Appeared in *Proceedings, 28th Annual ACM Symposium on Theory of Computing* (May 1996), pp. 338–347.

Efficient Approximation Algorithms for Semidefinite Programs Arising from MAX CUT and COLORING

Philip Klein
Department of Computer Science
Brown University
klein@cs.brown.edu

Hsueh-I Lu
Department of Computer Science
Brown University
hil@cs.brown.edu

Revised on July 30, 1996

Abstract

The best known approximation algorithm for graph MAX CUT, due to Goemans and Williamson, first finds the optimal solution of a semidefinite program and then derives a graph cut from that solution. Building on this result, Karger, Motwani, and Sudan gave an approximation algorithm for graph coloring that also involves solving a semidefinite program. Solving these semidefinite programs using known methods (ellipsoid, interior-point), though polynomial-time, is quite expensive. We show how they can be approximately solved in $\tilde{O}(nm)$ time for graphs with n nodes and m edges.

1 Introduction

It is well-established that linear programs can be useful in (1) estimating the value of an integer program, and (2) obtaining approximately optimum solutions to an integer program. Similar use of *semidefinite programming* is emerging as an important technique. Lovász [17] showed semidefinite programming could be used to compute the Shannon capacity of a graph, often referred to as the theta function; this is a number that lies between the size of the maximum clique and the minimum number of colors. Lovász and Schrijver [18] described a way to use semidefinite programming to estimate the value of integer programs.

In an important recent breakthrough, Goemans and Williamson [5] discovered an approximation algorithm for graph MAX CUT whose accuracy is significantly better than that of the previously known algorithms. Their algorithm is based on obtaining a near-optimum solution to a semidefinite program.

Building on this work, Karger, Motwani, and Sudan [10] discovered an approximation algorithm for graph coloring. If the input graph is 3-colorable, their algorithm obtains a coloring that uses $O(n^{1/4}\log n)$ colors. More generally, if the input graph is k-colorable, the coloring obtained by the algorithm uses $\tilde{O}(n^{1-1/(k+1)})$ colors. Their algorithm, like that of Goemans and Williamson, is based on obtaining a near-optimum solution to a semidefinite program.

Semidefinite programming is a generalization of linear programming, and a special case of convex programming. Essentially, what is added to linear programming is the ability to specify constraints of the form "X is a positive-semidefinite matrix", where X is a symmetric matrix whose entries are variables. Such a constraint is written $X \succeq 0$. A positive-semidefinite matrix is a matrix X all of whose eigenvalues are nonnegative. An equivalent condition is that for any column vector u, the value of $u^T X u$ is nonnegative.

Semidefinite programming is a special case of convex programming. The first algorithm proposed for semidefinite programming was based on the ellipsoid method. Lovász gave a subroutine that, given a matrix X that is not positive-semidefinite, finds a hyperplane separating X from the set of positive-semidefinite matrices. This subroutine, combined with the ellipsoid method, provided a polynomial-time algorithm for semidefinite programming. However, the complexity of this algorithm is quite high.

Nesterov and Nemirovsky showed [19] how to use the interior-point method to solve semidefinite programs. Alizadeh [1] showed how an interior-point algorithm for linear programming could be directly generalized to handle semidefinite programming. Since the work of Alizadeh, there has been a great deal of research into such algorithms [9, 26, 22, 28, 7, 14, 3]. Essentially, the number of iterations to achieve an approximately optimal solution is $\tilde{O}(\sqrt{n})$

depending on the quality of the initial solution, and each iteration involves either factoring a matrix or solving a least-squares problem.

Goemans and Williamson's algorithm involves solving a semidefinite program with $O(n)$ linear constraints. Each iteration takes $O(n^3)$ time. Thus the time required to solve this problem is $O(n^{3.5})$. Furthermore, after the algorithm computes a solution matrix X, it must compute the Cholesky factorization of X; this can be done in $O(n^3)$ time. The algorithm of Karger, Motwani, and Sudan requires the solution to a semidefinite program with $O(m)$ linear constraints. The time required using a straightforward approach is $O(\sqrt{n}m^3)$.

In this paper, we propose an alternative approach. We apply the method of Plotkin, Shmoys, and Tardos [20] to find an approximate solution to each of these semidefinite programs. An iteration involves a simpler semidefinite program. We show in solving this program, we can restrict our attention to rank-one solutions X, i.e. solutions of the form $X = uu^T$, where u is an n-element vector. When we make this substitution, the resulting program turns out to be equivalent to an eigenvector problem, for which we can use an approach called the *power method.*

The power method can exploit the sparsity of the matrix, which in turn reflects the sparsity of the graph. Using a randomized method for selecting an initial vector, we bound the number of iterations of the power method required to find a eigenvalue solution good enough for our purposes. As a result, for fixed ϵ, we obtain randomized algorithms that find an ϵ-optimal solutions to the semidefinite programs in only $\tilde{O}(nm)$ time, where m is the number of edges. Furthermore, the form in which this solution is given makes it trivial to find its Cholesky factorization (or rather a decomposition that serves the same purpose). Thus by using our algorithms as subroutines, one implement the MAX CUT algorithm of Goemans and Williamson and the graph coloring algorithm of Karger, Motwani, and Sudan in $\tilde{O}(nm)$ time.

The difference between our $\tilde{O}(nm)$ bound and the $\tilde{O}(n^{3.5})$ bound of the interior-point method is particularly striking when the graph is very sparse, i.e. $m = O(n)$. Thus one of the key advantages of our approach is that we can provably exploit the sparsity of the input graph.

The catch, of course, is that our algorithm finds only an ϵ-approximate solution to the semidefinite programs. The ellipsoid method and interior-point method also find approximate solutions, but their running-times' dependence on ϵ is much nicer. Consequently, our algorithm is useful only when one is willing to sacrifice a little in the quality of the output in order to get the solution more quickly—or when the input graph is so big (and yet sparse) that the interior-point method would take more time than feasible.

2 Background and overview

The basis of our approach is the method of Plotkin, Shmoys, and Tardos for approximately solving what they call fractional packing and covering problems. Their method is analogous to Lagrangean relaxation, in which constraints are replaced by a "penalty" component of the objective function, such that violations of the constraint are penalized. The difficulty in Lagrangean relaxation is in choosing the weights of the penalties. In the method of Plotkin, Shmoys, and Tardos, the penalties are determined directly from a current candidate solution. These penalties are used to select a direction to move from the current candidate solution to obtain the next candidate solution, and the process is repeated.

Some of the particulars of their approach originated in the work of Shahrokhi and Matula [24] on approximate solution of a multicommodity flow problem. Shahrokhi and Matula proved a polynomial but rather high bound on the running time of their algorithm. A much faster algorithm for this problem was developed by Klein, Plotkin, Stein, and Tardos [11]. One important ingredient in the improvement is a formulation of approximate optimality (relaxed complementary slackness conditions) that works well in this setting. This multicommodity flow algorithm was generalized by Leighton, Makedon, Plotkin, Stein, Tardos, and Tragoudas [16]. Plotkin, Shmoys, and Tardos then took a final step and generalized this multicommodity flow algorithm, obtaining not a single algorithm but a whole framework in which algorithms for a variety of problems could be formulated.

This framework, like that of the ellipsoid algorithm [6] and the method of Vaidya [25], casts algorithms in terms of a subroutine, an oracle. For the ellipsoid algorithm, the subroutine is called a separation oracle. For Plotkin, Shmoys, and Tardos, the subroutine must find an optimum (or near-optimum) solution to a simpler optimization problem. Specifically, if the goal is to find a vector that satisfies some linear inequalities and in addition lies in a given convex body P, the subroutine must find a vector of minimum cost of all those in P.

2.1 Solving the MAX CUT semidefinite program

To cast the MAX CUT semidefinite program in this framework, we take P to be the set of positive-semidefinite matrices X satisfying the linear inequality $\sum_{ij} L_{ij} X_{ij} = 1$. To obtain an algorithm, we must supply a subroutine to find such a matrix X minimizing a weighted sum of its diagonal elements. That is, we must find the minimizer of

$$\min\{\sum_i y_i X_{ii} : \sum_{ij} L_{ij} X_{ij} = 1, X \succeq 0\}.$$

We show that the minimum is achieved by a rank-one matrix, a matrix X of the form uu^T, where u is an n-element column vector. Furthermore, we show that the best such matrix is that defined by the vector u that is the eigenvector corresponding to the maximum eigenvalue of a matrix derived from L.

Finding an eigenvector of a matrix would take too much time. We exploit the fact that we need only an approximate solution, and can therefore use a few iterations of a method called the *power method.*

Since L is positive-semidefinite, all its eigenvalues are nonnegative. Hence its maximum eigenvalue is the eigenvalue of maximum absolute value. The power method uses the fact that the eigenvalues of A^k are the k^{th} powers of the eigenvalues of A. If k is big enough, therefore, small differences in eigenvalues of A result in large differences in eigenvalues of A^k. Instead of computing A^k directly, we can make do with the matrix-vector product $A^k x_{(0)}$, where $x_{(0)}$ is a suitably chosen initial vector. We can compute this product by computing a series of k matrix-vector products $Ax_{(i)}$. Computing each of these products takes time proportional to the number of nonzero elements of A. In this case, the number is $O(m)$. We show that a good enough solution is obtained by taking $k = O(\epsilon^{-1} \log n)$. We also show that a random selection of the initial vector is sufficiently good with high probability. Thus we obtain a fast subroutine to optimize over P.

Some additional work is needed. We need to ensure that the vectors obtained by the power method have reasonably small components. We do this by perturbing the initial problem in a way that does not modify the optimum value too much; an approximately optimal solution to the perturbed problem yields and approximately optimal solution to the original problem. We also need to show how to obtain a good initial solution to the semidefinite program. For this, we use the previously known approximation algorithm for MAX CUT, due to Sahni and Gonzalez [23]. Finally, we show that after only only $O(\epsilon^{-2} n \log n)$ iterations of optimizing over P, we obtain an ϵ-optimal solution X to the semidefinite program. The time required is thus $O(\epsilon^{-3} nm \log^2 n)$. Furthermore, in practice we could run the power method for many fewer iterations by using as its initial vector the output obtained from the power method the last time. Thus one factor of ϵ^{-1} in our analysis seems superfluous.

Since in each iteration the matrix output by the subroutine has the form uu^T, we obtain X as the sum of such rank-one matrices:

(1) $X = u^{(1)} {u^{(1)}}^T + \cdots + u^{(r)} {u^{(r)}}^T.$

That is,

(2) $X_{ij} = {u^{(1)}}_i {u^{(1)}}_j + \cdots + {u^{(r)}}_i {u^{(r)}}_j.$

The approximation algorithm of Goemans and Williamson requires that we compute a matrix H such that $H^T H$ equals X. That is, the i^{th} column of H is a vector h_i such that

$$X_{ij} = h_i \cdot h_j = h_i^{(1)} h_j^{(1)} + \cdots + h_i^{(t)} h_j^{(t)}.$$

As evident from (2), we can derive such vectors h_i directly from the $u^{(j)}$'s appearing in (1), namely we set $h_i^{(j)} := u_i^{(j)}$.

2.2 The graph coloring semidefinite program

In casting this problem in the framework of Plotkin, Shmoys, and Tardos, we take the convex body P to be the set of positive-definite matrices X whose diagonal elements are 1. Thus in this case P involves n linear constraints instead of just one, as in the case of MAX CUT. Moreover, the function we must optimize over P involves all the components of X rather than just the diagonal elements (as was the case in MAX CUT). The objective function has the form $\sum_{ij} C_{ij} X_{ij}$. Thus optimization over P cannot be done using a rank-one matrix. However, by making a small perturbation in this optimization problem, we transform it into precisely the MAX CUT semidefinite program. Thus we can use our algorithm for that problem to approximately optimize over P. (Once, again we show that the perturbation is such that an approximately optimal solution to the perturbed problem yields an approximately optimal solution to the original problem.)

This time, we need to optimize over P only $O(\epsilon^{-2} \log n)$ times. Thus we obtain an algorithm for the coloring semidefinite program that takes $O(\epsilon^{-5} nm \log^3 n)$ time. As mentioned in Subsection 2.1, by being more clever about how we use the power method, we can probably remove one factor of ϵ^{-1} in practice.

3 Preliminaries

All vectors in the paper are $n \times 1$ column vectors. All matrices in this paper are $n \times n$ and symmetric. Let u and v be two vectors. Define $u \cdot v$ to be the dot product $u^T v$ of u and v. Let A and B be two matrices. Define $A \bullet B$ to be the trace of AB, which is equal to $\sum_{1 \le i,j \le n} A_{ij} B_{ij}$. One can easily verify that $A \bullet (uu^T) = u^T A u$ for any vector u. Let I be the identity matrix and let J be the all-one matrix. Let $\vec{1}$ be the all-one vector.

A matrix X is *positive semidefinite*, denoted $X \succeq 0$, if every eigenvalue of X is nonnegative. It is well-known that $X \succeq 0$ if and only if X can written as $\sum_{u \in U} uu^T$, where U is a set of at most n orthogonal vectors.

Let G be a simple undirected graph of n nodes with nonnegative costs on edges. Let C be the cost matrix of G, where C_{ij} is the cost of the edge ij of G. Let $\bar{L}$ be the *Laplacian* of G, defined as

$$\bar{L}_{ij} = \begin{cases} -C_{ij} & i \neq j \\ \sum_{1 \le k \le n} C_{ik} & i = j. \end{cases}$$

For convenience we refer to the MAX CUT semidefinite program as the VECTOR MAX CUT problem. The VECTOR MAX CUT problem is the following semidefinite program as shown in [5].

$$\begin{array}{lll} \max & \frac{1}{4} C \bullet (J - X) & \\ \text{s.t.} & X_{ii} = 1 & \text{for every } 1 \le i \le n \\ & X \succeq 0. & \end{array}$$

Lemma 1 ensures that the above semidefinite program can be rewritten as follows.

$$(3) \quad \begin{array}{lll} \max & \frac{1}{4} \bar{L} \bullet X & \\ \text{s.t.} & X_{ii} = 1 & \text{for every } 1 \le i \le n \\ & X \succeq 0. & \end{array}$$

Lemma 1 Let $\bar{L}$ be the Laplacian of a graph corresponding to the cost matrix C. Let X be a matrix such that every diagonal element of X is one. Then $C \bullet (J - X) = \bar{L} \bullet X$.

Proof Let $D = \bar{L} + C$. Clearly D is a diagonal matrix. Since every diagonal element of X is one, we know that $D \bullet X = C \bullet J$. Therefore

$$L \bullet X = D \bullet X - C \bullet X = C \bullet J - C \bullet X.$$

□

The VECTOR COLORING problem is the following semidefinite program as shown in [10].

$$(4) \quad \begin{array}{lll} \min & \lambda & \\ \text{s.t.} & X_{ii} = 1 & \text{for every } 1 \le i \le n \\ & X_{ij} \le \lambda & \text{for every edge } ij \text{ of } G \\ & X \succeq 0. & \end{array}$$

For a mathematical program, e.g. (3), an assignment to the variables is said to be a *feasible solution* if it satisfies the constraints. The *value* of such an assignment is the value of the objective function at that assignment. Let $f(\cdot)$ be the objective function of a mathematical program Q. Let X^* be an optimal solution to P. A feasible solution X to Q is ϵ-*optimal* if $(1+\epsilon)^{-1} \le \frac{f(X^*)}{f(X)} \le (1+\epsilon)$. For simplicity of time-bounds, we assume throughout that $\epsilon > 1/n$.

4 VECTOR MAX CUT

4.1 Problem reduction

For simplicity of the analysis, we assume that the edge costs of G are scaled such that the sum of edge costs of G is exactly one. It follows that that $\bar{L} \bullet I = 2$. Define $L = \bar{L} + I/n$. Consider the following semidefinite program.

$$(5) \quad \begin{array}{lll} \min & \lambda & \\ \text{s.t.} & X_{ii} \le \lambda & \text{for every } 1 \le i \le n \\ & L \bullet X = 1 & \\ & X \succeq 0. & \end{array}$$

It is easy to see that this program is equivalent to (3), in that a feasible solution to one can be easily transformed into a feasible solution to the other. For example, let (X^*, λ^*) be an optimal solution to (5). Since the diagonal elements of L are nonnegative, we can assume without loss of generality that all the diagonal elements of X are exactly the maximum diagonal element, λ^* (by increasing them and then rescaling if necessary). If we divide every element of X^* by λ^*, we get a matrix X' in which each diagonal element is at most one. Increase each diagonal element until it is one. Since $L \bullet X^* = 1$, we obtain $L \bullet X' = 1/\lambda^*$. Finally, since $L = \bar{L} + I/n$, we have $L \bullet X' = \bar{L} \bullet X' + 1$, so $\frac{1}{4} \bar{L} \bullet X' = \frac{1}{4}(L \bullet X' - 1)$. Since this transformation is invertible, we conclude that an optimum solution to one program yields an optimum solution to the other. We can say more, however. Because the identify matrix I is a feasible solution to (3), the optimum value of that program is at least $\frac{1}{4} \bar{L} \bullet I$, which is $1/2$ by our assumption about the edge costs of G. It follows that an ϵ-optimal solution to (5) yields a 2ϵ-optimal solution to (3), at least as long as $\epsilon \le 0.5$.

More formally, let (X, λ) be the ϵ-optimal solution to (5), and let (X^*, λ^*) the optimal solution. Let $\bar{X}$ denote the solution to (3) corresponding to (X, λ), and let $\bar{X}^*$ be the optimal solution. Then we have

$$\begin{aligned} \bar{L} \bullet \bar{X} &= L \bullet \bar{X} - 1 \\ &\ge 1/\lambda - 1 \\ &\ge (1+\epsilon)^{-1}/\lambda^* - 1 \\ &\ge (1+\epsilon)^{-1}(1/\lambda^* - 1 - \epsilon) \\ &\ge (1+\epsilon)^{-1}(1 - 0.5\epsilon)\bar{L} \bullet \bar{X}^* \\ &\ge (1+2\epsilon)^{-1}\bar{L} \bullet \bar{X}^*, \end{aligned}$$

where the last inequality follows because $\epsilon \le 0.5$.

For the rest of the section we focus on finding an ϵ-optimal solution to (5).

4.2 Bounding the values

We first give a lower bound on the optimum value based on the assumption that the sum of edge costs of G is one. We then give an upper bound on the feasible values, which also holds for the optimal value.

Lemma 2 The optimum value is at least 0.17.

Proof Let $\bar{X}^*$ be an optimal solution to (3). The results in [5] ensure that $\frac{0.878}{4}\bar{L} \bullet \bar{X}^*$ is at most the sum of edge costs. Since we assume that the sum of edge costs of G is one, it follows that $\bar{L} \bullet \bar{X}^* \le \frac{4}{0.878}$. One can easily verify that $\lambda^* = \frac{1}{\bar{L} \bullet \bar{X}^* + 1}$. Therefore $\lambda^* \ge \frac{1}{1+4/0.878} > 0.17$. □

The following lemma upperbounds the value of any feasible solution.

Lemma 3 For any X satisfying $X \succeq 0$ and $L \bullet X = 1$, each diagonal element of X is at most n.

Proof Since $L = \bar{L} + I/n$, it follows that $I \bullet X = n(1 - \bar{L} \bullet X)$. Note that $\bar{L} \succeq 0$ by Geršgorin's theorem (e.g. see §10.6 of [15]). Since also $X \succeq 0$, we obtain $\bar{L} \bullet X \ge 0$, and thus $X_{11} + \cdots + X_{nn} = I \bullet X \le n$. Since $X \succeq 0$, each X_{ii} is nonnegative. It follows that $X_{ii} \le n$ for every $1 \le i \le n$. □

We now apply the framework of Plotkin, Shmoys, and Tardos [20]. We view (5) as

$$\begin{array}{lll} & \min & \lambda \\ (6) & \text{s.t.} & X_{ii} \le \lambda \quad \text{for every } 1 \le i \le n \\ & & X \in P, \end{array}$$

where $P = \{X : L \bullet X = 1, X \succeq 0\}$.

The algorithm depends on a parameter ρ that Plotkin, Shmoys, and Tardos call the *width* of the formulation. The width in this case is by definition $\max\{X_{ii} : X \in P\}$. By Lemma 3, the width is at most n. Now we can describe the algorithm.

4.3 The algorithm

Let C be the cost matrix for the given graph G. Let ϵ be the given positive precision parameter. We give an algorithm that produces an ϵ-optimal solution. The algorithm starts with finding a 1-optimal initial solution (X, λ). It then iteratively updates (X, λ) until it is ϵ-optimal. In each iteration a procedure IMPROVE is called to improve the precision of (X, λ). Specifically, after the k^{th} iteration (X, λ) is guaranteed to be 2^{-k}-optimal with high probability. The algorithm VECTORMAXCUT(C, ϵ) is as follows.

1. Scale C such that the sum of edge costs is one, and then compute L from C.
2. Let $(X, \lambda) = $ INITIAL(C).
3. Let $\epsilon' = 1$.
4. While $\epsilon' > \epsilon$ do
   (a) Let $\epsilon' = \epsilon'/2$.
   (b) Let $(X, \lambda) = $ IMPROVE$(X, \lambda, \epsilon'/7)$.

The procedure INITIAL(C) obtains an initial solution (X, λ) from a greedy solution to the MAX CUT problem. Using the greedy method by Sahni and Gonzalez [23], one can obtain a 1-optimal solution to the MAX CUT problem in time linear in the number of edges. Moreover such a greedy cut has value at least one half of the sum of edge costs. Let u be the characteristic vector for the greedy solution to the MAX CUT problem, where $u_i = 1$ for every node i on one side of the cut, and $u_j = -1$ for every node j on the other side of the cut. Clearly uu^T is the corresponding feasible solution to (3). Let $c = \bar{L} \bullet (uu^T)$. We know $\frac{1}{4}c \ge \frac{1}{2}$, since the sum of edge costs is scaled to be one. Hence $c \ge 2$. Let the return value (X, λ) of INITIAL(C) be $(\frac{uu^T}{c+1}, \frac{1}{c+1})$. Clearly (X, λ) is feasible to (5). Since $c \ge 2$, $\lambda \le 1/3$. Therefore the initial solution (X, λ) is 1-optimal by Lemma 2.

At the beginning of the procedure IMPROVE(X, λ, ϵ), we assume that $\lambda/\lambda^* = 1 + O(\epsilon)$. The solution (X', λ') output by this procedure satisfies $\lambda'/\lambda^* \le 1 + \epsilon$, so our assumption holds for the next call to the procedure, when ϵ' has been halved.

The procedure IMPROVE uses a positive vector y defined as a function of the current solution (X, λ). The procedure proceeds iteratively to update the current solution. In each iteration a procedure DIRECTION(y, ϵ) is called to obtain a matrix solution $\tilde{X}$ that with high probability is an approximate minimizer for

(7) $\mu(y) = \min\{y_1 X_{11} + \cdots + y_n X_{nn} : X \in P\}$.

The current solution is moved towards $\tilde{X}$ by a small amount σ. Namely let $X := (1-\sigma)X + \sigma\tilde{X}$, and let λ be the maximum diagonal element of the new X. One can easily verify that the new (X, λ) is still feasible. Let us define the notation $\langle X \rangle_y$ by

$$\langle X \rangle_y = y_1 X_{11} + \cdots + y_n X_{nn}.$$

The procedure can stop when the following condition holds.

(8) $\langle X \rangle_y - \langle \tilde{X} \rangle_y \le \epsilon(\langle X \rangle_y + \lambda y \cdot \vec{1})$

The procedure IMPROVE(X, λ_0, ϵ) is as follows.

1. Let $\lambda = \lambda_0$.
   Let $\alpha = 12\epsilon^{-1}\ln(2n\epsilon^{-1})$.
   Let $\sigma = \epsilon/(4\alpha n)$.
2. Repeat
   (a) Let $y_i = e^{\alpha X_{ii}}$ for every $i = 1, \ldots, n$.
   (b) Let $\tilde{X} = $ DIRECTION(y, ϵ).
   (c) If condition (8) is satisfied then
       Return (X, λ).
     else
       Let $X = (1-\sigma)X + \sigma\tilde{X}$.
       Let $\lambda = \max\{X_{11}, \ldots, X_{nn}\}$.

It follows from the framework of Plotkin, Shmoys, and Tardos that the number of iterations is $O(\epsilon^{-2} n \log n)$. In the next subsection, we give an implementation of DIRECTION that takes $\tilde{O}(\epsilon^{-1} m)$ time. Thus we will obtain the following lemma.

Lemma 4 The running time required by the algorithm VECTORMAXCUT(C, ϵ) is $\tilde{O}(\epsilon^{-3} mn)$.

4.4 The power method

It remains to describe DIRECTION(y, ϵ), which produces an approximate minimizer for (7) with high probability. One can easily verify that $\mu(y)$ can be achieved by a rank-one matrix $uu^T \in P$. Therefore it suffices for DIRECTION(y, ϵ) to obtain a vector u such that

$$\langle uu^T \rangle_y \le (1+\epsilon)\mu(y).$$

We show that it is in fact an eigenvector problem to find a feasible matrix uu^T such that $\langle uu^T \rangle_y = \mu(y)$. The eigenvector problem can be approximately solved by a well-known numerical algorithm called the *power method* (e.g. see §5.3 of [2]), which is one of the oldest methods for computing the dominant eigenpair of a matrix. Let L' be a matrix defined as $L'_{ij} = L_{ij}/\sqrt{y_i y_j}$. One can easily verify that $L' \succeq 0$. Let v be a vector defined by $v_i = u_i\sqrt{y_i}$. Note that y is positive, so L' and v are both well-defined. Clearly $L \bullet (uu^T) = L' \bullet (vv^T)$ and $\langle uu^T \rangle_y = v \cdot v$. It follows that

$$\mu(y) = \min\{v \cdot v : L' \bullet (vv^T) = 1\}.$$

Therefore

$$1/\mu(y) = \max\{L' \bullet (ww^T) : w \cdot w = 1\}.$$

By Rayleigh-Ritz's theorem (e.g. see Theorem 4.2.2 of [8]) we know $1/\mu(y)$ is the maximum eigenvalue of L'. A normalized eigenvector w in the eigenspace of L' corresponding to $1/\mu(y)$ is a vector that maximizes $L' \bullet (ww^T)$ over all normalized vectors. Let $v = \mu(y)w$, and let u be obtained by $u_i = v_i/\sqrt{y_i}$. By the above discussion we know that uu^T is a feasible solution such that $\langle uu^T \rangle_y = \mu(y)$. Clearly if w is close to the eigenspace of L' corresponding to $1/\mu(y)$, then the corresponding $L \bullet (uu^T)$ would be close to $\mu(y)$. Specifically if w is a normalized vector such that $L' \bullet (ww^T) \ge 1/((1+\epsilon)\mu(y))$, then the corresponding uu^T would be an ϵ-optimal minimizer for (7). Therefore the problem of finding an ϵ-optimal minimizer for (7) is reduced to the following problem:

> Let λ be the maximum eigenvalue of a positive semidefinite matrix A. Find a normalized vector x such that $A \bullet (xx^T) \ge (1+\epsilon)^{-1}\lambda$.

For the rest of the section we focus on adapting the power method to solve the above problem.

We first describe the power method as follows. The method consists of a series of k iterations, where k is a parameter to be determined. Each iteration consists of a matrix-vector multiplication. In particular, initialize by choosing a random normalized vector x_0, and then compute $x_{(k)} = A^k x_{(0)}/\|A^k x_{(0)}\|_2$ using the recurrence $x_{(i+1)} = Ax_{(i)}/\|Ax_{(i)}\|_2$.

Now we analyze the algorithm. Since $A \succeq 0$, it has n orthonormal eigenvectors $w_{(1)}, \ldots, w_{(n)}$. Let $\lambda_1, \ldots, \lambda_n$ be the corresponding eigenvectors, and suppose they are indexed so that $\lambda_1 \ge \cdots \ge \lambda_n \ge 0$. It is known that A can be written as $\lambda_1 w_{(1)} w_{(1)}^T + \cdots + \lambda_n w_{(n)} w_{(n)}^T$. Since $\{w_{(1)}, \ldots, w_{(n)}\}$ are orthonormal, every arbitrary normalized vector x can be written as

$\alpha_1 w_{(1)} + \cdots + \alpha_n w_{(n)}$, where $\alpha_1^2 + \cdots + \alpha_n^2 = 1$. Note that

$$\begin{aligned} A^k x &= \sum_{1 \le i \le n} \alpha_i \lambda_i^k w_{(i)} \\ \|A^k x\|_2 &= \sqrt{\sum_{1 \le i \le n} \alpha_i^2 \lambda_i^{2k}} \end{aligned}$$

for any $k \geq 0$. It is not hard to see that as k grows larger, $x_{(k)}$ gets closer to $u_{(1)}$. The following lemma gives an upper on the k that is required for $x_{(k)}$ to be a normalized vector x such that $A \bullet (xx^T) \geq (1+\epsilon)^{-1}\lambda_1$.

Lemma 5 Let $x_{(k)}$ be as defined above. Suppose $\alpha_1 \geq 0$. If $k \geq \epsilon^{-1} \ln(\epsilon^{-1}\alpha_1^{-2})$, then $A \bullet (x_{(k)} x_{(k)}^T) \geq (1+\epsilon)^{-1}\lambda_1$.

Proof Assume $A \bullet (x_{(k)} x_{(k)}^T) < \lambda_1/(1+\epsilon)$. We show $k < \epsilon^{-1}\ln(\epsilon^{-1}\alpha_1^{-2})$. Note that

$$A \bullet (x_{(k)} x_{(k)}^T) = \frac{\sum_{1 \le i \le n} \alpha_i^2 \lambda_i^{2k+1}}{\sum_{1 \le i \le n} \alpha_i^2 \lambda_i^{2k}}.$$

It follows from the assumption that

$$(1+\epsilon) \sum_{1 \le i \le n} \alpha_i^2 \lambda_i^{2k+1} < \lambda_1 \sum_{1 \le i \le n} \alpha_i^2 \lambda_i^{2k}.$$

Let $\delta_i = \lambda_1 - (1+\epsilon)\lambda_i$ for every $2 \leq i \leq n$. The above inequality can be rewritten as follows by moving the first terms of both summations to the left, and the remaining terms to the right.

$$(9) \quad \epsilon \alpha_1^2 \lambda_1^{2k+1} < \sum_{2 \le i \le n} \alpha_i^2 \lambda_i^{2k} \delta_i.$$

Clearly $\delta_2 \leq \cdots \leq \delta_n$. If δ_n were less than or equal to zero, then the RHS of (9) would be less than or equal to zero, which contradicts the fact that the LHS of (9) is nonnegative. Therefore we know $\delta_n > 0$. Let t be the smallest index such that $\delta_t > 0$. It follows that

$$\begin{aligned} \epsilon \alpha_1^2 \lambda_1^{2k+1} &< \sum_{t \le i \le n} \alpha_i^2 \lambda_i^{2k} \delta_i \\ &\le \sum_{t \le i \le n} \alpha_i^2 \lambda_t^{2k} \lambda_1 \\ &\le \lambda_t^{2k} \lambda_1. \end{aligned}$$

Therefore $\epsilon\alpha_1^2(\lambda_1/\lambda_t)^{2k} < 1$. Since $\delta_t > 0$, we know that $\lambda_1/\lambda_t > (1+\epsilon)$. Thus $\epsilon\alpha_1^2(1+\epsilon)^{2k} < 1$. Since $\ln(1+\epsilon) \geq \epsilon/2$ for any $0 \leq \epsilon \leq 1$, it follows that $k < \epsilon^{-1}\ln(\epsilon^{-1}\alpha_1^{-2})$. □

Since the number of iterations required by the power method is determined by the α_1^2 of $x_{(0)}$, it suffices to show how to choose the initial vector x such that its α_1^2 is sufficiently large. It turns out that randomly choosing $x_{(0)}$ uniformly from the surface of the n-dimensional unit sphere is sufficient. Note that the α_1 of x is exactly $x \cdot w_{(1)}$, the projection of x on $w_{(1)}$. The following lemma lowerbounds the probability that $\alpha_1^2 \geq O(n^{-c-1})$ for any $c \geq 0$.

Lemma 6 Suppose $n \geq 2$. Let x' be a fixed normalized vector in n-dimensional space. Let x be a random vector that is uniformly distributed on the surface of the n-dimensional unit sphere. Then the probability that $(x \cdot x')^2 \leq \frac{1}{2n^{c+1}}$ is at most n^{-c} for any $c \geq 0$.

Proof Let $a = \frac{1}{2n^{c+1}}$. Clearly $\Pr((x \cdot x')^2 \geq a^2) = \Pr(|x \cdot x'| \geq a)$. We show $\Pr(|x \cdot x'| \leq a) \leq n^{-c}$.

Let $\hat{\theta} = \arccos a$. It is known by multivariable analysis (e.g. see §60 of [21]) that

$$\begin{aligned} \Pr(|x \cdot y| \le a) &= \frac{1}{S}\int_{\hat{\theta}}^{\pi/2} \sin^{n-2}\theta \, d\theta \\ &\le \frac{1}{S}\left(\frac{\pi}{2} - \hat{\theta}\right), \end{aligned}$$

where $S = \int_0^{\pi/2} \sin^{n-2}\theta \, d\theta$. Note that $\theta \leq 2\sin\theta$, for any $\theta \leq \frac{\pi}{2}$. It follows that $\frac{\pi}{2} - \hat{\theta} \leq 2\sin(\frac{\pi}{2} - \hat{\theta}) = 2\cos\hat{\theta} = 2a = n^{-c-1}$. Therefore it suffices to show that $S \geq \frac{1}{n}$.

It is known (e.g. see (131) of §60 in [21]) that

$$S = \frac{\Gamma(\frac{n-1}{2}+1) n \pi^{n/2}}{2(n-1)\pi^{(n-1)/2}\Gamma(\frac{n}{2}+1)}$$

for any $n \geq 2$. Therefore

$$S \geq \frac{\Gamma(\frac{n}{2})}{2\Gamma(\frac{n}{2}+1)} = \frac{1}{n}.$$

□

A uniformly distributed random vector x on the surface of the n-dimensional unit sphere can be obtained by randomly generating n values $x_1, \ldots, x_n$ independently from the standard normal distribution, and then normalizing the resulting vector (e.g. see page 130 of [13].) The standard normal distribution can be simulated using the uniform distribution between 0 and 1 (e.g. see page 117 of [13].) Let $x = x_{(k)}$, where $k = \lceil \frac{1}{\epsilon} \ln(\frac{2n^c}{\epsilon}) \rceil$. It follows from Lemma 5 and Lemma 6 that $A \bullet (xx^T) \geq (1+\epsilon)^{-1}\lambda_1$ holds with probability at least $1 - n^{-c}$. Since each iteration of the power method involves a matrix-vector multiplication, which takes time $O(m)$, the running time of Direction(y, ϵ) is $O(m\epsilon^{-1}\ln n)$.

5 VECTOR COLORING

In this section we show how to obtain an $O(\epsilon)$-optimal solution to (4), which we restate here for convenience.

$$(10)\quad \begin{array}{lll} \min & \lambda & \\ \text{s.t.} & X_{ii} = 1 & \text{for every } 1 \leq i \leq n \\ & X_{ij} \leq \lambda & \text{for every edge } ij \text{ of } G \\ & X \succeq 0. & \end{array}$$

In order to put the problem into the framework of Plotkin, Shmoys, and Tardos [20], we can reformulate this program as follows.

$$\begin{array}{lll} \max & \gamma & \\ \text{s.t.} & X'_{ij} \geq \gamma & \text{for every edge } ij \text{ of } G \\ & X \in P', & \end{array}$$

where $P' = \{X' : X'_{ii} = -1, -X' \succeq 0\}$. Let $P = -P'$. Like P, P' is a convex body. The framework of [20] requires that we optimize over P'. Equivalently, we can optimize over P.

5.1 Definitions

In this section, we say a matrix X is *feasible* if $X \in P$. We say (X, λ) is feasible if X is feasible and every component X_{ij} of X, where ij is an edge of G, is at most λ.

A matrix Y is a *cost matrix for G* if Y is a nonnegative matrix such that $Y_{ij} = 0$ for every ij that is not an edge of G. Let

$$\mu(Y) = \min\{Y \bullet X : X \in P\}.$$

Since I is feasible and $Y \bullet I = 0$, we know that $\mu(Y) \leq 0$.

5.2 Bounding the values

The following lemma bounds the value of any feasible solution, including the value of an optimal solution.

Lemma 7 Let (X, λ) be a feasible solution. Then $\lambda \leq 1$.

Proof By the feasibility of (X, λ) we know $X \succeq 0$, which implies that $u^T X u \geq 0$ for any vector u. If $X_{ij} > 1$ for some edge ij of G, then $u^T X u < 0$, where u is defined by

$$u_k = \begin{cases} 1 & \text{if } k = i \\ -1 & \text{if } k = j \\ 0 & \text{otherwise.} \end{cases}$$

The lemma thus follows. $\square$

5.3 The algorithm

We give an algorithm that produces an ϵ-optimal solution. The algorithm starts with finding an initial solution (X, λ). It then iteratively updates (X, λ) until it is ϵ-optimal. In each iteration a procedure ImproveC is called to improve the precision of (X, λ). Specifically each call to ImproveC either reduces the value of the current solution by a factor of two or yields a solution that is near-optimal with respect to the precision parameter given to ImproveC. The algorithm VectorColoring(G, ϵ) is as follows.

1. Let $(X, \lambda) = $ InitialC(G).
   Let $\epsilon' = 2$.

2. While $\epsilon' > \epsilon$ do
   (a) Let $\epsilon' = \epsilon'/2$.
   (b) While (X, λ) is not ϵ'-optimal do
       Let $(X, \lambda) = $ ImproveC$(X, \lambda, \epsilon'/4)$.

The procedure InitialC(G) obtains an initial solution (X, λ) as follows:

$$X_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\frac{1}{n} & \text{if } ij \text{ is an edge of } G \\ 0 & \text{otherwise.} \end{cases}$$

Clearly $X \succeq 0$ (e.g. see the Geršgorin's theorem in §10.6 of [15]), and thus the initial solution $(X, -\frac{1}{n})$ is feasible.

At the beginning of the procedure ImproveC(X, λ, ϵ), we assume that $\lambda^*/\lambda = 1 + O(\epsilon)$. The solution (X', λ') output by this procedure satisfies $\lambda^*/\lambda' \leq 1 + \epsilon$, so our assumption holds for the next call to the procedure, when ϵ' has been halved.

The procedure ImproveC uses a nonnegative matrix Y defined as a function of the current solution (X, λ). The procedure proceeds iteratively to update the current solution. A procedure DirectionC(Y, ϵ) is called in each iteration to obtain a matrix $\tilde{X}$ that with high probability is an approximate minimizer for

$$(11)\quad \mu(Y) = \min\{Y \bullet X : X \in P\}.$$

The current solution is moved towards $\tilde{X}$ by a small amount σ. Namely let $X = (1 - \sigma)X + \sigma\tilde{X}$, and let λ be the new maximum X_{ij} over all ij that are edges of G. One can easily verify that the new (X, λ) is still feasible. The procedure can stop when the following condition holds.

$$(12)\quad \langle X \rangle_y - \langle \tilde{X} \rangle_y \leq \epsilon(\langle X \rangle_y + \lambda y \cdot \vec{1})$$

The procedure ImproveC(X, λ_0, ϵ) is as follows.

1. Let $\lambda = \lambda_0$.
   Let $\alpha = 4\epsilon^{-1}\lambda_0^{-1}\ln(4n^2\epsilon^{-1})$.
   Let $\sigma = \epsilon/(4\alpha)$.
2. Repeat
   (a) Let $Y_{ij} = e^{\alpha X_{ij}}$ for every edge ij of G.
   (b) Let $\tilde{X} = \textsc{DirectionC}(Y, \epsilon)$.
   (c) If $\lambda \leq \lambda_0/2$ or (12) is satisfied then
   Return (X, λ).
   else
   Let $X = (1-\sigma)X + \sigma\tilde{X}$.
   Let $\lambda = \max\{X_{ij} : ij \text{ is an edge of } G\}$.

By Lemma 7 we know the width of (10) is one. It follows from the framework of [20] that the number of iterations is $O(\epsilon^{-2}\log n)$. Suppose G is k-colorable. It follows from Lemma 9 that DirectionC can be implemented as VectorMaxCut$(Y, \frac{\epsilon}{6k})$, which takes time $\tilde{O}(\epsilon^{-3}k^3mn)$. Therefore we obtain the following lemma.

Lemma 8 The running time required by the algorithm VectorColoring(G, ϵ) is $\tilde{O}(\epsilon^{-5}mn)$.

Lemma 9 Suppose G is k-colorable, where $k \geq 2$. Suppose $\epsilon \leq 1$. Let $(\tilde{X}, \tilde{\lambda})$ an $\frac{\epsilon}{6k}$-optimal solution to the graph MAX CUT problem corresponding to the cost matrix Y. Then $\tilde{X}$ is an ϵ-optimal minimizer for (11).

Proof Let (X, λ) be an optimal solution to the VECTOR COLORING problem (10). Since G is k-colorable, it follows from the results in [10] that $\lambda \leq \frac{1}{1-k} \leq -\frac{1}{k}$. By the feasibility of (X, λ) we know that

(13) $Y \bullet X \leq -\dfrac{1}{k} Y \bullet J.$

Let X^* be an optimal solution to the VECTOR MAXCUT problem (3) corresponding to the cost matrix Y. Clearly $\mu(Y) = Y \bullet X^*$. Since X is also feasible for (3), it follows from Lemma 1 that

(14) $-Y \bullet X^* \geq -Y \bullet X.$

By the assumption we know

$$Y \bullet (J - \tilde{X}) \geq (1 + \frac{\epsilon}{3k})^{-1} Y \bullet (J - X^*),$$

and thus

$$\begin{aligned} -(1 + \frac{\epsilon}{3k}) Y \bullet \tilde{X} &\geq -\frac{\epsilon}{3k} Y \bullet J - Y \bullet X^* \\ &\geq -(1 - \frac{\epsilon}{3}) Y \bullet X^*, \end{aligned}$$

using (13) and (14). Since $\epsilon \leq 1$, we know that $(1 - \frac{\epsilon}{3})(1 + \frac{\epsilon}{3k})^{-1} \geq (1+\epsilon)^{-1}$. Therefore $-Y \bullet \tilde{X} \geq -(1+\epsilon)^{-1} Y \bullet X^*$. The lemma follows because $Y \bullet X^* = \mu(Y)$. $\square$

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[2] F. Chatelin. *Eigenvalues of Matrices.* John Wiley & Sons, 1993.

[3] R. M. Freund. Complexity of an Algorithm for Finding an Approximate Solution of a Semi-Definite Program with no Regularity Assumption. Technical Report OR-302-94, Operations Research Center, M.I.T., 1994.

[4] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 422–431, Montréal, Québec, Canada, 23–25 May 1994.

[5] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, Nov. 1995. The preliminary version is [4].

[6] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer Verlag, 1988.

[7] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. An interior point method for semidefinite programming. *SIAM Journal on Optimization*, 6(2):342–361, May 1996.

[8] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[9] F. Jarre. An interior-point method for minimizing the maximum eigenvalue of a linear combination of matrices. *SIAM Journal on Control and Optimization*, 31(5):1360–1377, 1993.

[10] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *35th Annual Symposium on Foundations of Computer Science*, pages 2–13, Santa Fe, New Mexico, 20–22 Nov. 1994. IEEE.

[11] P. Klein, S. Plotkin, C. Stein, and E. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, June 1994. The preliminary version is [12].

[12] P. Klein, C. Stein, and É. Tardos. Leighton-Rao might be practical: Faster approximation algorithms for concurrent flow with uniform capacities. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 310–321, Baltimore, Maryland, 14–16 May 1990.

[13] D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, 1973.

[14] M. Kojima, S. Shindoh, and S. Hara. Interior–Point Methods for the Monotone Linear Complementarity Problem in Symmetric Matrices. Technical Report B-282, Department of Information Sciences, Tokyo Institute of Technology, 1994.

[15] P. Lancaster and M. Tismenetsky. *The Theory of Matrices.* Academic Press, 1985.

[16] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 101–111, New Orleans, Louisiana, 6–8 May 1991.

[17] L. Lovász. On the Shannon Capacity of a graph. *IEEE Transactions on Information Theory*, IT-25:1–7, 1979.

[18] L. Lovász and A. Schrijver. Cones of Matrices and Setfunctions, and 0-1 Optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

[19] Y. Nesterov and A. Nemirovskii. *Interior Point Polynomial Methods in Convex Programming: Theory and Applications.* Society for Industrial and Applied Mathematics, Philadelphia, 1994.

[20] S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *32nd Annual Symposium on Foundations of Computer Science*, pages 495–504, San Juan, Puerto Rico, 1–4 Oct. 1991. IEEE.

[21] G. B. Price. *Multivariable Analysis.* Springer-Verlag, 1984.

[22] F. Rendl, R. Vanderbei, and H. Wolkowicz. A primal-dual interior-point method for the max-min eigenvalue problem. Technical report, University of Waterloo, Dept. of Combinatorics and Optimization, 1993.

[23] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, July 1976.

[24] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *Journal of the ACM*, 37(2):318–334, Apr. 1990.

[25] P. M. Vaidya. Speeding-up linear programming using fast matrix multiplication (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 332–337, Research Triangle Park, North Carolina, 30 Oct.–1 Nov. 1989. IEEE.

[26] L. Vandenberghe and S. Boyd. A primal-dual potential reduction method for problems involving matrix inequalities. Technical report, Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA, Jan. 1993. The journal version is [27].

[27] L. Vandenberghe and S. Boyd. A primal-dual potential reduction method for problems involving matrix inequalities. *Mathematical Programming, Series B*, 69:205–236, 1995.

[28] A. Yoshise. An optimization method for convex programs — interior-point method and analytical center. *Systems, Control and Information*, 38(3):155–160, Mar. 1994.