D)|}{\mu(D)} \leq 7\phi = \theta.$$

Let $j_{max} = 12m\lceil \lg(1/p) \rceil$. To prove property (P.3), let S satisfy (28), and consider what happens if we ignore the second condition in the while loop and run `Partition` for all potential j_{max} iterations, obtaining cuts $D_1, \ldots, D_{12m\lceil \lg(1/p) \rceil}$. Let

$$D^{\leq j} = \bigcup_{i \leq j} D_i.$$

We will prove that if neither

$$\mu\left(D^{\leq k}\right) \geq \frac{\mu(V)}{4} \quad \text{nor} \quad \mu\left(S \cap D^{\leq k}\right) \geq \frac{\mu(S)}{2}$$

holds at iteration k, then with probability at least $1/2$, one of these conditions will be satisfied by iteration $k + 12m$. Thus, after all j_{max} iterations, one of properties (P.3.a) or (P.3.b) will be satisfied with probability at least $1 - p$. If the algorithm runs for fewer iterations, then condition (P.3.a) is satisfied.

To simplify notation, let $C_i = D_{k+i}$ and $U_i = W_{k+i}$ for $0 \leq i \leq 12m$. Assume that

$$\mu(U_0) \geq \frac{3}{4}\mu(V) \quad \text{and} \quad \mu(S \cap U_0) < \frac{1}{2}\mu(S).$$

For $1 \leq i \leq 12m$, define the random variable

$$X_i = \frac{\mu(C_i \cap S)}{\mu(U_0 \cap S)}.$$

As each set C_i is a subset of U_0, and the C_i are mutually disjoint, we will always have

$$\sum_{i=1}^{12m} X_i \leq 1.$$

Define β to satisfy

$$(1 - \beta)\mu(S \cap U_0) = \frac{1}{2}\mu(S),$$

and note that this ensures that $0 < \beta \leq 1/2$. Moreover, if $\sum X_i \geq \beta$, then $\mu\left(S \cap D^{\leq k+12m}\right) \geq \mu(S)/2$ will hold.

Let E_j be the event

$$\mu(U_j) < \frac{3}{4}\mu(V).$$

We need to show that, with probability at least $1/2$, either an event E_j holds, or $\sum X_i \geq \beta$. To this end, we now show that if neither E_j nor $\sum_{i \leq j} X_i \geq \beta$ holds, then $\mathbf{E}[X_{j+1}] \geq 1/8m$. If $\sum_{i \leq j} X_i < \beta$, then

$$\begin{aligned}\mu(S \cap U_j) = \mu(S \cap U_0) - \sum_{i \leq j} \mu(S \cap C_i) &= \mu(S \cap U_0)\left(1 - \sum_{i \leq j} X_i\right) \\ &> \mu(S \cap U_0)(1 - \beta) = \frac{1}{2}\mu(S).\end{aligned}$$

If E_j does not hold, then

$$\mu(U_j - S \cap U_j) = \mu(U_j) - \mu(S \cap U_j) \geq \frac{3}{4}\mu(V) - \mu(S) \geq \frac{1}{4}\mu(V) \geq \frac{1}{2}\mu(S).$$

So,

$$\begin{aligned}\Phi^G_{U_j}(S \cap U_j) &= \frac{|E(S \cap U_j, U_j - S \cap U_j)|}{\min(\mu(S \cap U_j), \mu(U_j - S \cap U_j))} \\ &\leq \frac{|E(S, V - S)|}{(1/2)\mu(S)} \leq 2\Phi_G(S) \leq 2f_2(\theta) = f_1(\phi).\end{aligned}$$

We also have $\mu(S \cap U_j) \leq \mu(S) \leq (2/3)\mu(U_j)$, so the conditions of property (R.3) of Lemma 3.1 are satisfied and

$$\mathbf{E}[X_{j+1}] \geq 1/4\mu(U_j) \geq 1/8m.$$

Now, set

$$Y_j = \begin{cases} 1/8m & \text{if } \sum_{i<j} X_i \geq \beta, \text{ or if } E_{j-1}, \\ X_j & \text{otherwise.} \end{cases}$$

So, for all j, we have $\mathbf{E}[Y_j] \geq 1/8m$, and so $\mathbf{E}[\sum_{j \leq 12m} Y_j] \geq 3/2$. On the other hand,

$$\sum_{i \leq 12m} Y_j \leq \sum X_i + \frac{12m}{8m} \leq 5/2.$$

So, with probability at least $1/2$,

$$\sum_{j \leq 12m} Y_j \geq 1/2 \geq \beta.$$

This implies that with probability at least $1/2$ either $\sum_i X_i \geq \beta$, or some event E_j holds, which is what we needed to show. ∎

4. Final remarks.

4.1. An open question. The main technical question left open by this work is how quickly one can find a cluster of low conductance given a random vertex in such a cluster. Andersen, Chung, and Lang [ACL06] and Andersen and Peres [AP09] have greatly reduced the polylogarithmic dependence on the size of the graph and the dependence on the conductance of the cluster. We ask whether these factors can be eliminated entirely. That is, does there exist an algorithm that when given a random node in a cluster of low conductance probably finds a cluster of similarly low conductance in time that depends only on the volume of the cluster it discovers?

Acknowledgment. We would like to thank the anonymous referees for their helpful suggestions.

REFERENCES

[ABC+07] R. Andersen, C. Borgs, J. T. Chayes, J. E. Hopcraft, V. S. Mirrokni, and S.-H. Teng, *Local computation of PageRank contributions*, in Algorithms and Models for the Web-Graph, Lecture Notes in Comput. Sci. 4863, A. Bonato and F. R. K. Chung, eds., Springer, Berlin, 2007, pp. 150–165.

[ACL06] R. Andersen, F. Chung, and K. Lang, *Local graph partitioning using PageRank vectors*, in Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS '06), 2006, pp. 475–486.

[ACL07] R. Andersen, F. R. K. Chung, and K. J. Lang, *Local partitioning for directed graphs using PageRank*, in Algorithms and Models for the Web-Graph, Lecture Notes in Comput. Sci. 4863, A. Bonato and F. R. K. Chung, eds., Springer, Berlin, 2007, pp. 166–178.

[AHK04] S. Arora, E. Hazan, and S. Kale, *$O(\sqrt{\log n})$ approximation to Sparsest Cut in $\tilde{O}(n^2)$ time*, SIAM J. Comput., 39 (2010), pp. 1748–1771.

[AK07] S. Arora and S. Kale, *A combinatorial, primal-dual approach to semidefinite programs*, in Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC '07), 2007, pp. 227–236.

[And08] R. Andersen, *A local algorithm for finding dense subgraphs*, in Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '08), 2008, pp. 1003–1009.

[AP09] R. Andersen and Y. Peres, *Finding sparse cuts locally using evolving sets*, in Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09), 2009, pp. 235–244.

[ARV04] S. Arora, S. Rao, and U. Vazirani, *Expander flows, geometric embeddings and graph partitioning*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC '04), 2004, pp. 222–231.

[Che70] J. Cheeger, *A lower bound for smallest eigenvalue of the Laplacian*, in Problems in Analysis, Princeton University Press, Princeton, NJ, 1970, pp. 195–199.

[Cor] Intel Corporation, *Moore's Law*, ftp://download.intel.com/museum/Moores_Law/Printed_Materials/Moores_Law_Poster_Ltr.pdf.

[Fac10] Facebook, *Statistics*, http://www.facebook.com/press/info.php?statistics (August, 2010).

[Goo08] Google, *We Knew the Web Was Big...*, http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html (2008).

[HK92] L. Hagen and A. B. Kahng, *New spectral methods for ratio cut partitioning and clustering*, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., 11 (1992), pp. 1074–1085.

[KRV06] R. Khandekar, S. Rao, and U. Vazirani, *Graph partitioning using single commodity flows*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06), 2006, pp. 385–390.

[KVV04] R. Kannan, S. Vempala, and A. Vetta, *On clusterings: Good, bad and spectral*, J. ACM, 51 (2004), pp. 497–515.

[LH08] J. Leskovec and E. Horvitz, *Planetary-scale views on a large instant-messaging network*, in Proceeding of the 17th ACM International Conference on World Wide Web (WWW '08), 2008, pp. 915–924.

[LR99] T. Leighton and S. Rao, *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*, J. ACM, 46 (1999), pp. 787–832.

[LS90] L. Lovász and M. Simonovits, *The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume*, in Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, Vol. 1, St. Louis, MO, 1990, pp. 346–354.

[LS93] L. Lovász and M. Simonovits, *Random walks in a convex body and an improved volume algorithm*, Random Structures Algorithms, 4 (1993), pp. 359–412.

[Mih89] M. Mihail, *Conductance and convergence of Markov chains—A combinatorial treatment of expanders*, in Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS '89), 1989, pp. 526–531.

[NSW+09] K. Nomura, R. Seymour, W. Wang, H. Dursun, R. K. Kalia, A. Nakano, P. Vashishta, F. Shimojo, and L. H. Yang, *A metascalable computing framework for large spatiotemporal-scale atomistic simulations*, in Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing (IPDPS '09), 2009, pp. 1–10.

[OSVV08] L. Orecchia, L. J. Schulman, U. V. Vazirani, and N. K. Vishnoi, *On partitioning graphs via single commodity flows*, in Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08), 2008, pp. 461–470.

[OV11] L. Orecchia and N. K. Vishnoi, *Towards an SDP-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition*, in Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete

Algorithms, 2011, pp. 532–545.

[PBMW98] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, Technical report, Stanford Digital Library Technologies Project, Stanford University, Stanford, CA, 1998.

[She09] J. Sherman, *Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cut*, in Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '09), 2009, pp. 363–372.

[SS06] J. Síma and S. E. Schaeffer, *On the NP-completeness of some graph cluster measures*, in SOFSEM 2006: Theory and Practice of Computer Science, Lecture Notes in Comput. Sci. 3831, J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková, and J. Stuller, eds., Springer, Berlin, 2006, pp. 530–537.

[ST96] D. A. Spielman and S.-H. Teng, *Spectral partitioning works: Planar graphs and finite element meshes*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS '96), 1996, pp. 96–105.

[ST04] D. A. Spielman and S.-H. Teng, *Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, 2004, pp. 81–90; full version available from http://arxiv.org/abs/cs.DS/0310051.

[ST08] D. A. Spielman and S.-H. Teng, *Nearly-Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems*, http://www.arxiv.org/abs/cs.NA/0607105 (2008), SIAM J. Matrix Anal. Appl., submitted.

[ST11] D. A. Spielman and S.-H. Teng, *Spectral sparsification of graphs*, SIAM J. Comput., 40 (2011), pp. 981–1025.

[VTX09] V. Konstantin, S.-H. Teng, and Y. Xia, *Finding local communities in protein networks*, BMC Bioinformatics, 10 (2009), 297.

[Wik] Wikipedia, *Transistor Count*, http://en.wikipedia.org/wiki/Transistor_count (August, 2010).