On Approximation of Max-Vertex-Cover *

Qiaoming Han †
School of Mathematics and Computer Science
Nanjing Normal University
Nanjing, 210097, P.R.China

Yinyu Ye
Department of Management Sciences
Henry B. Tippie College of Business
The University of Iowa
Iowa City, IA 52242, USA

Hantao Zhang
Department of Computer Science
The University of Iowa
Iowa City, IA 52242, USA

Jiawei Zhang
Department of Management Sciences
Henry B. Tippie College of Business
The University of Iowa
Iowa City, IA 52242, USA

April 24, 2000

*Research supported in part by NSF grants CCR-9357851, DMI-9908077 and DMS-9703490.

†The author is currently visiting Computational Optimization Laboratory, Department of Management Sciences, The University of Iowa.

Abstract

We consider the Max-Vertex-Cover (MVC) problem, i.e., find k vertices from a undirected and edge-weighted graph $G = (V, E)$, where $|V| = n \geq k$, such that the total edge weight covered by the k vertices is maximized. This problem was introduced by Petrank who proved that the existence of a $(1-\epsilon)$-approximation algorithm for MVC implies P=NP for some $\epsilon > 0$. There is a 3/4-approximation algorithm for MVC, based on a linear programming (LP) relaxation. We illustrate that the guaranteed ratio can be improved by a simple greedy algorithm for $k > \frac{3}{4}n$, and a simple randomized algorithm for $k > \frac{1}{2}n$. Most importantly, we study semidefinite programming (SDP) relaxation based approximation algorithms for MVC. We prove that, for a range of k, our SDP-based algorithms achieve the best performance guarantee among the four types of algorithms described in this paper.

Key words. Max-vertex-cover, polynomial approximation algorithm, linear programming, semidefinite programming.

1 Introduction

A vertex cover of an undirected graph $G = (V, E)$ is a set, denoted by S, of vertices such that $i \in S$ or $j \in S$ for every edge $(i, j) \in E$. The (minimum) Vertex Cover (VC) problem is to find a vertex cover containing the smallest number of vertices. The VC problem has been considered one of the most prominent problems in combinatorial optimization. It was among the first to be proved NP-complete [13]. For the related results on the VC problem, we recommend the excellent survey by Hochbaum [12].

In this paper, we consider a "dual" variation of the VC problem, which was introduced by Petrank [18], and it is called the Max-Vertex-Cover (MVC) problem. That is, given a positive integer k, $k \leq n$, find k vertices from G such that they cover as many edges as possible. It has been proved that the MVC problem is NP-hard ([18]). Hence, one should not expect to find a polynomial time algorithm for solving this problem exactly. Therefore we are interested in developing polynomial time approximation algorithms for MVC. A (randomized) algorithm for the maximization problem is called (randomized) r-approximation algorithm, where $0 < r \leq 1$, if it outputs a feasible solution with its (expected) value at least r times the optimum value for all instances of the problem.

By using the concept of r-approximation algorithm, Petrank's complexity result in [18] can be stated in a stronger way: There exists an $\epsilon > 0$ such that it is NP-hard to find a $(1 - \epsilon)$-approximation algorithm for MVC.

The first positive result for MVC is a 3/4-approximation algorithm due to Ageev and Sviridenko [1]. A key step in designing a good approximation algorithm for such a maximization problem is to establish a good upper bound on the maximal objective value. They show that MVC can be formulated as an $0 - 1$ integer programming problem and thus they can relax it to a linear program (LP). The maximal value of the LP relaxation provides the desired upper bound, and they go on to establish a 3/4-approximation algorithm for MVC.

The algorithm to use linear programming (and duality) was devised for the (minimization) Vertex Cover and Set Cover problems by Hochbaum [11]. LP relaxations of integer programs have been used extensively in designing approximation algorithms for many other fundamental combinatorial optimization problems, e.g., scheduling and facility location (Shmoys [19]), $b-$balanced cut (Leighton and Rao [15]), Max SAT (Goemans and Williamson [8]), etc.

In this paper we describe two simple algorithms: a greedy algorithm and a randomized algorithm. We illustrate that the greedy algorithm possesses a worst-case performance ratio $\frac{k}{n}$, and the randomized algorithm possesses $2(\frac{k}{n}) - (\frac{k}{n})^2$.

Our main results presented in this paper are two semidefinite programming (SDP) relaxation based approximation algorithms for MVC: one is a straightforward SDP relaxation of MVC and the other is a strengthened SDP relaxation by adding triangle inequalities. We will prove that, for a wide range of k, the SDP-based algorithms achieve the best performance guarantees among the four types of approximation algorithms described here. For instance, when $k = n/2$, the provable performance guarantees are 0.75 for the LP-based algorithm, 0.5 for the greedy algorithm, 0.75 (roughly) for the randomized algorithm, 0.77 for the SDP-based algorithm, and .81 for the strengthened SDP-based algorithm. We have recently learnt that Feige and Langberg [5] also studied SDP-based algorithms on MVC and obtained improved results. Our approximation ratios (see Table 2) are slightly better than the 0.8 result, for $k \geq n/2$, established by them in [5].

Since the seminar work of Goemans and Williamson [8] for MAX-CUT, SDP relaxations have recently proved useful in obtaining improved approximation algorithms for several graph optimization problems, including Vertex Coloring (Karger et.al. [14]), Maximum Independent Set (Alon and Kahale [2]), MAX-DICUT (Feige and Goemans [4]), MAX-k-CUT and MAX-BISECTION (Frieze and Jerrum [7], Ye [21]), Dense-k-Subgraph (Feige and Seltser [6], Srivastav and Wolf [20], Ye and Zhang [23], Han et.al. [10]), etc. Nesterov [17], Zwick [24] and Ye [22] have extended SDP to solving more general problems. Also see Mahajan and Ramesh [16] for some derandomization techniques for these SDP based randomized algorithms.

Our results presented in this paper actually hold for more generally weighted MVC problems: Given an undirected graph $G = (V, E)$ and a non-negative weight $w_{ij} = w_{ji}$ on each edge $(i, j) \in E$, the problem is to determine a subset $S \subset V$ of k vertices such that the total weight of the edges covered by S is maximized, i.e.,

$$w^* := \max_{S \subset V,\ |S|=k} \sum_{i \in S \text{ or } j \in S} w_{ij}. \tag{1}$$

The unweighted MVC problem is a special case of the general MVC problem when $w_{ij} = 1$ if $(i, j) \in E$ and $w_{ij} = 0$ otherwise.

2 LP relaxation of MVC

We first present Ageev and Sviridenko's LP-based algorithm. The (weighted) MVC problem could be formulated as the following $0-1$ integer program

$$
\text{(MVC)} \qquad \begin{aligned} w^* := \text{Maximize} \quad & \sum_{i<j} w_{ij} \cdot (x_i + x_j - x_i x_j) \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = k \\ & x_i \in \{0,1\}, \quad i = 1, \ldots, n, \end{aligned}
$$

where vertex i belongs to S if and only if $x_i = 1$. Replacing the quadratic terms $x_i x_j$ of the objective function by $0-1$ variables x_{ij}, we derive

$$
\text{(MVC)} \qquad \begin{aligned} w^* := \text{Maximize} \quad & \sum_{i<j} w_{ij} \cdot (x_i + x_j - x_{ij}) \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = k, \\ & x_{ij} \le x_i, \\ & x_{ij} \le x_j, \\ & x_{ij} \ge x_i + x_j - 1, \\ & x_i \in \{0,1\}, \\ & x_{ij} \in \{0,1\}, \quad 1 \le i, j \le n. \end{aligned}
$$

Relaxing the integral constraints, we have an LP relaxation of (MVC),

$$
\text{(LP-MVC)} \qquad \begin{aligned} w^{LP} := \text{Maximize} \quad & \sum_{i<j} w_{ij} \cdot (x_i + x_j - x_{ij}) \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = k, \\ & x_{ij} \le x_i, \\ & x_{ij} \le x_j, \\ & x_{ij} \ge x_i + x_j - 1, \\ & 0 \le x_i \le 1, \\ & 0 \le x_{ij} \le 1, \quad 1 \le i, j \le n. \end{aligned}
$$

Let

$$
F(x_1, x_2, \cdots, x_n) = \sum_{i<j} w_{ij}(x_i + x_j - x_i x_j).
$$

We now present the Ageev and Sviridenko LP-based Algorithm.

LP-Algorithm

1. **LP Solving:** Solve the linear program (LP-MVC) and obtain an optimal solution x^*. Let $\hat{x} = x^*$.

2. **Deterministic Rounding:** While there exists an i ($1 \le i \le n$) such that $0 < \hat{x}_i < 1$, there must be another $j \ne i$ (w.l.o.g., assume that $i < j$) such that $0 < \hat{x}_j < 1$ (since the constraint $\sum_{i=1}^n \hat{x}_i = k$ and k being an integer). Then, choose either $\min\{1 - \hat{x}_i, \hat{x}_j\}$ or $-\min\{\hat{x}_i, 1 - \hat{x}_j\}$ as δ such that

$$F(\hat{x}_1, \cdots, \hat{x}_i + \delta, \cdots, \hat{x}_j - \delta, \cdots, \hat{x}_n) \ge F(\hat{x}_1, \cdots, \hat{x}_i, \cdots, \hat{x}_j, \cdots, \hat{x}_n).$$

   Then let $\hat{x}_i = \hat{x}_i + \delta$ and $\hat{x}_j = \hat{x}_j - \delta$, and return to Step 2.

3. **Solution:** Let $S = \{i \mid \hat{x}_i = 1\}$. ∎

The following Lemma shows that the value of $F(\cdot)$ is non-decreasing in the rounding procedure.

Lemma 1 *If $0 < x_i, x_j < 1$, let $\delta_1 = \min\{1 - x_i, x_j\}$ and $\delta_2 = -\min\{x_i, 1 - x_j\}$, then either*

$$F(x_1, \cdots, x_i + \delta_1, \cdots, x_j - \delta_1, \cdots, x_n) \ge F(x_1, \cdots, x_i, \cdots, x_j, \cdots, x_n)$$

or

$$F(x_1, \cdots, x_i + \delta_2, \cdots, x_j - \delta_2, \cdots, x_n) \ge F(x_1, \cdots, x_i, \cdots, x_j, \cdots, x_n).$$

∎

This is because that, for any $\delta \in R$ such that $-\min\{x_i, 1 - x_j\} \le \delta \le \min\{1 - x_i, x_j\}$,

$$g(\delta) = F(x_1, \cdots, x_i + \delta, \cdots, x_j - \delta, \cdots, x_n) - F(x_1, \cdots, x_i, \cdots, x_j, \cdots, x_n)$$

can be reduced to a convex quadratic function in δ.

Note that since w_{ij} are non-negative for all i, j, (LP-MVC) is equivalent to

$$\begin{array}{rll} w^{LP} := \text{Maximize} & \sum_{i<j} w_{ij} \cdot (x_i + x_j - \max\{0, x_i + x_j - 1\}) \\ \text{subject to} & \sum_{i=1}^{n} x_i = k, \\ & 0 \le x_i \le 1 \quad \text{for} \quad i = 1, \ldots, n. \end{array}$$

Moreover, the following Lemma holds:

Lemma 2 *Let x^* be an optimal solution to (LP-MVC), then*

$$x_i^* + x_j^* - x_i^* x_j^* \geq \frac{3}{4}\left(x_i^* + x_j^* - \max\{0, x_i^* + x_j^* - 1\}\right). \qquad (2)$$

∎

Combining Lemmas 1 and 2, one can see that the final $\hat{x}$ resulted from the rounding procedure satisfies

$$F(\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n) \geq F(x_1^*, x_2^*, \cdots, x_n^*) \geq \frac{3}{4} \cdot w^{LP} \geq \frac{3}{4} \cdot w^*,$$

which leads to the Ageev and Sviridenko result:

Theorem 1 *The worst-case performance ratio of the* LP-Algorithm *for approximating the (weighted) MVC problem is at least* $3/4$ *for any given* $1 \leq k \leq n$.

3 Two simple approximation algorithms

In this section, we describe two simple approximation algorithms for MVC: a greedy algorithm and a randomized algorithm. The two algorithms have performance guarantees $\frac{k}{n}$ and $2(\frac{k}{n}) - (\frac{k}{n})^2$, respectively. Although these performance guarantees are not constant, they are better than the LP-Algorithm when $k > \frac{3}{4}n$ for the greedy algorithm and $k > \frac{1}{2}n$ for the randomized algorithm.

3.1 Greedy Algorithm

The algorithm simply does the following:

1. For each $i \in V$, let $\zeta(i) = \sum_{j \neq i} w_{ij}$ and renumber the vertices by the order such that $\zeta(1) \geq \zeta(2) \geq ... \geq \zeta(n)$.
2. Let $S = \{1, 2, \cdots, k\}$.

∎

For any $U \subset V$, denote the total weights within the subgraph induced by U as $w_1(U)$, that is, $w_1(U) = \sum_{i<j,\ i \in U,\ j \in U} w_{ij}$, and the total weights of the crossing edges as $w_2(U)$, that is, $w_2(U) = \sum_{i \in U, j \notin U} w_{ij}$, and $w(U) = w_1(U) + w_2(U)$.

Theorem 2 *The Greedy Algorithm has a performance guarantee of at least* $\frac{k}{n}$.

Proof. Let S be the subset of vertices selected by Greedy Algorithm, and recall that $\zeta(i) = \sum_{j \neq i} w_{ij}$ for each i. It is clear that

$$\sum_{i=1}^{n} \zeta(i) = 2 \sum_{1 \leq i < j \leq n} w_{ij} \geq 2w^*. \tag{3}$$

On the other hand,

$$\sum_{i=1}^{k} \zeta(i) = 2w_1(S) + w_2(S) \leq 2w(S). \tag{4}$$

Note that

$$\zeta(1) \geq \zeta(2) \geq ... \geq \zeta(n),$$

and thus we have

$$\sum_{i=1}^{k} \zeta(i) \geq \frac{k}{n} \sum_{i=1}^{n} \zeta(i). \tag{5}$$

Therefore, by (3), (4) and (5) we get

$$\frac{w(S)}{w^*} \geq \frac{k}{n}. \qquad \blacksquare$$

3.2 Randomized Algorithm

The algorithm is to run the following two steps many times and output the best solution.

1. Let each vertex i be selected to S independently with probability $\frac{k}{n}$.

2. Let $\tilde{S} = S$. Do one of the following two cases:

   If $|\tilde{S}| > k$, for each $i \in \tilde{S}$, let $\zeta(i) = \sum_{j \notin \tilde{S}} w_{ij}$ and $\tilde{S} = \{i_1, i_2, ..., i_{|\tilde{S}|}\}$, where $\zeta(i_1) \geq \zeta(i_2) \geq ... \geq \zeta(i_{|\tilde{S}|})$. Then, remove vertex $i_{|\tilde{S}|}$ from $\tilde{S}$ and reassign $\tilde{S} = \{i_1, i_2, ..., i_{|\tilde{S}|-1}\}$. Repeat this swapping procedure till $|\tilde{S}| = k$.

   If $|\tilde{S}| \leq k$, arbitrarily add $k - |S|$ vertices from outside of S into $\tilde{S}$. $\blacksquare$

Let $\hat{x}_i = 1$ if vertex i is selected to S, otherwise $\hat{x}_i = 0$, for $i = 1, 2, \cdots, n$. Then $\hat{x}_i$ is a random variable with

$$\Pr\{\hat{x}_i = 1\} = \frac{k}{n},$$

and thus

$$\mathrm{E}[\hat{x}_i] = \frac{k}{n}, \quad \text{for} \quad 1 \le i \le n.$$

Therefore by the independence of $\hat{x}_i$ and $\hat{x}_j$ for any $i \neq j$, it is easy to see that

$$\begin{array}{rcl} \mathrm{E}[w(S)] & = & \sum w_{ij}(\mathrm{E}[\hat{x}_i] + \mathrm{E}[\hat{x}_j] - \mathrm{E}[\hat{x}_i]\mathrm{E}[\hat{x}_j]) \\ & = & \sum w_{ij}\left(\frac{k}{n} + \frac{k}{n} - (\frac{k}{n})^2\right) \\ & \ge & \left(2(\frac{k}{n}) - (\frac{k}{n})^2\right) \cdot w^*. \end{array} \tag{6}$$

But this is not sufficient to prove that the randomized algorithm is a $(2(\frac{k}{n}) - (\frac{k}{n})^2)$-approximation algorithm, since $|S|$ may contain more than k vertices. Therefore, we have to give an estimate of the expected value of $w(\tilde{S})$—the final output of the algorithm, where $|\tilde{S}| = k$.

To bound $\mathrm{E}[w(\tilde{S})]$, we need to bound $|S| = \sum_{i=1}^n \hat{x}_i$. The following version of the Chernoff inequality will be useful since $\hat{x}_i, 1 \le i \le n$, are independent Bernoulli random variables.

Lemma 3 *Let $X_i, 1 \le i \le n$, be independent Bernoulli random variables with* $\Pr\{X_i = 1\} = p_i$. *Then*

$$\Pr\{\sum_{i=1}^n X_i > (1+r)\sum_{i=1}^n p_i\} < e^{-\frac{r^2}{3}\sum_{i=1}^n p_i}$$

where $0 < r < \frac{1}{3}$.

The lemma tells that $|S|$ is close to k with high probability. This, combining with the next lemma, implies that the weight difference between $\tilde{S}$ and S is very small.

Lemma 4

$$w(\tilde{S}) \ge \frac{k}{max\{|S|, k\}} \cdot w(S).$$

Proof. We only need to consider the case $|S| > k$, since otherwise $w(\tilde{S}) \ge w(S)$. Note that an edge is removed in the swapping procedure if and only if both of its endpoints are not in $\tilde{S}$. It follows that

$$\sum_{i \in \tilde{S}} w(\tilde{S} \setminus \{i\}) = |\tilde{S}|w_1(\tilde{S}) + (|\tilde{S}| - 1)w_2(\tilde{S}),$$

since each edge in $\tilde{S}$ is counted $|\tilde{S}|$ times, and the crossing edges is counted $|\tilde{S}|-1$ times. Suppose v is the vertex that is removed in the swapping procedure, then

$$\sum_{j \notin \tilde{S}} w_{vj} \leq \sum_{j \notin \tilde{S}} w_{ij}$$

for any other $i \in \tilde{S}$. Thus,

$$w(\tilde{S} \setminus \{v\}) \geq \frac{1}{|\tilde{S}|} \sum_{i \in \tilde{S}} w(\tilde{S} \setminus \{i\}) = \frac{|\tilde{S}|-1}{|\tilde{S}|} \cdot w(\tilde{S}).$$

By induction we get that the final $\tilde{S}$ satisfies

$$w(\tilde{S}) \geq \frac{k}{|S|} \cdot w(S). \qquad \blacksquare$$

Now we derive the final result for the randomized algorithm.

Theorem 3
$$\frac{w(\tilde{S})}{w^*} \geq \frac{1}{1+k^{-\frac{1}{3}}} \cdot \left(2(\frac{k}{n}) - (\frac{k}{n})^2 - \frac{1}{n}\right)$$
for $k \geq \frac{n}{2}$ and n sufficiently large.

Proof. For simplicity, let $\max\{w_{ij} : (i,j) \in E\} = 1$. Thus, $w^* \geq 1$.

Since $k \geq \frac{n}{2}$, by Lemma 3,

$$\Pr\{|S| > (1+k^{-\frac{1}{3}}) \cdot k\} = \Pr\{\sum_{i=1}^{n} \hat{x}_i > (1+k^{-\frac{1}{3}}) \cdot k\} < e^{-\frac{(\frac{n}{2})^{\frac{1}{3}}}{3}} < \frac{1}{n^3},$$

for n large enough. We now rewrite (6) as

$$\begin{aligned} \mathrm{E}[w(S)] &= \mathrm{E}[w(S)|\ |S| \leq (1+k^{-\frac{1}{3}}) \cdot k] + \mathrm{E}[w(S)|\ |S| > (1+k^{-\frac{1}{3}}) \cdot k] \\ &\geq \left(2(\frac{k}{n}) - (\frac{k}{n})^2\right) \cdot w^* \end{aligned}$$

However, since $w(S) \leq kn \leq n^2$,

$$\begin{aligned} \mathrm{E}[w(S)|\ |S| > (1+k^{-\frac{1}{3}}) \cdot k] &\leq \Pr\{|S| > (1+k^{-\frac{1}{3}}) \cdot k\} \cdot n^2 \\ &< \frac{1}{n} \end{aligned}$$

Thus

$$\begin{aligned} \mathrm{E}[w(S) \mid |S| \le (1+k^{-\frac{1}{3}}) \cdot k] &> \Big(2(\tfrac{k}{n}) - (\tfrac{k}{n})^2\Big)\, w^* - \tfrac{1}{n} \\ &\ge \Big(2(\tfrac{k}{n}) - (\tfrac{k}{n})^2 - \tfrac{1}{n}\Big) \cdot w^*. \end{aligned} \tag{7}$$

Note that we output the best $\tilde{S}$ among all solutions S. We may underestimate it by selecting $\tilde{S}$ only from S such that

$$|S| \le (1+k^{-\frac{1}{3}}) \cdot k.$$

(We are able to generate such S since the probability $\Pr\{|S| \le (1+k^{-\frac{1}{3}}) \cdot k\}$ is almost one.) Thus, by (7) and Lemma 4, we have

$$\begin{aligned} \frac{\mathrm{E}[w(\tilde{S})]}{w^*} &\ge \frac{k}{\max\{|S|, k\}} \cdot \frac{\mathrm{E}[w(S) \mid |S| \le (1+k^{-\frac{1}{3}}) \cdot k]}{w^*} \\ &\ge \frac{1}{1+k^{-\frac{1}{3}}} \cdot \frac{\mathrm{E}[w(S) \mid |S| \le (1+k^{-\frac{1}{3}}) \cdot k]}{w^*} \\ &\ge \frac{1}{1+k^{-\frac{1}{3}}} \cdot \left(2(\frac{k}{n}) - (\frac{k}{n})^2 - \frac{1}{n}\right). \blacksquare \end{aligned}$$

As $k \ge n/2$ and n increases, $\frac{1}{1+k^{-\frac{1}{3}}} \cdot \Big(2(\frac{k}{n}) - (\frac{k}{n})^2 - \frac{1}{n}\Big)$ approaches $2(\frac{k}{n}) - (\frac{k}{n})^2$, which gives a worst-case ratio better than those from the LP-based and greedy algorithms.

4 SDP relaxation of MVC

The MVC problem can be represented as the following binary integer program:

$$\begin{aligned} w^* := \text{Maximize} \quad & \frac{1}{4}\sum_{i<j} w_{ij}(3 + x_i + x_j - x_i x_j) \\ \text{subject to} \quad & \sum_{j=1}^{n} x_j = 2k - n, \\ & x_j^2 = 1, \ j = 1, \ldots, n, \end{aligned}$$

which can be written as

$$
\text{(BMVC)} \qquad \begin{aligned} w^* := \text{Maximize} \quad & \frac{1}{4}\sum_{i<j} w_{ij}(3 + x_0 x_i + x_0 x_j - x_i x_j) \\ \text{subject to} \quad & \sum_{j=1}^{n} x_0 x_j = 2k - n, \\ & x_j^2 = 1,\ \ j = 0, 1, \cdots, n, \end{aligned} \tag{8}
$$

by introducing a "reference" binary variable x_0, where $i \in S$ if and only if $x_i = x_0$ for $i = 1, 2, \cdots, n$.

The SDP relaxation of MVC becomes:

$$
\text{(SDP-MVC)} \qquad \begin{aligned} w^{SDP} := \quad \text{Maximize} \quad & \frac{1}{4}\sum_{i<j} w_{ij}(3 + X_{0i} + X_{0j} - X_{ij}) \\ \text{subject to} \quad & \sum_{i \geq 1, j \geq 1} X_{ij} = (2k - n)^2, \\ & \sum_{i \geq 1} X_{0i} = 2k - n, \\ & X_{jj} = 1, \ \text{for } j = 0, 1, \cdots, n, \\ & X \succeq 0, \end{aligned} \tag{9}
$$

where $X \in R^{(n+1)\times(n+1)}$ is a unknown symmetric matrix, $\bullet$ is the matrix inner product, that is, $Q \bullet X = \text{trace}(Q^T X)$, and $X \succeq Z$ means that $X - Z$ is positive semidefinite.

Obviously, $w^{SDP} \geq w^*$ holds, since for any feasible solution $x \in R^{n+1}$ of (BMVC), $X = xx^T$ is feasible for (SDP-MVC).

Let $\bar{X}$ be an optimal solution of (SDP-MVC), and $\sigma = \frac{k}{n}$ and $\kappa = 2\sigma - 1$ (note that $-1 \leq \kappa \leq 1$) be given. We present an SDP-based approximation algorithm for MVC as follows.

SDP-Algorithm

1. **SDP Solving:** Solve (SDP-MVC) to obtain an optimal semidefinite matrix $\bar{X}$.

   Repeat the following two steps many times, and output the best block.

2. **Randomized Rounding:** Generate a vector u from a multivariate normal distribution with 0 mean and the covariance matrix $\theta\bar{X} + (1 - \theta)P$ for a $0 \leq$

$\theta \leq 1$, where P is the positive semidefinite matrix

$$P = \begin{pmatrix} 1 & \kappa & \kappa & \cdots & \kappa \\ \kappa & 1 & \kappa^2 & \cdots & \kappa^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \kappa & \kappa^2 & \cdots & \kappa^2 & 1 \end{pmatrix},$$

that is, $u \in N(0, \theta\bar{X} + (1-\theta)P)$. Then assign $\hat{x} = \text{sign}(u)$, i.e.,

$$\hat{x}_i = \begin{cases} 1 & \text{if } u_i \geq 0, \\ -1 & \text{if } u_i < 0. \end{cases}.$$

Select the block $S = \{i : \hat{x}_i = \hat{x}_0\}$. Let $\tilde{S} = S$.

3. **Vertex Swapping:** Do one of the following two cases:

   If $|\tilde{S}| > k$, for each $i \in \tilde{S}$, let $\zeta(i) = \sum_{j \notin \tilde{S}} w_{ij}$ and $\tilde{S} := \{i_1, i_2, ..., i_{|\tilde{S}|}\}$, where $\zeta(i_1) \geq \zeta(i_2) \geq ... \geq \zeta(i_{|\tilde{S}|})$. Then, remove vertex $i_{|\tilde{S}|}$ from $\tilde{S}$ and reassign $\tilde{S} := \{i_1, i_2, ..., i_{|\tilde{S}|-1}\}$. Repeat this swapping procedure till $|\tilde{S}| = k$.

   If $|\tilde{S}| \leq k$, arbitrarily add $k - |S|$ vertices from outside of S into $\tilde{S}$. ∎

Note that the swapping procedure is identical to that in the randomized algorithm, so that Lemma 4 still holds in the SDP-algorithm.

The performance of the SDP-Algorithm is determined by two factors: the expected quality of S and how much the size of S exceeding k. We want high $w(S)$; but, at the same time, small $|S|$ since a larger size of S deteriorates $w(\tilde{S})$ at the end. The convex combination parameter θ used in $\theta\bar{X} + (1-\theta)P$ provides a balance for the two factors. Typically, the more use of $\bar{X}$ in the combination results in higher expected quality $w(S)$ but larger expected size $|S|$; and the more use of P results in less $w(S)$ and smaller $|S|$.

5 Analysis of the SDP-algorithm

Recall

$$w(S) = \frac{1}{4} \sum_{1 \leq i < j \leq n} w_{ij} \left(3 + \hat{x}_0\hat{x}_i + \hat{x}_0\hat{x}_j - \hat{x}_i\hat{x}_j\right)$$

and note

$$|S|(n-|S|) = \frac{n^2}{4} - \frac{(e^T\hat{x})^2}{4} = \frac{1}{4}\sum_{i,j}(1-\hat{x}_i\hat{x}_j).$$

Let

$$\begin{array}{rl} \alpha = \alpha(\theta) := \min & \frac{3+\frac{2}{\pi}(\arcsin(\theta x+(1-\theta)\kappa)+\arcsin(\theta y+(1-\theta)\kappa)-\arcsin(\theta z+(1-\theta)\kappa^2))}{3+x+y-z} \\ \text{subject to} & \begin{pmatrix} 1 & x & y \\ x & 1 & z \\ y & z & 1 \end{pmatrix} \succeq 0 \end{array} \tag{10}$$

for a given $0 \leq \theta \leq 1$. Then, we have the following lemma:

Lemma 5 *The expectation of $w(S)$ of the SDP-algorithm satisfies the following inequality*

$$\mathrm{E}[w(S)] \geq \alpha \cdot w^{SDP} \geq \alpha \cdot w^*.$$

Proof. From Goemans and Williamson [9], Frieze and Jerrum [7], and Bertsimas and Ye [3], we have

$$\begin{aligned} \mathrm{E}[\hat{x}_0\hat{x}_j] &= \frac{2}{\pi}\arcsin(\theta\bar{X}_{0j} + (1-\theta)\kappa), \quad j = 1,\ldots,n, \\ \mathrm{E}[\hat{x}_i\hat{x}_j] &= \frac{2}{\pi}\arcsin(\theta\bar{X}_{ij} + (1-\theta)\kappa^2), \quad i,j = 1,\ldots,n. \end{aligned}$$

Thus

$$\begin{aligned} \mathrm{E}[w(S)] &= \frac{1}{4}\sum_{1\leq i<j\leq n} w_{ij}\left(3 + \mathrm{E}[\hat{x}_0\hat{x}_i] + \mathrm{E}[\hat{x}_0\hat{x}_j] - \mathrm{E}[\hat{x}_i\hat{x}_j]\right) \\ &= \frac{1}{4}\sum_{1\leq i<j\leq n} w_{ij}\Big(3 + \frac{2}{\pi}(\arcsin(\theta\bar{X}_{0j} + (1-\theta)\kappa) \\ &\quad + \arcsin(\theta\bar{X}_{0i} + (1-\theta)\kappa) - \arcsin(\theta\bar{X}_{ij} + (1-\theta)\kappa^2))\Big) \\ &\geq \frac{\alpha}{4}\sum_{1\leq i<j\leq n} w_{ij}\left(3 + \bar{X}_{0j} + \bar{X}_{0i} - \bar{X}_{ij}\right) \\ &= \alpha \cdot w^{SDP} \geq \alpha \cdot w^*. \quad \blacksquare \end{aligned}$$

For a given $0 \leq \theta \leq 1$, let

$$\tau^+ = \tau^+(\theta) := \quad \min_{-1\leq y\leq 1} \frac{2}{\pi}\frac{\arcsin(\theta - (1-\theta)\kappa) + \arcsin(\theta y + (1-\theta)\kappa)}{1+y}$$

$$\beta^+ = \beta^+(\theta) := \quad 1 - \tfrac{2}{\pi}\arcsin(\theta - (1-\theta)\kappa) + \tau^+(\theta)$$

$$\tau^- = \tau^-(\theta) := \quad \min_{-1\le y\le 1} \frac{2}{\pi}\frac{\arcsin(\theta + (1-\theta)\kappa) - \arcsin(\theta y + (1-\theta)\kappa)}{1-y}$$
$$\beta^- = \beta^-(\theta) := \quad 1 - \tfrac{2}{\pi}\arcsin(\theta + (1-\theta)\kappa) + \tau^-(\theta)$$

$$\tau = \tau(\theta) := \quad \min_{-1\le y\le 1} \frac{2}{\pi}\frac{\arcsin(\theta + (1-\theta)\kappa^2) - \arcsin(\theta y + (1-\theta)\kappa^2)}{1-y}$$
$$\beta = \beta(\theta) := \quad 1 - \tfrac{2}{\pi}\arcsin(\theta + (1-\theta)\kappa^2) + \tau(\theta).$$

Then, we have the following lemma which is proved in [10].

Lemma 6 *The SDP-Algorithm yields S satisfying the following inequalities:*

$$\mathrm{E}(|S|) \ge (\beta^+ + \kappa\tau^+)\frac{n}{2},$$
$$\mathrm{E}(|S|) \le (2 - \beta^- + \kappa\tau^-)\frac{n}{2},$$
$$\mathrm{E}(|S|(n-|S|) \ge (\beta - \frac{\beta-\tau}{n} - \kappa^2\tau)\frac{n^2}{4}. \qquad \blacksquare$$

Define random variable

$$z(\gamma,\eta) := \frac{w}{w^*} + \eta\gamma\frac{|S|}{n} + \gamma\frac{|S|(n-|S|)}{n^2}, \quad \eta \in [-1, 2\sigma - 1], \tag{11}$$

and let

$$\gamma = \begin{cases} \dfrac{\alpha}{(2\sigma - (1+\eta)/2)^2 - \mu}, & \text{if} \quad 4\sigma - 3 \le \eta \le 2\sigma - 1 \\[2ex] \dfrac{\alpha(1-\sigma)}{\sigma(\mu - \eta) + (1+\eta)^2/4 - \mu}, & \text{if} \quad -1 \le \eta < 4\sigma - 3. \end{cases} \tag{12}$$

where μ ($\le \sigma^2$) is a lower bound of $\mathrm{E}[\eta\frac{|S|}{n} + \frac{|S|(n-|S|)}{n^2}]$, then we have

$$\mathrm{E}[z(\gamma,\eta)] \ge \alpha + \gamma\mu \quad \text{and} \quad z \le 1 + \frac{\gamma(1+\eta)^2}{4}. \tag{13}$$

Lemma 7 *If random variable $z(\gamma,\eta)$ fulfills its expectation, i.e., $z(\gamma,\eta) \ge \alpha + \gamma\mu$, then*

$$w(\tilde{S}) \ge R(\theta,\eta)\cdot w^*,$$

where

$$R(\theta,\eta)=\begin{cases}\dfrac{\alpha\sigma(4\sigma-2(1+\eta))}{(2\sigma-(1+\eta)/2)^2-\mu}, & \textit{if} \quad 4\sigma-3\le\eta\le 2\sigma-1\\[2ex] \dfrac{\alpha\sigma((1+\eta)^2/4-\eta)}{\sigma(\mu-\eta)+(1+\eta)^2/4-\mu}, & \textit{if} \quad -1\le\eta<4\sigma-3.\end{cases}$$

Proof. Suppose that

$$w(S)=\lambda w^* \quad \text{and} \quad |S|=\delta k=\delta\sigma n.$$

Note that $\delta\le\frac{1}{\sigma}$.

Consider the case $\delta>1$, it follows from Lemma 4 that

$$\begin{aligned} w(\tilde{S}) &\ge \frac{1}{\delta}w(S)\\ &= \frac{\lambda}{\delta}\cdot w^*\\ &\ge \frac{\alpha+\gamma\mu-\eta\gamma\delta\sigma-\gamma\delta\sigma(1-\delta\sigma)}{\delta}\cdot w^* \end{aligned}$$

If $4\sigma-3\le\eta\le 2\sigma-1$, then

$$\frac{\alpha+\gamma\mu-\eta\gamma\delta\sigma-\gamma\delta\sigma(1-\delta\sigma)}{\delta}\ge\left(2\sqrt{(\alpha+\gamma\mu)\gamma}-(1+\eta)\gamma\right)\sigma,$$

since from simple calculus and (12)

$$\delta=\frac{1}{\sigma}\sqrt{\frac{\alpha+\gamma\mu}{\gamma}}=2-\frac{1+\eta}{2\sigma}\le\frac{1}{\sigma}$$

yields the minimal value for

$$\frac{\alpha+\gamma\mu-\eta\gamma\delta\sigma-\gamma\delta\sigma(1-\delta\sigma)}{\delta}.$$

If $-1\le\eta<4\sigma-3$, then

$$\frac{\alpha+\gamma\mu-\eta\gamma\delta\sigma-\gamma\delta\sigma(1-\delta\sigma)}{\delta}\ge\sigma(\alpha+\mu\gamma-\eta\gamma),$$

since from simple calculus, (12) and the fact $\sigma \geq 1/2$,

$$\delta = \frac{1}{\sigma}$$

yields the minimal value for

$$\frac{\alpha + \gamma\mu - \eta\gamma\delta\sigma - \gamma\delta\sigma(1-\delta\sigma)}{\delta}.$$

For the case $\delta \leq 1$, the number of vertices in S is no more than k and no vertex has to be removed from S. Then,

$$\begin{aligned} w(\tilde{S}) &\geq w(S) \\ &= \lambda w^* \\ &\geq (\alpha + \gamma\mu - \eta\gamma\delta\sigma - \gamma\sigma\delta(1-\delta\sigma)) \cdot w^* \\ &\geq (\alpha + \gamma\mu - \gamma(1+\eta)^2/4) \cdot w^*, \end{aligned}$$

where the third inequality follows from the simple fact that

$$\delta = \frac{1+\eta}{2\sigma} \leq 1$$

yields the maximal value, $(1+\eta)^2/4$, of

$$\eta\delta\sigma + \sigma\delta(1-\delta\sigma).$$

Using (12) we have, for $4\sigma - 3 \leq \eta \leq 2\sigma - 1$,

$$\left(2\sqrt{(\alpha+\gamma\mu)\gamma} - (1+\eta)\gamma\right)\sigma = \alpha + \gamma\mu - \gamma(1+\eta)^2/4 = \frac{\alpha(4\sigma^2 - 2(1+\eta)\sigma)}{(2\sigma - (1+\eta)/2)^2 - \mu}$$

and, for $-1 \leq \eta < 4\sigma - 3$,

$$\sigma(\alpha + \mu\gamma - \eta\gamma) = \alpha + \gamma\mu - \gamma(1+\eta)^2/4 = \sigma\alpha\left(1 + \frac{(1-\sigma)(\mu-\eta)}{\sigma(\mu-\eta) + (1+\eta)^2/4 - \mu}\right).$$

This completes the proof. ∎

For a given vertex number k, σ and κ are fixed. If we choose $\eta \geq 0$, then from Lemma 6,

$$\mathrm{E}[\eta\frac{|S|}{n} + \frac{|S|(n-|S|)}{n^2}] \geq \eta(\beta^+ + \kappa\tau^+)\frac{1}{2} + (\beta - \frac{\beta-\tau}{n} - \kappa^2\tau)\frac{1}{4},$$

which establishes a lower bound

$$\mu = \mu(\theta) := \min\{\sigma^2, \eta(\beta^+ + \kappa\tau^+)\frac{1}{2} + (\beta - \frac{\beta - \tau}{n} - \kappa^2\tau)\frac{1}{4}\}.$$

Using this μ, we solve the maximization problem

$$R^+(\sigma) = \max_{0 \le \theta \le 1,\ 0 \le \eta \le 1} R(\theta, \eta). \tag{14}$$

Note that α, β^+, τ^+, β and τ in the formula of μ and $R(\theta, \eta)$ of Lemma 7 are all functions of θ.

If we choose $\eta \le 0$, then from Lemma 6,

$$\mathrm{E}[\eta\frac{|S|}{n} + \frac{|S|(n - |S|)}{n^2}] \ge \eta(2 - \beta^- + \kappa\tau^-)\frac{1}{2} + (\beta - \frac{\beta - \tau}{n} - \kappa^2\tau)\frac{1}{4},$$

which establishes a lower bound

$$\mu = \mu(\theta) := \min\{\sigma^2, \eta(2 - \beta^- + \kappa\tau^-)\frac{1}{2} + (\beta - \frac{\beta - \tau}{n} - \kappa^2\tau)\frac{1}{4}\}.$$

Using this μ, we solve the maximization problem

$$R^-(\sigma) = \max_{0 \le \theta \le 1,\ -1 \le \eta \le 0} R(\theta, \eta). \tag{15}$$

Note that β^- and τ^- in the formula of μ are also functions of θ.

The motivation to solve these two maximization problems is to yield the largest value of $R(\theta, \eta)$—the performance guarantee. Finally, let $R^*(\sigma) = \max\{R^+(\sigma), R^-(\sigma)\}$. Then we have the following conclusion.

Theorem 4 *The worst-case performance ratio of the SDP-Algorithm for the MVC problem is at least $R^*(\sigma)$ for sufficiently large n and k, where $\sigma = \frac{k}{n}$.*

We have calculated the first table, Table 1, which shows that $R(\theta, \eta)$ for a wide range of k.

Recall that the performance ratio of the SDP-Algorithm is determined by two factors: the expected quality of $w(S)$, (Lemma 5), which is represented by the ratio α (see Lemma 5), and expected how many vertices being removed form S to generate $\tilde{S}$, which is reflected by the estimates of Lemma 6. The convex combination parameter θ used in the rounding procedure balances the two factors—the greater of θ, the higher expected quality of $w(S)$ but larger size of S, which would make $w(\tilde{S})$ deteriorate

$\sigma = \frac{k}{n}$	θ	α	η	γ	μ	$R(\theta, \sigma)$
0.40	0.88	0.8518	-0.4457	3.5916	0.0362	0.7058
0.42	0.86	0.8478	-0.3801	3.9058	0.0639	0.7221
0.44	0.86	0.8498	-0.3270	4.0463	0.0854	0.7372
0.46	0.84	0.8465	-0.2650	4.4135	0.1135	0.7511
0.48	0.82	0.8437	-0.2056	4.8025	0.1353	0.7637
0.50	0.82	0.8460	-0.1582	4.9000	0.1627	0.7750
0.52	0.82	0.8482	-0.1130	4.9314	0.1838	0.7848
0.54	0.82	0.8503	-0.0703	4.8866	0.2044	0.7933
0.56	0.84	0.8563	-0.0372	4.5475	0.2195	0.8005
0.58	0.84	0.8580	0.0	4.3462	0.2382	0.8067
0.60	0.86	0.8633	0.0	3.3791	0.2345	0.8110
0.62	0.88	0.8681	0.0	2.7335	0.2300	0.8135
0.64	0.90	0.8724	0.0	2.2734	0.2246	0.8148
0.66	0.90	0.8733	0.0138	3.4010	0.2282	0.8154
0.68	0.90	0.8742	0.0359	3.2569	0.2366	0.8157
0.70	0.90	0.8750	0.0571	3.1098	0.2445	0.8158
0.72	0.90	0.8758	0.0774	2.9643	0.2518	0.8157
0.74	0.90	0.8765	0.0970	2.8239	0.2584	0.8154
0.76	0.90	0.8771	0.1157	2.6913	0.2643	0.8149
0.78	0.90	0.8777	0.1336	2.5684	0.2694	0.8143
0.80	0.90	0.8783	0.1294	1.0737	0.2588	0.8138
0.82	0.92	0.8808	0.1006	0.8838	0.2274	0.8141
0.84	0.92	0.8811	0.0761	0.7280	0.1996	0.8157
0.86	0.94	0.8833	0.0252	0.5610	0.1477	0.8187
0.88	0.96	0.8853	0.0	0.4491	0.1120	0.8233
0.90	0.96	0.8854	0.0	0.3686	0.0979	0.8293
0.92	0.98	0.8872	0.0	0.2912	0.0779	0.8371
0.94	0.98	0.8872	0.0	0.2161	0.0618	0.8465
0.96	1.0	0.8889	0.0	0.1430	0.0337	0.8580
0.98	1.0	0.8889	0.0	0.0712	0.0172	0.8723

Table 1: Worst-case performance ratios of the SDP-algorithm for $k \geq 0.40n$

more. We see from the table that as k increases, θ approaches one which implies that the SDP solution $\bar{X}$ is weighted more and more in the rounding procedure. This is because that the more use of $\bar{X}$ in the covariance matrix combination results in better $w(S)$ than the use of P; and, for large k, the size of S becomes a less important factor since it cannot exceed n anyway. We also see that the performance ratio of the SDP-Algorithm is better than all other algorithms described earlier for $.46n \le k \le .54n$.

6 Strengthened SDP relaxation of MVC

The SDP relaxations considered in the previous two sections can be strengthened by adding valid inequalities. Consider the -1 or 1 variable x_i for $i = 0, 1, 2, \cdots, n$ in the formulation (MVC). For any given x_0, x_i, and x_j the following four inequalities must hold:

$$x_0x_i + x_0x_j + x_ix_j \ge -1 \tag{16}$$

$$-x_0x_i - x_0x_j + x_ix_j \ge -1 \tag{17}$$

$$-x_0x_i + x_0x_j - x_ix_j \ge -1 \tag{18}$$

$$x_0x_i - x_0x_j - x_ix_j \ge -1. \tag{19}$$

To get a stronger SDP relaxation, we can relax (16)-(19) to the following, respectively,

$$X_{0i} + X_{0j} + X_{ij} \ge -1 \tag{20}$$

$$-X_{0i} - X_{0j} + X_{ij} \ge -1 \tag{21}$$

$$-X_{0i} + X_{0j} - X_{ij} \ge -1 \tag{22}$$

$$X_{0i} - X_{0j} - X_{ij} \ge -1. \tag{23}$$

Note that the basic inequalities used in the proof of the performance guarantee of the SDP-algorithm are Lemma (5) and Lemma (6), in which we established bound for $\mathrm{E}[w(S)]$, $\mathrm{E}[|S|]$ and $\mathrm{E}[|S|(n-|S|)]$. We observed that by adding the valid inequalities (20)-(23), the bound of $\mathrm{E}[|S|]$ and $\mathrm{E}[|S|(n-|S|)]$ will remain the same. However as we will see next, the bound for $\mathrm{E}[w(S)]$ will be improved.

Recall that, in the last section, we proved that

$$\mathrm{E}[w(S)] \ge \alpha \cdot w^{SDP} \ge \alpha \cdot w^*,$$

where α is defined by (10).

We observed that the optimal solution (x, y, z) of (10) must satisfy

$$x = y > 0 \quad \text{and} \quad z < 0$$

and

$$-x - y + z < -1.$$

Thus, it violates the inequality (21). This means the minimal value of (10) will improve if we add the inequality (21) as a constraint into (10). (However, we find that the inequality (20), (22) and (23) seem useless for this problem.)

Therefore, we derive a strengthened SDP relaxation, (SSDP-MVC), for MVC:

$$\text{(SSDP-MVC)} \quad \begin{array}{lll} w^{SDP} := & \text{Maximize} & \frac{1}{4}\sum_{i<j} w_{ij}(3 + X_{0i} + X_{0j} - X_{ij}) \\ & \text{subject to} & \sum_{i\geq 1, j\geq 1} X_{ij} = (2k-n)^2, \\ & & \sum_{i\geq 1} X_{0i} = 2k - n, \\ & & -X_{0i} - X_{0j} + X_{ij} \geq -1 \text{ for } 1 \leq i < j \leq n, \\ & & X_{jj} = 1, \text{ for } j = 0, 1, \cdots, n, \\ & & X \succeq 0, \end{array}$$

The SSDP-based algorithm is identical to the SDP-Algorithm, except that we solve (SSDP-MVC) rather than (SDP-MVC). The analysis of the SSDP-algorithm is the same as before, except α is now defined by:

$$\begin{array}{ll} \alpha = \alpha(\theta) := \min & \frac{3 + \frac{2}{\pi}(\arcsin(\theta x + (1-\theta)\kappa) + \arcsin(\theta y + (1-\theta)\kappa) - \arcsin(\theta z + (1-\theta)\kappa^2))}{3 + x + y - z} \\ \text{subject to} & -x - y + z \geq -1, \\ & \begin{pmatrix} 1 & x & y \\ x & 1 & z \\ y & z & 1 \end{pmatrix} \succeq 0 \end{array}$$

We have calculated the worst-case ratio of the SSDP-based approximation algorithm in the second table—Table 2. Comparing the two tables, we see that the performance ratios are significantly improved for all selections of k, which is directly resulted from the improved α. One noticeable change is that as k increases, θ is decreasing. This is because, for large k, both covariance matrices $\bar{X}$ and P used in the rounding procedure produce high quality $w(S)$, and the SDP solution $\bar{X}$ is no longer that useful. Note that the performance ratio of the SSDP-based algorithm is better than all other algorithms for $.42n \leq k \leq .60n$.

$\sigma = \frac{k}{n}$	θ	α	η	γ	μ	$R(\theta, \sigma)$
0.40	0.82	0.8691	-0.4157	4.2531	0.0535	0.7338
0.42	0.80	0.8675	-0.3520	4.6605	0.0801	0.7518
0.44	0.78	0.8664	-0.2901	5.1341	0.1069	0.7687
0.46	0.76	0.8657	-0.2304	5.6700	0.1337	0.7844
0.48	0.74	0.8655	-0.1733	6.2403	0.1601	0.7987
0.50	0.72	0.8657	-0.1198	6.7714	0.1856	0.8113
0.52	0.70	0.8663	-0.0708	7.1330	0.2096	0.8220
0.54	0.70	0.8700	-0.0322	6.8569	0.2284	0.8307
0.56	0.72	0.8761	-0.0016	6.1515	0.2430	0.8377
0.58	0.76	0.8840	0.0	4.5398	0.2409	0.8425
0.60	0.78	0.8887	0.0	3.5221	0.2377	0.8453
0.62	0.78	0.8910	0.0226	4.4361	0.2475	0.8471
0.64	0.76	0.8914	0.0500	4.1206	0.2607	0.8485
0.66	0.76	0.8921	0.0748	3.7722	0.2689	0.8495
0.68	0.74	0.8944	0.0931	3.4349	0.2799	0.8502
0.70	0.72	0.8953	0.1143	3.1240	0.2922	0.8507
0.72	0.70	0.8965	0.1336	2.8414	0.2977	0.8511
0.74	0.68	0.8977	0.1513	2.5910	0.3048	0.8512
0.76	0.64	0.8983	0.1694	2.3500	0.3124	0.8512
0.78	0.60	0.8992	0.1854	2.1365	0.3183	0.8512
0.80	0.02	0.8964	0.2794	1.6791	0.3877	0.8612
0.82	0.00	0.9039	0.3048	1.5784	0.4026	0.8676
0.84	0.00	0.9115	0.3245	1.4596	0.4130	0.8742
0.86	0.00	0.9191	0.3375	1.3074	0.4183	0.8814
0.88	0.00	0.9268	0.3499	1.1479	0.4231	0.8895
0.90	0.00	0.9346	0.3618	0.9807	0.4271	0.8989
0.92	0.00	0.9428	0.3732	0.8053	0.4304	0.9098
0.94	0.00	0.9515	0.3841	0.6210	0.4326	0.9227
0.96	0.00	0.9612	0.3946	0.4268	0.4334	0.9386
0.98	0.00	0.9731	0.4046	0.2209	0.4315	0.9595

Table 2: Worst-case performance ratios of the SSDP-algorithm for $k \geq 0.40n$

References

[1] A. A. Ageev and M. I. Sviridenko, "Approximation algorithms for maximum coverage and max cut with given sizes of parts", IPCO, G. Cornuejols, R. E. Burkard, G. J. Woeginger (Eds.), 17-30, 1999.

[2] N. Alon and N. Kahale, "Approximating the independence number via the θ-function", *Math. Programming*, 80(1998)253-264.

[3] D. Bertsimas and Y. Ye, "Semidefinite relaxations, multivariate normal distributions, and order statistics," *Handbook of Combinatorial Optimization (Vol. 3)*, D.-Z. Du and P.M. Pardalos (Eds.) pp. 1-19, (1998 Kluwer Academic Publishers).

[4] U. Feige and M. X. Geomans, "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) pp. 182-189.

[5] U. Feige and M. Langberg, "Approximation Algorithms for Maximization Problems arising in Graph Partitioning" manuscript and Ph. D. thesis, 1999.

[6] U. Feige and M. Seltser, "On the densest k-subgraph problem," Technical report, Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot, September 1997.

[7] A. Frieze and M. Jerrum, "Improved approximation algorithms for max k-cut and max bisection," *Proc. 4th IPCO Conference* (1995) 1-13.

[8] M. X. Geomans and D. P. Williamson, " New 3/4-approximation algorithms for MAX SAT," *SIAM J. of Discrete Mathematics* 7 (1994), 656-666.

[9] M. X. Geomans and D. P. Williamson, " Improved approximation algorithms for Maximum Cut and Satisfiability problems using semidefinite programming," *Journal of ACM* 42 (1995) 1115–1145.

[10] Q. Han, Y. Ye and J. Zhang, "An improved rounding method and semidefinite programming relaxation for graph partition," *Manuscript*, February 2000.

[11] D. S. Hochbaum, "Approximation algorithms for the set covering and vertex cover problems," *SIAM J. Comput.* 11 (1982), 555-556.

[12] D. S. Hochbaum, "Approximating covering and packing problem: set cover, vertex cover, independent set, and related problem," *Approximation Algorithm for NP-hard Problems*, Hochbaum (eds), (1996),94-143, PWS Publishing Company.

[13] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (eds.), Plenum Press, New York (1972) 85-103.

[14] D. Karger, R. Motwani and M. Sudan, "Approximate graph coloring by semidefinite programming," in *35th FOCS* (1994) 2-13.

[15] T. Leighton and S. Rao, "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms," to appear in *Journal of ACM.*

[16] S. Mahajan and H. Ramesh, "Derandomizing semidefinite programming based approximation algorithms," in *36th FOCS*, (1995) pp. 162-169.

[17] Y. E. Nesterov, "Semidefinite relaxation and nonconvex quadratic optimization," *Optimization Methods and Software* 9 (1998) 141-160.

[18] E. Petrank, "The hardness of approximation: gap location," *Computational Complexity*, 4 (1994), 133-157.

[19] D. B. Shmoys, "Using linear programming in the design and analysis of approximation algorithms: two illustrative problems," *Approximation Algorithms for Combinatorial Optimization*, K. Jansen and J. Rolim (Eds.) (1998) pp. 15-32.

[20] A. Srivastav and K. Wolf, "Finding dense subgraphs with semidefinite programming," in *Approximation Algorithms for Combinatorial Optimization*, K. Jansen and J. Rolim (Eds.) (1998) pp. 181-191.

[21] Y. Ye, "A .699-approximation algorithm for Max-Bisection," *Manuscript*, March 1999.

[22] Y. Ye, "Approximating quadratic programming with bound and quadratic constraints," *Mathematical Programming* 84 (1999) 219-226.

[23] Y. Ye and J. Zhang, "Approximation of Dense-$\frac{n}{2}$-Subgraph and the complement of Min-Bisection," *Manuscript*, November 1999.

[24] U. Zwick, "Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to Max-Cut and other problems," in *31th STOC* (1999) 679-687.