*Sampling Subproblems of Heterogeneous Max-Cut Problems and Approximation Algorithms**

Petros Drineas,[1] Ravi Kannan,[2,†] Michael W. Mahoney[3]
[1] *Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York 12180; e-mail: drinep@cs.rpi.edu*
[2] *Department of Computer Science, Yale University, New Haven, Connecticut 06520; e-mail: kannan@cs.yale.edu*
[3] *Department of Mathematics, Yale University, New Haven, Connecticut 06520; e-mail: mahoney@cs.yale.edu*

Received 16 November 2005; revised in final form 25 July 2006; accepted 20 November 2006
Published online 4 September 2007 in Wiley InterScience (www.interscience.wiley.com).
DOI 10.1002/rsa.20196

ABSTRACT: Recent work in the analysis of randomized approximation algorithms for *NP*-hard optimization problems has involved approximating the solution to a problem by the solution of a related subproblem of constant size, where the subproblem is constructed by sampling elements of the original problem uniformly at random. In light of interest in problems with a heterogeneous structure, for which uniform sampling might be expected to yield suboptimal results, we investigate the use of nonuniform sampling probabilities. We develop and analyze an algorithm which uses a novel sampling method to obtain improved bounds for approximating the Max-Cut of a graph. In particular, we show that by judicious choice of sampling probabilities one can obtain error bounds that are superior to the ones obtained by uniform sampling, both for unweighted and weighted versions of Max-Cut. Of at least as much interest as the results we derive are the techniques we use. The first technique is a method to compute a compressed approximate decomposition of a matrix as the product

Correspondence to: Michael W. Mahoney
*A preliminary version of this paper with the same title appeared in the Proceedings of the 22nd Annual International Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 3404, pp. 57–68, 2005 [13].
†Supported in part by the NSF.
© 2007 Wiley Periodicals, Inc.

WILEY InterScience® DISCOVER SOMETHING GREAT

of three smaller matrices, each of which has several appealing properties. The second technique is a method to approximate the feasibility or infeasibility of a large linear program by checking the feasibility or infeasibility of a nonuniformly randomly chosen subprogram of the original linear program. We expect that these and related techniques will prove fruitful for the future development of randomized approximation algorithms for problems whose input instances contain heterogeneities.
© 2007 Wiley Periodicals, Inc. Random Struct. Alg., 32, 307–333, 2008

Keywords: Max-Cut algorithm; CUR matrix decomposition; Sampling Linear Programs

1. INTRODUCTION

1.1. Background

We are interested in developing improved methods to compute approximate solutions to certain *NP*-hard optimization problems that arise in applications of graph theory and that have significant heterogeneities and nonuniformities. The methods we present here are a first step in that direction. They make use of random sampling according to certain judiciously chosen nonuniform probability distributions and they depend heavily on our recent work on designing and analyzing fast Monte-Carlo algorithms for performing useful computations on large matrices [14–16].

As an application of these methods, we design and analyze an algorithm to compute an approximation for the Max-Cut problem. In a Max-Cut problem, also known as a maximum weight cut problem, the input consists of the $n \times n$ adjacency matrix A of an undirected graph $G = (V, E)$ with n vertices, and the objective of the problem is to find a cut, i.e., a partition of the vertices into two subsets V_1 and V_2, such that the number of edges of E that have one endpoint in V_1 and one endpoint in V_2 is maximized. In its weighted version, the input consists of an $n \times n$ weighted adjacency matrix A, and the objective is to find a cut such that the sum of the weights of the edges of E that have one endpoint in V_1 and one endpoint in V_2 is maximized. The Max-Cut problem has applications in such diverse fields as statistical physics and circuit layout design [7], and it has been extensively studied theoretically [20, 31].

The Max-Cut problem is known to be *NP*-hard, both for general graphs and when restricted to dense graphs [3], where a graph on n vertices is dense if it contains $\Omega(n^2)$ edges. Thus, much effort has gone into designing and analyzing approximation algorithms for the Max-Cut problem. It is known that there exists a 0.878-approximation algorithm [20]. Håstad's classic inapproximability results [23], following the PCP results of Arora et al., [4] proved that (unless $P = NP$) there exists a constant α, bounded away from 1 such that there does not exist a polynomial time α-approximation algorithm. In particular, this means that there does not exist a polynomial time approximation scheme (PTAS) for the general Max-Cut problem. (A PTAS is an algorithm that for every fixed $\epsilon > 0$ achieves an approximation ratio of $1 - \epsilon$ in time which is poly(n); such a scheme is a fully polynomial time approximation scheme (FPTAS) if the running time is poly$(n, 1/\epsilon)$.) Very recently, results by Khot et al. [27] argue that approximating Max-Cut with a better than 0.878-approximation algorithm is hard, unless a conjecture (introduced by Khot in [26] and called the "Unique Games Conjecture") is false. Finally, we should mention that Max-Cut belongs to the class of Max-2-CSP problems, where the input consists of a set E of m distinct Boolean functions $f_1, f_2, \ldots, f_m$ of n Boolean variables $x_1, x_2, \ldots, x_n$, where each f_i is a function of only 2 of the n

variables, and where the objective of the problem is to maximize the number of functions that are satisfied by an assignment to the n variables. This Max-2-CSP class of problems is a subclass of the more general Max-CSP class of problems; see [1, 25]. Numerous other graph and combinatorial problems are contained in the Max-2-CSP class of problems.

Work originating with [3] has focused on designing PTASs for a large class of *NP*-hard optimization problems, such as the Max-Cut problem, when the problem instances are dense [1, 3, 8, 17, 18, 21]. [3] and [8], using quite different methods, designed approximation algorithms for Max-Cut (and other problems) that achieve an additive error of ϵn^2 (where $\epsilon > 0$, $\epsilon \in \Omega(1)$ is an error parameter) in a time poly(n) (and exponential in $1/\epsilon$); this implies relative error for dense instances of these problems. In [21] it was shown that a constant-sized (with respect to n) sample of a graph is sufficient to determine whether a graph has a cut close to a certain value. This work investigated dense instances of *NP*-hard problems from the viewpoint of query complexity and property testing and yielded an $O(1/\epsilon^5)$ time algorithm to approximate, among other problems, dense instances of Max-Cut. [17] and [18] examined the regularity properties of dense graphs and developed a new method to approximate matrices with respect to the cut-norm. This led to a PTAS for dense instances of all Max-2-CSP, and more generally for dense instances of all Max-CSP, problems. [1] extended this and developed a PTAS for dense instances of all Max-CSP problems in which the sample complexity was poly($1/\epsilon$) and independent of n; when applied to the Max-Cut problem this led to an $O\left(\frac{\log 1/\epsilon}{\epsilon^4}\right)$ time approximation algorithm.

In all these cases, these approximation algorithms involve sampling elements of the input uniformly at random in order to construct a subproblem which is then used to compute an approximation to the original problem with additive error at most ϵn^2 [1, 3, 8, 17, 18, 21]; this then translates into a relative error bound for dense graphs. These methods are not useful for nondense graphs since with such an error bound a trivial approximate solution would always suffice. This uniform sampling does have the advantage that it can be carried out "blindly" since the "coins" can be tossed before seeing the data; then, given either random access or one pass, i.e., one sequential read, through the data, samples from the data may be drawn and then used to compute. Such uniform sampling is appropriate for problems that have nice uniformity or regularity properties [17].

In many applications of graph theory problems, however, significant heterogeneities are present [30]. For instance, the graph may have a power-law structure, or a large part of the graph may be very sparse and a small subset of vertices (sometimes, but not always a $o(n)$-sized subset) may have most of the edges ($1 - o(1)$ of the edges) incident to them. Similarly, in a weighted graph, the total weight of edges incident to most vertices may be small, while among the remaining vertices the total weight of incident edges may be quite large. Neither the adjacency matrix nor the adjacency list representation of a graph used in property testing capture well this phenomenon [21].

With the additional flexibility of several passes over the data, we may use one pass to assess the "importance" of a piece (or set of pieces) of data and determine the probability with which it (or they) should be sampled, and a second pass to actually draw the sample. Such importance sampling has a long history [29]. In recent work, we have shown that by sampling columns and/or rows of a matrix according to a judiciously-chosen and data-dependent nonuniform probability distribution, we may obtain better (relative to uniform sampling) bounds for approximation algorithms for a variety of common

matrix operations [14–16]; see also [10–12]. The power of using information to construct nonuniform sampling probabilities has also been demonstrated in recent work examining so-called oblivious versus so-called adaptive sampling [5, 6]. For instance, it was demonstrated that in certain cases approximation algorithms (for matrix problems such as those discussed in [14–16]) which use oblivious uniform sampling cannot achieve the error bounds that are achieved by adaptively constructing nonuniform sampling probabilities [5, 6].

1.2. Summary of Main Result

In this article, we apply these methods [14–16] to develop an approximation algorithm for both unweighted and weighted versions of the Max-Cut problem. We do so by using nonuniform sampling probabilities in the construction of the sub-problem to be solved. For unweighted graphs, we show that at the cost of substantial additional sampling, these methods lead to an additive error improvement over previous results [1, 21]; for weighted graphs, these methods lead to substantial improvements when the average edge weight is much less than the maximum edge weight. We view our new results as a proof of principle and expect that further work will lead to substantial additional improvement when these methods are applied to problems with a heterogeneous and/or nonuniform structure.

Let A be the $n \times n$ weighted adjacency matrix of a graph $G = (V, E)$, let ϵ be a constant independent of n, and recall that $\|A\|_F^2 = \sum_{ij} A_{ij}^2$. A main result of this article, which is presented in a more precise form in Theorem 4, is that there exists an algorithm that, upon being input A, returns an approximation Z to the Max-Cut of A such that with high probability

$$|Z - \textbf{MAX-CUT}[A]| \leq \epsilon n \|A\|_F. \tag{1}$$

The algorithm makes three passes, i.e., three sequential reads, through the matrix A and then needs constant additional space and constant additional time (constant, that is, with respect to n) in order to compute the approximation. The algorithm uses a judiciously-chosen and data-dependent nonuniform probability distribution in order to obtain bounds of the form (1); these probabilities are computed in the first two passes through the matrix.

For a complete graph $\|A\|_F^2 = n(n-1)$ since $A_{ij} = 1$ for every $i \neq j$. For general unweighted graphs $\sqrt{2|E|} = \|A\|_F < n$, where $|E|$ the cardinality of the edge set. Thus, in general, the additive error bound (1) becomes $\epsilon n\sqrt{2|E|}$, which is an improvement over the previous results of ϵn^2 [1, 21]. In addition, from this bound we obtain a PTAS for graphs with $|E| = \Omega(n^2)$. Unfortunately, this does not translate into a PTAS for any class of subdense graphs. Demonstrating that such a PTAS exists would be significant application of our methodology and is the object of current work. It has been shown by other methods that there does exist a PTAS for Max-Cut and other Max-2-CSP problems restricted to slightly subdense, i.e., $\Omega(n^2/\log n)$ edges, graphs [9]. Also, more recently, Alon and Naor [2] gave an approximation algorithm for the cut-norm, which yields a PTAS for Max-Cut on graphs with $\Omega(n^\alpha)$ edges with α slightly less than 2. Since we are primarily interested in presenting a methodology to deal with heterogeneities and nonuniformities that arise in applications of graph theory problems, we make no effort to optimize constants or polynomial factors. In particular, although we have a PTAS, both the sampling complexity and the running time of the algorithm are exponential in $1/\epsilon$,

which is substantially larger than previous results [1, 21]; we expect that this may be improved.

Our results are of particular interest for weighted graphs. Note that for weighted problems, the ϵn^2 error bound of previous work for unweighted graphs extends easily to $\epsilon n^2 W_{\max}$, where $W_{\max}$ is the maximum edge weight. For these problems, $\|A\|_F/n$ may be thought of as the average weight over all the edges; one may then view our error bounds as replacing $W_{\max}$ in the $\epsilon n^2 W_{\max}$ error bound by $\|A\|_F/n$. If only a few of the weights are much higher than this average value, the bound of $\epsilon n\|A\|_F$ given in (1) is much better than the bound of $\epsilon n^2 W_{\max}$.

1.3. Summary of Intermediate Results

To prove Theorem 4 we use two intermediate results that are of independent interest. The first intermediate result has been presented in much more generality previously [16] and is thus only summarized in Section 2. It involves a new compressed approximate decomposition of a large matrix $A \in \mathbb{R}^{m\times n}$ as the product of three smaller matrices $A' = C\tilde{U}R$, where C is an $m \times c$ matrix consisting of c randomly picked columns of A, R is an $r \times n$ matrix consisting of r randomly picked rows of A, and $\tilde{U}$ is a $c \times r$ matrix computed from C and R. The sampling probabilities are crucial features of the algorithm; if they are chosen carefully then by sampling $s = O(1/\epsilon^8)$ columns and rows of A we have from Theorem 1 that with high probability $\|A - C\tilde{U}R\|_2 \le \epsilon\|A\|_F$. The approximation can be constructed after making three passes through the whole matrix A and the matrix $\tilde{U}$ can be constructed using additional RAM space and time that is constant.

The second intermediate result relates the feasibility or infeasibility of a given Linear Program (LP) on n variables to the feasibility or infeasibility of an induced sub-LP involving only the variables in Q, where Q is a (small) randomly picked subset of the n variables. In particular, if $P \in \mathbb{R}^{r\times n}$ and $b \in \mathbb{R}^r$, we consider a set of constraints of the form $Px \le b$, $0 \le x_i \le c_i$, where $x \in \mathbb{R}^n$. In Theorem 2 and Theorem 3 we show that (i) if the n-variable LP is feasible, then the LP induced on Q is approximately feasible in the sense that a slight relaxation of it is feasible and (ii) if the n-variable LP is infeasible, then the LP induced on Q is approximately infeasible in the sense that a slight tightening of it is infeasible. A similar result using uniform sampling appeared in [1] for the case when $x_i \in [0, 1]$. The current result uses nonuniform sampling probabilities, is tighter, and applies to general bounds on the variables; the proof of this lemma is a nontrivial extension of the previous result of [1] and makes critical use of previous work on approximating the product of matrices by nonuniformly sampling a small number of columns and rows of the matrices [14].

1.4. Outline of the Paper

In Section 2, we provide a review of relevant linear algebra and of our first intermediate result which is the approximate $C\tilde{U}R$ decomposition results from [16] that will be needed for the proofs of our results. In Section 3, we then present our second intermediate result that deals with approximating the feasibility of a LP by considering random sub-programs of the LP. Then, in Section 4 we present and analyze an algorithm to approximate the Max-Cut of a matrix; in particular, we prove Theorem 4 which establishes (1). Finally, in Section 5 we provide a brief conclusion.

2. REVIEW OF RELEVANT BACKGROUND

This section contains a review of linear algebra that will be useful throughout the article; for more detail, see [22, 24, 36] and references therein. This section also contains a review of the compressed approximate $C\tilde{U}R$ decomposition of a matrix. The $C\tilde{U}R$ result is presented in much more generality in [16] and depends critically on related work on computing an approximation to the Singular Value Decomposition (SVD) of a matrix and on computing an approximation to the product of two matrices; see [14–16] for more details. All of the results of [14–16], and thus of the present paper, may be formulated within the framework of the Pass-Efficient computational model in which the three scarce computational resources are number of passes over the data and the additional RAM space and additional time required by the algorithm; see [12, 14]. This model of data-streaming computation may be viewed in terms of sublinear models of computation; see [12, 14, 28] and [15, 19].

2.1. Review of Linear Algebra

For a vector $x \in \mathbb{R}^n$ we let x_i, $i = 1, \ldots, n$, denote the i-th element of x and we let $|x| = (\sum_{i=1}^n |x_i|^2)^{1/2}$. For a matrix $A \in \mathbb{R}^{m \times n}$ we let $A^{(j)}, j = 1, \ldots, n$, denote the j-th column of A as a column vector and $A_{(i)}$, $i = 1, \ldots, m$, denote the i-th row of A as a row vector. We denote matrix norms by $\|A\|_\xi$, using subscripts to distinguish between various norms. Of particular interest will be the Frobenius norm, the square of which is $\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n A_{ij}^2$, and the spectral norm, which is defined by $\|A\|_2 = \sup_{x \in \mathbb{R}^n,\, x \neq 0} \frac{|Ax|}{|x|}$. These norms are related to each other as: $\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2$. If the SVD of A is

$$A = U \Sigma V^T = \sum_{t=1}^{\rho} \sigma_t u^t v^{t^T},$$

where ρ is the rank of A, u^t are the left singular vectors of A, v^t are the right singular vectors of A, and σ_t are the singular values of A such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\rho$, then for $k \leq \rho$ the matrix $A_k = \sum_{t=1}^k \sigma_t u^t v^{t^T}$ is the best rank-k approximation to A with respect to all unitarily invariant norms, such as the spectral and the Frobenius norms.

2.2. Review of Approximating a Matrix as the Product $C\tilde{U}R$

In [16] we presented and analyzed two algorithms to compute compressed approximate decompositions of a matrix $A \in \mathbb{R}^{m \times n}$. We will only use the second one here, which is computed with the ConstantTimeCUR algorithm of [16]. This decomposition is of the form

$$A' = C\tilde{U}R,$$

where C is an $m \times s$ matrix consisting of s randomly picked (and suitably rescaled) columns of A and R is an $s \times n$ matrix consisting of s randomly picked (and suitably rescaled) rows of A. More specifically, define two sets of probabilities $\{p_i\}_{i=1}^m$ and $\{q_j\}_{j=1}^n$ such that

$$p_i = \frac{|A_{(i)}|^2}{\|A\|_F^2} \quad \text{and} \quad q_j = \frac{|A^{(j)}|^2}{\|A\|_F^2}$$

and notice that $\sum_{i=1}^m p_i = \sum_{j=1}^n q_j = 1$. These probabilities will be used to pick columns and rows of A—to include in C and R—with probability proportional to their Euclidean

lengths squared. To form C (resp. R), we pick in s i.i.d. trials s columns (resp. rows) of A, such that in the t-th trial the j-th column (resp. i-th row) of A is picked with probability q_j (resp. p_i). If the j_t-th column of A (resp. i_t-th row of A) is picked in the t-th trial, then $A^{(j_t)}/\sqrt{sq_{j_t}}$ (resp. $A_{(i_t)}/\sqrt{sp_{i_t}}$) is included as a column of C (resp. row of R).

Given our construction of C and R, the construction of $\tilde{U}$ is somewhat more complicated, and involves an additional level of sampling. First, let $\{i_1, i_2, \ldots, i_s\}$ denote the indices of the s rows of A that were sampled above in order to construct R. We will pick the same rows of C in order to construct a $s \times s$ matrix Ψ such that the t-th row of Ψ is

$$\Psi_{(t)} = \frac{C_{(i_t)}}{\sqrt{sp_{it}}}.$$

Second, we will define yet another set of probabilities $\{\pi_i\}_{i=1}^m$ such that

$$\pi_i = \frac{|C_{(i)}|^2}{\|C\|_F^2}$$

and notice that $\sum_{i=1}^m \pi_i = 1$. We will use these probabilities to pick s rows of C in order to construct an $s \times s$ matrix W. More specifically, we pick in s i.i.d. trials s rows of C, such that in the t-th trial the i-th row of C is picked with probability π_i. If the i_t-th row of C is picked in the t-th trial, then $C_{(i_t)}/\sqrt{s\pi_{i_t}}$ is included as a row of W. We then compute $W^T W$ and its singular value decomposition,

$$W^T W = \sum_{t=1}^{c} \sigma_t^2(W) z^t z^{tT}.$$

Now, we compute the Moore-Penrose [22] pseudoinverse of a low-rank approximation to $W^T W$. More specifically, first we discard all singular values of $W^T W$ that are smaller than $(\epsilon/100)\|W\|_F^2$ and then we compute the pseudovinverse of the truncated matrix, which we denote by $\tilde{\Phi}$. Formally, if $\ell = \max\{t : \sigma_t^2(W) \geq \gamma \|W\|_F^2\}$,

$$\tilde{\Phi} = \sum_{t=1}^{\ell} \frac{1}{\sigma_t^2(W)} z^t z^{tT}.$$

Finally, we define $\tilde{U} = \tilde{\Phi}\Psi^T$.

Notice that we need two passes through the matrix A to construct C and R. In the first pass we compute the probabilities $\{p_i\}_{i=1}^m$ and $\{q_j\}_{j=1}^n$, and in the second pass we pick the actual rows and columns of A to form C and R. We will not need to store C and R in RAM, instead all we will need for our Max-Cut algorithm is $\tilde{U}$. So, it suffices to store $\tilde{\Phi}$ and Ψ in RAM, which we can do by computing the probabilities $\{\pi_i\}_{i=1}^m$ in the second pass and forming $\tilde{\Phi}$ and Ψ in a third pass through A. Thus, $\tilde{U}$ can be computed and stored using only $O(1)$ additional space and time, after making three passes over the input matrix A. To clarify the constant space claim, we note that a naive computation of the sampling probabilities would require $O(m + n)$ space. However, by employing a variant of reservoir sampling we can easily implement the sampling using only $O(1)$ additional space. This standard technique is analyzed thoroughly in [16].

Our main theorem regarding this CONSTANTTIMECUR algorithm of [16] is the following.

Theorem 1. *Suppose $A \in \mathbb{R}^{m\times n}$ and let C, $\tilde{U}$, and R be constructed as described earlier. Let $\delta, \epsilon > 0$. If we set $s = \Omega(1/\epsilon^8)$, then with probability at least $7/8$ each of the following holds*:

$$\|C\|_F = \|A\|_F \tag{2}$$

$$\|\tilde{U}\|_2 \leq O(1/\epsilon)/\|A\|_F \tag{3}$$

$$\|R\|_F = \|A\|_F \tag{4}$$

$$\|A - C\tilde{U}R\|_2 \leq \|A - A_k\|_2 + \epsilon\|A\|_F. \tag{5}$$

If we let $\epsilon' = 2\epsilon$ and choose $k = 1/\epsilon^2$ then (5) *becomes*

$$\|A - C\tilde{U}R\|_2 \leq \epsilon'\|A\|_F. \tag{6}$$

Notice that (6) follows from (5) since $\|A - A_k\|_2$ is exactly equal to the $(k+1)$-st singular value of A, and since $\sigma_{k+1}(A) \leq \|A\|_F/\sqrt{k}$. Also notice that (5) implies that the $C\tilde{U}R$ decomposition may be thought of as a low rank approximation to the matrix A. (See [16] for a discussion of this issue and a comparison of this low rank approximation with that of the Singular Value Decomposition.)

3. SAMPLING SUBPROGRAMS OF A LINEAR PROGRAM

In this section, we examine relating the feasibility or infeasibility of a Linear Program (LP) to the feasibility or infeasibility of a randomly sampled version of that LP. More specifically, we sample a small number of variables, and prove that determining the feasibility or infeasibility of the resulting subprograms can provide information as to whether the original LP was feasible or infeasible. The main novel aspect of our algorithms is that the sampling is done nonuniformly. Indeed, we keep variables whose associated coefficients in the original LP are large with higher probability. This allows us to improve on the results of [1] which employed uniform sampling to achieve weaker results than ours. This improvement is critical in obtaining the improved Max-Cut algorithm.

We now state and prove Theorems 2 and 3; these two theorems provide results that are complementary, as we discuss at the end of this section. An important ingredient of our proofs is to bound the loss in accuracy when sampling variables from the LPs by reducing the problem to approximating matrix-vector products by sampling a few columns of the matrix and the corresponding elements of the vectors. The latter is much easier to bound directly, and has been thoroughly studied in [11, 14]. The proof in [1] employs only first principles from probability theory and does not seem to easily generalize to our non-uniform sampling.

Theorem 2. *Let P be a $r \times n$ matrix and b be a $r \times 1$ vector. Let $P^{(i)}$ denote the i-th column of P and consider the following LP*:

$$Px = \sum_{i=1}^{n} P^{(i)}x_i \leq b \quad 0 \leq x_i \leq c_i. \tag{7}$$

Suppose q is a positive integer and Q is a random subset of $\{1, 2, \ldots n\}$ with $|Q| = q$ formed by picking elements of $\{1, 2, \ldots n\}$ in q i.i.d. trials, where, in each trial, the probability of picking the i-th element is

$$p_i = \mathbf{Pr}[i_t = i] = \frac{c_i |P^{(i)}|}{\mathcal{N}}, \tag{8}$$

where $\mathcal{N} = \sum_{i=1}^{n} c_i |P^{(i)}|$. Let $\vec{\mathbf{1}}_r$ denote the $r \times 1$ all-ones vector and $\eta = 1 + \sqrt{8 \log(1/\delta)}$. If the LP (7) *is feasible then, with probability at least $1 - \delta$*

$$\sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i \leq b + \frac{\eta \mathcal{N}}{\sqrt{q}} \vec{\mathbf{1}}_r \quad 0 \leq x_i \leq c_i, i \in Q \tag{9}$$

is feasible as well. If the LP (7) *is infeasible then, with probability at least $1 - \delta$*

$$\sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i \leq b - \frac{\eta \mathcal{N}}{\sqrt{q}} \vec{\mathbf{1}}_r \quad 0 \leq x_i \leq c_i \, , \, i \in Q \tag{10}$$

is infeasible as well.

Proof. We first show that the vector $\sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i$ in (9) and (10) is with high probability close to the vector $\sum_{i=1}^{n} P^{(i)} x_i$ in (7); then we prove (9) and finally we prove (10).

Construct $\tilde{P}$ and $\tilde{x}$ as follows. Randomly pick in i.i.d. trials q columns of P and the corresponding q "rows" of x (here "rows" of x are elements of x), where the probability of picking the i-th column-row pair is p_i. Let $\tilde{P}$ be the $r \times q$ matrix with columns equal to the columns that were picked divided by $\sqrt{q p_i}$, and $\tilde{x}$ be the $q \times 1$ vector whose elements are those elements of x that were picked divided by $\sqrt{q p_i}$. Thus if $\{i_t\}_{t=1}^{c}$ is the sequence of elements of $\{1, \cdots, n\}$ chosen, then $\tilde{P^{(t)}} = \frac{1}{\sqrt{q p_{i_t}}} P^{(i_t)}$ and $\tilde{x_t} = \frac{1}{\sqrt{q p_{i_t}}} x_{i_t}$. Furthermore, $\tilde{P}\tilde{x} = \sum_{t=1}^{c} \frac{1}{q p_{i_t}} P^{(i_t)} x_{i_t} = \sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i$.

We first claim that Px is well approximated by $\tilde{P}\tilde{x}$, i.e., that

$$\mathbf{Pr}\left[\|Px - \tilde{P}\tilde{x}\|_F \leq \frac{\eta}{\sqrt{q}} \sum_{i=1}^{n} c_i |P^{(i)}| \right] \geq 1 - \delta. \tag{11}$$

To establish the claim, note that

$$\mathbf{E}[\|Px - \tilde{P}\tilde{x}\|_F] \leq \left(\frac{1}{q} \sum_{i=1}^{n} \frac{1}{p_i} |P^{(i)}|^2 |x_i|^2 \right)^{1/2} \leq \frac{1}{\sqrt{q}} \sum_{i=1}^{n} c_i |P^{(i)}|. \tag{12}$$

The first inequality can be derived by analyzing

$$\mathbf{E}\left[\left\| \sum_{i=1}^{n} P^{(i)} x_i - \sum_{t=1}^{q} \tilde{P}^{(i_t)} \tilde{x}_{i_t} \right\|_F^2 \right] = \mathbf{E}\left[\left\| \sum_{i=1}^{n} P^{(i)} x_i - \sum_{t=1}^{q} \frac{1}{q p_{i_t}} P^{(i_t)} x_{i_t} \right\|_F^2 \right],$$

where $i_1, i_2, \ldots, i_q$ are the indices of the columns of P that were included in the sample Q. Notice that $\sum_{t=1}^{q} \frac{1}{q p_{i_t}} P^{(i_t)} x_{i_t}$ is an unbiased estimator for $\sum_{i=1}^{n} P^{(i)} x_i$, and the result follows

by bounding the variance of the estimator. The second inequality follows from the form of the probabilities of (8) and since $x_i \leq c_i$. Next, define the event $\mathcal{E}_1$ to be

$$\|Px - \tilde{P}\tilde{x}\|_F \geq \frac{\eta}{\sqrt{q}} \sum_{i=1}^{n} c_i |P^{(i)}|$$

and the event $\mathcal{E}_2$ to be

$$\|Px - \tilde{P}\tilde{x}\|_F \geq \mathbf{E}[\|Px - \tilde{P}\tilde{x}\|_F] + \sqrt{(8/q)\log(1/\delta)} \sum_{i=1}^{n} c_i |P^{(i)}|,$$

and note that $\mathbf{Pr}[\mathcal{E}_1] \leq \mathbf{Pr}[\mathcal{E}_2]$ due to (12). Thus, to prove the claim it suffices to prove that $\mathbf{Pr}[\mathcal{E}_2] \leq \delta$. To that end, note that $\tilde{P}$ and $\tilde{x}$ and thus $\tilde{P}\tilde{x}$ are formed by randomly selecting c elements $\{i_t\}_{t=1}^{c}$ from $\{1, \ldots, n\}$, independently and with replacement; thus, $\tilde{P}\tilde{x} = \tilde{P}\tilde{x}(i_1, \ldots, i_c)$. Consider the function

$$F(i_1, \ldots, i_c) = \|Px - \tilde{P}\tilde{x}\|_F.$$

We will show that changing one i_t at a time does not change F too much; this will enable us to apply a martingale inequality. To this end, consider changing one of the i_t to i_t' while keeping the other i_t's the same. Then, construct the corresponding $\tilde{P}'$ and $\tilde{x}'$. Note that $\tilde{P}'$ differs from $\tilde{P}$ in only a single column, $\tilde{x}'$ differs from $\tilde{x}$ in only a single row. Thus,

$$\|\tilde{P}\tilde{x} - \tilde{P}'\tilde{x}'\|_F = \left\| \frac{P^{(i_t)} x_{i_t}}{q p_{i_t}} - \frac{P^{(i_t')} x_{i_t'}}{q p_{i_t'}} \right\|_F \tag{13}$$

$$\leq \frac{2}{q} \max_{\xi} \left\| \frac{P^{(\xi)} x_{\xi}}{p_{\xi}} \right\|_F \tag{14}$$

$$\leq \frac{2}{q} \sum_{i=1}^{n} c_i \left| P^{(i)} \right| \tag{15}$$

(15) follows from the particular form of the probabilities used and since $x_i \leq c_i$. Define $\Delta = \frac{2}{q} \sum_{i=1}^{n} c_i |P^{(i)}|$. Then, we see that

$$\|Px - \tilde{P}\tilde{x}\|_F \leq \|Px - \tilde{P}'\tilde{x}'\|_F + \Delta$$

and that

$$\|Px - \tilde{P}'\tilde{x}'\|_F \leq \|Px - \tilde{P}\tilde{x}\|_F + \Delta.$$

Thus F satisfies

$$|F(i_1, \ldots, i_k, \ldots, i_q) - F\big(i_1, \ldots, i_k', \ldots, i_q\big)| \leq \Delta.$$

Define $\gamma = \sqrt{2q \log(1/\delta)}\,\Delta$ and consider the associated Doob martingale. By the Hoeffding-Azuma inequality,

$$\mathbf{Pr}[\mathcal{E}_2] \leq \exp\left(-\gamma^2 / 2q\Delta^2\right) = \delta.$$

This completes the proof of the claim.

In order to prove (9), note that (11) implies that with probability at least $1-\delta$ there exists some vector v such that $\tilde{P}\tilde{x} = Px + v$ and such that $|v| \leq \frac{\eta}{\sqrt{q}}(\sum_{i=1}^{n} c_i|P^{(i)}|)$. (9) follows since $Px \leq b$ by assumption and since $v \leq \frac{\eta}{\sqrt{q}}(\sum_{i=1}^{n} c_i|P^{(i)}|)\vec{\mathbf{1}}_r$ componentwise. Next, we prove (10).

Using a variant of Farkas' lemma (Corollary 7.1e, p. 89 of [34]), the hypothesis of the theorem implies that there exists a non-negative $r \times 1$ vector u such that the following LP is infeasible

$$\sum_{i=1}^{n} u^T P^{(i)} x_i \leq u^T b \quad 0 \leq x_i \leq c_i, \quad i = 1, 2, \ldots n$$

Choosing x^* such that $x_i^* = c_i$ if $(u^T P)_i < 0$ and $x_i^* = 0$ if $(u^T P)_i \geq 0$, this implies that

$$\sum_{i=1}^{n} c_i (u^T P)_i^- > u^T b \tag{16}$$

Since $|u^T Px - u^T \tilde{P}\tilde{x}| \leq \|u\|_F \|Px - \tilde{P}\tilde{x}\|_F$ and $\|u\|_F = \|u\|_2 = (\sum_{i=1}^{r} u_i^2)^{1/2} \leq \sum_{i=1}^{r} |u_i| = \sum_{i=1}^{r} u_i$ we can use (11) to see that with probability at least $1-\delta$

$$|u^T Px - u^T \tilde{P}\tilde{x}| \leq \left(\sum_{i=1}^{r} u_i\right) \frac{\eta}{\sqrt{q}} \sum_{i=1}^{n} c_i |P^{(i)}|.$$

Thus, with probability at least $1-\delta$,

$$u^T \tilde{P}\tilde{x} \geq u^T Px - \left(\sum_{i=1}^{r} u_i\right) \frac{\eta}{\sqrt{q}} \sum_{i=1}^{n} c_i |P^{(i)}| \tag{17}$$

Define event $\mathcal{E}_3$ to be the event that

$$\sum_{i \in Q} \frac{c_i}{q p_i} (u^T P)_i^- > u^T b - \left(\sum_{i=1}^{r} u_i\right) \frac{\eta}{\sqrt{q}} \sum_{i=1}^{n} c_i |P^{(i)}|$$

By applying (17) to the x^* defined above and using (16) it follows that $\mathbf{Pr}[\mathcal{E}_3] \geq 1-\delta$. Under $\mathcal{E}_3$, we claim that the following Linear Program is infeasible:

$$\sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i \leq b - \frac{\eta}{\sqrt{q}} \left(\sum_{i=1}^{n} c_i |P^{(i)}|\right) \vec{\mathbf{1}} \quad 0 \leq x_i \leq c_i, \ \forall i \in Q \tag{18}$$

This is because, if the above LP had a feasible solution x, then (since u is a non-negative vector) x would also satisfy

$$\sum_{i \in Q} \frac{1}{q p_i} (u^T P)_i x_i \leq u^T b - \left(\sum_{i=1}^{r} u_i\right) \frac{\eta}{\sqrt{q}} \sum_{i=1}^{n} c_i |P^{(i)}|$$

contradicting $\mathcal{E}_3$. The theorem then follows. ∎

We next present Theorem 3, which is quite similar to Theorem 2, except that we construct the sample Q according to the probability distribution (20).

Theorem 3. *Let P be a $r \times n$ matrix and b be a $r \times 1$ vector. Let $P^{(i)}$ denote the i-th column of P and consider the following Linear Program:*

$$Px = \sum_{i=1}^{n} P^{(i)} x_i \leq b \quad 0 \leq x_i \leq c_i. \tag{19}$$

Suppose q is a positive integer and Q is a random subset of $\{1, 2, \ldots n\}$ with $|Q| = q$ formed by picking elements of $\{1, 2, \ldots n\}$ in q i.i.d. trials, where, in each trial, the probability of picking the i-th element is

$$p_i = \mathbf{Pr}[i_t = i] = \frac{|P^{(i)}|^2}{\|P\|_F^2}. \tag{20}$$

Let $\vec{\mathbf{1}}_r$ denote the $r \times 1$ all-ones vector. If the LP (19) *is feasible then, with probability at least $1 - \delta$*

$$\sum_{i \in Q} \frac{1}{qp_i} P^{(i)} x_i \leq b + \frac{1}{\delta\sqrt{q}} |x| \|P\|_F \vec{\mathbf{1}}_r \quad 0 \leq x_i \leq c_i \,,\, i \in Q \tag{21}$$

is feasible as well. If the Linear Program (19) *is infeasible then, with probability at least $1 - \delta$*

$$\sum_{i \in Q} \frac{1}{qp_i} P^{(i)} x_i \leq b - \frac{1}{\delta\sqrt{q}} |x| \|P\|_F \vec{\mathbf{1}}_r \quad 0 \leq x_i \leq c_i, i \in Q \tag{22}$$

is infeasible as well.

Proof. We follow reasoning similar to that used in the proof of Theorem 2. We first claim that Px is well approximated by $\tilde{P}\tilde{x} = \sum_{i \in Q} \frac{1}{qp_i} P^{(i)} x_i$, i.e., that

$$\mathbf{Pr}\left[\|Px - \tilde{P}\tilde{x}\|_F \leq \frac{1}{\delta\sqrt{q}} |x| \|P\|_F \right] \geq 1 - \delta. \tag{23}$$

To establish the claim, first note that

$$\mathbf{E}[\|Px - \tilde{P}\tilde{x}\|_F] \leq \left(\frac{1}{q} \sum_{i=1}^{n} \frac{1}{p_i} |P^{(i)}|^2 |x_i|^2 \right)^{1/2} \leq \frac{1}{\sqrt{q}} |x| \|P\|_F, \tag{24}$$

and then apply Markov's inequality. (21) then follows immediately since there exists a vector v such that with high probability $\tilde{P}\tilde{x} = Px + v$, where $|v| \leq \frac{1}{\delta\sqrt{q}} |x| \|P\|_F$. (22) follows by using LP duality in a manner similar to the proof of Theorem 2. ∎

Note that establishing (23) in Theorem 3 (respectively, (11) in Theorem 2) uses ideas that are very similar to those used in [14] for approximating the product of two matrices. Once we are given (23) (respectively, (11)) then the proof of (21) (respectively, (9)) is immediate; we simply show that if the original LP has a solution then the sampled LP also has a solution since $\tilde{P}\tilde{x}$ is sufficiently close to Px. On the other hand, proving (22) (respectively, (10)) is more difficult; we must show that the nonexistence of a solution of the original LP implies the same for the randomly sampled version of the LP. Fortunately, by LP duality theory the nonexistence of a solution in the LP implies the existence of a certain solution in a related LP.

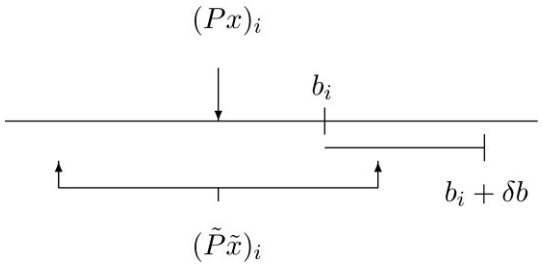

Fig. 1. Diagram illustrating feasible subprograms (9) and (21).

To illustrate the action of Theorems 2 and 3, consider Figs. 1 and 2. Figure 1 illustrates that when LP, e.g., (19), is feasible each of the $(Px)_i$ is less than the corresponding b_i. Let $\delta b = \frac{\eta\mathcal{N}}{\sqrt{q}}$ for Theorem 2 and let $\delta b = \frac{1}{\delta\sqrt{q}}|x|\|P\|_F$ for Theorem 3. Recall that with high probability $(\tilde{P}\tilde{x})_i$ is not much more than $(Px)_i$; thus, if the right hand side of the constraint is increased by that small amount, i.e., b_i is replaced by $b_i + \delta b$ then we will still have that $(\tilde{P}\tilde{x})_i \le b_i + \delta b$. Similarly, Fig. 2 illustrates that when LP (7) is infeasible at least one of the $(Px)_i$ is greater than the corresponding b_i. Recall also that with high probability $(\tilde{P}\tilde{x})_i$ is not much less than $(Px)_i$; thus, if the right hand side of the constraint is decreased by that small amount, i.e., b_i is replaced by $b_i - \delta b$ then we will still have that $(\tilde{P}\tilde{x})_i \ge b_i - \delta b$ and the constraint will still not be satisfied.

A special case of these results occurs when $c_i = 1$ for all i, since in that case the Cauchy-Schwartz inequality implies $\sum_{i=1}^{n} |P^{(i)}| \le \sqrt{n}\|P\|_F$. The induced LPs (21) and (22) in Theorem 3 may then be replaced by

$$\sum_{i\in Q} \frac{1}{qp_i} P^{(i)} x_i \le b \pm \frac{1}{\delta}\sqrt{\frac{n}{q}}\|P\|_F \vec{\mathbf{1}}_r \quad 0 \le x_i \le c_i\,,\, i \in Q. \tag{25}$$

Similarly, the induced LPs (9) and (10) in Theorem 2 may then be replaced by

$$\sum_{i\in Q} \frac{1}{qp_i} P^{(i)} x_i \le b \pm \eta\sqrt{\frac{n}{q}}\|P\|_F \vec{\mathbf{1}}_r \quad 0 \le x_i \le 1\,,\, i \in Q. \tag{26}$$

Note that these two theorems are complementary in the following sense. Using the SELECT algorithm of [14] (a simple variant of reservoir sampling) we see that computing the probabilities (20) of Theorem 3 requires $O(1)$ space [14]. On the other hand, computing

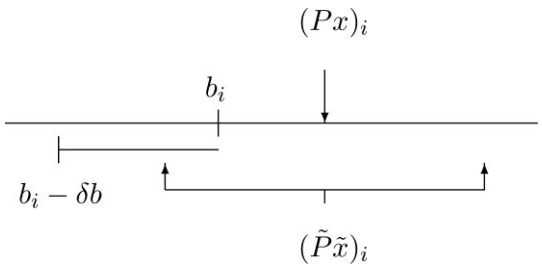

Fig. 2. Diagram illustrating infeasible subprograms (10) and (22).

the probabilities (8) of Theorem 2 requires, to the best of our knowledge, $O(m+n)$ space. Also, notice that using (8) we can obtain the results we want with probability at least $1-\delta$ with a sampling complexity of $q = O(\eta^2) = O(\log 1/\delta)$, while with (20) we require $q = O(1/\delta^2)$ samples. In our Max-Cut application, since we want a sampling complexity and running time that are constant, independent of n, we will use the probabilities (20) and Theorem 3. As a result, this will require $q = O(1/\delta^2)$ samples, which will translate into substantially worse sampling complexity as a function of $1/\epsilon$. It is an open problem whether, for an LP problem, one can construct sampling probabilities in $O(1)$ space while needing a sampling complexity of only $q = O(\log 1/\delta)$.

4. AN APPROXIMATION ALGORITHM FOR MAX-CUT

In this section we present and analyze a new approximation algorithm for the Max-Cut problem. Recall that the Max-Cut of a graph G with weighted adjacency matrix A is:

$$\textbf{MAX-CUT}[G] = \textbf{MAX-CUT}[A] = \max_{x\in\{0,1\}^n} x^T A(\vec{\mathbf{1}}_n - x), \tag{27}$$

where $\vec{\mathbf{1}}_n$ is the all-ones $n \times 1$ vector and x is the characteristic vector of the cut, i.e., it is a 0–1 vector whose i-th entry denotes whether vertex i is on the left or right side of the cut. In Section 4.1 we introduce and describe the ApproximateMaxCut algorithm, in Section 4.2 we provide a discussion of sampling and running time issues, and in Section 4.3 we present Theorem 4, which establishes the correctness of the algorithm. Then, in Section 4.4 we present the intuition behind the proof of Theorem 4 and in Section 4.5 we present the proof of Theorem 4.

4.1. The Algorithm

Consider the ApproximateMaxCut algorithm which is presented in Fig. 3 and which takes as input an $n \times n$ matrix A, which is the weighted adjacency matrix of a weighted undirected graph G on n vertices, and computes an approximation Z_{LPQ} to **MAX-CUT**$[A]$. To compute Z_{LPQ}, the ApproximateMaxCut algorithm uses the ConstantTimeCUR algorithm, as described in [16] and Section 2, to compute a constant-sized description of three matrices, C, $\tilde{U}$, and R, whose product $C\tilde{U}R \approx A$. In addition, from the (not explicitly constructed) matrices C and R two constant-sized matrices, denoted $\tilde{C}$ and $\tilde{R}$, consisting of q (see (52) for the exact value of q) rows of C and the corresponding q columns of R, each appropriately rescaled, are constructed. These matrices are used in the construction of the linear programs $\mathbf{LP}_Q(u, v)$; the algorithm then checks whether a constant number of these LPs (each on a constant number q of variables) are feasible and returns the maximum of an easily-computed function of the feasible vectors as the approximation Z_{LPQ} of **MAX-CUT**$[A]$. Notice that we are *only* returning an approximation to the value of the maximum cut, and not an approximation to the actual cut; indeed, it is not clear that our techniques can return the latter approximation in either an explicit or an implicit form.

To prove Theorem 4, which establishes the correctness of the algorithm, we will require four levels of approximation and we will have to show that each level does not introduce too much error. At the first level of approximation we will use the ConstantTimeCUR algorithm of [16] and Section 2 in order to compute a constant-sized description of C, $\tilde{U}$, and R. This will enable us to reduce approximating **MAX-CUT**$[A]$

APPROXIMATEMAXCUT Algorithm

Input: $A \in \mathbb{R}^{n\times n}$, the weighted adjacency matrix of a graph $G = (V, E)$ with n vertices, and ϵ, an error parameter.

Output: Z_{LPQ}, an approximation to **MAX-CUT**$[A]$.

- Let $s = \Theta(1/\epsilon^8)$ be the number of columns/rows of A that are sampled for the $C\tilde{U}R$ approximation, let $q = \text{poly}(1/\epsilon)\exp(\text{poly}(1/\epsilon))$ be the dimension of the randomly sampled Linear Program, and let Q be the set of indices of the sampled variables.
- Compute, using the CONSTANTTIMECUR algorithm (as described in Section 2), and store the $s \times s$ matrix $\tilde{U}$.
- Compute the matrices C and R (as described in Section 2), but *do not* store them. Instead, compute sampling probabilities $\{w_i\}_{i=1}^n$

$$w_i = \frac{|C_{(i)}, R^{(i)}|^2}{\sum_{j=1}^n |C_{(j)}, R^{(j)}|^2}, \tag{28}$$

  where $|C_{(i)}, R^{(i)}|^2 = \sum_{j=1}^s (C_{ij}^2 + R_{ji}^2)$. Pick q integers from $[n]$ in q i.i.d. trials, where in the t-th trial, i is picked with probability w_i. Let Q be the set of the q sampled integers.
- Compute and store $\tilde{C}$ (resp. $\tilde{R}$), a $q \times s$ (resp. $s \times q$) matrix that contains the rows of C (resp. columns of R) whose indices are in Q, rescaled by $1/qw_i$ for $i \in Q$.
- Construct all possible vector pairs $(u, v) \in [-\sqrt{n}\|A\|_F, \sqrt{n}\|A\|_F]^{2s}$ in increments of $(\epsilon/4s)\sqrt{n}\|A\|_F$. Let Ω_Δ be the set of all such pairs.
- For every pair $(u, v) \in \Omega_\Delta$ check whether the Linear Program $\mathbf{LP}_Q(u, v)$

$$\begin{aligned} u - \tfrac{\epsilon}{4s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \le \quad & \textstyle\sum_{i\in Q} \frac{1}{qw_i} C_{(i)}x_i \quad \le u + \tfrac{\epsilon}{4s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \\ v - \tfrac{\epsilon}{4s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \le \quad & R\vec{\mathbf{1}}_n - \textstyle\sum_{i\in Q} \frac{1}{qw_i} R^{(i)}x_i \quad \le v + \tfrac{\epsilon}{4s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \end{aligned}$$

$$x_i \in [0, 1], \quad i \in Q$$

  is feasible, and select $(\bar{u}, \bar{v})$ such that $u^T\tilde{U}v$ is maximized among all feasible pairs.
- Return $Z_{LPQ} = \bar{u}^T\tilde{U}\bar{v}$.

Fig. 3. The APPROXIMATEMAXCUT algorithm.

to **MAX-CUT**$[C\tilde{U}R]$. At the second level of approximation we will reduce approximating **MAX-CUT**$[C\tilde{U}R]$ to checking the feasibility of a large but constant number of Integer Programs (IPs) $\mathbf{IP}(u, v)$ and returning the maximum of an easily-computed function of them. At the third level of approximation we will consider the LP relaxation of these IPs and construct a large but constant number of LPs $\mathbf{LP}(u, v)$. Finally, at the fourth level of approximation we will sample from each of these LPs and construct the constant-sized randomly-sampled $\mathbf{LP}_Q(u, v)$. By combining these results, we will see that approximating **MAX-CUT**$[A]$ can be reduced via four approximation steps to testing the feasibility of $\mathbf{LP}_Q(u, v)$ for every $(u, v) \in \Omega_\Delta$, where Ω_Δ is a large but constant-sized set that is defined below. Note that sampling and thus failure probabilities enter only at the first and fourth level of approximation, i.e., they enter when approximating **MAX-CUT**$[A]$ by **MAX-CUT**$[C\tilde{U}R]$ since with some (small) probability $C\tilde{U}R$ may fail to provide a

good approximation to A, and they enter when approximating $\mathbf{LP}(u, v)$ by $\mathbf{LP}_Q(u, v)$ since with some (small) probability the sampled version of the LP may fail to be close to the original LP.

4.2. Analysis of the Implementation and Running Time

In the APPROXIMATEMAXCUT algorithm, three passes over the matrix are performed. In the first pass, the row and column probabilities with which to sample from A in order to construct C and R are chosen in constant additional space and time. In the second pass, probabilities are chosen with which to sample rows of C in order to construct a matrix from which the approximate SVD may be performed and thus from which $\tilde{U}$ may be computed; this requires constant additional space and time. In addition, in the second pass probabilities are chosen with which to sample rows of C and the corresponding columns of R in order to construct a matrix for the sampled version of the linear program; since we use probabilities (20) this also requires constant additional space and time. Then, in the third pass, the relevant quantities are extracted and the computations are performed to compute the $C\tilde{U}R$ decomposition. In addition, in the third pass, rows of C and the corresponding columns of R are chosen and from them $\tilde{C}$ and $\tilde{R}$ and thus the sampled LP (48) are constructed; in particular, $R\vec{\mathbf{1}}_n \in \mathbb{R}^s$ (and thus the right hand side of the LP written as (45)) can be computed in constant additional space and time. Note that the sampling for the sampled LP is independent of and thus does not conflict with the sampling for $C\tilde{U}R$. Overall, our algorithm keeps the $s \times s$ matrix $\tilde{U}$, an $O(s)$-lengthed description of C and R and the $q \times s$ matrix $\tilde{C}$ and the $s \times q$ matrix $\tilde{R}$. Since $s = O(1/\epsilon^8)$ and $q = \text{poly}(1/\epsilon)\exp(\text{poly}(1/\epsilon))$ the total additional space and time is $\exp(\text{poly}(1/\epsilon))$.

4.3. The Main Theorem

Theorem 4 is our main theorem; note that the 3/4 is arbitrary and can be boosted to any number less than 1 using standard methods.

Theorem 4. *Let A be the $n \times n$ weighted adjacency matrix of a graph $G = (V, E)$, let ϵ be fixed, and let Z_{LPQ} be the approximation to the* **MAX-CUT**$[A]$ *returned by the* APPROXIMATEMAXCUT *algorithm. Then, with probability at least* 3/4

$$|Z_{\text{LPQ}} - \mathbf{MAX\text{-}CUT}[A]| \leq \epsilon n \|A\|_F.$$

The algorithm makes three passes, i.e., three sequential reads, through the matrix A and then uses constant (with respect to n) additional space and constant additional time. The algorithm chooses a random sample of A according to a nonuniform probability distribution.

4.4. Intuition behind the Proof of Theorem 4

To prove Theorem 4, we will require four levels of approximation, and we will have to show that each level does not introduce too much error. We note that the algorithm and its analysis use ideas similar to those used in [1, 18] for the case of uniform sampling. In particular, [18] introduced the idea of using linear programming relaxations of integer programs in a grid in order to approximate the value of the Max-Cut. [1] use the same ideas, but tailored to the cut-norm matrix approximation scheme that they employed. Here, we

use the same general outline, appropriately modified to fit the $C\tilde{U}R$ matrix decomposition that we employ to approximate the adjacency matrix of weighted graphs. Thus, although the general algorithmic framework outlining the use of linear algebraic approaches for combinatorial problems appeared in [18], all the implementation details are significantly different depending on the matrix decomposition that is used, and thus merit a detailed exposition.

In the *first* level of approximation we will use the ConstantTimeCUR algorithm of [16] and Section 2, and sample $s = \Theta(1/\epsilon^8)$ rows and columns of the matrix A in order to compute a constant-sized description of C, $\tilde{U}$, and R. The description consists of the explicit matrix $\tilde{U}$ and labels indicating which s columns of A and which s rows of A are used in the construction of C and R, respectively. From Theorem 1 we see that under appropriate assumptions on the sampling probabilities (which we will satisfy) $C\tilde{U}R$ is close to A in the sense that $\|A - C\tilde{U}R\|_2 \leq \epsilon\|A\|_F$ with high probability. A good approximation to **MAX-CUT**$[A]$ is provided by **MAX-CUT**$[C\tilde{U}R]$, which is the Max-Cut of the graph whose weighted adjacency matrix is $C\tilde{U}R$. This is shown in Lemma 1. Thus, it suffices to approximate well

$$\textbf{MAX-CUT}[C\tilde{U}R] = \max_{x\in\{0,1\}^n} x^T C\tilde{U}R(\vec{\mathbf{1}}_n - x).$$

We will see that with high probability $|C^T x| \leq \sqrt{n}\|A\|_F$ and $|R(\vec{\mathbf{1}}_n - x)| \leq \sqrt{n}\|A\|_F$. Thus, both $C^T x$ and $R(\vec{\mathbf{1}}_n - x)$ lie in $[-\sqrt{n}\|A\|_F, \sqrt{n}\|A\|_F]^s$. Let $\Omega = [-\sqrt{n}\|A\|_F, \sqrt{n}\|A\|_F]^{2s}$ and consider the set of vectors $(u, v) \in \Omega$. Suppose we pick the vector pair $(\bar{u}, \bar{v})$ that satisfies the following two conditions:

1. Feasibility: There exists a vector $\bar{x} \in \{0, 1\}^n$ such that the pair $(\bar{u}, \bar{v})$ satisfies
$$C^T\bar{x} = \bar{u} \quad \text{and} \quad R(\vec{\mathbf{1}}_n - \bar{x}) = \bar{v},$$
2. Maximization: $(\bar{u}, \bar{v})$ maximizes the value $u^T\tilde{U}v$ over all such possible pairs.

For such a $(\bar{u}, \bar{v})$ the vector $\bar{x}$ defines a Max-Cut of the graph with weighted adjacency matrix $C\tilde{U}R$ and thus $\bar{u}^T\tilde{U}\bar{v} =$ **MAX-CUT**$[C\tilde{U}R]$.

We will then perform a *second* level of approximation and discretize the set of candidate vector pairs. Let Ω_Δ denote the discretized set of vector pairs, where the discretization Δ is defined below. Consider the set of vectors $(u, v) \in \Omega_\Delta$ and suppose we pick the vector pair $(\bar{u}, \bar{v})$ that satisfies the following two conditions:

1′. Approximate feasibility: There exists a vector $\bar{x} \in \{0, 1\}^n$ such that the pair $(\bar{u}, \bar{v})$ satisfies
$$\bar{u} - \frac{\epsilon}{s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \leq C^T\bar{x} \leq \bar{u} + \frac{\epsilon}{s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \text{ and}$$
$$\bar{v} - \frac{\epsilon}{s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \leq R(\vec{\mathbf{1}}_n - \bar{x}) \leq \bar{v} + \frac{\epsilon}{s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s,$$
i.e., there exists a vector $\bar{x} \in \{0, 1\}^n$ such that the pair $(\bar{u}, \bar{v})$ satisfies
$$C^T\bar{x} \approx \bar{u} \quad \text{and} \quad R(\vec{\mathbf{1}}_n - \bar{x}) \approx \bar{v},$$
2′. Maximization: $(\bar{u}, \bar{v})$ maximizes the value $u^T\tilde{U}v$ over all such possible pairs.

In this case, we are checking whether every vector pair $(u, v) \in \Omega_\Delta$ in the discretized set is approximately feasible in the sense that a nearby vector pair $(u, v) \in \Omega$ satisfies the feasibility condition exactly. This test of approximate feasibility is formalized as the integer program IP (36). If we choose the discretization in each dimension as $\Delta = \frac{\epsilon}{4s}\sqrt{n}\|A\|_F$, then every vector pair $(u, v) \in \Omega$ is near a vector pair $(u, v) \in \Omega_\Delta$. Thus, (subject to a small failure probability) we will not "miss" any feasible vector pairs, i.e., for every exactly feasible $(u, v) \in \Omega$ there exists some approximately feasible $(u, v) \in \Omega_\Delta$. Equally importantly, with this discretization, we only have to check a large but constant number of candidate vector pairs. Taking the maximum of $u^T\tilde{U}v$ over the feasible vector pairs $(u, v) \in \Omega_\Delta$ will lead to an approximation of **MAX-CUT**$[C\tilde{U}R]$. This is formalized in Lemma 3. At this point, we have reduced the problem of approximating **MAX-CUT**$[A]$ to that of checking the feasibility of a large but constant number of IPs (of the form (36)) and returning the maximum of an easily computed function of them.

Next, we reduce this to the problem of checking the feasibility of a large but constant number of constant-sized LPs on a constant number q of variables (of the form (48)) and returning the maximum of an easily computed function of them. We do this in two steps. First, we will perform a *third* level of approximation and consider the LP relaxation of the IP. Since this LP has a very special structure, i.e., even though the LP lies in an n-dimensional space the number of the constraints is a constant independent of n, there exists a feasible solution for the LP that has at most a constant number of non-integral elements. We will exploit this and will consider an LP (see LP (44)) which is a slight tightening of the LP relaxation of the IP; in Lemma 4 we will prove that if the IP is infeasible then the LP is infeasible as well. Then, we will perform a *fourth* level of approximation and construct a constant-sized randomly-sampled LP on a constant number q of variables, such that the infeasibility of the LP on n variables implies, with probability at least $1-\delta^*$, the infeasibility of the LP on q variables; see LP (48) and Lemma 5. This last level of approximation involves sampling and thus a failure probability. Since there are a constant number $\left(\frac{8s}{\epsilon}\right)^{2s}$ of values of $(u, v) \in \Omega_\Delta$ to check, by choosing δ^* to be a sufficiently small constant independent of n, the probability that any one of the large but constant number of sampling events involved in the construction of the constant-sized LPs will fail can be bounded above by $1/8$. From Lemmas 6 and 7, Theorem 4 will then follow.

4.5. Proof of Theorem 4

We will prove that $|Z_{\text{LPQ}} - \textbf{MAX-CUT}[A]| \leq O(\epsilon)n\|A\|_F$, where Z_{LPQ} is the output of the ApproximateMaxCut algorithm and is also defined in (50); by suitable choice of constants it will then follow that $|Z_{\text{LPQ}} - \textbf{MAX-CUT}[A]| \leq \epsilon' n\|A\|_F$, establishing Theorem 4. The first step of our proof is to compute the $C\tilde{U}R$ approximation to A. Lemma 1 guarantees that **MAX-CUT**$[C\tilde{U}R]$ and **MAX-CUT**$[A]$ are close.

Lemma 1. *With probability at least* $7/8$

$$|\textbf{MAX-CUT}[C\tilde{U}R] - \textbf{MAX-CUT}[A]| \leq \epsilon n\|A\|_F/2.$$

Proof. By submultiplicitivity we have that for all $x \in \mathbb{R}^n$

$$|x^T(A - C\tilde{U}R)(\vec{\mathbf{1}}_n - x)| \leq |x||\vec{\mathbf{1}}_n - x|\|A - C\tilde{U}R\|_2.$$

In addition, for every $x \in \{0,1\}^n$ we have that $|x||\vec{\mathbf{1}}_n - x| \leq n/2$. Since $C\tilde{U}R$ is constructed with the CONSTANTTIMECUR algorithm of [16] and Section 2, $\|A - C\tilde{U}R\|_2 \leq \epsilon\|A\|_F$ holds with probability at least $7/8$. In the remainder of the proof, let us condition on this event holding. Thus, for all $x \in \{0,1\}^n$,

$$|x^T(A - C\tilde{U}R)(\vec{\mathbf{1}}_n - x)| \leq \epsilon n\|A\|_F/2. \tag{29}$$

Let x_1 be a vector that maximizes **MAX-CUT**$[C\tilde{U}R]$ and let x_2 be a vector that maximizes **MAX-CUT**$[A]$, i.e., x_1 and x_2 are such that

$$x_2^T A(\vec{\mathbf{1}}_n - x_2) = \textbf{MAX-CUT}[A] \tag{30}$$

$$x_1^T C\tilde{U}R(\vec{\mathbf{1}}_n - x_1) = \textbf{MAX-CUT}[C\tilde{U}R]. \tag{31}$$

Clearly then

$$x_1^T A(\vec{\mathbf{1}}_n - x_1) \leq \textbf{MAX-CUT}[A] \tag{32}$$

$$x_2^T CUR(\vec{\mathbf{1}}_n - x_2) \leq \textbf{MAX-CUT}[C\tilde{U}R]. \tag{33}$$

Since (29) holds for every $x \in \{0,1\}^n$ it follows from (29), (30), and (33) that

$$\textbf{MAX-CUT}[A] - \textbf{MAX-CUT}[C\tilde{U}R] \leq \epsilon n\|A\|_F/2. \tag{34}$$

Similarly, it follows from (29), (31), and (32) that

$$\textbf{MAX-CUT}[C\tilde{U}R] - \textbf{MAX-CUT}[A] \leq \epsilon n\|A\|_F/2. \tag{35}$$

The lemma follows by combining (34) and (35). ∎

At this point, it is not obvious that computing the **MAX-CUT**$[C\tilde{U}R]$ is any faster or simpler than computing the **MAX-CUT**$[A]$. However, we can exploit the special structure in $C\tilde{U}R$. Towards that end, let u, v be s dimensional vectors such that $u, v \in [-\sqrt{n}\|A\|_F, \sqrt{n}\|A\|_F]^s$, where $s = O(1/\epsilon^8)$ is the number of rows/columns that determines the size of the matrices in the $C\tilde{U}R$ decomposition, and define

$$\textbf{IP}(u,v) = \begin{cases} u - \frac{\epsilon}{s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \leq C^T x \leq u + \frac{\epsilon}{s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \\ v - \frac{\epsilon}{s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \leq R(\vec{\mathbf{1}}_n - x) \leq v + \frac{\epsilon}{s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \\ x \in \{0,1\}^n. \end{cases} \tag{36}$$

The vectors u and v will be used to approximate possible values for $C^T x$ and $R(\vec{\mathbf{1}}_n - x)$, respectively. Recall that

$$\Omega = [-\sqrt{n}\|A\|_F, \sqrt{n}\|A\|_F]^{2s}$$

and that Ω_Δ is the discretized set of vector pairs, where the discretization is $\Delta = \frac{\epsilon}{4s}\sqrt{n}\|A\|_F$ in each of the $2s$ directions. Let

$$\mathcal{F}_{\text{IP}} = \{(u,v) \in \Omega_\Delta : \textbf{IP}(u,v) \text{ is feasible}\} \tag{37}$$

denote the set of all pairs (u, v) such that the $\mathbf{IP}(u, v)$ is feasible and let $(\bar{u}, \bar{v})$ be the pair that maximizes $u^T \tilde{U} v$ over all pairs $(u, v) \in \mathcal{F}_{\text{IP}}$, and let

$$Z_{\text{IP}} = \max_{(u,v)\in\mathcal{F}_{\text{IP}}} u^T \tilde{U} v. \tag{38}$$

By the fineness of the discretization, $\mathcal{F}_{\text{IP}}$ is not empty. (Note that with the error permitted in IP (36) it would suffice to discretize Ω with the coarser discretization $\frac{\epsilon}{s}\sqrt{n}\|A\|_F$; in particular, doing so would suffice (up to failure probability which can be made small) to check whether there existed any $(u, v) \in \Omega$ for which x was exactly feasible. We opt for the finer discretization since, although it is wasteful in terms of constant factors at this point in the discussion, we will need the finer discretization later and in the interests of simplicity we prefer to work with a single grid.)

The next step of the proof is to show that Z_{IP} is close to $\mathbf{MAX\text{-}CUT}[C\tilde{U}R]$, where Z_{IP} is defined in (38). To this end, we first note the following lemma.

Lemma 2. *After accounting for the failure probability of Lemma* 1, *both of the following hold:*

$$|C^T x| \leq \sqrt{n}\|A\|_F \tag{39}$$

$$|R(\vec{\mathbf{1}}_n - x)| \leq \sqrt{n}\|A\|_F. \tag{40}$$

Proof. Note that $|C^T x| \leq \|C\|_F |x|$, that by Theorem 1 $\|C\|_F = \|A\|_F$, and also that $|x| \leq \sqrt{n}$ since $x \in \{0, 1\}^n$. Similarly for $|R(\vec{\mathbf{1}}_n - x)|$. ∎

The following lemma shows that Z_{IP} is an approximation to $\mathbf{MAX\text{-}CUT}[C\tilde{U}R]$.

Lemma 3.

$$|Z_{\text{IP}} - \mathbf{MAX\text{-}CUT}[C\tilde{U}R]| \leq O(\epsilon)n\|A\|_F.$$

Proof. Let x_1 be such that $x_1^T C\tilde{U}R(\vec{\mathbf{1}}_n - x_1) = \mathbf{MAX\text{-}CUT}[C\tilde{U}R]$. Then, there will exist some point (u_1, v_1) in $\mathcal{F}_{\text{IP}}$ such that the integer program $\mathbf{IP}(u_1, v_1)$ is feasible with $x = x_1$. Thus,

$$\begin{aligned}
\left|x_1^T C\tilde{U}R(\vec{\mathbf{1}}_n - x_1) - u_1^T \tilde{U}R(\vec{\mathbf{1}}_n - x_1)\right| &\leq \left|x_1^T C - u_1^T\right| \|\tilde{U}\|_2 \|R\|_2 \left|\vec{\mathbf{1}}_n - x_1\right| \\
&\leq \left(\frac{\epsilon}{\sqrt{s}}\sqrt{n}\|A\|_F\right)\left(\frac{O(1/\epsilon)}{\|A\|_F}\right)\|A\|_F\sqrt{n} \qquad (41)\\
&\leq O(\epsilon)n\|A\|_F. \qquad (42)
\end{aligned}$$

(41) follows from Theorem 1 and since

$$\left|x_1^T C - u_1^T\right| \leq \frac{\epsilon}{s}\sqrt{n}\|A\|_F \left(\sum_{i=1}^{s} 1\right)^{1/2} = \frac{\epsilon}{s}\sqrt{n}\|A\|_F\sqrt{s},$$

and (42) follows since $s = O(1/\epsilon^8)$. Similarly,

$$\begin{aligned}
\left|u_1^T \tilde{U}R(\vec{\mathbf{1}}_n - x_1) - u_1^T \tilde{U}v_1\right| &\leq \left|u_1^T\right| \|\tilde{U}\|_2 \left|R(\vec{\mathbf{1}}_n - x_1) - v_1\right| \\
&\leq (\sqrt{n}\|A\|_F)\left(\frac{O(1/\epsilon)}{\|A\|_F}\right)\left(\frac{\epsilon}{\sqrt{s}}\sqrt{n}\|A\|_F\right) \\
&\leq O(\epsilon)n\|A\|_F. \qquad (43)
\end{aligned}$$

By summing (42) and (43) and noticing that

$$\textbf{MAX-CUT}[C\tilde{U}R] \geq Z_{\text{IP}} = \max_{(u,v)\in\mathcal{F}_{\text{IP}}} u^T\tilde{U}v \geq u_1^T\tilde{U}v_1$$

the lemma follows. ∎

By combining Lemmas 1 and 3, we have reduced the problem of approximating **MAX-CUT**$[A]$ to the problem of finding the pair $(\bar{u},\bar{v})$ that maximizes $u^T\tilde{U}v$ over all pairs $(u,v)\in\mathcal{F}_{\text{IP}}$ and then returning Z_{IP}. Thus, to prove Theorem 4, it suffices to prove that Z_{LPQ}, which is returned by the ApproximateMaxCut algorithm and which is defined in (50), is with high probability an approximation to Z_{IP} to within additive error $O(\epsilon)n\|A\|_F$.

To this end, let us consider a single (u,v) pair. We relax the IP (36) by removing the integrality constraints and we also slightly tighten the other constraints by $\frac{\epsilon}{2s}\sqrt{n}\|A\|_F$. Note that we do not need a finer discretization since the original discretization was overly fine for the IPs (36). Define

$$\textbf{LP}(u,v) = \begin{cases} u - \frac{\epsilon}{2s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \leq \sum_{i=1}^n C_{(i)}x_i \leq u + \frac{\epsilon}{2s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \\ v - \frac{\epsilon}{2s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \leq \sum_{i=1}^n R^{(i)}(1-x_i) \leq v + \frac{\epsilon}{2s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \\ x_i \in [0,1]. \end{cases} \tag{44}$$

The particular tightening chosen above guarantees that if $\textbf{IP}(u,v)$ is infeasible then $\textbf{LP}(u,v)$ is infeasible, as is proven in Lemma 4. Let us first write LP (44) in the form (19):

$$Px = \begin{bmatrix} C^T \\ -R \\ -C^T \\ R \end{bmatrix} x \leq \begin{pmatrix} u + \frac{\epsilon}{2s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \\ v + \frac{\epsilon}{2s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s - R\vec{\mathbf{1}}_n \\ -u + \frac{\epsilon}{2s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s \\ -v + \frac{\epsilon}{2s}\sqrt{n}\|A\|_F\vec{\mathbf{1}}_s + R\vec{\mathbf{1}}_n \end{pmatrix} = b, \tag{45}$$

where $P\in\mathbb{R}^{4s\times n}$ and $b\in\mathbb{R}^{4s}$. Note that the following lemma does not involve randomization and thus always holds.

Lemma 4. *If* $\textbf{IP}(u,v)$ *is infeasible, then* $\textbf{LP}(u,v)$ *is infeasible.*

Proof. Assume that $\textbf{LP}(u,v)$ has a feasible solution. Then, since there are at most $4s$ constraints in (45), there exists a feasible solution $\textbf{LP}(u,v)$ with at most $4s$ fractional values. By rounding these fractional values up to 1, the constraints of (45) may be violated, but not by more than $4s\max_{i,j}\{|C_{ij}|,|R_{ij}|\} \leq 4s\|A\|_F$. Since asymptotically $s^2 \leq \epsilon\sqrt{n}/8$, if the constraints of LP (45) are relaxed by $(\epsilon/2s)\sqrt{n}\|A\|_F$, then the new constraints are satisfied by the integral-valued solution. Thus, $\textbf{IP}(u,v)$ has a feasible solution, and the lemma follows. ∎

Since each $\textbf{LP}(u,v)$ of the form (44) or (45) is an LP over n variables, checking its feasibility is easy, but unfortunately requires $poly(n)$ time. (Recall that we will provide a constant time algorithm for Max-Cut.) Thus, we will now use the results of Section 3 to reduce the problem

to checking the feasibility of an induced linear program with a constant (independent of n) number of variables. So, we perform random sampling and construct a q-variable LP which is a subprogram of the $\mathbf{LP}(u, v)$. To sample from the original LP, we randomly sample q columns of P; we do this by choosing q elements from $\{1, 2, \ldots n\}$ in i.i.d. trials, where the probability of picking i is

$$w_i = \frac{|C_{(i)}, R^{(i)}|^2}{\sum_{j=1}^{n} |C_{(j)}, R^{(j)}|^2}, \tag{46}$$

where $|C_{(i)}, R^{(i)}|^2 = \sum_{j=1}^{s} (C_{ij}^2 + R_{ji}^2)$. Since x is such that $0 \leq x_i \leq 1$ for every i, these probabilities are of the form (20) for the P matrix given by (45). Let Q denote the subset of $\{1, 2, \ldots, n\}$ that is picked. Let $\tilde{P}$ be the corresponding sampled (and appropriately rescaled) matrix, and let $\tilde{C}$ and $\tilde{R}$ be the corresponding sampled and rescaled versions of C and R (formed from the sampled q rows of C and the corresponding q columns of R). Since $\|P\|_F^2 = 4\|A\|_F^2$, if we tighten the constraints by $\frac{2}{\delta^* \sqrt{q}} \sqrt{n} \|A\|_F$, where δ^* is a failure probability in Lemma 5 below, then the constraints from the sampled version of the LP take the form:

$$u - \left(\frac{\epsilon}{2s} - \frac{2}{\delta^* \sqrt{q}} \right) \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s \leq \sum_{i \in Q} \frac{1}{q w_i} C_{(i)} x_i \leq u + \left(\frac{\epsilon}{2s} - \frac{2}{\delta^* \sqrt{q}} \right) \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s$$

$$v - \left(\frac{\epsilon}{2s} - \frac{2}{\delta^* \sqrt{q}} \right) \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s \leq R\vec{\mathbf{1}}_n - \sum_{i \in Q} \frac{1}{q w_i} R^{(i)} x_i \leq v + \left(\frac{\epsilon}{2s} - \frac{2}{\delta^* \sqrt{q}} \right) \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s,$$

or equivalently,

$$\tilde{P}\tilde{x} = \begin{bmatrix} \tilde{C}^T \\ -\tilde{R} \\ -\tilde{C}^T \\ \tilde{R} \end{bmatrix} \tilde{x} \leq \begin{pmatrix} u + \left(\frac{\epsilon}{2s} - \frac{2}{\delta^* \sqrt{q}} \right) \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s \\ v + \left(\frac{\epsilon}{2s} - \frac{2}{\delta^* \sqrt{q}} \right) \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s - R\vec{\mathbf{1}}_n \\ -u + \left(\frac{\epsilon}{2s} - \frac{2}{\delta^* \sqrt{q}} \right) \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s \\ -v + \left(\frac{\epsilon}{2s} - \frac{2}{\delta^* \sqrt{q}} \right) \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s + R\vec{\mathbf{1}}_n \end{pmatrix}. \tag{47}$$

With this tightening, the infeasibility of LP (45) implies, with high probability, the infeasibility of the sampled LP, as is proven in Lemma 5. Since δ^* and q are at our disposal, let us choose them such that $1/\delta^* q = \epsilon/4s$; then we may define

$$\mathbf{LP}_Q(u, v) = \begin{cases} u - \frac{\epsilon}{4s} \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s \leq \sum_{i \in Q} \frac{1}{q w_i} C_{(i)} x_i \leq u + \frac{\epsilon}{4s} \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s \\ v - \frac{\epsilon}{4s} \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s \leq R\vec{\mathbf{1}}_n - \sum_{i \in Q} \frac{1}{q w_i} R^{(i)} x_i \leq v + \frac{\epsilon}{4s} \sqrt{n} \|A\|_F \vec{\mathbf{1}}_s \\ x_i \in [0, 1], \quad i \in Q \end{cases} \tag{48}$$

to be the sampled version of the LP (44). The following lemma is an immediate consequence of Theorem 3 when LP (44) is written in the form (45).

Lemma 5. *If* $\mathbf{LP}(u, v)$ *is infeasible, then with probability at least* $1 - \delta^*$ $\mathbf{LP}_Q(u, v)$ *is infeasible.*

Proof. Apply Theorem 3, noting that the sampling probabilities (46) are of the form (20). ∎

Thus, by combining Lemmas 4 and 5 we see that, for every (u, v) pair, subject to a failure probability which is less than δ^*, the feasibility of LP (48) implies the feasibility of IP (36).

Let us now consider again all (u, v) pairs. Let

$$\mathcal{F}_{\mathrm{LPQ}} = \{(u, v) \in \Omega_\Delta : \mathbf{LP}_Q(u, v) \text{ is feasible}\} \tag{49}$$

denote the set of all pairs (u, v) such that the $\mathbf{LP}_Q(u, v)$ is feasible, let $(\bar{u}, \bar{v})$ be the pair that maximizes $u^T \tilde{U} v$ over all pairs $(u, v) \in \mathcal{F}_{\mathrm{LPQ}}$, and let

$$Z_{\mathrm{LPQ}} = \max_{(u,v)\in\mathcal{F}_{\mathrm{LPQ}}} u^T \tilde{U} v \tag{50}$$

be the value returned by the ApproximateMaxCut algorithm. Thus, to prove Theorem 4, it suffices to prove that Z_{LPQ} is a good approximation to Z_{IP}; we do this in Lemmas 6 and 7.

To this end, note that $|\Omega_\Delta| = (8s/\epsilon)^{2s}$. Since $s = \kappa/\epsilon^8$ for some constant κ, let us choose

$$\delta^* = \frac{1}{8}\left(\frac{\epsilon^9}{8\kappa}\right)^{2\kappa/\epsilon^8}, \tag{51}$$

and then let us choose

$$q = \frac{32\kappa}{\epsilon^9}\left(\frac{8\kappa}{\epsilon^9}\right)^{2\kappa/\epsilon^8} = \mathrm{poly}(1/\epsilon)\exp(\mathrm{poly}(1/\epsilon)). \tag{52}$$

Lemma 6. *With probability at least* $7/8$, $Z_{\mathrm{LPQ}} \le Z_{\mathrm{IP}}$.

Proof. By combining Lemmas 4 and 5 we see that the for every $(u, v) \in \Omega_\Delta$ the feasibility of $\mathbf{LP}_Q(u, v)$ implies, with probability at least $1 - \delta^*$ the feasibility of $\mathbf{IP}(u, v)$. The probability that there exists a $(u, v) \in \Omega_\Delta$ for which the implication fails to hold is bounded above by $\sum_{i=1}^{|\Omega_\Delta|} \delta^*$, which by the choice of δ in (51) is at most $1/8$. Thus, with probability at least $7/8$, $\mathcal{F}_{\mathrm{LPQ}} \subset \mathcal{F}_{\mathrm{IP}}$; in this case, the maximization in (38) is over a larger set than the maximization in (50), and the lemma follows. ∎

Lemma 7. *After accounting for the failure probability of Lemma* 6 *and Lemma* 1,

$$Z_{\mathrm{IP}} - O(\epsilon) n \|A\|_F \le Z_{\mathrm{LPQ}}.$$

Proof. Let $(\bar{u}, \bar{v})$ be a pair that achieves the maximum of $u^T \tilde{U} v$ over all $(u, v) \in \mathcal{F}_{\mathrm{IP}}$; i.e., $\bar{u}^T \tilde{U} \bar{v} = Z_{\mathrm{IP}}$. Although $(\bar{u}, \bar{v})$ may not be in $\mathcal{F}_{\mathrm{LPQ}}$, note that by the error permitted in IP (36) and by the choice of the discretization Δ, there exists a $(u, v) \in \mathcal{F}_{\mathrm{LPQ}}$ that is close to $(\bar{u}, \bar{v})$. More precisely, subject to the failure probability of Lemma 6, there exists a $(u, v) \in \mathcal{F}_{\mathrm{LPQ}}$ such that $u = \bar{u} + \alpha$ and $v = \bar{v} + \beta$ for some vectors $\alpha, \beta \in \mathbb{R}^s$, where every element of α and β is no greater than $\frac{\epsilon}{s}\sqrt{n}\|A\|_F$. In this case, $|\alpha| \le \frac{\epsilon}{\sqrt{s}}\sqrt{n}\|A\|_F$ and $|\beta| \le \frac{\epsilon}{\sqrt{s}}\sqrt{n}\|A\|_F$. Recall that $\|\tilde{U}\|_2 \le O(1/\epsilon)/\|A\|_F$, subject to the failure probability of Lemma 1. In addition, note that

$$|\bar{u}| \le |C^T x + \frac{\epsilon}{s}\sqrt{n}\|A\|_F \vec{\mathbf{1}}_s| \le \left(1 + \frac{\epsilon}{\sqrt{s}}\right)\sqrt{n}\|A\|_F,$$

and similarly that

$$|\bar{v}| \leq \left| R(\vec{\mathbf{1}}_s - x) + \frac{\epsilon}{s}\sqrt{n}\|A\|_F \vec{\mathbf{1}}_s \right| \leq \left(1 + \frac{\epsilon}{\sqrt{s}}\right)\sqrt{n}\|A\|_F.$$

Thus, since $u^T\tilde{U}v = (\bar{u}^T + \alpha^T)\tilde{U}(\bar{v} + \beta)$, we have that

$$\begin{aligned} |u^T\tilde{U}v - \bar{u}^T\tilde{U}\bar{v}| &\leq |\bar{u}^T\tilde{U}\beta| + |\alpha^T\tilde{U}\bar{v}| + |\alpha\tilde{U}\beta| \\ &\leq O(\epsilon)n\|A\|_F. \end{aligned}$$

Thus,

$$Z_{\text{IP}} = \bar{u}^T\tilde{U}\bar{v} \leq u^T\tilde{U}v + O(\epsilon)n\|A\|_F \leq Z_{\text{LPQ}} + O(\epsilon)n\|A\|_F,$$

where the second inequality is since $(u, v) \in \mathcal{F}_{\text{LPQ}}$. The lemma follows. ∎

Theorem 4 follows by combining Lemmas 6 and 7 with Lemmas 1 and 3.

5. CONCLUSION

We have been interested in developing improved methods to compute approximate solutions to certain *NP*-hard optimization problems that arise in applications of graph theory and that have significant heterogeneities and/or nonuniformities. We should note that our results do not assume that the matrix A has positive elements; we note also that, although it is less relevant to the Max-Cut problem, our $C\tilde{U}R$ decomposition and our LP lemmas do not assume that the matrix A is symmetric; thus, e.g., they would apply to problems involving directed and weighted directed graphs.

It is worth considering how our Linear Programming results relate to previous work on perturbed LPs, e.g., [32, 33, 35]. Renegar was interested in developing a complexity theory in which problem instance data is allowed to consist of real, and not simply rational, data; customary measures of size were replaced with condition measures [32, 33]. If one considers the linear program with constraints $Px \leq b$, $x \geq 0$, then in order to decide whether an instance $\mathcal{D} = (P, b)$ is a consistent system of constraints, the condition measure of instance $\mathcal{D}$ with respect to the feasibility decision problem is defined such that its inverse is the minimal relative perturbation of the data vector $\mathcal{D}$ to obtain a system from $\mathcal{D}$ whose answer for the decision problem is different than the answer for $\mathcal{D}$. Spielman and Teng were interested in studying the performance of algorithms under small random perturbation of their inputs in order to, e.g., explain why certain algorithms with inconclusive or negative theoretical results perform quite well in practice [35]. They considered LP of the form: max z^Tx s.t. $Px \leq b$, and they studied the performance of the algorithm under slight random perturbations on the inputs. In particular, they replace the previous LP with one of the form: max z^Tx s.t. $(P + \sigma G)x \leq b$, where G is a matrix of independently chosen Gaussian random variables of mean 0 and variance 1 and where σ is a polynomially small variance parameter.

Our perturbed LP results are somewhat different in that we only consider perturbations of the left hand side of $Px \leq b$ of a particular form and we then construct a right hand side, i.e., $b \pm \delta b\vec{\mathbf{1}}_r$, where $\delta b \in \mathbb{R}$, such that the new LP has a feasibility status that is relatable to that of the original LP. This is different than but related to the intuition that if the LP is "very feasible" or "very infeasible" (in the sense that all of the inequalities are easily satisfied or

that there exists an inequality that is far from being satisfied) then a slight perturbation of the data matrix (P, b) should not change the feasibility status of the LP.

Several directions present themselves for future work. The first and most obvious is to improve upon the present results by improving the sampling complexity with respect to $1/\epsilon$. A more ambitious improvement would be developing a PTAS for certain classes of subdense graphs. Another extension would be to construct nonuniformly an *induced* sub-problem, from which the solution to the original problem could be approximated; this would be more in line with the results of [1]. A final extension involves generalizing this work to all Max-2-CSP and more generally to all Max-r-CSP problems. Extending our results to all problems in Max-r-CSP, $r > 2$, necessitates the design of an efficient $C\tilde{U}R$-type approximation for multi-dimensional arrays. The significant challenges involved in the design of a multi-dimensional version of our algorithm are worth considering since such an algorithm might be useful in applications where multi-dimensional data are involved.

ACKNOWLEDGEMENTS

We would like to thank an anonymous reviewer for carefully reading the manuscript and providing numerous comments that significantly improved the presentation of our work.

REFERENCES

[1] N. Alon, W. F. de la Vega, R. Kannan, and M. Karpinski, Random sampling and approximation of MAX-CSPs, J Comput Syst Sci 67 (2003), 212–243.

[2] N. Alon and A. Naor, Approximating the cut-norm via Grothendieck's inequality, SIAM J Comput 35 (2006), 787–803.

[3] S. Arora, D. Karger, and M. Karpinski, Polynomial time approximation schemes for dense instances of NP-hard problems, In Proceedings of the 27th Annual ACM Symposium on Theory of Computing, 1995, pp. 284–293.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and the hardness of approximation problems, In Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, 1992, pp. 14–23.

[5] Z. Bar-Yossef, The complexity of massive data set computations, PhD thesis, University of California, Berkeley, 2002.

[6] Z. Bar-Yossef, Sampling lower bounds via information theory, In Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 335–344.

[7] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design, Oper Res 36 (1988), 493–513.

[8] W. F. de la Vega, MAX-CUT has a randomized approximation scheme in dense graphs, Random Struct Algorithms 8 (1996), 187–198.

[9] W. F. de la Vega and M. Karpinski, A polynomial time approximation scheme for subdense MAX-CUT, Technical Report TR02-044, Electronic Colloquium on Computational Complexity, 2002.

[10] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, Clustering in large graphs and matrices, In Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999, pp. 291–299.

[11] P. Drineas and R. Kannan, Fast Monte-Carlo algorithms for approximate matrix multiplication, In Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 452–459.

[12] P. Drineas and R. Kannan, Pass efficient algorithms for approximating large matrices, In Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 223–232.

[13] P. Drineas, R. Kannan, and M. W. Mahoney, Sampling sub-problems of heterogeneous Max-Cut problems and approximation algorithms, In Proceedings of the 22nd Annual International Symposium on Theoretical Aspects of Computer Science, 2005, pp. 57–68.

[14] P. Drineas, R. Kannan, and M. W. Mahoney, Fast Monte Carlo algorithms for matrices. I. Approximating matrix multiplication, SIAM J Comput 36 (2006), 132–157.

[15] P. Drineas, R. Kannan, and M. W. Mahoney, Fast Monte Carlo algorithms for matrices. II. Computing a low-rank approximation to a matrix, SIAM J Comput 36 (2006), 158–183.

[16] P. Drineas, R. Kannan, and M. W. Mahoney, Fast Monte Carlo algorithms for matrices. III. Computing a compressed approximate matrix decomposition, SIAM J Comput 36 (2006), 184–206.

[17] A. Frieze and R. Kannan, The regularity lemma and approximation schemes for dense problems, In Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 1996, pp. 12–20.

[18] A. Frieze and R. Kannan, Quick approximation to matrices and applications, Combinatorica 19 (1999), 175–220.

[19] A. Frieze, R. Kannan, and S. Vempala, Fast Monte-Carlo algorithms for finding low-rank approximations, In Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, 1998, pp. 370–378.

[20] M. X. Goemans and D. P. Williamson, .878-approximation algorithms for MAX CUT and MAX 2SAT, In Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 422–431.

[21] O. Goldreich, S. Goldwasser, and D. Ron, Property testing and its connection to learning and approximation, In Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 1996, pp. 339–348.

[22] G. H. Golub and C. F. Van Loan, Matrix computations, Johns Hopkins University Press, Baltimore, 1989.

[23] J. Håstad, Some optimal inapproximability results, J ACM 48 (2001), 798–859.

[24] R. A. Horn and C. R. Johnson, Matrix analysis, Cambridge University Press, New York, 1985.

[25] S. Khanna, M. Sudan, L. Trevisan, and D. P. Williamson, The approximability of constraint satisfaction problems, SIAM J Comput 30 (2001), 1863–1920.

[26] S. Khot, On the power of unique 2-prover 1-round games, In Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 767–775.

[27] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell, Optimal inapproximability results for Max-Cut and other 2-variable CSPs? In Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, 2004, pp. 146–154.

[28] R. Kumar and R. Rubinfeld, Sublinear time algorithms, Samir Khuller's Algorithms column in SIGACT News 2003.

[29] J. S. Liu, Monte Carlo strategies in scientific computing, Springer-Verlag, New York, 2001.

[30] M. E. J. Newman, The structure and function of complex networks, SIAM Rev 45 (2003), 167–256.

[31] S. Poljak and Z. Tuza, "Maximum cuts and large bipartite subgraphs," In Combinatorial optimization, DIMACS series in discrete mathematics and theoretical computer Science,

Vol. 20, W. Cook, L. Lovász, P. Seymour (Editors), American Mathematical Society, 1995, pp. 181–244.

[32] J. Renegar, Some perturbation theory for linear programming, Math Program 65 (1994), 73–91.

[33] J. Renegar, Incorporating condition measures into the complexity theory of linear programming, SIAM J Optimization 5 (1995), 506–524.

[34] A. Schrijver, Theory of linear and integer programming, Wiley, New York, 1998.

[35] D. Spielman and S. H. Teng, Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time, In Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 296–305.

[36] G. W. Stewart and J. G. Sun, Matrix perturbation theory, Academic Press, New York, 1990.