Sampling Sub-problems of Heterogeneous Max-cut Problems and Approximation Algorithms

Petros Drineas[1], Ravi Kannan[2], and Michael W. Mahoney[3]

[1] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York 12180
`drinep@cs.rpi.edu`

[2] Department of Computer Science, Yale University, New Haven, CT 06520
`kannan@cs.yale.edu`

[3] Department of Mathematics, Yale University, New Haven, CT 06520
`mahoney@cs.yale.edu`

Extended Abstract

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 sub-problem of constant size, where the sub-problem 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 sub-optimal 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 weighted and unweighted 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 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 sub-program 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.

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/or 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 [12, 13, 14].

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.

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 [3, 6, 16, 18, 17, 1, 2]. [3] and [6], 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 [18] 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. [16] and [17] examined the regularity properties of dense graphs and developed a new method to approximate matrices; 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, 2] 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 sub-problem which is then used to compute an approximation to the original problem with additive error at most ϵn^2 [3, 6, 16, 18, 17, 1, 2]; 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 [16].

In many applications of graph theory problems, however, significant heterogeneities are present [22]. 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 captures well this phenomenon [18].

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 [21]. 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 [12, 13, 14]; see also [8, 9, 10]. 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 [4, 5]. For instance, it was demonstrated that in certain cases approximation algorithms (for matrix problems such as those discussed in [12, 13, 14]) which use oblivious uniform sampling cannot achieve the error bounds that are achieved by adaptively constructing nonuniform sampling probabilities [4, 5].

1.2 Our Main Result

In this paper we develop an approximation algorithm for both weighted and unweighted 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 weighted graphs, these methods lead to substantial improvements when the average edge weight is much less than the maximum edge weight; for unweighted graphs, we show that at the cost of substantial additional sampling, these methods lead to an additive error improvement over previous results [18, 2].

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 paper, which is presented in a more precise form in Theorem 3, 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

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

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.

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 $\epsilon n^2 W_{max}$.

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 [18, 2]. 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 sub-dense 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 recently 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 [7]. 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 [18, 2]; we expect that this may be substantially improved.

1.3 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 [14] 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 summarize a proof of Theorem 3 which establishes (1).

1.4 Technical Report and Journal Paper

For the proofs of the results presented here, as well as for more details and discussion related to these results, see the associated technical report [15]. Also, see the associated journal paper [11].

2 Review of Relevant Background

This section contains a review of linear algebra that will be useful throughout the paper; for more detail, see [19, 20, 23] 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 [14] 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 [12, 13, 14] for more details.

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 $|x| = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}$. 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, then for $k \leq \rho$ define $A_k = \sum_{t=1}^{k} \sigma_t u^t {v^t}^T$.

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

In [14] we presented and analyzed two algorithms to compute compressed approximate decompositions of a matrix $A \in \mathbb{R}^{m\times n}$. The second approximation (computed with the ConstantTimeCUR algorithm of [14]) is of the form $A' = C\tilde{U}R$, where C is an $m \times c$ matrix consisting of c randomly picked (and suitably rescaled) columns of A, R is an $r \times n$ matrix consisting of r randomly picked (and suitably rescaled) rows of A; the algorithm constructs the $w \times c$ matrix W consisting of w randomly picked (and suitably rescaled) rows of C, and from the SVD of W^TW constructs the $c \times r$ matrix $\tilde{U}$. The $C\tilde{U}R$ approximation may be defined after making three passes through the data matrix A, and $\tilde{U}$ can be constructed using additional RAM space and time that is $O(1)$. In the following theorem we let $c = w = r = s$ for simplicity. Note also that γ is a parameter and k is the rank of the approximation; see [14] for a full discussion and definition of notation.

Theorem 1. *Suppose $A \in \mathbb{R}^{m\times n}$ and let C, $\tilde{U}$, and R be constructed from the* ConstantTimeCUR *algorithm by sampling s columns of A (and then sampling s rows of C) and s rows of A. Let $\delta, \epsilon > 0$. If a spectral norm bound is desired, and hence the* ConstantTimeCUR *algorithm of [14] is run with $\gamma = \Theta(\epsilon)$ and $s = \Omega\left(1/\epsilon^8\right)$, then under appropriate assumptions on the sampling probabilities we have that with probability at least $1 - \delta$ each of the following holds: $\|C\|_F = \|A\|_F$, $\left\|\tilde{U}\right\|_2 \leq O(1/\epsilon)/\|A\|_F$, $\|R\|_F = \|A\|_F$, and $\left\|A - C\tilde{U}R\right\|_2$*

$\leq \|A - A_k\|_2 + \epsilon \|A\|_F$. Thus, if we choose $k = 1/\epsilon^2$ (and $s = \Omega\left(1/\epsilon^8\right)$) then with probability at least $1 - \delta$

$$\left\|A - C\tilde{U}R\right\|_2 \leq \epsilon \|A\|_F . \tag{2}$$

3 Sampling Sub-programs of a Linear Program

In this section, we examine relating the feasibility or infeasibility of a Linear Program to the feasibility or infeasibility of a randomly sampled version of that LP.

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 Linear Program:*

$$Px = \sum_{i=1}^{n} P^{(i)} x_i \leq b \qquad 0 \leq x_i \leq c_i. \tag{3}$$

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

$$p_i = \mathbf{Pr}\left[i_t = i\right] = \frac{\left|P^{(i)}\right|^2}{\|P\|_F^2}. \tag{4}$$

Let $\mathbf{1}_r$ denote the $r \times 1$ all-ones vector. If the Linear Program (3) is feasible then, with probability at least $1 - \delta$

$$\sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i \leq b + \frac{1}{\delta\sqrt{q}} |x| \, \|P\|_F \mathbf{1}_r \qquad 0 \leq x_i \leq c_i \ , \, i \in Q \tag{5}$$

is feasible as well. If the Linear Program (3) is infeasible then, with probability at least $1 - \delta$

$$\sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i \leq b - \frac{1}{\delta\sqrt{q}} |x| \, \|P\|_F \mathbf{1}_r \qquad 0 \leq x_i \leq c_i \ , \, i \in Q \tag{6}$$

is infeasible as well.

Proof: We first claim that Px is well approximated by $\tilde{P}\tilde{x} = \sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i$, i.e., that

$$\mathbf{Pr}\left[\left\|Px - \tilde{P}\tilde{x}\right\|_F \leq \frac{1}{\delta\sqrt{q}} |x| \, \|P\|_F\right] \geq 1 - \delta. \tag{7}$$

To establish the claim, first note that

$$\mathbf{E}\left[\left\|Px - \tilde{P}\tilde{x}\right\|_F\right] \leq \left(\frac{1}{q}\sum_{i=1}^{n} \frac{1}{p_i} \left|P^{(i)}\right|^2 |x_i|^2\right)^{1/2} \leq \frac{1}{\sqrt{q}} |x| \, \|P\|_F , \tag{8}$$

and then apply Markov's inequality. (5) then follows immediately since there exists a vector v such that with high probability $\tilde{P}\tilde{x} = Px + v$, where $|v| \leq \frac{1}{\delta\sqrt{q}} |x| \, \|P\|_F$. (6) follows by using LP duality. For details (and for a related theorem), see [15].

$\diamond$

Note that establishing (7) in Theorem 2 uses ideas that are very similar to those used in [12] for approximating the product of two matrices. Once we are given (7) then the proof of (5) 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 (6) is more difficult; we must show that the non-existence of a solution of the original LP implies the same for the randomly sampled version of the LP. Fortunately, by LP duality theory the non-existence of a solution in the LP implies the existence of a certain solution in a related LP.

A special case of these results occurs when $c_i = 1$ for all i, since in that case the Cauchy-Schwartz inequality implies $\sum_{i=1}^{n} \left|P^{(i)}\right| \leq \sqrt{n} \, \|P\|_F$. The induced LPs (5) and (6) in Theorem 2 may then be replaced by

$$\sum_{i \in Q} \frac{1}{qp_i} P^{(i)} x_i \leq b \pm \frac{1}{\delta} \sqrt{\frac{n}{q}} \, \|P\|_F \, \mathbf{1}_r \qquad 0 \leq x_i \leq c_i \, , \, i \in Q. \tag{9}$$

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:

$$\textbf{MAX-CUT}\,[G] = \textbf{MAX-CUT}\,[A] = \max_{x \in \{0,1\}^n} x^T A(\mathbf{1}_n - x), \tag{10}$$

where $\mathbf{1}_n$ 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.

4.1 The Algorithm

Consider the ApproximateMaxCut algorithm which is presented in Figure 1 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 $\textbf{MAX-CUT}\,[A]$. In order to compute Z_{LPQ}, the ApproximateMaxCut algorithm uses the ConstantTimeCUR algorithm of [14] 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 rows of C and the corresponding q columns of R, each appropriately rescaled, are

ApproximateMaxCut Algorithm

Input: $A \in \mathbb{R}^{n \times n}$, the weighted adjacency matrix of a graph $G = (V, E)$, 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 of [14]) and store the $s \times s$ matrix $\tilde{U}$.
- Compute and store the matrices $\tilde{C}$ and $\tilde{R}$.
- 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 $\mathbf{LP}_Q(u, v)$

$$u - \tfrac{\epsilon}{4s}\sqrt{n}\|A\|_F \mathbf{1}_s \le \quad \textstyle\sum_{i \in Q} \tfrac{1}{qw_i} C_{(i)} x_i \quad \le u + \tfrac{\epsilon}{4s}\sqrt{n}\|A\|_F \mathbf{1}_s$$

$$v - \tfrac{\epsilon}{4s}\sqrt{n}\|A\|_F \mathbf{1}_s \le R\mathbf{1}_n - \textstyle\sum_{i \in Q} \tfrac{1}{qw_i} R^{(i)} x_i \le v + \tfrac{\epsilon}{4s}\sqrt{n}\|A\|_F \mathbf{1}_s$$

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

  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. 1. The ApproximateMaxCut Algorithm

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** $[A]$.

4.2 Analysis of the Implementation and Running Time

The ApproximateMaxCut algorithm may be implemented with three passes over the matrix and constant (with respect to n, but exponential in $1/\epsilon$) additional space and time. For details, see [15].

4.3 The Main Theorem

Theorem 3 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 3. *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* $3/4$

$$|Z_{LPQ} - \textbf{MAX-CUT}\,[A]| \leq \epsilon n \left\|A\right\|_F .$$

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 3

In order to prove Theorem 3, 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 [17, 2] for the case of uniform sampling.

In the *first* level of approximation we will use the ConstantTimeCUR algorithm of [14] 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. As discussed in [14] 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 $\left\|A - C\tilde{U}R\right\|_2 \leq \epsilon \left\|A\right\|_F$ with high probability. A good approximation to $\textbf{MAX-CUT}\,[A]$ is provided by $\textbf{MAX-CUT}\left[C\tilde{U}R\right]$, which is the Max-Cut of the graph whose weighted adjacency matrix is $C\tilde{U}R$. Thus, it suffices to approximate well

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

Since with high probability $\left|C^T x\right| \leq \sqrt{n}\left\|A\right\|_F$ and $|R(\mathbf{1}_n - x)| \leq \sqrt{n}\left\|A\right\|_F$, both $C^T x$ and $R(\mathbf{1}_n - x)$ lie in $[-\sqrt{n}\left\|A\right\|_F, \sqrt{n}\left\|A\right\|_F]^s$. Consider the set of vectors $(u, v) \in \Omega$, where $\Omega = [-\sqrt{n}\left\|A\right\|_F, \sqrt{n}\left\|A\right\|_F]^{2s}$. 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
$$C^T\bar{x} = \bar{u} \qquad \text{and} \qquad R(\mathbf{1}_n - \bar{x}) = \bar{v},$$
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 $\bar{u}^T\tilde{U}\bar{v} = \textbf{MAX-CUT}\left[C\tilde{U}R\right]$.

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

$$\begin{aligned} \bar{u} - \frac{\epsilon}{s}\sqrt{n}\,\|A\|_F\,\mathbf{1}_s \le \quad C^T\bar{x} \quad &\le \bar{u} + \frac{\epsilon}{s}\sqrt{n}\,\|A\|_F\,\mathbf{1}_s \text{ and} \\ \bar{v} - \frac{\epsilon}{s}\sqrt{n}\,\|A\|_F\,\mathbf{1}_s \le R(\mathbf{1}_n - \bar{x}) &\le \bar{v} + \frac{\epsilon}{s}\sqrt{n}\,\|A\|_F\,\mathbf{1}_s, \end{aligned}$$

i.e., there exists a vector $\bar{x} \in \{0,1\}^n$ such that the pair $(\bar{u}, \bar{v})$ satisfies

$$C^T\bar{x} \approx \bar{u} \qquad \text{and} \qquad R(\mathbf{1}_n - \bar{x}) \approx \bar{v},$$

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. 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 $\mathbf{MAX\text{-}CUT}\left[C\tilde{U}R\right]$. At this point we have reduced the problem of approximating $\mathbf{MAX\text{-}CUT}\,[A]$ to that of checking the feasibility of a large but constant number of IPs 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 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 which is a slight tightening of the LP relaxation of the IP; 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.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 a small constant. Theorem 3 will then follow.

For details of the proof, see [15].

References

1. N. Alon, W.F. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of MAX-CSP problems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 232–239, 2002.
2. N. Alon, W.F. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of MAX-CSPs. *Journal of Computer and System Sciences*, 67:212–243, 2003.
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*, pages 284–293, 1995.
4. Z. Bar-Yossef. *The Complexity of Massive Data Set Computations.* PhD thesis, University of California, Berkeley, 2002.
5. Z. Bar-Yossef. Sampling lower bounds via information theory. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 335–344, 2003.
6. W.F. de la Vega. MAX-CUT has a randomized approximation scheme in dense graphs. *Random Structures and Algorithms*, 8(3):187–198, 1996.
7. 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.
8. 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*, pages 291–299, 1999.
9. 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*, pages 452–459, 2001.
10. P. Drineas and R. Kannan. Pass efficient algorithms for approximating large matrices. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 2003.
11. P. Drineas, R. Kannan, and M.W. Mahoney. Sampling sub-problems of heterogeneous MAX-2-CSP problems and approximation algorithms. *manuscript.*
12. P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. Technical Report YALEU/DCS/TR-1269, Yale University Department of Computer Science, New Haven, CT, February 2004.
13. P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. Technical Report YALEU/DCS/TR-1270, Yale University Department of Computer Science, New Haven, CT, February 2004.
14. P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. Technical Report YALEU/DCS/TR-1271, Yale University Department of Computer Science, New Haven, CT, February 2004.
15. P. Drineas, R. Kannan, and M.W. Mahoney. Sampling sub-problems of heterogeneous Max-Cut problems and approximation algorithms. Technical Report YALEU/DCS/TR-1283, Yale University Department of Computer Science, New Haven, CT, April 2004.
16. 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*, pages 12–20, 1996.

17. A. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.
18. 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*, pages 339–348, 1996.
19. G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 1989.
20. R.A. Horn and C.R. Johnson. *Matrix Analysis.* Cambridge University Press, New York, 1985.
21. J.S. Liu. *Monte Carlo Strategies in Scientific Computing.* Springer-Verlag, New York, 2001.
22. M.E.J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.
23. G.W. Stewart and J.G. Sun. *Matrix Perturbation Theory.* Academic Press, New York, 1990.