Space-efficient Approximation Algorithms for MAXCUT and COLORING Semidefinite Programs

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

Hsueh-I Lu
Brown University (USA) and
National Chung-Cheng University (Taiwan)
hil@cs.brown.edu or hil@cs.ccu.edu.tw

Abstract

The essential part of the best known approximation algorithm for graph MAXCUT is approximately solving MAXCUT's semidefinite relaxation. For a graph with n nodes and m edges, previous work on solving its semidefinite relaxation for MAXCUT requires space $\tilde{O}(n^2)$. We show how an approximate solution can be found in space $O(m+n^{1.5})$, where $O(m)$ comes from the input; and therefore reduce the space required by the best known approximation algorithm for graph MAXCUT.

Using the above space-efficient algorithm as a subroutine, we show an approximate solution for COLORING's semidefinite relaxation can be found in space $O(m)+\tilde{O}(n^{1.5})$. This reduces not only the space required by the best known approximation algorithm for graph COLORING, but also the space required by the only known polynomial-time algorithm for finding a maximum clique in a perfect graph.

1 Introduction

Semidefinite programming [34] is the mathematical programming for optimizing a linear function of a matrix X subject to linear constraints and the constraint that X is symmetric and positive semidefinite. (The eigenvalues of a symmetric matrix are all real. A symmetric matrix is positive semidefinite if all of its eigenvalues are nonnegative.) It is a special case of convex programming, and a generalization of linear programming. Its application on combinatorial optimization problems was pioneered by Lovász's work on the Shannon capacity of a graph, which is also known as the theta function [23]. The polynomial-time solvability of semidefinite programs leads to the only known polynomial-time algorithms for some optimization problems for perfect graphs, such as MAXCLIQUE (and therefore MAX STABLE SET) [12], and COLORING [13].

Recently semidefinite programming is emerging as an important technique for designing approximation algorithms. Goemans and Williamson [11] gave an approximation algorithm for graph MAXCUT, whose approximation ratio is significantly better than that of the previously known algorithms. The essential part of their algorithm is obtaining a near-optimal solution to a semidefinite program.

Based on this work, Karger, Motwani, and Sudan [18] discovered the best known approximation algorithm for coloring a k-colorable graph. Besides MAXCUT and COLORING, the technique of using semidefinite programming has been successful in designing approximation algorithms for many other optimization problems, such as MAX DICUT [11, 8], MAX SAT [11, 8], MAX BISECTION [10], MAX k-CUT [10], MAX STABLE SET [2], BETWEENNESS [6], and COLORING for 3-colorable graphs [3]. Each of these improved algorithms is based on obtaining a near-optimal solution to a semidefinite program, which is referred as the *semidefinite relaxation* of the underlining optimization problem. Therefore how to approximately solve these semidefinite relaxations efficiently, both in time and space, is practically important.

The first algorithm proposed for solving semidefinite programs was based on the ellipsoid method [12]. Nesterov and Nemirovsky showed [25] how to use the interior-point methods 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 [17, 33, 29, 35, 14, 22, 9]. A simplex-type method was also discovered by Pataki [27]. Among these varieties, the performance of interior-point methods are the best in both theory and practice. Suppose there are ℓ constraints in a semidefinite program, whose variable is an $n \times n$ matrix. A near-optimal solution for the semidefinite program can be obtained in time $\tilde{O}(\sqrt{n}\ell^3)$ using interior-point methods. For example, the time required for interior-point methods to find near-optimal solutions for MAXCUT's and COLORING's semidefinite relaxations are $\tilde{O}(n^{3.5})$ and $\tilde{O}(\sqrt{n}m^3)$, respectively, where n is the number nodes and m is the number of edges.

In a very recent work, we [20] show how to obtain an approximate solution for some semidefinite relaxations more quickly by exploiting the structure of the underlining optimization problems. Specifically we show how to obtain near-optimal solutions for MAXCUT's and COLORING's semidefinite relaxations in time $\tilde{O}(mn)$, where n is the number of nodes and m is the number of edges.

In this paper we explore the efficiency of algorithms for semidefinite relaxations in another domain, i.e. the space complexity. The space required by interior-point methods is $\Omega(n^2)$ because it has to maintain a full-rank $n \times n$ matrix during its execution. Our algorithm for MAXCUT's semidefinite relaxation given in [20] is potentially better because it does not have to maintain a full-rank matrix all the time. Specifically, it starts with an initial rank-one matrix, and the proceeds iteratively. In each iteration it apply a rank-one update to the current solution matrix. Unfortunately the number of iterations can be as large as n. Therefore the output tends to be full-rank, and its space complexity is thus asymptotically the same as that of interior-point methods. Similarly the output of our previous algorithm for COLORING's semidefinite relaxation tends to have full rank.

In this paper we show how to equip our previous time-efficient algorithm for MAXCUT with a *rank-reduction procedure*. The purpose of the rank-reduction procedure is to keep the rank of the current solution matrix at $O(\sqrt{n})$. As a result, the space required by the new algorithm during the execution can be kept at $(n^{1.5})$. The space-efficient algorithm for MAXCUT's semidefinite relaxation can be applied to solving COLORING's semidefinite program, as shown in our previous paper [20], and results in a space-efficient approximation algorithm for COLORING's semidefinite relaxation. The space-efficient algorithm for COLORING's semidefinite program can then be applied to the only known polynomial-time algorithm for MAXCLIQUE on perfect graphs, and results in a space-efficient algorithm for finding a maximum clique in a perfect graph.

2 Overview

Space efficiency is a practical issue in solving mathematical programs. Usually the program for solving a mathematical program has to keep everything in the main memory. Therefore the $O(n^2)$ space requirement for solving semidefinite programs has been keeping the size of a practically solvable problem small, even if we are willing to spend more time. In this paper we show how to obtain solutions for three problems in space $\tilde{O}(m + n^{1.5})$. Our results could increase the size of practically solvable problem by a significant factor. For example, let the size of main memory be M. When $m = O(n^{1.5})$, the solvable size n will be increased by a factor of $M^{\frac{2}{3}-\frac{1}{2}} = M^{\frac{1}{6}}$ using our space-efficient algorithms

2.1 MAXCUT

Our previous time-efficient algorithm for MAXCUT's semidefinite program [20] starts with a rank-one feasible solution, and then proceeds iteratively to improve its quality. In each iteration a rank-one update is applied to the current solution, so the rank of the current solution is increased by at most one. Since the number of iterations could as large as $\tilde{O}(\epsilon^{-2}n)$, the resulting solution is likely to be full-rank. However, it follows from Pataki's results that no matter how big the rank of the current solution is, there always exists a solution of rank $O(\sqrt{n})$ that has the same quality. Therefore our space-efficient algorithm is basically the time-efficient algorithm augmented with a number of *rank-reduction* steps, which was first proposed by Pataki [27]. Specifically, whenever the rank of the current solution is one more than the bound given by Pataki, we replace the current solution matrix by a new matrix that has one less rank, and has (almost) the same quality. That way we keep the rank of the current solution low. during the execution of their algorithm. As a result we reduce the space required by the currently best approximation algorithm for MAXCUT down to $O(m + n^{1.5})$, where $O(m)$ comes from the input.

2.2 COLORING

Our previous time-efficient algorithm for COLORING's semidefinite program [20] runs in $\tilde{O}(\epsilon^{-2})$ iterations. In each iteration an approximate solution for a MAXCUT semidefinite program is obtained. The resulting approximation solution of the COLORING semidefinite program is a linear combination of the solutions obtained from all iterations. If we resort to our space-efficient algorithm for MAXCUT semidefinite program in each of those iterations, then we are guaranteed to find an approximate solution for COLORING semidefinite program whose rank is $\tilde{O}(\epsilon^{-2}\sqrt{n})$. This gives an approximation algorithm for COLORING's semidefinite relaxation in space $O(m)+\tilde{O}(\epsilon^{-2}n^{1.5})$. It is interesting to observe that Pataki's results only guarantee a rank-$O(\sqrt{m})$ solution for the COLORING semidefinite program. Therefore the rank of the solution output by our algorithm is asymptotically less than the bound given by Pataki when ϵ is a constant and the graph is dense.

2.3 MAXCLIQUE

The only known polynomial-time algorithm for MAXCLIQUE on perfect graphs is based on the observation that the clique number for a perfect graph can be computed in polynomial time [12]. It follows from a result in the journal version of [18] that the clique number for a perfect graph is strongly related to the optimal value of its semidefinite relaxation of COLORING. Therefore we can use our space-efficient algorithm for COLORING's semidefinite relaxation to reduce the space requirement of the only known polynomial-time algorithm for finding a maximum clique on a perfect graph.

2.4 Outline

Here is the structure for the rest of the paper. We first give some preliminaries in the following two sections. In Section 5 we explain the rank-reduction procedure used by our space-efficient algorithms. It is interesting to note that our explanation can be regarded as an alternative proof for Pataki's rank bound on the basic solutions of semidefinite programs. In Section 6, 7, 8, and 9 we give the space-efficient algorithm for MAXCUT's semidefinite program. In Section 10 we show the space-efficient algorithm for COLORING's semidefinite program. In Section 11 we show how to reduce the space required by the polynomial-time algorithm for solving MAXCLIQUE on perfect graphs.

Due to the page limit, we omit proofs of Lemma 1, 3, and 4. They can all be easily proved.

3 Preliminaries

All matrices and vectors are real in the paper. All vectors are column vectors in the paper. Let x be an n-element vector. Define

$$\|x\|_2 = \sqrt{\sum_{1\le i\le n} x_i^2}.$$

Let A and B be two $m \times n$ matrices. Define

$$\|A\|_F = \sqrt{\sum_{1\le i\le m}\sum_{1\le j\le n} A_{ij}^2}.$$

$$A \bullet B = \sum_{1 \le i \le m} \sum_{1 \le j \le n} A_{ij} B_{ij}.$$

We use $X \succeq 0$ to signify that X is symmetric and positive semidefinite. Vectors $v_1, \ldots, v_n$ are *orthonormal* if the following holds for every $1 \le i, j \le n$.

$$v_i^T v_j = \begin{cases} 1 & \text{when } i = j \\ 0 & \text{when } i \ne j. \end{cases}$$

Clearly if the columns of an $n \times k$ matrix U are orthonormal, then $U^T U$ is the $k \times k$ identity matrix.

It is well-known that every $n \times n$ symmetric rank-k matrix X can be written as $X = VDV^T$ for some $k \times k$ diagonal matrix D and $n \times k$ matrix V, where the columns of V are orthonormal (see e.g. Theorem 2.5.4 of [16]).

We also need the following lemmas in the paper.

Lemma 1 Let X be a symmetric matrix. Suppose $X = UDU^T$, where the columns of U are linearly independent, and D is $k \times k$ and diagonal.

- If the rank of X is k, then $D_{ii} \ne 0$ for every $1 \le i \le k$.
- $X \succeq 0$ if and only if $D_{ii} \ge 0$ for every $1 \le i \le k$.

Lemma 2 Let S and $\hat{S}$ be two symmetric $k \times k$ matrices, where $S \succeq 0$ and $\hat{S} \not\succeq 0$. Then

$$\hat{\xi} = \max\{\xi : S + \hat{\xi}\hat{S} \succeq 0\}.$$

is well-defined. Moreover the rank of $S + \hat{\xi}\hat{S}$ is at most $k - 1$.

Proof Since $S \succeq 0$, it follows from Theorem 8.7.1 of [15] that there exists a nonsingular $k \times k$ matrix R such that $S = RDR^T$ and $\hat{S} = R\hat{D}R^T$, where D and $\hat{D}$ are diagonal. Since R is nonsingular, we know the columns of R are linearly independent. Since $S + \xi\hat{S} = R(D + \xi\hat{D})R^T$, it follows from Lemma 1 that $S + \xi\hat{S} \succeq 0$ if and only if all (diagonal) elements of $D + \xi\hat{D}$ are nonnegative. By the second statement of Lemma 1 we know $D_{ii} \ge 0$ for every i and $\hat{D}_{jj} < 0$ for some j. Therefore $\hat{\xi}$ is well-defined as

$$\hat{\xi} = \min\{-D_{ii}/\hat{D}_{ii} : \hat{D}_{ii} < 0\}.$$

Suppose $\hat{\xi} = -D_{jj}/\hat{D}_{jj}$ for some $1 \le j \le k$. Clearly the j^{th} diagonal element of $D + \hat{\xi}\hat{D}$ is zero. It follows from Lemma 1 that the rank of $S + \hat{\xi}\hat{S}$ is at most $k - 1$. $\square$

Lemma 3 Let X, U, and S be three nonzero matrices such that $X = USU^T$. The following statements hold.

- The rank of X is no more than the rank of S.
- X is positive semidefinite if S is positive semidefinite.

4 Compact representation of low-rank matrices

A procedure for the following problem is required by our space-efficient algorithm.

> Given an $n \times n$ symmetric matrix X of rank k, where k is unknown, but $k \le \ell$ for some known $\ell \le n$, compute an $n \times k$ matrix U and a symmetric $k \times k$ matrix S such that $X = USU^T$.

Since the rank of X is k, there exists a $k \times k$ diagonal matrix D and an $n \times k$ matrix V, whose columns are orthonormal, such that $X = VDV^T$. Let v_i be the i^{th} column of V. Namely $X = \sum_{1 \le i \le k} \lambda_i v_i v_i^T$. In order to solve the above problem, we first compute ℓ vectors $w_1, \ldots, w_\ell$, where each w_i is obtained by multiplying X by a random vector r_i. Namely $w_i = Xr_i$, for every $1 \le i \le \ell$. We have the following lemma.

Lemma 4 The linear space spanned by $v_1, \ldots, v_k$ is equal to the linear space spanned by $w_1, \ldots, w_\ell$ with probability one.

Now we orthonormalize $w_1, \ldots, w_\ell$ using the Gram-Shmidt procedure (see e.g. Chapter 5 of [15]), and get a set of orthonormal vectors $u_1, \ldots, u_k$ such that the linear space spanned by

$w_1, \ldots, w_\ell$ is equal to the linear space spanned by $u_1, \ldots, u_k$. Let U be the $n \times k$ matrix whose columns are $u_1, \ldots, u_k$. Since the linear space spanned by $u_1, \ldots, u_k$ is equal to the linear space spanned by $v_1, \ldots, v_k$, we know there exists a $k \times k$ matrix R such that $V = UR$. It follows that $X = URDR^TU^T$. Therefore we know there exists a symmetric $k \times k$ matrix $S = RDR^T$ such that $X = USU^T$. In fact $S = U^TXU$, since the columns of U are orthonormal. We then have a USU^T factoring for the given matrix X.

If the given matrix X is in the form of $\bar{U}\bar{S}\bar{U}^T$, where $\bar{U}$ is $n \times \bar{k}$ and $\bar{S}$ is $\bar{k} \times \bar{k}$, then the rank of X is at most $\bar{k}$ by Lemma 3. One can easily verify from the above that the USU^T decomposition of X can be found in time $O(n\bar{k}^2)$ and space $O(n\bar{k})$.

5 Pataki's rank bound on the basic solutions of semidefinite programs

Consider the following set of symmetric positive-semidefinite matrices

$$Q = \{X \succeq 0 : A^{(i)} \bullet X = b_i,\ i = 1, \ldots, m\},$$

where X and $A^{(1)}, \ldots, A^{(m)}$ are $n \times n$ matrices. Pataki [26] shows that if Q nonempty, then Q contains a matrix of k, for some k such that $k(k+1) \leq 2m$. The low-rank matrix is called a *basic solution*, which is the analogy of a basic solution for a linear program [7]. His proof relies on the facial structure of the positive-semidefinite cone, which requires background from convex analysis [30, 4]. Based on the existence of basic solutions, Pataki [27] also shows how to obtain a basic solution from a feasible solution by performing a sequence of rank-reduction procedure.

In the section we explain Pataki's rank-reduction procedure. Our explanation can be regarded as an alternative proof for Pataki's rank bound on basic matrix solutions. The key for the rank-reduction procedure is the following straightforward observation.

> A homogeneous linear system has a nonzero solution if the number of variables is more than the number of constraints.

Pataki's bound on the rank of a basic solution follows inductively from the following lemma.

Lemma 5 Suppose Q contains a matrix X whose rank is k, where $k(k+1) \geq 2m+1$. Then Q contains a matrix X' of rank at most $k-1$.

Proof The proof is constructive, which explains Pataki's rank-reduction procedure. Since the rank of X is k, it follows from §4 that $X = USU^T$ for some $k \times k$ symmetric matrix S and $n \times k$ matrix U, where the columns of U are orthonormal. Since $X \succeq 0$ and $S = U^TXU$, we know $S \succeq 0$ by the second statement of Lemma 3. Suppose we can find a $k \times k$ symmetric matrix $\hat{S}$ such that

1. $\hat{S} \not\succeq 0$, and
2. $A^{(i)} \bullet (U\hat{S}U^T) = 0$ for every $i = 1, \ldots, m$.

It follows from Lemma 2 that the rank of $S + \hat{\xi}\hat{S}$ is at most $k-1$, where $\hat{\xi} = \max\{\xi : S + \xi\hat{S}\}$. Let $S' = S + \hat{\xi}\hat{S}$. Let $X' = US'U^T$. It follows from Lemma 3 that $X' \succeq 0$ and the rank of X' is at most $k-1$. Since $A^{(i)} \bullet (U\hat{S}U^T) = 0$ for every $i = 1, \ldots, m$, we know $A^{(i)} \bullet X' = b_i$ for every $i = 1, \ldots, m$. Therefore X' is a matrix in $Q_{\bar{X}}$ whose rank is at most $k-1$.

It remains to show the existence of a $k \times k$ symmetric matrix $\hat{S}$ with the required properties. Let us focus on the second property first. Note that each

$$A^{(i)} \bullet (U\hat{S}U^T) = 0$$

is a homogeneous linear constraint on $\hat{S}$. Since $\hat{S}$ is symmetric and $k \times k$, it has to satisfies $\frac{k^2-k}{2}$ other homogeneous linear constraints. As a result, we have a homogeneous linear system with k^2 variables and $m + \frac{k^2-k}{2}$ constraints. Clearly

$$\begin{aligned} k^2 - \left(m + \frac{k^2-k}{2}\right) &= \frac{k^2+k}{2} - m \\ &\geq 1. \end{aligned}$$

The number of variables is at least one more than the number of constraints. Therefore the homogeneous linear system has a nonzero solution $\check{S}$. Clearly $-\check{S}$ is also a nonzero solution for the system. At least one of $\check{S}$ and $-\check{S}$ is not positive semidefinite. Therefore at least one of them can be the $\hat{S}$ that has the required properties. $\square$

6 The algorithm for VECTOR MAXCUT

Let G be a graph composed of n nodes and m edges with positive weights. Let C be the matrix such that C_{ij} is the weight of edge ij. In [20] we give an algorithm VectorMaxCut(C, ϵ) for solving the following semidefinite program to within relative error ϵ.

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

where L is the perturbed Laplacian matrix for the graph. At the end of the algorithm, the Cholesky decomposition of an ϵ-optimal solution is reported. The space required by VectorMaxCut is $\tilde{O}(n^2)$. In this section we show how to modify that algorithm and obtain an algorithm solving the same problem whose space requirement is only $O(m + n^{1.5})$, where $O(m)$ comes from the input.

CompactVectorMaxCut(C, ϵ), the compact version of the approximation algorithm, is as follows, where the procedure Initial(C) is given in [20].

1. Scale C such that the sum of edge weights 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 $(X, \lambda) =$ CompactImprove$(X, \lambda, \epsilon'/7)$.

The algorithm starts with finding an initial rank-one matrix (X, λ). It then iteratively calls the subroutine CompactImprove to improve the quality of the current solution. The only thing that CompactVectorMaxCut differs from the original version VectorMaxCut is that the new subroutine CompactImprove runs in space $O(n^{1.5})$ and always outputs the matrix X in the form of USU^T, where U is $n \times k$ and S is $k \times k$, for some $k = O(\sqrt{n})$.

Define $\langle X \rangle_y = y_1 X_{11} + \cdots + y_n X_{nn}$. We give the subroutine CompactImprove(X, λ_0, ϵ) 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 $\langle X \rangle_y - \langle \tilde{X} \rangle_y \le \epsilon(\langle X \rangle_y + \lambda y \cdot \vec{1})$
       Return (X, λ).
     else
       Let $X = (1 - \sigma)X + \sigma \tilde{X}$.
       Let $X = $ RankReduction(X).
       Let $\lambda = \max\{X_{11}, \ldots, X_{nn}\}$.

The subroutine Direction is given in [20], which runs in time $\tilde{O}(m\epsilon^{-1})$ and space $O(n)$. It always outputs a rank-one matrix in the form of $\tilde{u}\tilde{u}^T$ for some vector $\tilde{u}$.

The only thing that CompactImprove differs from the original version is that a new procedure RankReduction is called in each iteration when the rank-one update is applied to the current matrix. The purpose is to ensure that the rank k of the current matrix X satisfies that $k(k + 1) \le 2n + 2$. Specifically, RankReduction$(\bar{X})$ computes the USU^T decomposition of $\bar{X}$ using the procedure described in §4, and therefore determines the rank k of $\bar{X}$. If $k(k + 1) \ge 2n + 3$, then it uses Pataki's rank-reduction procedure described in the proof of Lemma 5 to find a matrix $X' \in Q_{\bar{X}}$ whose rank is at most $k - 1$, where $Q_{\bar{X}}$ is the following set of matrices

$$\{X \succeq 0 : L \bullet X = L \bullet \bar{X},\, X_{ii} = \bar{X}_{ii},\, 1 \le i \le n\}.$$

Since $\bar{X} \in Q_{\bar{X}}$, we know $Q_{\bar{X}}$ is not empty. Lemma 5 guarantees the existence of X', which is also the output of RankReduction$(\bar{X})$.

In order to claim that CompactVectorMaxCut is a space-efficient algorithm, it remains to show how to implement RankReduction to run in space $O(m + n^{1.5})$.

7 The rank-reduction step

RankReduction($\bar{X}$) has the following two steps.

1. Use the procedure given in §4 to compute the USU^T decomposition of $\bar{X}$ where the columns of U are orthonormal, and S is a $k \times k$ full-rank matrix.

2. If $k(k+1) \leq 2n + 2$, then return $\bar{X}$ in the form of USU^T. Otherwise,

   (a) Find a symmetric $k \times k$ matrix $\hat{S}$ such that $\hat{S} \not\succeq 0$, $L \bullet (U\hat{S}U^T) = 0$ and the diagonal elements of $U\hat{S}U^T$ are all zero.

   (b) Find a nonsingular matrix R and two diagonal matrices D and $\hat{D}$ such that $S = RDR^T$ and $\hat{S} = R\hat{D}R^T$.

   (c) Compute $\hat{\xi} = \min\{-D_{ii}/\hat{D}_{ii} : \hat{D}_{ii} < 0\}$. Let $S' = S + \hat{\xi}\hat{S}$. Let $X' = US'U^T$.

   (d) Return X' in the form of $US'U^T$.

The input matrix $\bar{X}$ is $(1 - \sigma)$ times the output of the previous iteration plus a rank-one update $\sigma\tilde{u}\tilde{u}^T$. Therefore is can be put in the form $\bar{U}\bar{S}\bar{U}^T$, where U is $n \times \bar{k}$ and $\bar{S}$ is $\bar{k} \times \bar{k}$ for some $\bar{k} = O(\sqrt{n})$. It follows from §4 that the first step can be done in time $O(n^2)$ and space $(n^{1.5})$.

Step 2-(b) can be done in time $O(k^3)$ and space $O(k^2)$ using, for example, Algorithm 8.7.1 in [15].

The real challenge comes from Step 2-(a), which requires to find a nonzero solution for the underconstrained homogeneous linear system $Ax = 0$ with $O(n)$ variables and $O(n)$ constraints. Note that $L \bullet (U\hat{S}U^T)$ is equal to $(U^TLU) \bullet \hat{S}$, where U^TLU is a $k \times k$ obtainable in time $O(mk + nk^2)$. Also Note that the i^{th} diagonal element of $U\hat{S}U^T$ is exactly $u_i^T\hat{S}u_i$ where u_i is the i^{th} column of U^T. Therefore the corresponding matrix A can be represented sparsely in space $O(n^{1.5})$, although it is $O(n) \times O(n)$.

Direct methods such as Gaussian Elimination cannot be applied here to find a nonzero solution for $Ax = 0$, since it would take $O(n^2)$ space. Therefore we have to resort to iterative methods which can find an approximate solution in space $O(n)$. Since iterative methods only give us approximate solution, we have to ensure that the error does not affect the quality of the current solution matrix by too much.

When $k(k+1) \geq 2n + 3$, ideally we want to find a rank-$(k-1)$ matrix X' in $Q_{\bar{X}}$. Namely we want X' and $\bar{X}$ to have exactly the same potential, so the numbers of iteration required by CompactImprove and Improve will have the same bound as. Since the solution for the linear system has error, the corresponding X' will not be in $Q_{\bar{X}}$. However, it would be OK if the potential of X' is slightly more than that of $\bar{X}$.

Specifically by the analysis shown in [28] each iteration of the original Improve(X, λ_0, ϵ) given in [20] reduces the potential by a factor of $\Delta = \frac{\epsilon^2\lambda^*}{4n}$. The number of iterations can therefore be shown to be $O(\epsilon^{-2}n\log(n\epsilon^{-1}))$. The number of iterations will be at most twice as many, even if each iteration reduces the potential only by a factor of $\Delta/2$. Clearly $(1-\Delta)(1+\Delta/2) \geq (1-\Delta/2)$. Therefore it would be OK if the approximate solution for $Ax = 0$ gives a $\hat{S}$ such that adding $\hat{\xi}\hat{X}$ to $\bar{X}$ increases the potential of $\bar{X}$ by at most a factor of $\Delta/2$.

As defined in [20], the potential of a solution matrix X, where $L \bullet X = 1$, is $\sum_{1\leq i\leq n} e^{\alpha X_{ii}}$. Let $\delta = \max_i |\hat{X}_{ii}|$. If we add $\hat{\xi}\hat{X}$ to the current matrix $\bar{X}$, then the increase of potential is at most a factor of $2\hat{\xi}\delta$, because

$$\begin{aligned}
&\sum_{1\leq i\leq n} (e^{\alpha\bar{X}_{ii}+\hat{\xi}\delta} - e^{\alpha\bar{X}_{ii}}) \\
&= (e^{\hat{\xi}\delta} - 1) \sum_{1\leq i\leq n} e^{\alpha\bar{X}_{ii}} \\
&\leq 2\hat{\xi}\delta \sum_{1\leq i\leq n} e^{\alpha\bar{X}_{ii}},
\end{aligned}$$

when $\hat{\xi}\delta$ is small. Therefore we can reduce the goal to be ensuring that $\hat{\xi}\delta \leq \frac{\Delta}{4}$. By the following lemma we know it suffices to guarantee that $\|\hat{X}\|_F \geq 1$ and $\delta \leq \frac{\Delta}{16n\lambda}$, since $\frac{\Delta}{16n\lambda}$ is clearly no more than $\frac{1}{2n^2}$.

Lemma 6 If $\delta \leq \frac{1}{2n^2}$ and $\|\hat{X}\|_F \geq 1$ then $\hat{\xi} \leq 4n\bar{\lambda}$, where $\bar{\lambda}$ is the maximum diagonal element of $\bar{X}$.

Proof By the weak duality of cone-LP, (see e.g. [1]), we have

$$\hat{\xi} = \max\{\xi : \bar{X} + \xi\hat{X} \succeq 0\} \leq \bar{X} \bullet Y,$$

for any $Y \succeq 0$ such that $\hat{X} \bullet Y = -1$. We prove the lemma by presenting a matrix $\hat{Y} \succeq 0$ such that $\hat{X} \bullet \hat{Y} \leq -1$ and $\bar{X} \bullet \hat{Y} \leq 4n\bar{\lambda}$.

Since $\|\hat{X}\|_F \geq 1$, $\hat{X}$ must have one element, say $\hat{X}_{ij}$, whose absolute value is no less than $\frac{1}{n}$. Choose $\hat{Y}$ to be the $n \times n$ matrix with exactly four nonzero elements as follows: $\hat{Y}_{ii} = \hat{Y}_{jj} = n$, and $\hat{Y}_{ij} = \hat{Y}_{ji} = -n\frac{\hat{X}_{ij}}{|\hat{X}_{ij}|}$. Clearly $\hat{Y} \succeq 0$. We also know that $\hat{X} \bullet \hat{Y} \leq -2 + \frac{1}{n} \leq -1$, since $\delta \leq \frac{1}{2n^2}$.

It remains to show $\bar{X} \bullet \hat{Y} \leq 4n\bar{\lambda}$. Note that both $\bar{X}_{ii}$ and $\bar{X}_{jj}$ are less than or equal to $\bar{\lambda}$ by definition of $\bar{\lambda}$. Since $\bar{X} \succeq 0$, we know $|\bar{X}_{ij}| = |\bar{X}_{ji}| \leq \bar{\lambda}$. Therefore $\bar{X} \bullet \hat{Y} \leq 4n\bar{\lambda}$. □

Therefore in each rank-reduction step, we find a nonzero solution $\hat{x}$ for a homogeneous linear system $Ax = 0$, and $\hat{S}$ can be obtained from $\hat{x}$. By the above argument we know that the norm of Ax, which is an upper bound of every $\hat{X}_{ii}$, has so small that the matrix $\hat{X} = U\hat{S}U^T$ has to satisfy the following conditions.

1. $\|\hat{X}\|_F \geq 1$, and
2. $|\hat{X}_{ii}| \leq \frac{\Delta}{16n\bar{\lambda}} = \frac{\epsilon^2\lambda^*}{64n^2\bar{\lambda}} \leq \frac{\epsilon^2}{192n^2}$.

One can verify that $\|A\|_F \leq 2n$. Therefore to finding a sufficiently good $\hat{S}$, it suffices to find a nonzero vector x such that

1. $\|x\|_2 \geq 1$, and
2. $\|Ax\|_2 \leq \frac{\epsilon^2}{384n^3}\|A\|_F$.

8 Approximately solving a homogeneous linear system in exact arithmetic

Given a homogeneous linear system $\bar{A}x = b$, where $\bar{A}$ is a nonsingular $n \times n$ matrix, and b is nonzero. Then an approximate solution to the system with error at most ϵ' can be found in time $O(n^3 \log(1/\epsilon'))$ in exact arithmetic and space $O(n)$ using iterative methods like the *conjugate gradient method* or the *steepest descent method* [32]. It can be shown that in our homogeneous system $Ax = 0$, the number of variables is $O(\sqrt{n})$ more than the number of constraints. Therefore we can throw in $O(\sqrt{n})$ linear constraints with random coefficients to make the system nonsingular and nonhomogeneous, and then use iterative methods to find an approximate solution of the new system, which also clearly gives a nonzero approximate solution for the original homogeneous system. Note that throw in those $O(\sqrt{n})$ constraints requires $O(n^{1.5})$ extra space, which fortunately does not exceed the space bound we are aiming for. As for exactly how many extra constraints have to be added to make the system nonsingular but still consistent, we can use binary search, which will add an $O(\log n)$ factor to the required running time.

Therefore we can achieve the goal shown at the end of the previous section in time $O(n^3 \log^2(n\epsilon^{-1}))$ and space $O(n^{1.5})$ using exact arithmetic. (In practice, however, the running time of iterative methods depends polynomially on the condition number of the matrix A, i.e. the largest ratio of the absolution values of two eigenvalues of A. Researchers have been developing various kinds of preconditioning techniques to bring down the condition number of the given linear system [5, 19, 31].)

9 Overall complexity

As shown in [20], the total number of iterations required by CompactVectorMaxCut(C, ϵ) is $O(\epsilon^{-2} n \log(n\epsilon^{-1}))$ and the running time of each iteration is dominated by the time required by solving the homogeneous linear system, which is $O(n^3 \log^2(n\epsilon^{-1}))$ using exact arithmetic. It follows that the time complexity of our space-efficient algorithm is $O(\epsilon^{-2} n^4 \log^3(n/\epsilon))$ using exact arithmetic. As shown in the previous sections, the space required is $O(m + n^{1.5})$.

10 The algorithm for VECTOR COLORING

If the given graph is k-colorable, in [20] we show that an ϵ-optimal solution for COLORING's semidefinite relaxation can be found by making $O(\epsilon^{-1}\log(n\epsilon^{-1}))$ subroutine calls to VECTORMAXCUT, each of which finds an $\frac{\epsilon}{6k}$-optimal solution to a semidefinite relaxation for some graph MAXCUT. The resulting ϵ-optimal solution is simply a linear combination of the approximate solutions reported by those $\tilde{O}(\epsilon^{-2})$ subroutine calls to VECTORMAXCUT. Clearly we can replace VECTORMAXCUT by COMPACTVECTORMAXCUT. We therefore have an algorithm COMPACTVECTORCOLORING(G,ϵ), which runs in space $O(m+n^{1.5}\epsilon^{-2}\log(n\epsilon^{-1}))$.

Since the matrix output by COMPACTVECTORMAXCUT has rank at most $\sqrt{2n}$, the rank of the solution output by COMPACTVECTORCOLORING has rank $\tilde{O}(\epsilon^{-2}\sqrt{n})$. It is interesting to note that if ϵ is a constant, then the rank of the ϵ-optimal solution output by COMPACTVECTORCOLORING is asymptotically lower than Pataki's rank bound $O(\sqrt{m})$ on the optimal solutions.

11 The algorithm for MAXCLIQUE on perfect graphs

The only known polynomial-time algorithm for MAXCLIQUE on perfect graphs [12] is based on the observation that the clique number for a perfect graph can be computed in polynomial time [23, 21]. In each iteration of the algorithm, the clique number of an induced subgraph, which is therefore perfect [24] and has less edges, has to be computed. Let G be a perfect graph. Let $\omega(G)$ be G's clique number. Let $\lambda(G)$ be the optimal value of G's semidefinite relaxation for COLORING. It is shown in the journal version of [18] that $\lambda(G) = 1/(1-\omega(G))$. If the given perfect graph is k-colorable, then it is well-known that $\omega(G) \leq k$. It follows that an $O(1/k)$-optimal approximation to $\lambda(G)$ can be used to determine $\omega(G)$, since $\omega(G)$ is an integer. It follows that our COMPACTVECTORCOLORING gives a way to implement the algorithm given in [12] in space $O(m+k^2n^{1.5}\log(nk))$.

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] N. Alon and N. Kahale. Approximating the independence number via the θ-fucntion. Unpublished manuscript, Nov. 1994.

[3] A. Blum and D. Karger. An $\tilde{O}(n^{3/14})$-coloring for 3-colorable graphs. Submitted to *Information Processing Letters*, Jan. 1996.

[4] A. Brøndsted. *An Introduction to Convex Polytopes.* Springer-Verlag, New York Heidelberg Berlin, 1983.

[5] A. M. Bruaset. *A Survey of Preconditioned Iterative Methods.* Longman Scientific & Technical, 1995.

[6] B. Chor and M. Sudan. A geometric approach to betweenness. In *Proceedings of the Third Annual European Symposium Algorithms*, pages 227–237, 1995.

[7] V. Chvátal. *Linear Programming.* Freeman, New York, 1983.

[8] U. Feige and M. X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the Third Israel Symposium on Theory of Computing and Systems*, 1995.

[9] 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.

[10] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, to appear.

[11] 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.

[12] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[13] M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. *Annal of Discrete Mathematics*, 21:325–357, 1985.

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

[15] G. H. Holub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, second edition, 1989.

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

[17] 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.

[18] 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. IEEE, 1994.

[19] C. T. Kelly. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, 1995.

[20] P. Klein and H.-I. Lu. Efficient approximation algorithms for semidefinite programs arising from MAXCUT and COLORING. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 338–347, 1996.

[21] D. E. Knuth. The sandwich theorem. *The Electronic Journal of Combinatorics*, 1(A1):1–48, 1994.

[22] 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 Inst. of Technology, 1994.

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

[24] L. Lovász. Perfect graphs. In L. W. Beineke and R. J. Wilson, editors, *Selected Topics in Graph Theory 2*, pages 55–87. Academic Press, London, 1983.

[25] Y. Nesterov and A. Nemirovskii. *Self-Concordant Functions and Polynomial Time Methods in Covex Programming*. Central Economic and Mathematical Institute, USSR Academy of Science, Moscow, 1989.

[26] G. Pataki. On the facial structure of cone-LP's and semi-definite programs. Technical Report MSRR-595, Graduate School of Industrial Administration, Carnegie–Mellon University, 1994.

[27] G. Pataki. Cone-LP's and semidefinite programs: Geometry and a simplex-type method. In *Proceedings of the Fifth IPCO Conference on Integer Programming and Combinatorial Optimization*, 1996.

[28] 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. IEEE, 1991.

[29] 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.

[30] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.

[31] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, 1996.

[32] J. R. Shewchuk. An introduction to the Conjugate Gradient Method without the agonizing pain. Technical Report CMU-CS-94-125, Shool of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, Mar. 1994.

[33] 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. *Math. Prog.* (to appear).

[34] L. Vandenberghe and S. Boyd. Semidefinite programming. Technical report, Information Systems Laboratory, Stanford University, 1994.

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