Approximation Algorithms for Maximization Problems arising in Graph Partitioning

Uriel Feige Michael Langberg

February 15, 2001

Abstract

Given a graph $G = (V, E)$, a weight function $w : E \rightarrow R^+$ and a parameter k, we consider the problem of finding a subset $U \subseteq V$ of size k that maximizes :

Max-Vertex Cover$_k$	:	the weight of edges incident with vertices in U.
Max-Dense Subgraph$_k$	:	the weight of edges in the subgraph induced by U.
Max-Cut$_k$	:	the weight of edges cut by the partition $(U, V \setminus U)$.
Max-Uncut$_k$	:	the weight of edges not cut by the partition $(U, V \setminus U)$.

For each of the above problems we present approximation algorithms based on semidefinite programming, and obtain approximation ratios better than those previously published. In particular we show that if a graph has a vertex cover of size k, then one can select in polynomial time a set of k vertices that covers over 80% of the edges.

1 Introduction

Given a graph $G = (V, E)$ with $V = \{v_1 \dots v_n\}$, a weight function $w : E \rightarrow R^+$, and a parameter k the Max-Vertex Cover$_k$ (Max-VC$_k$) problem is the problem of finding a subset $U \subseteq V$ of size k such that the weight of edges incident to U is maximal. The Max-VC$_k$ problem is NP hard to solve exactly, hence we consider algorithms that yield approximate solutions. An *approximation algorithm* for NP-hard maximization problems is an algorithm that for every instance produces a solution whose value is guaranteed to be within a ratio of at least α from the value of the optimal solution. The parameter $0 \leq \alpha \leq 1$ is known as the *approximation ratio* of the algorithm, and the larger α is, the better.

A simple approximation algorithm for Max-VC$_k$ which uniformly picks a random subset $U \subseteq V$ of size k has an expected approximation ratio of at least $1 - (1 - k/n)^2$. A greedy algorithm which iteratively constructs the set U by adding vertices that increase the weight of edges incident to U by most, has an approximation ratio of at least $1-1/e$ [Hoc95]. An algorithm based on a linear program relaxation of Max-VC$_k$ is shown in [AS99] to have an approximation ratio of at least 3/4 (for every k). In the following work we consider an algorithm based on a *semidefinite* program relaxation for the Max-VC$_k$ problem.

Semidefinite programming is the problem of maximizing (or minimizing) a linear objective function subject to a set of linear constraints (as in linear programming) with the additional constraint that the variables involved in such a program form the entries of a symmetric positive semidefinite matrix. Semidefinite programming has become a common tool in the approximation of combinatorial optimization problems, much due to the work of Goemans and Williamson [GW95] in which an approximation algorithm for the Max-Cut problem based on semidefinite programming was presented.

In our work we use a semidefinite program relaxation of the Max-VC$_k$ problem in the design of an approximation algorithm with an approximation ratio greater than that achieved by the algorithm suggested in [AS99]. In general our algorithm combines ideas and techniques used in several previous works based on semidefinite programming [GW95, FJ97, FG95].

In addition to the Max-VC$_k$ problem, we consider three other maximization problems related to covering edges when a set of k vertices are selected in a graph. Given an instance $G = (V, E)$, k, and w, instead of finding a subset U of size k incident to the maximal weight of edges as in the Max-VC$_k$ problem, one can seek to maximize :

Max-Dense Subgraph$_k$	:	the weight of edges in the subgraph induced by U.
Max-Cut$_k$	:	the weight of edges cut by the partition $(U, V \setminus U)$.
Max-Uncut$_k$	:	the weight of edges not cut by the partition $(U, V \setminus U)$.

These four problems are illustrated in Figure 1.

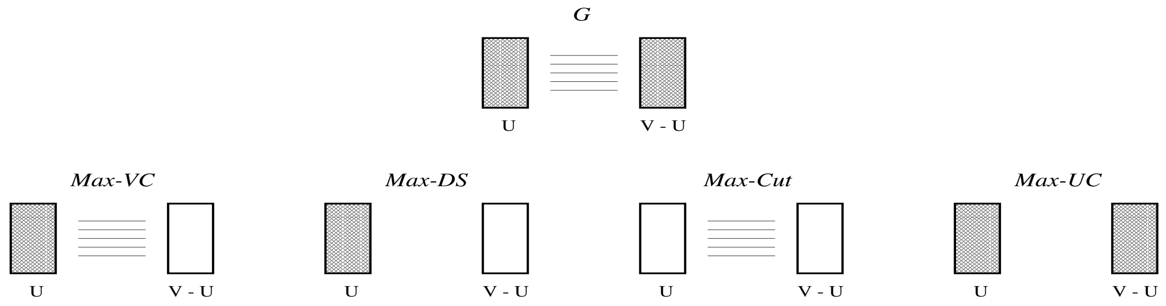

Figure 1: Let $G = (V, E)$ and $U \subseteq V$ such that $|U| = k$. The value of the solution associated with U on the four maximization problems : (a) Max-VC$_k$: the weight of edges incident with vertices in U. (b) Max-DS$_k$: the weight of edges in the subgraph induced by U. (c) Max-Cut$_k$: the weight of edges cut by the partition $(U, V \setminus U)$. (d) Max-UC$_k$: the weight of edges not cut by the partition $(U, V \setminus U)$.

As in the Max-VC$_k$ problem, it is not hard to verify that the additional three optimization problems are NP-hard. We present approximation algorithms for these problems based on semidefinite programming, and improve the approximation ratios previously published.

Our algorithms are all BPP-algorithms, that is they are randomized, run in polynomial time, and succeed with overwhelming probability. The basic techniques used on all four problems are similar, and are based on the work of Frieze and Jerrum [FJ97] on the Max-Cut$_{\frac{n}{2}}$ problem (better known as the Max-Bisection problem). Roughly speaking, given an instance $G = (V, E)$, k, and w, we solve a semidefinite relaxation to obtain a set of n unit vectors in R^n corresponding to the n vertices of G. We then round this vector configuration using the random hyperplane rounding technique presented in [GW95] or variations of the technique, to obtain a subset $U \subseteq V$. Recall that we are interested in a subset U of size k. The subset of vertices U obtained in this process is not necessarily of size k, thus an additional step is added to our algorithm in which we *fix* the size of U to be exactly k.

A recurrent theme in our work is that algorithms based on semidefinite programming have approximation ratios that are strictly better than those known to be achievable using linear programming. A new aspect in our work (applicable at the moment only to Max-VC$_k$ and Max-Cut$_k$) is our ability to handle arbitrary values of k, and not just values of k close to $n/2$. This requires techniques beyond those used in earlier related work (such as [FJ97]).

1.1 Previous work and our results

Our results, as well as previous work, are discussed below and summarized in Table 1.

Max-Vertex Cover$_k$: Max-VC$_k$ is a maximization version of the well known Vertex Cover minimization problem (Min-VC). Whereas in Min-VC we must cover all edges using as few vertices as possible, in Max-VC$_k$ we must use exactly k vertices and cover as many edges as possible.

A generalization of Max-VC$_k$ is the Max-k-Coverage problem. Given a set system $\mathcal{S}$ over the universe U, a parameter k, and a weight function $w : U \to R^+$ the Max-k-Coverage problem is the problem of finding k sets such that the total weight of elements in their union is maximized. By associating with every vertex v_i in V the set of edges incident with v_i, the Max-VC$_k$ problem can be viewed as a special case of the Max-k-Coverage problem.

Several algorithms approximate Min-VC within a ratio of 2, and it is a long standing open problem whether an approximation ratio of $2 - \epsilon$ for some fixed $\epsilon > 0$ can be achieved in polynomial time.

For Max-VC$_k$ we are not yet in a position to formulate a conjecture about the best possible approximation ratio. The simple algorithm that uniformly picks a random subset $U \subseteq V$ of size k has an expected approximation ratio of $1 - (1 - k/n)^2$. A greedy approximation algorithm presented in [Hoc95] has an approximation ratio of $\max(1 - 1/e, 1 - (1 - k/n)^2)$. An algorithm based on linear programming was shown in [AS99] to have an approximation ratio of 3/4.

We present an algorithm based on semidefinite programming that has an approximation ratio of at least $3/4+\varepsilon$ for some universal constant $\varepsilon > 0$ and all values of k. When $k \geq n/2$, or when k is at least the size of the minimum vertex cover in the input graph, we achieve an approximation ratio above 0.8. Our algorithm and its analysis use ideas from [NT75, GW95, FG95, FJ97].

Max-Dense Subgraph$_k$: The simple algorithm that uniformly picks a random subset of k vertices has an expected approximation ratio of $\frac{k(k-1)}{n(n-1)}$. Asahiro et al. [AITT95] analyze a greedy algorithm for the Max-DS$_k$ problem and prove an approximation ratio of $O(k/n)$ for all values of k. Feige, Kortsarz and Peleg present in [KP93, FKP98] an algorithm with an approximation

Problem	Technique	Approximation ratio	Range
Max-VC$_k$	Random	$1-(1-k/n)^2$	all k
	Greedy	$\max(1-1/e, 1-(1-k/n)^2)$	all k
	LP	$\frac{3}{4}$	all k
	SDP	0.8	$k \geq n/2$
			k size of minimum VC
		$3/4+\varepsilon$	all k, universal $\varepsilon > 0$
Max-DS$_k$	Random	$\frac{k(k-1)}{n(n-1)}$	all k
	Greedy	$O(k/n)$	all k
	LP	$\frac{k}{n}(1-\varepsilon)$	all k, every $\varepsilon > 0$
	SDP	$k/n + \varepsilon_k$	$k \sim n/2$
Max-Cut$_k$	Random	$\frac{2k(n-k)}{n(n-1)}$	all k
	LP	$\frac{1}{2}$	all k
	SDP	$1/2+\varepsilon$	all k, universal $\varepsilon > 0$
Max-UC$_k$	Random/LP	$1-\frac{2k(n-k)}{n(n-1)}$	all k
	SDP	$1/2+\varepsilon_k$	$k \sim n/2$

Table 1: Approximation ratios achieved on our four maximization problems, our results appear in columns in which the SDP technique is mentioned.

ratio of at least $\max(k/2n, n^{\varepsilon-1/3})$ for all k and some constant $\varepsilon > 0$. An algorithm based on linear programming (attributed to Goemans) achieves the ratio of $(1-\varepsilon)\frac{k}{n}$ for all values of k and any constant $\varepsilon > 0$. Though these ratios are constant when k is linear in n, it is not known if for all k there exists a polynomial algorithm able to approximate the Max-DS$_k$ problem beyond a ratio of r for some constant $r > 0$. Furthermore, the above ratios are all strictly less than $\frac{k}{n}$ when k is linear in n.

Srivastav and Wolf [SW98] use analysis based on ideas of [GW95, FJ97] and claim that for some values of k, a certain semidefinite program approximates Max-DS$_k$ within a ratio better than k/n. In Section 3.1 we demonstrate that their claim is incorrect. We also present a semidefinite program different than the one used in [SW98], and for it obtain an approximation ratio slightly above k/n when k is close to $n/2$. For example we achieve the ratio of $\alpha_{SDP} = 0.517$ and $\alpha_{SDP} = 0.3502$ for $k = n/2$ and $k = n/3$ respectively.

Max-Cut$_k$: The general Max-Cut problem, of finding a subset $U \subseteq V$ of arbitrary size that maximizes the weight of edges cut by the partition $(U, V \setminus U)$, is studied by Goemans and Williamson [GW95] who achieve an approximation ratio above 0.8785 for this problem. Frieze and Jerrum [FJ97], using techniques introduced in [GW95], study the Max-Cut$_k$ problem for k half the size of the given vertex set V (denoted as the Max-Bisection problem), and achieve an approximation ratio of 0.6511 for such k. Using linear programming [AS99] show that for all values of k an approximation ratio of $1/2$ can be achieved.

We use semidefinite programming to achieve the ratio of at least $1/2+\varepsilon$ for all k and some universal constant $\varepsilon > 0$. In particular for $k = n/2$ we slightly improve the ratio stated in [FJ97] and obtain a ratio of $\alpha_{SDP} = 0.6514$ and for $k = n/3$ we obtain the ratio of $\alpha_{SDP} = 0.58$.

Max-Uncut$_k$: Little is known about the Max-UC$_k$ problem. A closely related minimization problem is the Min-Bisection problem of finding a subset $U \subseteq V$ of size $n/2$ such that the minimum weight of edges are cut by the partition $(U, V \setminus U)$. Much research has been done regarding the Min-Bisection problem. As in the Max-DS$_k$ problem, the existence of a polynomial approximation algorithm for the Min-Bisection problem with a constant approximation ratio remains open.

The straightforward use of linear programming on the Max-UC$_k$ problem does not help us achieve an approximation ratio better than that achieved by picking a random subset $U \subseteq V$ of size k. Such a random process yields the approximation ratio of $1 - \frac{2k(n-k)}{n(n-1)}$. We improve this trivial ratio when k is close to n/2 using semidefinite programming and achieve the ratio of $\alpha_{SDP} = 0.5417$ for $k = n/2$.

1.2 Structure

In Sections 2 through 5 we concentrate on the four maximization problems, one problem per section. Note that as we use related algorithmic ideas in all four of our problems, later sections build upon analysis and intuition given in previous ones. In the Appendix we present some results regarding the integrality gaps of the semidefinite relaxations we use throughout our work, and fill in some technical parts of our proofs.

2 Max-Vertex Cover$_k$

Throughout this section we denote an instance of the Max-VC$_k$ problem by $G = (V, E)$, w and k, where $V = \{v_1 \dots v_n\}$ and each edge $e_{ij} \in E$ is of weight w_{ij}. We define $Opt(G)$ to be the maximal weight of edges covered by (*i.e.* incident to) a set of k vertices in G, and Z^* to be the optimal value of various relaxations of the Max-VC$_k$ problem that will be stated later. We also define $U \subseteq V$ to be the vertex set generated by our algorithms with $U = \{u_1 \dots u_k\}$, and $w(U)$ to be the total weight covered by U.

The Max-VC$_k$ problem is clearly *NP-Hard* which can be seen by a reduction from the Min-VC problem. It has been shown in [Pet94] that for some k it is *NP-Hard* to approximate Max-VC$_k$ beyond some fixed constant less than one (using hardness results on the Min-VC problem restricted to graphs of bounded degree [PY91]). This in fact holds for a wide range of values of k (see [Lan98], for example).

Proposition 2.1 *For every constant $\varepsilon > 0$ there is some constant $c < 1$ such that for every $n^\varepsilon < k < (1-\varepsilon)n$ it is NP-Hard to approximate the Max-VC$_k$ problem within ratio c.*

Our algorithm for Max-VC$_k$ is based on combining semidefinite programming with a greedy algorithm and a linear program. We review the latter approaches first.

2.1 A greedy algorithm

Consider the following algorithm A_{Greedy} :

1. Start with a set U which is empty.
2. Identify a vertex $v \in V \setminus U$ that maximizes the weight of edges covered by $U \cup \{v\}$, and add v to U.
3. Repeat the above until U is of size k.

In [Hoc95] it is shown that algorithm A_{Greedy} has an approximation ratio α of at least $(1-1/e)$. In Lemma B.1 of the appendix we show that A_{Greedy} will return a set U with $w(U) \geq (1-(1-k/n)^2)w(V)$, thus implying that algorithm A_{Greedy} has an approximation ratio of $\alpha \geq \max(1-1/e, 1-(1-k/n)^2)$. This analysis is tight, namely, there are infinitely many graphs on which the greedy approach yields an approximation ratio no better than $(1+\varepsilon)\alpha$ for any constant $\varepsilon > 0$. (See [Lan98], for example.)

2.2 A linear programming based algorithm

Given a graph $G = (V, E)$ consider the following integer program :

(IP-VC)	Maximize	$\sum_{e_{ij} \in E} w_{ij} z_{ij}$	
	subject to:		
	(1)	$z_{ij} \leq x_i + x_j$	for all $e_{ij} \in E$
	(2)	$\sum_{i=1}^{n} x_i = k$	
	(3)	$x_i \in \{0,1\}, \; z_{ij} \in \{0,1\}$	for $1 \leq i \leq n$ and $e_{ij} \in E$

By choosing the vertex v_i to be in the cover U if and only if $x_i = 1$, the integer program *(IP-VC)* corresponds to the Max-VC$_k$ problem on G and the optimal integer solution Z^*_{IP-VC} is exactly $Opt(G)$. As it is *NP-Hard* to solve such a program in polynomial time, we consider a relaxation of *(IP-VC)* by allowing x_i (and z_{ij}) to receive values in the interval $[0, 1]$. We denote the relaxed linear program by *(LP-VC)*. Clearly $Z^*_{LP-VC} \geq Z^*_{IP-VC} = Opt(G)$.

Given an optimal fractional solution $(x_1^* \dots x_n^*)$ to *(LP-VC)* we wish to obtain a valid integer solution to *(IP-VC)*. In [AS99], Ageev and Sviridenko present a rounding technique that yields a valid integer solution of value at least

$$\sum_{e_{ij} \in E} w_{ij}(1-(1-x_i^*)(1-x_j^*)).$$

In [GW94] it is shown that for any $x_i^*, x_j^* \in [0, 1]$ we have

$$1-(1-x_i^*)(1-x_j^*) \geq \frac{3}{4} z_{ij}$$

with equality only in the case $x_i^* = x_j^* = 1/2$. Combining these results, [AS99] obtain an approximation algorithm based on linear programming with an approximation ratio of $3/4$. Note that the analysis in [AS99] is tight as there are graphs for which the integrality gap of relaxation *(LP-VC)* is arbitrarily close to $\frac{4}{3}$.

2.3 Approximation algorithms based on semidefinite programming

Goemans and Williamson [GW95] have studied a semidefinite relaxation to the Max-2SAT problem. Their ideas were extended by Feige and Goemans [FG95], resulting in an 0.931 approximation algorithm for Max-2SAT. Combining this with ideas from [FJ97, SW98], we show that semidefinite programming can be used in order to obtain improved approximation ratios for Max-VC$_k$. The following theorem is proven.

Theorem 2.2 *For the following values of α_{SDP} and k, there exists a BPP algorithm based on semidefinite programming that approximates the Max-VC$_k$ problem within the ratio of α_{SDP}.*

(a) : $\alpha_{SDP} > 0.8$, for $k \geq n/2$.

(b) : $\alpha_{SDP} > 0.8$, for k greater than or equal to the size of the minimum vertex cover in a given graph G.

(c) : $\alpha_{SDP} \geq 3/4 + \varepsilon$, for all k and some universal constant $\varepsilon > 0$.

Proof of Theorem 2.2(a) : Given an instance $G = (V, E)$, w and k of the Max-VC$_k$ problem, consider the following quadratic integer program.

(QI-VC)
$$\begin{array}{lll} \text{Maximize} & \sum_{e_{ij}\in E} w_{ij} \left(\frac{3+x_i+x_j-x_ix_j}{4}\right) & \\ \text{subject to:} & & \\ (1) & \sum_{i=1}^{n} x_i = 2k-n & \\ (2) & x_i \in \{-1,1\} & \text{for } 1 \leq i \leq n \end{array}$$

Notice that for each edge e_{ij} the objective function is zero if both x_i and x_j are equal to -1 and w_{ij} otherwise. Furthermore, constraint *(1)* guarantees that exactly k variables will be of value 1. Thus by choosing the vertex $v_i \in V$ to be in the cover U if and only if $x_i = 1$, the quadratic program *(QI-VC)* corresponds to the Max-VC$_k$ problem on G. As in the case of linear integer programming we consider a relaxation of *(QI-VC)* that is known to be solvable (up to an arbitrarily small additive error) in polynomial time.

The semidefinite relaxation :
Consider the following relaxation for the Max-VC$_k$ problem :

(SDP-VC)
$$\begin{array}{lll} \text{Maximize} & \sum_{e_{ij}\in E} w_{ij} u(i,j) & \\ \text{subject to:} & & \\ (1) & \sum_{i=1}^{n} v_iv_0 = 2k-n & \\ (2) & v_i \in S_n & \text{for } 0 \leq i \leq n \\ (3) & \left.\begin{array}{ll} v_iv_j + v_jv_k + v_kv_i & \geq -1 \\ v_iv_j - v_jv_k - v_kv_i & \geq -1 \end{array}\right\} & \forall i,j,k \in \{0 \ldots n\} \end{array}$$

where S_n is the unit sphere in R^{n+1}, v_i is an $(n+1)$-dimensional vector in S_n representing the vertex v_i, and

$$u(i,j) = \frac{3 + v_iv_0 + v_jv_0 - v_iv_j}{4}$$

Without constraint *(3)* relaxation *(SDP-VC)* is obtained from the quadratic integer program *(QI-VC)* by associating a unit vector v_i with each vertex i. The inner product $v_i v_0$ corresponds to the variable of x_i, and each product $x_i x_j$ is replaced by the inner product $v_i v_j$. *(SDP-VC)*, including constraint *(3)*, is indeed a relaxation to the Max-VC$_k$ problem. Namely, given an instance of Max-VC$_k$ and a subset $U \subseteq V$ of size k, a valid vector solution to *(SDP-VC)* of value $w(U)$ can be constructed. This is done by replacing the vector v_0 with the vector $(1, 0, \ldots, 0)$, the vector v_i with the vector $(1, 0, \ldots, 0)$ if the vertex v_i is in U and $(-1, 0, \ldots, 0)$ otherwise. With these values for $\{v_0, \ldots v_n\}$ it follows that $u(i,j) = 1$ if the edge e_{ij} is covered by U and 0 otherwise. Note that all constraints in *(SDP-VC)* are fulfilled by this vector solution. The addition of constraint *(3)* to our relaxation ensures that $u(i,j) \leq 1$, enables us to use the results of [FG95] stated later, and is crucial to the proof of Theorem 2.2*(c)*.

Solving *(SDP-VC)* on a given graph $G = (V, E)$ and integer k, we obtain an optimal vector solution of value Z^*_{SDP}. Clearly from the above discussion we have that $Z^*_{SDP} \geq Opt(G)$. This solution is *rounded* into a subset $U \subseteq V$ by the following *rounding* technique analyzed in [FG95].

The rounding technique :

Given a vector solution to *(SDP-VC)* of $n+1$ vectors, $v_i \in S_n$ for $i \in \{0 \ldots n\}$, we map v_i to a new vector $\tilde{v}_i$ using the following mapping. Let $f(\theta) = \theta + 0.8067(\frac{\pi}{2}(1 - \cos(\theta)) - \theta)$. (The function f is a convex combination of the functions $f_0(\theta) = \theta$ and $f_1(\theta) = \frac{\pi}{2}(1 - \cos(\theta))$.) If the angle between v_i and v_0 is θ, we define $\tilde{v}_i$ to be the vector in S_n that forms an angle of $f(\theta)$ with v_0, and lies on the plane created by v_0 and v_i on the same side of v_0 as v_i. This has the effect of moving each vector v_i towards the vector v_0 if the angle between v_i and v_0 is less than $\pi/2$ and towards $-v_0$ otherwise. We now choose a random hyperplane passing through the origin and set the vertex v_i to be in the cover U if and only if the corresponding vector $\tilde{v}_i$ is on the same side of the hyperplane as the vector $\tilde{v}_0$. This is done by choosing a vector r uniformly distributed in the unit sphere S_n and defining U to be the vertex set $\{v_i \mid \text{sgn}(\tilde{v}_i \cdot r) = \text{sgn}(\tilde{v}_0 \cdot r)\}$.

Recall that our objective, after the rounding scheme, is to obtain a subset U of *small* size and *large* weight (*i.e.* the size of U is to be at most k and the weight covered by U is to be as close to the optimal cover as possible). Unfortunately the set U obtained by this rounding technique may be of arbitrary size, but using computer assisted analysis, [FG95] have bounded from above the probability that a vertex v_i is chosen to be in the cover U, and from below the probability that an edge e_{ij} is covered.

$$\Pr(v_i \notin U) \geq \alpha_1 \left(\frac{1 - v_i v_0}{2} \right)$$

$$\Pr(e_{ij} \text{ is covered by } U) \geq \alpha_2 u(i,j)$$

with $\alpha_1 = 0.976$, and $\alpha_2 = 0.931$. These bounds are used in order to analyze the following algorithm.

Algorithm A_{SDP-VC} **:**

1. Solve the relaxation *(SDP-VC)*, corresponding to a given graph $G = (V, E)$, to obtain an optimal set $\{v_0^*, \ldots v_n^*\}$ of vectors in S_n.

2. Round the optimal set of vectors using the technique described above to obtain a subset U of V.
3. Repeat step two above a polynomial number of times and set $\tilde{U}$ to be the *best* solution found. (The *best* solution will be defined shortly using a new random variable Z.)
4. Greedily *fix* the size of $\tilde{U}$ to be exactly k. This is done by iteratively removing or adding vertices from/to $\tilde{U}$. If $|\tilde{U}| > k$, iteratively remove the vertex which is adjacent to the least weight of edges in the cut $(\tilde{U}, V \setminus \tilde{U})$. If $|\tilde{U}| < k$, iteratively add the vertex v that maximizes $w(\tilde{U} \cup v)$. Denote the resulting set by U_k.

We now analyze the quality of the set U_k returned by algorithm A_{SDP-VC}.

Analysis :
Let Z^*_{SDP} be the optimal value and $\{v^*_0, \dots v^*_n\}$ the optimal vector solution obtained by solving the relaxation *(SDP-VC)* in step *(1)* of algorithm A_{SDP-VC}. Let U be the subset of V obtained by rounding these vectors in step *(2)*, and $k = n/c$ for $c > 1$. Using ideas of [FJ97, SW98] define

$$Z = \frac{w(U)}{Z^*_{SDP}} + \theta_k \frac{n - |U|}{n - k}$$

where $\theta_k \geq 0$ is some fixed constant.

Intuitively we would like the total weight covered by U to be large, while the size of U remains smaller than k. Thus it is reasonable to find a set U that maximizes Z which is a weighed and normalized sum of $w(U)$ and $(n - |U|)$. We start by analyzing the expectation of Z.

Lemma 2.3 *The expected value of Z is at least $\alpha_2 + \alpha_1 \theta_k$ with α_1, and α_2 as defined earlier.*

Proof : It suffices to compute the expected values of $w(U)$ and $|U|$. From [FG95] we have that

$$E[w(U)] = \sum_{e_{ij} \in E} w_{ij} \Pr(e_{ij} \text{ is covered by } U) \geq \sum_{e_{ij} \in E} \alpha_2 w_{ij} u(i,j) = \alpha_2 Z^*_{SDP}.$$

Furthermore, by constraint *(1)* of *(SDP-VC)* we have that $\sum_{i=1}^n v_i v_0 = 2k - n$ thus

$$\begin{aligned} E[|U|] &= \sum_{i=1}^n \Pr(v_i \in U) = \sum_{i=1}^n (1 - \Pr(v_i \notin U)) \leq n - \alpha_1 \left(\sum_{i=1}^n \frac{1 - v_i v_0}{2} \right) = \\ &= n\left(1 - \frac{\alpha_1}{2}\right) + \frac{\alpha_1}{2} \sum_{i=1}^n v_i v_0 = n\left(1 - \frac{\alpha_1}{2}\right) + \frac{\alpha_1}{2}(2k - n) = n(1 - \alpha_1) + \alpha_1 k. \end{aligned}$$

Which gives us the desired result.

■

We now show for any $\varepsilon = 1/poly(n)$ that with non-negligible probability our rounding scheme returns a set U with a corresponding Z of value at least $(1 - \varepsilon)E[Z]$. Using results stated in Section 2.1 (the greedy algorithm) we have that

$$Z^*_{SDP} \geq Opt(G) \geq \left(1 - (1 - k/n)^2\right) w(V) = \left(1 - (1 - 1/c)^2\right) w(V).$$

Hence from the fact that $w(U)$ is less than $w(V)$ and $|U|$ is non-negative we conclude that Z is bounded by

$$Z \leq \frac{w(V)}{w(V)\,(1-(1-1/c)^2)} + \theta_k \frac{n}{n-n/c} = \frac{c^2}{(2c-1)} + \theta_k \frac{c}{c-1} = Z_{max}$$

Thus using the Markov inequality

$$\Pr(Z \leq (1-\varepsilon)E[Z]) \leq \frac{Z_{max} - E[Z]}{Z_{max} - (1-\varepsilon)E[Z]} = p < 1.$$

Repeating step *(2)* of algorithm A_{SDP-VC} independently a polynomial number of times and returning the set U with the maximal corresponding Z (*i.e.* the *best* U), we have with high probability that Z is at least $(1-\varepsilon)E[Z]$. We are now ready to prove our main lemma which concludes the proof of Theorem 2.2(a).

Lemma 2.4 *For $k \geq \frac{n}{2}$, algorithm A_{SDP-VC} returns, with high probability, a vertex set U_k of size k and weight αZ^*_{SDP} with $\alpha > 0.8$.*

Proof : Recall that $c = \frac{n}{k}$ and observe that by the conditions of the lemma $1 \leq c \leq 2$. Let $\tilde{U}$ be the set returned by step *(3)* of algorithm A_{SDP-VC} with $w(\tilde{U}) = \lambda Z^*_{SDP}$, $|\tilde{U}| = \mu n$ for some $\lambda \geq 0$, $\mu \in [0,1]$. From the above analysis we have with high probability that for any $\varepsilon \geq 1/poly(n)$ the corresponding random variable Z is of size at least $(1-\varepsilon)E[Z]$ meaning :

$$Z = \lambda + \theta(1-\mu)\frac{c}{c-1} > (1-\varepsilon)(\alpha_2 + \theta\alpha_1)$$

Hence :

$$\lambda > (1-\varepsilon)(\theta\alpha_1 + \alpha_2) - \theta(1-\mu)\frac{c}{c-1}$$

We now discuss two different possibilities. If the set $\tilde{U}$ is of size greater than or equal to k (*i.e.* $\mu \geq 1/c$) we *fix* the size of $\tilde{U}$ by the following greedy process. At each step identify the vertex in $\tilde{U}$ whose removal from $\tilde{U}$ reduces $w(\tilde{U})$ by the least amount, and remove it from $\tilde{U}$. It can be seen that the resulting set U_k has weight at least $\frac{k}{|\tilde{U}|}w(\tilde{U}) = \frac{1}{c\mu}\lambda Z^*_{SDP}$. Thus in this case, ignoring the negligible factor of $(1-\varepsilon)$, our algorithm returns a valid vertex set U_k of weight at least $\alpha_+ Z^*_{SDP}$ with

$$\alpha_+ = \frac{1}{c\mu}\lambda \geq \min_{\mu \in [\frac{1}{c},1]} \left(\frac{1}{c\mu} \left(\theta\alpha_1 + \alpha_2 - \theta(1-\mu)\frac{c}{c-1} \right) \right)$$

The above minimum is achieved when $\mu = 1/c$ or $\mu = 1$ thus

$$\alpha_+ \geq \min\left(\theta\alpha_1 + \alpha_2 - \theta \;\;,\;\; \frac{1}{c}(\theta\alpha_1 + \alpha_2)\right).$$

On the other hand, if the size of $\tilde{U}$ is less than k (*i.e.* $\mu < 1/c$) we must add $k - |\tilde{U}|$ vertices to $\tilde{U}$ to obtain a subset U_k of size exactly k. Recall the greedy algorithm reviewed in Section 2.1. Using this algorithm on the subgraph induced by $V \setminus \tilde{U}$ we are able to obtain a subset $\hat{U} \subseteq V \setminus \tilde{U}$ of size $k - |\tilde{U}|$ that covers the weight of at least $(1-(1-p)^2)(w(V) - w(\tilde{U}))$ where $p = \frac{1/c - \mu}{1-\mu}$ and $w(V) - w(\tilde{U})$ is the weight of edges in the subgraph induced by $V \setminus \tilde{U}$. Setting U_k to be the union of the original set $\tilde{U}$ and the set $\hat{U}$ above, we conclude that U_k is of size exactly k and covers the weight of at least

$$\begin{aligned} w(U_k) &\geq w(\tilde{U}) + (w(V) - w(\tilde{U}))\left(1-(1-p)^2\right) \\ &\geq \lambda Z^*_{SDP} + (Z^*_{SDP} - \lambda Z^*_{SDP})\left(1-(1-p)^2\right) \\ &= \left(1-(1-\lambda)(1-p)^2\right) Z^*_{SDP}. \end{aligned}$$

Thus, in this case, our algorithm returns a valid cover of weight $\alpha_- Z^*_{SDP}$ with

$$\alpha_- = 1 - (1-\lambda)(1-p)^2.$$

All in all, with high probability algorithm A_{SDP-VC} returns a set U_k of weight at least αZ^*_{SDP} with $\alpha = \min(\alpha_-, \alpha_+)$. By computing the exact value of α for $k = \frac{n}{c}$ with $c \in (1,2]$, our lemma is proven (detailed analysis is given in Lemma B.2 of the appendix). For $c = 1$ (*i.e.* $k = n$) our algorithm trivially returns the optimal vertex set V.

■

The second part of Theorem 2.2 deals with the special case of Max-VC$_k$ in which k is greater than or equal to the minimum vertex cover of a given graph G.

Theorem 2.2(b) *Given a graph G, there exists a BPP algorithm based on semidefinite programming that approximates the Max-VC$_k$ problem within the ratio of $\alpha > 0.8$ for all k greater than or equal to the size of the minimum vertex cover in G.*

Proof : Consider the following well known linear relaxation to the *Min-Vertex Cover* problem (the problem of finding a minimum set of vertices covering the entire edge set).

(Min-VC)	Minimize	$\sum_{i=1}^{n} x_i$	
	subject to:		
	(1)	$x_i + x_j \geq 1$	for all $e_{ij} \in E$
	(2)	$0 \leq x_i \leq 1$	for $1 \leq i \leq n$

For a given graph $G = (V, E)$ one can efficiently find an optimal fractional solution to the above linear relaxation such that $x_i \in \{0, 1/2, 1\}$ for all $1 \leq i \leq n$ [NT75]. Furthermore, [NT75] show that there exists some minimum vertex cover in G that agrees with this *half integral* solution on its integer values (*i.e.* there exists some minimum vertex cover U such that if x_i is equal to 1 then the corresponding vertex v_i is in U and if x_i is equal to 0 then the corresponding vertex v_i in not in U). Given an optimal half integral solution to *(Min-VC)*, these two facts are used to partition the graph G into three parts. The set U_1 of all vertices v_i with corresponding fractional values x_i that are equal to 1, the set R with vertices of fractional value 0, and the remaining set Q with vertices of fractional value 1/2. It is not hard to see that the following properties hold for the above partition :

1. At least one optimal vertex cover of G contains all vertices in U_1.
2. Each vertex in R has all its neighbored vertices in U_1.
3. The minimum vertex cover of the subgraph $H_Q = (Q, E_Q)$, induced by the vertices in Q, is of size at least $|Q|/2$.

Thus if $G = (V, E)$, w and k is an instance of the Max-VC$_k$ problem with k greater than or equal to the size of the minimum vertex cover in G, we partition G as above and use algorithm A_{SDP-VC} on the subgraph H_Q. As described below this procedure assures us the desired results.

Set k_Q to be $k - |U_1|$. We assume that k_Q is less than $|Q|$ otherwise we trivially take $U_1 \cup Q$ as our desired cover. In addition $|U_1|$ is at most the size of the minimum vertex cover in G thus k_Q is non-negative. From the above we conclude that the induced subgraph H_Q has a vertex cover of size k_Q with $k_Q \geq |Q|/2$. Using algorithm A_{SDP-VC} on H_Q, we find a subset U_2 in Q of size k_Q with weight at least $\alpha w(E_Q)$, where α is the approximation ratio of algorithm A_{SDP-VC} for k_Q and $w(E_Q)$ is the weight of edges in the induced subgraph H_Q. From Theorem 2.2*(a)* and the fact that $k_Q \geq |Q|/2$ we conclude that α is at least 0.8. Hence setting U to be $U_1 \cup U_2$ we receive a subset of V with size exactly k and weight at least $\alpha w(V) = \alpha Opt(G)$ (recall that there are no edges from R to Q).

■

Combining SDP and linear programming : We now show how part *(a)* of Theorem 2.2 can be used in order to obtain an approximation ratio better than 3/4 for all values of k. Let $G = (V, E)$, w and k be an instance of the Max-VC$_k$ problem. Given a valid vector configuration $\{v_0 \ldots v_n\}$ to our semidefinite relaxation *(SDP-VC)* of value Z^*, we are able to construct a valid fractional solution (x, z) to the previous linear relaxation *(LP-VC)* of value at least Z^*. This is done by setting x_i to be equal to $\frac{1+v_i v_0}{2}$ for $i = 1 \ldots n$ and z_{ij} to be $\min(1, x_i + x_j)$ for all $e_{ij} \in E$. Due to the fact that $v_i v_0 \in [-1, 1]$ we have that $x_i \in [0, 1]$. We also conclude using constraint *(1)* of relaxation *(SDP-VC)* that $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} \frac{1+v_i v_0}{2} = k$. Finally by constraint *(3)* of *(SDP-VC)* we have that $u(i,j) = \frac{3+v_i v_0 + v_j v_0 - v_i v_j}{4} \leq \min(1, x_i + x_j) = z_{ij}$. These three observations show us that the fractional solution (x, z) is indeed a valid one for *(LP-VC)* of value at least Z^*. Using the rounding technique mentioned in Section 2.2 on (x, z) we may obtain a subset $U_k \subseteq V$ of size exactly k that covers the weight of at least

$$\frac{3}{4} \sum_{e_{ij} \in E} w_{ij} z_{ij} \geq \frac{3}{4} Z^*.$$

As mentioned earlier, if for each edge e_{ij} one of the values x_i or x_j differ from 1/2, an approximation ratio greater than 3/4 can be achieved. This fact and the fact that our semidefinite algorithm A_{SDP-VC} has an advantage over the approximation ratio of 3/4 when k is of size approximately $n/2$ results in the following algorithm.

Given an instance $G = (V, E)$, w and k of the Max-VC$_k$ problem, solve the semidefinite relaxation *(SDP-VC)* to obtain an optimal set of vectors $\{v_0^* \dots v_n^*\}$ of total value Z^*. Compute the corresponding fractional solution (x^*, z^*), to relaxation *(LP-VC)*, as above. Define an edge e_{ij} to be *good* if at least one of the values x_i^* or x_j^* differ from $1/2$ by at least some small constant value. Now assume that the contribution of these *good* edges to the value Z^* is non-negligible (*i.e.* some constant factor of Z^* is achieved by these edges). Thus rounding our vector solution $\{v_1^* \dots v_n^*\}$ using (x^*, z^*) computed above and the rounding technique from [AS99], we achieve an approximation ratio strictly greater than $3/4$ on the set of *good* edges and an approximation ratio of at least $3/4$ on the remaining edges. As the contribution of the *good* edges is non-negligible we obtain, in this case, an approximation ratio slightly above $3/4$.

Otherwise the contribution of the *good* edges to the value Z^* is negligible and we may ignore them. Thus we concentrate on the subgraph $H = (V_h, E_h)$ of G, induced by the remaining *bad* edges. Recall that for each *bad* edge e_{ij} both x_i^* and x_j^* are close to $1/2$, hence the sum of x_i^* for all vertices v_i in the graph H is approximately **half** the size of the vertex set V_h. We now find ourselves in the exact case dealt with by part *(a)* of Theorem 2.2. Thus we conclude that ignoring all *good* edges and rounding the remaining vector configuration using the rounding technique described in algorithm A_{SDP-VC}, we obtain a subset U_k of size exactly k that covers the weight of approximately $0.8Z^*$.

All in all, when the above ideas are analyzed carefully we achieve part *(c)* of Theorem 2.2 stated below (detailed proof of the above is given in Lemma B.3 of the appendix).

Theorem 2.2(c) *There exists a BPP algorithm based on semidefinite programming that approximates the Max-VC$_k$ problem within the ratio of $\alpha \geq 3/4 + \varepsilon$ for all k and some universal constant $\varepsilon > 0$.*

3 Max-Dense Subgraph$_k$

Srivastav and Wolf [SW98] have studied an approximation algorithm based on semidefinite programming and claim to achieve an approximation ratio slightly above k/n on Max-DS$_k$ for various values of k. We show that this claim is incorrect for their particular semidefinite program, and that it is correct for a modified semidefinite program.

As before we denote an instance of the Max-DS$_k$ problem by $G = (V, E)$, w and k. We define $Opt(G)$ to be the maximal weight of edges induced by a subgraph of k vertices in G, and Z^* to be the optimal value of relaxations stated later. We also define $U \subseteq V$ to be the vertex set generated by our algorithms with $U = \{u_1 \dots u_k\}$, and $w(U)$ to be the total weight of edges in the subgraph induced by U.

3.1 An approximation algorithm based on semidefinite programming

Using ideas from [GW95, FJ97, FS97, SW98] we analyze a semidefinite relaxation to the Max-DS$_k$ problem, and achieve an approximation algorithm with a ratio that beats the target of k/n when k is close to $n/2$. In particular, using the following theorem we prove that for

$k = n/2$ we obtain an approximation ratio of at least 0.517.

Theorem 3.1 *For k 'close' to $n/2$ there exists some constant $\varepsilon_k > 0$ such that an approximation ratio of $\alpha_{SDP} \geq k/n + \varepsilon_k$ for the Max-DS$_k$ problem can be achieved by a BPP algorithm based on semidefinite programming.*

Proof : Consider the following quadratic integer program.

(QI-DS)	Maximize	$\sum_{e_{ij} \in E} w_{ij} \left(\frac{1+x_i+x_j+x_i x_j}{4} \right)$	
	subject to:		
	(1)	$\sum_{i=1}^{n} x_i = 2k - n$	
	(2)	$x_i \in \{-1, 1\}$	for $1 \leq i \leq n$

Given a vertex set U of size k, by setting x_i to be of value 1 if and only if the corresponding vertex v_i is in U, we obtain a valid integer solution to *(QI-DS)*. In this solution an edge e_{ij} in the subgraph induced by U contributes the weight of w_{ij} to the objective function, while the remaining edges contribute nothing. In [SW98] the following semidefinite relaxation of *(QI-DS)* is suggested.

(SDP-DS1)	Maximize	$\sum_{e_{ij} \in E} w_{ij} \left(\frac{1+v_i v_0+v_j v_0+v_i v_j}{4} \right)$	
	subject to:		
	(1)	$\sum_{i=1}^{n} v_i v_0 = 2k - n$	
	(2)	$v_i \in S_n$	for $0 \leq i \leq n$

As in Section 2.3, *(SDP-DS1)* is obtained from the quadratic integer program *(QI-DS)* by associating with a vertex i a unit vector v_i, replacing each variable x_i with the inner product $v_i v_0$ and each product $x_i x_j$ with the inner product $v_i v_j$. For reasons that will be discussed shortly, we define the following *improved* relaxation.

(SDP-DS2)	Maximize	$\sum_{e_{ij} \in E} w_{ij} \left(\frac{1+v_i v_0+v_j v_0+v_i v_j}{4} \right)$	
	subject to:		
	(1)	$\sum_{j=1}^{n} v_i v_j = v_i v_0 (2k - n)$	$\forall i \in \{0 \ldots n\}$
	(2)	$v_i \in S_n$	for $0 \leq i \leq n$

Notice the slight difference between these relaxations. Constraint *(1)* of the first relaxation is extended to the constraints $\sum_{j=1}^{n} v_i v_j = v_i v_0 (2k - n)$ of *(SDP-DS2)* where $i \in \{0 \ldots n\}$. For vectors v_i in the unit sphere S_n, these new constraints are equivalent to the constraint $\sum_{i=1}^{n} v_i = v_0 (2k - n)$. Namely, the vectors satisfying constraints *(1)* and *(2)* of *(SDP-DS2)* are spread out in a *symmetric* manner around the vector v_0. In our future discussion we will see that the generalized constraint *(1)* of *(SDP-DS2)* is crucial to the success of our algorithm. Furthermore, its absence in relaxation *(SDP-DS1)* enables us to contradict results stated in [SW98].

For a given graph $G = (V, E)$, an integer k, and a subset $U \subseteq V$ of size k, we obtain a valid vector solution to *(SDP-DS1,2)* of value $w(U)$ by replacing the vector v_0 with the vector $(1, 0, \ldots, 0)$, the vector v_i with the vector $(1, 0, \ldots, 0)$ if the vertex v_i is in U and $(-1, 0, \ldots, 0)$ otherwise. With these values for $\{v_0, \ldots v_n\}$ it follows that $\frac{1+v_iv_0+v_jv_0+v_iv_j}{4} = 1$ if the edge e_{ij} is in the subgraph induced by U and 0 otherwise. This, in addition to the fact that any valid vector configuration of *(SDP-DS2)* is a valid vector configuration of *(SDP-DS1)*, implies that $Opt(G) \leq Z^*_{SDP-DS2} \leq Z^*_{SDP-DS1}$ where Z^*_R is the optimal value of relaxation R.

Solving *(SDP-DS2)* we obtain an optimal vector solution $\{v_0^* \ldots v_n^*\}$. This solution is *rounded* into a subset $U \subseteq V$ by the *random hyperplane* rounding technique suggested in [GW95]. Choose a random hyperplane passing through the origin and set the vertex v_i to be in the set U if and only if the corresponding vector v_i is on the same side of the hyperplane as the vector v_0. This is done by choosing a vector r uniformly distributed in the unit sphere S_n and defining U to be the vertex set $\{v_i \mid \operatorname{sgn}(v_i \cdot r) = \operatorname{sgn}(v_0 \cdot r)\}$.

Recall, as in the previous section, that our objective after the rounding procedure is to obtain a subset U of *small* size with a *heavy* induced subgraph. The analysis of the random hyperplane rounding technique in [GW95] provides the following inequalities that are useful in establishing these properties (with high probability).

$$\Pr(v_i \notin U) \geq \alpha \left(\frac{1 - v_iv_0}{2}\right) \quad , \quad \Pr\,(e_{ij} \text{ is in the subgraph induced by } U) \geq \beta \left(\frac{1 + v_iv_0 + v_jv_0 + v_iv_j}{4}\right)$$

where $\alpha \geq 0.8785$ and $\beta \geq 0.796$. These bounds will help us analyze the following algorithm analogous to algorithm A_{SDP-VC}.

Algorithm A_{SDP-DS}

1. Solve the relaxation *(SDP-DS2)*, corresponding to a given graph $G = (V, E)$, to obtain an optimal set $\{v_0^*, \ldots v_n^*\}$ of vectors in S_n.
2. Round the optimal set of vectors using the *random hyperplane* rounding technique to obtain a subset U of V.
3. Repeat step two above a polynomial number of times and set $\tilde{U}$ to be the solution that maximizes the random variable Z that will be defined shortly.
4. If $|\tilde{U}| > k$, greedily fix the size of $\tilde{U}$ to be exactly k. The greedy procedure and its analysis are given in Lemma C.1 of the appendix (the fixing lemma). Otherwise, *add* $k - |\tilde{U}|$ vertices to $\tilde{U}$. This *addition* is not arbitrary and is described in Lemma C.2 of the appendix. Denote the resulting set by U_k.

Let Z^*_{SDP} be the optimal value and $\{v_0^* \ldots v_n^*\}$ be the optimal vector solution obtained by solving the relaxation *(SDP-DS2)* in step *(1)* of algorithm A_{SDP-DS}. Let U be the subset of V obtained by step *(2)* of A_{SDP-DS}, and $k = n/c$ for $c > 1$. Extending ideas from [FJ97, SW98] define

$$Z = \frac{w(U)}{Z^*_{SDP}} + \theta_1 \frac{n - |U|}{n - k} + \theta_2 \frac{|U|(2k - |U|)}{n^2}$$

where θ_1 and θ_2 are non-negative fixed constants.

The first and second components of the random variable Z are identical to those used previously in Section 2.3. The third component of Z adds extra complications in the analysis, but without it one cannot prove an approximation ratio better than k/n (at least not using techniques similar to those used in Section 2.3). This can be seen as follows. The first two components relate only to the expected weight and size of U. Consider for example the complete graph on n vertices, in which we seek a dense subgraph on k vertices. Clearly, such a subgraph has $k(k-1)/2$ edges. Assume now that the only information that we are using about U is that it has expected size k and expected weight (number of edges) W. How large might W be? If U is the whole graph with probability k/n and the empty set with probability $1-k/n$, then indeed its expected size is k, and moreover, its expected weight is $\frac{k}{n}\binom{n}{2} = k(n-1)/2$. Hence the ratio between the optimal weight and the expected weight is $(k-1)/(n-1)$. We want to prove better approximation ratios. Hence we add the third term to Z, which in a sense controls the variance of $|U|$. (See also discussion in Section 3.2.)

We now analyze the expected value of Z.

Lemma 3.2 *The expected value of Z is at least $\beta + \theta_1\alpha + \theta_2\left(\alpha(1-1/c)^2 - 1 + 2/c\right)$ when $k \le n/2$, and $\beta + \theta_1\alpha + \theta_2\alpha/c^2$ otherwise.*

Proof : We start by stating some additional bounds from [GW95] regarding the *random hyperplane* rounding technique.

$$\Pr(v_i \in U) \ge \alpha\left(\frac{1+v_iv_0}{2}\right) \quad , \quad \Pr\left(v_i \text{ and } v_j \text{ are } \textit{separated}\right) \ge \alpha\left(\frac{1-v_iv_j}{2}\right)$$

where α remains 0.8785, and 'v_i is *separated* from v_j' if one of the vertices is in U and the other in $V \setminus U$. We thus conclude using constraint *(1)* of *(SDP-DS2)* that $E[|U|] \ge \alpha k$, and using the proof in the previous section (Lemma 2.3) that $E[|U|] \le (1-\alpha)n + \alpha k$ and $E[w(U)] \ge \beta Z^*_{SDP}$. Let Λ_{ij} be the indicator function of the event 'v_i is *separated* from v_j'. It follows that

$$E[|U|(n-|U|)] = \sum_{i<j} E[\Lambda_{ij}] \ge \alpha \sum_{i<j}\left(\frac{1-v_iv_j}{2}\right) = \alpha k(n-k).$$

The last equality is derived from constraint *(1)* of *(SDP-DS2)* which implies

$$\sum_{i,j} v_iv_j = \sum_i \sum_j v_iv_j = (2k-n)\sum_i v_iv_0 = (2k-n)^2.$$

Note that this is the only place in our analysis where the *symmetric* nature of a valid vector configuration is used. Using the fact that $\alpha k \le E[|U|] \le (1-\alpha)n + \alpha k$ we conclude

$$E[|U|(2k-|U|)] = E[|U|(n-|U|)] - (n-2k)E[|U|] \ge \begin{cases} \alpha(n-k)^2 + 2kn - n^2 & \text{for } k \le \frac{n}{2} \\ \alpha k^2 & \text{otherwise} \end{cases}$$

■

From the fact that Z is bounded we conclude that for any constant $\varepsilon > 0$, repeating our rounding scheme a polynomial number of times we obtain a subset U such that its corresponding

Z is of value at least $(1-\varepsilon)E[Z]$. Using a proof technique similar to that of Lemma 2.4, we prove our Theorem 3.1. Detailed analysis for the case $k = n/2$ is given in Lemma C.1 (the fixing lemma) and Lemma C.2 of the appendix.

■

3.2 Counter example to [SW98]

In [SW98], Srivastav and Wolf analyze an algorithm analogous to A_{SDP-DS} which uses relaxation *(SDP-DS1)*. Essentially 'Theorem 1' of [SW98] claims that given a valid vector configuration for the relaxation *(SDP-DS1)*, algorithm A_{SDP-DS} yields a subset U of size k with an induced subgraph of weight $\alpha_k Z^*_{SDP-DS1}$. In particular for $k = n/3$, [SW98] claim that $\alpha_k = 0.3353 > 1/3$. The following example contradicts this claim and shows that the *symmetric* nature of the generalized constraint *(1)* in *(SDP-DS2)* is crucial to our analysis. Recall that this constraint is used to analyze the third component $|U|(2k - |U|)$ of the random variable Z defined previously.

Consider the complete graph on n vertices K_n. For any k the optimal set of k vertices induces a subgraph of weight $\frac{k^2}{2} - \frac{k}{2}$. Let $k = n/2$ and consider the vector configuration where all vectors $\{v_1 \dots v_n\}$ are equal to a single vector w such that w is perpendicular to v_0. This *non-symmetric* vector configuration is not a valid vector configuration for relaxation *(SDP-DS2)* used in our algorithm, but is a valid solution to *(SDP-DS1)* used in [SW98]. The value of the above vector configuration is $\frac{1}{2}\left(\frac{n^2}{2} - \frac{n}{2}\right)$. Thus implying that the value of $Opt(K_n)$ is less than or equal to $\frac{1}{2}Z^*_{SDP-DS1}$ (*i.e.* relaxation *(SDP-DS1)* on K_n has an integrality gap of at least 2 when $k = n/2$). We conclude that our analysis, guaranteeing an approximation ratio above 1/2, fails in the above example. Specifically we have that $E[|U|(2k - |U|)]$ equals zero in contrast to the positive lower bound stated in Lemma 3.2.

For general k, by setting the vector w such that the inner product $v_0 w$ is exactly $\frac{2k}{n} - 1$, it can be seen that the value of relaxation *(SDP-DS1)* is $\frac{k}{n}\left(\frac{n^2}{2} - \frac{n}{2}\right)$, while the optimal set of k vertices induces a subgraph of weight $\frac{k^2}{2} - \frac{k}{2}$. Thus in this case we conclude an integrality gap of $\frac{n}{k}$, which contradicts 'Theorem 1' of [SW98].

4 Max-Cut$_k$

Our third problem is the Max-Cut$_k$ problem. Similar to Section 2 we review an algorithm based on linear programming suggested by [AS99] and show how combining it with semidefinite programming improves the approximation ratio of 1/2. We denote an instance of the Max-Cut$_k$ problem by $G = (V, E)$, w and k. We define Z^*, U as before and $w(U)$ to be the weight of edges cut by the partition $(U, V \setminus U)$. In this case we fix $Opt(G)$ to be the maximal weight of edges cut by a partition $(U, V \setminus U)$ when U is of size k. Since the value $w(U)$ is equal to the value $w(V \setminus U)$ we only consider the case of $k \leq n/2$. A simple reduction from Max-Cut (adding a sufficiently large independent set) establishes the following proposition.

Proposition 4.1 *For any constant $\varepsilon > 0$ there is some constant $c < 1$ such that for every $n^\varepsilon \le k \le n/2$, it is NP-Hard to approximate the Max-Cut$_k$ problem within ratio c.*

4.1 An approximation algorithm based on linear programming

Consider the following linear program for the Max-Cut$_k$ problem.

(LP-Cut)	Maximize	$\sum_{(i,j)\in E} w_{ij} z_{ij}$	
	subject to:		
	(1)	$z_{ij} \le 2 - x_i - x_j$	for all $e_{ij} \in E$
	(2)	$z_{ij} \le x_i + x_j$	for all $e_{ij} \in E$
	(3)	$\sum_{i=1}^n x_i = k$	
	(4)	$0 \le x_i, z_{ij} \le 1$	for $1 \le i \le n$ and $e_{ij} \in E$

Given an optimal fractional solution $\{x_1^* \dots x_n^*\}$ to *(LP-Cut)*, [AS99] suggest a rounding technique which yields a corresponding integer solution of value at least

$$\sum_{e_{ij}\in E} x_i^*(1-x_j^*) + x_j^*(1-x_i^*).$$

It can be seen that for x_i^* and x_j^* in $[0,1]$ we have

$$x_i^*(1-x_j^*) + x_j^*(1-x_i^*) \ge \frac{1}{2} z_{ij}$$

with equality when $x_i^* = x_j^* = 1/2$. These results imply an approximation algorithm with a ratio of $1/2$ for the Max-Cut$_k$ problem. As in the case of Max-VC$_k$ it can be seen that these results are tight due to the integrality gap of relaxation *(LP-Cut)*.

4.2 Approximation algorithms based on semidefinite programming

Goemans and Williamson [GW95], Frieze and Jerrum [FJ97] have studied a semidefinite relaxation to the Max-Cut and Max-Cut$_k$ problems when k is exactly $n/2$. We extend their ideas and beat the approximation ratio of $1/2$ stated earlier for all values of k.

Lemma 4.2 *There exists a BPP algorithm based on semidefinite programming that approximates the Max-Cut$_k$ problem within the ratio of $\alpha_{SDP} \ge \frac{1}{2} + \varepsilon$ for $k \in [\frac{n}{3.1}, \frac{n}{2}]$ and some universal constant $\varepsilon > 0$.*

Proof : Consider the following quadratic integer program.

(QI-Cut)	Maximize	$\sum_{e_{ij}\in E} w_{ij}\left(\frac{1-x_i x_j}{2}\right)$	
	subject to:		
	(1)	$\sum_{i=1}^n x_i = 2k - n$	
	(2)	$x_i \in \{-1, 1\}$	for $1 \le i \le n$

Given a vertex set U of size k, by setting x_i to be of value 1 if and only if the corresponding vertex v_i is in U, we obtain a valid integer solution to *(QI-Cut)*. In this solution an edge e_{ij} cut by the partition $(U, V \setminus U)$ contributes the weight of w_{ij} to the objective function, while the remaining edges contribute nothing. We relax *(QI-Cut)* by the following semidefinite relaxation.

(SDP-Cut) Maximize $\sum_{e_{ij} \in E} w_{ij} \left(\frac{1 - v_i v_j}{2}\right)$

subject to:

$$
\begin{array}{lll}
(1) & \sum_{j=1}^{n} v_i v_j = v_i v_0 (2k - n) & \forall i \in \{0 \ldots n\} \\
(2) & v_i \in S_n & \text{for } 0 \le i \le n \\
(3) & \left.\begin{array}{ll} v_i v_j + v_k v_j + v_k v_i & \ge -1 \\ v_i v_j - v_k v_j - v_k v_i & \ge -1 \end{array}\right\} & \forall i, j, k \in \{0 \ldots n\}
\end{array}
$$

As in the Max-DS$_k$ problem, constraint *(1)* of *(QI-Cut)* is generalized to constraint *(1)* of *(SDP-Cut)*, and as in the Max-VC$_k$ problem a new constraint *(3)* is added.

With a few slight changes we are able to use algorithm A_{SDP-DS} from the previous section and the analysis given in Theorem 3.1 to achieve our assertion (we sketch the proof in Lemma D.1 (the fixing lemma) and Lemma D.2 of the appendix). In particular an approximation ratio of 0.6514 (a bit better than the ratio of 0.6511 presented in [FJ97]) can be achieved for k equal to $n/2$ and an approximation ratio of 0.5856 can be achieved for $k = n/3$.

∎

Independent of our previous result, we now show that the approximation ratio of 1/2 can be improved for all k less than $\frac{n}{3.1}$.

Lemma 4.3 *There exists a BPP algorithm based on semidefinite programming that approximates the Max-Cut$_k$ problem within the ratio of $\alpha_{SDP} \ge \frac{1}{2} + \varepsilon$ for any k less than $\frac{n}{3.1}$ and some universal constant $\varepsilon > 0$.*

Thus we conclude :

Theorem 4.4 *There exists a BPP algorithm that approximates the Max-Cut$_k$ problem within the ratio of $\alpha_{SDP} \ge \frac{1}{2} + \varepsilon$ for every k and some universal constant $\varepsilon > 0$.*

Sketch of Proof (Lemma 4.3) : Consider the following algorithm based on ideas introduced in Theorem 2.2*(c)*.

Given a valid vector configuration $\{v_0 \ldots v_n\}$ to our semidefinite relaxation *(SDP-Cut)* of value Z^*, we construct a valid fractional solution (x, z) to the previous linear relaxation *(LP-Cut)* of value at least Z^* by fixing x_i to be $\frac{1 + v_i v_0}{2}$ for all $i = 1 \ldots n$ and z_{ij} to be $\min(x_i + x_j, 2 - x_i - x_j)$ for all $e_{ij} \in E$. As in Theorem 2.2*(c)* we obtain $x_i \in [0, 1]$, $\sum_{i=1}^{n} x_i = k$ and $\frac{1 - v_i v_j}{2} \le z_{ij}$ for all $e_{ij} \in E$ (the latter due to constraint *(3)* of *(SDP-Cut)*). Thus $(x, z) = (\{x_1 \ldots x_n\}, \{ z_{ij} \mid e_{ij} \in E\})$ is a valid solution to our linear program *(LP-Cut)* of desired value. We conclude that performing the rounding technique mentioned in Section 4.1 on the fractional solution (x, z) we obtain a set U_k of size exactly k with weight at least

$$\frac{1}{2} \sum_{e_{ij} \in E} w_{ij} z_{ij} \ge \frac{1}{2} Z^*.$$

If x_i or x_j differs from $1/2$ then $x_i(1-x_j)+x_j(1-x_i)$ is strictly greater that $\frac{1}{2}z_{ij}$. We now use this fact in order to construct the following approximation algorithm.

Given an instance $G=(V,E)$, w and k of the Max-Cut$_k$ problem, solve the semidefinite relaxation *(SDP-Cut)* to obtain an optimal set of vectors $\{v_0^* \ldots v_n^*\}$ of total value Z^*. Compute the corresponding fractional solution (x^*, z^*) to relaxation *(LP-Cut)* as above. Define an edge e_{ij} to be *good* if at least one of the values x_i^* or x_j^* differs from $1/2$ by at least some small constant value. Now assume that the contribution of these *good* edges to the value Z^* is non-negligible (*i.e.* some constant factor of Z^* is achieved by these edges). Thus rounding our vector solution $\{v_1^* \ldots v_n^*\}$ using (x^*, z^*) computed above and the rounding technique suggested by [AS99], we achieve an approximation ratio strictly greater than $1/2$ on the set of *good* edges and an approximation ratio of at least $1/2$ on the remaining edges. As the contribution of the *good* edges is non-negligible we obtain, in this case, an approximation ratio slightly above $1/2$.

Otherwise the contribution of the *good* edges to the value Z^* is negligible and we may ignore them. Thus we concentrate on the subgraph $H=(V_h,E_h)$ of G, induced by the remaining *bad* edges, and the subset of optimal vectors $\{v_i^* \mid v_i \in V_h\}$ corresponding to the vertices of V_h. Using the *random-hyperplane* rounding technique analyzed in [GW95] we round this subset of optimal vectors achieving a partition $(U_h, V_h \setminus U_h)$ of weight greater than $0.8785Z_h^*$, where Z_h^* is equal to

$$\sum_{e_{ij}\in E_h} \frac{1-v_i^* v_j^*}{2}$$

which is approximately Z^* by our assumption. Without loss of generality we may assume that the size of U_h is less than or equal to half the size of the vertex set V_h. Recall that for each *bad* edge e_{ij} both x_i^* and x_j^* are close to $1/2$, hence the size of V_h is at most a bit above $2k$, and the size of U_h is at most a bit above k. Note that as $k \le \frac{n}{3.1}$, at least k vertices do not belong to V_h. These *leftover* vertices can be added to either side of the partition $(U_h, V_h \setminus U_h)$ without decreasing its weight. If $|U_h|$ is above k we greedily *fix* the size of U_h using Lemma D.1 (from the appendix) and add all *leftover* vertices to the right hand side of the partition $(U_h, V_h \setminus U_h)$. Otherwise we add $k-|U_h|$ arbitrary *leftover* vertices to U_h and the rest to $V_h \setminus U_h$. In both cases we obtain a subset $U \subseteq V$ of size k such that the weight of edges cut by the partition $(U, V\setminus U)$ is *close* to $0.8785Z_h^* \sim 0.8785Z^*$. We allow ourselves to omit the exact proof as it is similar to that of Theorem 2.2*(c)*.

■

5 Max-Uncut$_k$

Our last problem is the Max-UC$_k$ problem. It can be seen that the Max-UC$_k$ problem is *NP-Hard* by a trivial reduction from the Min-Bisection problem shown to be *NP-Hard* in [GJS76]. Recall that Min-Bisection is the problem of finding a partition $(U, V\setminus U)$ in which U is half the size of V and the minimal weight of edges are cut. It is not known whether Min-Bisection can be approximated within a factor arbitrarily close to one in polynomial time. This is a long standing open problem. Note that if Min-Bisection can be approximated within a factor arbitrarily close to one than so can Max-UC$_k$ when $k = n/2$.

Let $G = (V, E)$, w and k be an instance of the Max-UC$_k$ problem. Define the weight $w(U)$ of a vertex set U to be the weight of edges *not* cut by the partition $(U, V \setminus U)$. As in the Max-Cut$_k$ problem we have that $w(U)$ is equal to $w(V \setminus U)$ thus we only consider values of k less than or equal to $n/2$.

The simple random algorithm of picking a subset U in V of size k at random yields the approximation ratio of $1 - \frac{2k(n-k)}{n(n-1)}$. Using semidefinite programming we improve this ratio when k is close to $n/2$. Consider the following quadratic integer program.

(QI-UC)	Maximize	$\sum_{e_{ij} \in E} w_{ij} \left(\frac{1+x_i x_j}{2}\right)$	
	subject to:		
	(1)	$\sum_{i=1}^{n} x_i = 2k - n$	
	(2)	$x_i \in \{-1, 1\}$	for $1 \leq i \leq n$

Given a vertex set U of size k, by setting x_i to be of value 1 if and only if the corresponding vertex v_i is in U, we obtain a valid integer solution to *(QI-UC)*. In this solution an edge e_{ij} *not* cut by the partition $(U, V \setminus U)$ contributes the weight of w_{ij} to the objective function, while the remaining edges contribute nothing. We relax *(QI-UC)* by the following semidefinite relaxation similar to the relaxations presented earlier.

(SDP-UC)	Maximize	$\sum_{e_{ij} \in E} w_{ij} \left(\frac{1+v_i v_j}{2}\right)$	
	subject to:		
	(1)	$\sum_{j=1}^{n} v_i v_j = v_i v_0 (2k - n)$	$\forall i \in \{0 \ldots n\}$
	(2)	$v_i \in S_n$	for $0 \leq i \leq n$

We are now able to use a slight variant of algorithm A_{SDP-DS} from Section 3 and the analysis given in Theorem 3.1 to achieve an approximation ratio above 1/2 for k close to $n/2$. In particular we achieve a ratio of 0.54 when k is equal to $n/2$ (we sketch the proof for $k = n/2$ in Lemma E.1 (the fixing lemma) and Lemma E.2 of the appendix).

6 Summary and concluding remarks

In this paper we have analyzed various algorithmic techniques used in the approximation of four *NP-Hard* maximization problems arising on a given graph $G = (V, E)$ when considering a subset $U \subseteq V$ of restricted size k. In general we have seen, for various values of k, that using semidefinite programming one can obtain an approximation ratio strictly higher than that obtained using linear programming, which in turn improves the trivial ratio achieved by picking a subset U of size k at random. Table 1 summarizes the approximation ratios achieved on our four maximization problems using the algorithmic techniques discussed in our work.

We still do not fully understand how to exploit semidefinite programs as approximation algorithms. The actual numerical bounds that are presented in this paper are not meant to be best possible, but just an indication of the results one can achieve. For none of the four problems could we find examples of problem instances for which the integrality gap of their semidefinite relaxation matches (or nearly matches) the approximation ratio we achieve. In Section A of the appendix the integrality gaps of the semidefinite relaxations used throughout our work are analyzed with and without the addition of triangle inequalities. (The triangle inequalities were used as constraint number *(3)* in the relaxations of Max-VC$_k$ and Max-Cut$_k$, but were not used for the other two problems.)

For the Max-VC$_k$ and Max-Cut$_k$ problems, we present some universal constant $\varepsilon > 0$ such that our algorithms based on semidefinite programming achieve approximation ratios that are at least an additive factor of ε higher than those using linear programming, for all values of k. For Max-DS$_k$ and Max-UC$_k$ we show such results when k is roughly $n/2$, and leave open the question whether these results can be extended to arbitrary small values of k. Possibly, the use of triangle inequalities for these problems can help in obtaining a positive answer.

In some cases, details not appearing in the current paper can be found in [Lan98]. We remark that at the time this work was first done, the results of [AS99] were not yet available, and previous versions of this manuscript used the conventional randomized rounding technique to round linear programs, rather than the cleaner rounding technique of [AS99]. This has no effect on the main results of this paper. Techniques developed more recently [Zwi99, Ye99, HZ01, FL01] provide improved numerical values for the approximation ratios of the problems studied in this paper (with minor modifications to the algorithms).

Acknowledgments

The first author is the Incumbent of the Joseph and Celia Reskin Career Development Chair. This research was supported in part by a Minerva grant, Project 8354.

References

[Asa97] T. Asano. Approximation algorithms for MAX SAT: Yannakakis vs. Goemans-Williamson. *Proceedings of th 5th IEEE Israel Symposium on Theory of Computing and Systems, p. 24-37, 1997.*

[AITT95] Y. Asahiro, K. Iwama, H. Tamaki, T. Tokuyama. Greedily finding a dense subgraph. *Proceedings of th 5th Scandinavian Workshop on Algorithm Theory (SWAT). Lecture Notes in Computer Science, 1097, p. 136-148, Springer-Verlag, 1996.*

[AS99] A.A. Ageev, M.I. Sviridenko. Approximation algorithms for Maximum Coverage and Max Cut with given sizes of parts. *In Proceedings of the Conference on Integer Programming and Combinatorial Optimization (IPCO99), Lecture Notes in Computer Science v.1610, p. 17-30. 1999.*

[Fei94] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of ACM - to appear.*

[FG95] U. Feige, M.X. Goemans. Approximating the Value of Two Prover Proof Systems with Applications to Max-2-SAT and Max-Dicut. *Proceedings of th 3rd IEEE Israel Symposium on Theory of Computing and Systems, p. 182-189, 1995.*

[FJ97] A. Frieze, M. Jerrum. Improved approximation algorithms for Max-k-Cut and Max-Bisection. *Algorithmica 18, p. 67-81, 1997.*

[FKP98] U. Feige, G. Kortsarz, D. Peleg. The Dense k-Subgraph Problem. *To appear in Algorithmica.*

[FL01] U. Feige, M. Langberg. The RPR^2 rounding technique for semidefinite programs. *Manuscript, 2001.*

[FS97] U. Feige, M. Seltser. On the densest k-subgraph problem. *Technical Report CS97-16, Weizmann Institute of Science.*

[GJ79] M.R. Garey, D.S. Johnson. Computers and Intractability. A guide to the Theory of NP-Completeness. *Bell Telephone Laboratories, 1979.*

[GJS76] M.R. Garey, D.S. Johnson, L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science 1, p. 237-267, 1976.*

[GV83] G.H. Golub, C.F. Van Loan. Matrix computation. *The Johns Hopkins University Press - 1983, Baltimore, Md.*

[GW94] M.X. Goemans, D.P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics, 7(4), p. 656-666, 1994.*

[GW95] M.X. Goemans, D.P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of ACM, Vol 42, p. 1115-1145, 1995.*

[Hås97] J. Håstad. Some optimal inapproximability results. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, El Paso, Texas, p. 1-10, 1997.*

[Hoc95] D.S. Hochbaum. Approximation algorithms for NP-Hard problems. *PWS Publishing Company, 1995.*

[HR90] T. Hagerup, C. Rüb. A guided tour of chernoff bounds. *Information processing letters 33(6), p. 305-308, 1990.*

[HZ01] E. Halperin, U. Zwick. Improved approximation algorithms for maximum graph bisection problems. *To appear in IPCO 2001*

[Kar84] N. Karmarkar. A new polynomial-time algorithm for Linear Programming. *Combinatorica 4, p. 373-395, 1984.*

[Kha79] L.G. Khachiyan. A polynomial algorithm for Linear Programming (in Russian). *Doklady Akademiia Nauk USSR 244, 1997.* A translation appears in *Soviet Mathematics Doklady 20, p. 191-194, 1979.*

[KP93] G. Kortsarz, D. Peleg. On choosing a Dense Subgraph. *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, p. 692-701, 1993.*

[KZ97] B. Karloff, U. Zwick. A $(7/8-\varepsilon)$-approximation algorithm for Max-3-SAT? *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science, p. 406-415, 1997.*

[Lan98] M. Langberg. Approximation Algorithms for Maximization Problems arising in Graph Partitioning. *M.Sc. thesis, Weizmann Institute of Science, 1998.*

[NT75] G.L. Nemhauser, W.T. Trotter. Vertex packing: Structural properties and algorithms. *Math. programming 8, p. 232-248, 1975.*

[Pet94] E. Petrank. The hardness of approximation: gap location. *Computational Complexity 4, p. 133-157, 1994.*

[PY91] C.H. Papadimitriou, M. Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences 43, p. 425-440, 1991.*

[RT87] P. Raghavan, C.D. Thompson. Randomized Rounding : A technique for provably good algorithms and algorithmic proofs. *Combinatorica 7 (4), p. 365-374, 1987.*

[Svi98] M.I. Sviridenko. Best possible Approximation Algorithms for Max-SAT with Cardinality Constraint. *Proceedings of International Workshop Approx. '98, p. 193-199, 1998.*

[SW98] A. Srivastav, K. Wolf. Finding Dense Subgraphs with Semidefinite Programming. *Proceedings of International Workshop Approx. '98, p. 181-191, 1998.*
An erratum : A. Srivastav, K. Wolf. Erratum on Finding Dense Subgraphs with Semidefinite Programming. *Preprint, Mathematisches Seminar, Universitaet zu Kiel, 1999.*

[Yan94] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms, 17, p. 475-502, 1994.*

[Ye99] Y. Ye. A 0.699-approximation algorithm for Max-Bisection. *Manuscript, 1999.*

[Zwi99] U. Zwick. Outward rotations: a new tool for rounding solutions of semidefinite programming relaxations, with application to Max-Cut and other problems. *In Proceedings of the 31th ACM Symposium on Theory of Computing, p. 679-687, 1999.*

Appendix

A Integrality gaps

In the following section we present some results regarding the integrality gaps of the relaxations used throughout our work. We denote the constraints $v_iv_j + v_kv_i + v_kv_j \geq -1$ and $v_iv_j - v_kv_i - v_kv_j \geq -1$ (where $i, j, k \in \{0 \dots n\}$) as triangle constraints, and study the integrality gap of our relaxations with and without the addition of these constraints. We present detailed proof for the relaxations *(SDP-VC)* and *(SDP-DS2)*, and sketch our results regarding the relaxations *(SDP-Cut)* and *(SDP-UC)* as they are achieved using similar techniques.

I.G. of *(SDP-VC)* (with triangle constraints):

Proposition A.1 *There are instances of the Max-VC$_k$ problem with $k = n/2$ on which the integrality gap of relaxation* (SDP-VC) *is $\frac{65}{64}$.*

Proof : Consider the graph consisting of two disjoint cliques of size 5. Five vertices cover at most 16 edges, while the value of our semidefinite relaxation on the following symmetric vector configuration is 16.25. For both of the cliques above, set the five vectors corresponding to its five vertices such that one of the vectors is equal to v_0 and the inner product of any two of the five vertices is exactly $-\frac{1}{4}$. ■

I.G. when the triangle constraints are removed from *(SDP-VC)* : Recall that we obtain a $3/4+\varepsilon$ approximation algorithm for the Max-VC$_k$ problem. Following we show that the triangle constraints present in the relaxation *(SDP-VC)* of Section 2 are crucial to our analysis.

Proposition A.2 *For any $k \leq n/2$ there are instances of the Max-VC$_k$ problem on which the integrality gap of relaxation* (SDP-VC) *without triangle constraints is $\frac{3}{2} - \frac{k}{n}$.*

Proof : Consider the complete bipartite graph $(A, B; E)$ in which each side is of size $n/2$ (*i.e.* $|A| = |B| = n/2$). It is not hard to see that the optimal cover with k vertices consists of an independent set of size k thus having the weight of $\frac{nk}{2}$. On the other hand, a valid vector configuration can be achieved by setting all vectors corresponding to vertices of one of the sides, to be equal to a single vector w_1, and setting the remaining vectors to be equal to a different vector w_2 where w_1, w_2, v_0 all lie on the same plane and $\frac{1+w_1v_0}{2} = \frac{1+w_2v_0}{2} = \frac{k}{n}$. With this configuration the angle between w_1 and w_2 is two times the angle between w_1 and v_0, hence $w_1w_2 = 2(w_1v_0)^2 - 1$. We conclude that $u(i,j) = \frac{3+v_iv_0+v_jv_0-v_iv_j}{4} = \frac{3k}{n} - \frac{2k^2}{n^2}$ for every edge e_{ij} yielding a total value of $\frac{3}{4}nk - \frac{k^2}{2}$.

■

I.G. of *(SDP-DS2)* after adding triangle constraints :

Proposition A.3 *There exist instances on which the integrality gap of relaxation* (SDP-DS2) *with the addition of triangle constraints is $\frac{10}{9}$ for any $k \leq \frac{n}{2}$.*

Proof : Consider the graph consisting of five disjoint cliques each of size $\frac{2}{5}k$ and an independent set of size $n-2k$ (we denote such a graph by $5 * K_{\frac{2}{5}k} + IS_{n-2k}$). The optimal subset of size k consists of two and a half cliques and induces a subgraph of weight slightly less than $\frac{9}{20}$ the size of the full edge set. Consider the following valid vector configuration $\{v_0 \dots v_n\}$. For each vertex v_i in the independent set IS_{n-2k} set the vector v_i to be equal to $-v_0$. Divide the remaining vectors into five groups, each group containing the vectors corresponding to the $\frac{2}{5}k$ vertices in one of the five cliques. For each of the five groups set all the vectors in the group to be equal to a single vector $\hat{v}_i$. Spread these five vectors, $\{\hat{v}_1 \dots \hat{v}_5\}$, in a symmetric manner such that $\hat{v}_1$ is equal to v_0 and $\hat{v}_i\hat{v}_j$ is equal to $-1/4$ for $1 \leq i \neq j \leq 5$. With this configuration we achieve a solution of value half the edge set yielding the desired gap.

■

I.G. of *(SDP-DS2)* **:**

Proposition A.4 *There exist instances on which the integrality gap of relaxation* (SDP-DS2) *is at least* $\max(\frac{6}{5}, \frac{n}{2k})$ *for any* $k \leq \frac{n}{2}$.

Proof : The proof of the first result is similar to the above proof when using the graph $3 * K_{\frac{2}{3}k} + IS_{n-2k}$. In this case the corresponding vector configuration does not satisfy the triangle constraints. For the second result, consider the graph consisting of two disjoint cliques each of size $n/2$. The optimal subset with k vertices is of weight $\frac{k^2}{2} - \frac{k}{2}$. On the other hand, a valid vector configuration can be achieved by setting all vectors corresponding to vertices in one of the subgraphs, to be equal to a single vector w_1, and setting the remaining vectors to be equal to a different vector w_2 where w_1, w_2, v_0 all lie on the same plane and $\frac{1+w_1v_0}{2} = \frac{1+w_2v_0}{2} = \frac{k}{n}$. With this configuration $\frac{1+v_iv_0+v_jv_0+v_iv_j}{4} = \frac{k}{n}$ for every edge e_{ij} yielding a total value of approximately $\frac{nk}{4}$.

■

I.G. of *(SDP-Cut)* **(sketch) :** An integrality gap of $\frac{25}{24}$ can be achieved for $k \leq n/2$ using the graph $\left(IS_{\frac{2k}{5}}\right)^5 + IS_{n-2k}$, where IS_{n-2k} is an independent set of size $n-2k$ and $\left(IS_{\frac{2k}{5}}\right)^5$ is the complete five-partite graph on $2k$ vertices in which each independent set is of equal size. The corresponding vector configuration is identical to the one used in Proposition A.3. Set each vector corresponding to a vertex in the independent set IS_{n-2k} to be equal to $-v_0$. Divide the remaining vectors into five groups, each group containing the vectors corresponding to the $\frac{2}{5}k$ vertices in one of the five subgraphs $IS_{\frac{2k}{5}}$. For each of the five groups set all the vectors in the group to be equal to a single vector $\hat{v}_i$. Spread these five vectors, $\{\hat{v}_1 \dots \hat{v}_5\}$, in a symmetric manner such that $\hat{v}_1$ is equal to v_0 and $\hat{v}_i\hat{v}_j$ is equal to $-1/4$ for $1 \leq i \neq j \leq 5$.

I.G. when triangle constraints are removed from *(SDP-Cut)* **(sketch) :** An integrality gap of $\max\left(\frac{9}{8}, 2 - \frac{2k}{n}\right)$ can be achieved in this case. Consider the graphs $\left(IS_{\frac{2k}{3}}\right)^3 + IS_{n-2k}$ and $\left(IS_{\frac{n}{2}}\right)^2$ respectively. The corresponding vector configuration for the first graph is analogous to the one described above. For the second graph we use the vector configuration shown in

Proposition A.2.

I.G. of *(SDP-UC)* with and without the addition of triangle constraints (sketch) : For $k = n/2$ we obtain a gap of $\frac{10}{9}$ (for *(SDP-UC)* with additional triangle constraints) and $\frac{6}{5}$ (for *(SDP-UC)*) by considering the graphs consisting of 5 disjoint cliques each of size $\frac{2k}{5}$, and 3 disjoint cliques each of size $\frac{2k}{3}$ respectively. Our analysis is similar to that of Propositions A.3 and A.4.

B Max-VC$_k$

Proposition B.1 *Algorithm A_{Greedy} yields an approximation ratio of $\alpha \geq 1-(1-k/n)^2$.*

Proof : Let $u_1 \dots u_k$ be the vertices picked by the greedy algorithm and w_i be the weight *added* to the cover at stage i. We start by the following simple observation. Given a graph $G=(V,E)$, there exists a vertex that covers at least the weight of $2w(V)/n$ edges, where $w(V)$ is the total weight of edges in G, and n is the size of V. At stage θ of the greedy algorithm, the vertex u_θ is picked only if it covers the maximal weight of uncovered edges, *i.e.* u_θ covers the maximal weight of edges in the subgraph induced by $V \setminus \{u_1 \dots u_{\theta-1}\}$. Thus u_θ covers at least the weight of

$$\frac{2\left(w(V)-\sum_{i=1}^{\theta-1} w_i\right)}{n-\theta+1}.$$

We now prove by induction that for every θ :

$$\sum_{i=1}^{\theta} w_i \geq \left(1-\left(\frac{n-\theta}{n}\right)^2\right) w(V)$$

It is not hard to see that this inequality is true for $\theta=1$. For general θ we have :

$$\begin{aligned}\sum_{i=1}^{\theta+1} w_i &= \sum_{i=1}^{\theta} w_i + w_{\theta+1} \geq \sum_{i=1}^{\theta} w_i + \frac{2\left(w(V)-\sum_{i=1}^{\theta} w_i\right)}{n-\theta} = \sum_{i=1}^{\theta} w_i\left(1-\frac{2}{n-\theta}\right)+\frac{2w(V)}{n-\theta} \geq \\ &\geq \left(1-\left(\frac{n-\theta}{n}\right)^2\right)\left(1-\frac{2}{n-\theta}\right)w(V)+\frac{2w(V)}{n-\theta} \geq \left(1-\left(\frac{n-\theta-1}{n}\right)^2\right)w(V).\end{aligned}$$

Thus we conclude that $W_{Greedy} = \sum_{i=1}^{k} w_i \geq \left(1-(1-k/n)^2\right) Opt(G)$.

■

Lemma B.2 (Lemma 2.4) *For $k=\frac{n}{c}$ and $c \in (1,2]$, algorithm A_{SDP-VC} returns, with high probability, a vertex set U_k of size k and weight αZ^*_{SDP} with $\alpha > 0.8$.*

Proof : Based on analysis in section 2.3 we define the following functions :

$$\begin{aligned}
\lambda(\mu) &= (\theta\alpha_1 + \alpha_2) - \theta(1-\mu)\frac{c}{c-1} \\
p(\mu) &= \frac{\frac{1}{c} - \mu}{1-\mu} \\
f^+(\mu) &= \frac{1}{c\mu}\lambda(\mu) \\
f^-(\mu) &= 1 - (1-\lambda(\mu))(1-p(\mu))^2
\end{aligned}$$

for fixed $\alpha_1, \alpha_2, \theta$, and $c \in (1,2]$.

By analysis in Lemma 2.4 we have, for $k = \frac{n}{c}$, that with high probability algorithm A_{SDP-VC} yields an approximation ratio arbitrarily close to $\alpha = \min(\alpha_+, \alpha_-)$ where

$$\alpha_+ \geq \min_{\mu \in [\frac{1}{c}, 1]} f^+(\mu), \qquad \alpha_- \geq \min_{\mu \in [0, \frac{1}{c})} f^-(\mu)$$

We start by analyzing the function $f^+(\mu)$. The function $f^+(\mu)$ is equal to

$$\frac{1}{\mu}\left(\frac{\theta\alpha_1 + \alpha_2}{c} - \frac{\theta}{c-1}\right) + \frac{\theta}{c-1}$$

Clearly f^+ in monotone and thus obtains it's minimum in the interval $[1/c, 1]$ at $\mu = 1/c$ or $\mu = 1$, hence

$$\alpha_+ \geq \min\left(\theta\alpha_1 + \alpha_2 - \theta \quad , \quad \frac{1}{c}(\theta\alpha_1 + \alpha_2)\right).$$

The function $f^-(\mu)$ is equal to

$$1 - \left(1 - (\theta\alpha_1 + \alpha_2) + \theta(1-\mu)\frac{c}{c-1}\right)\left(\frac{c-1}{c}\right)^2 \frac{1}{(1-\mu)^2}$$

For simplicity we denote $X = \frac{c}{c-1}$, and $Y = 1 - (\theta\alpha_1 + \alpha_2)$. With this notation we have

$$f^-(\mu) = 1 - \frac{Y + \theta X(1-\mu)}{X^2(1-\mu)^2}$$

By taking the derivative of f^- we have that

$$(f^-(\mu))' = \frac{-2Y}{X^2(1-\mu)^3} - \frac{\theta}{X(1-\mu)^2}$$

Thus f^- has an extremum at $\mu^* = 1 + \frac{2Y}{X\theta}$. Furthermore, for $\frac{2Y}{\theta} \leq -1$ we also have that μ^* is less than $\frac{1}{c}$ and that this extremum is a minimum. Thus for $\frac{2Y}{\theta} \leq -1$ we conclude

$$\min_{\mu \in [0, \frac{1}{c})} f^-(\mu) = f^-(\mu^*) = 1 + \frac{\theta^2}{4Y}$$

By fixing $\theta = 0.69$, $\alpha_1 = 0.976$, and $\alpha_2 = 0.931$ we conclude that $\frac{2Y}{\theta} = -1.752 \leq -1$ thus for any $c \in (1,2]$ algorithm A_{SDP-VC} yields with high probability an approximation ratio of $\alpha(c)$ equal to

$$\min\left(\theta\alpha_1 + \alpha_2 - \theta \ , \ \frac{1}{c}(\theta\alpha_1 + \alpha_2) \ , \ 1 + \frac{\theta^2}{4\left(1 - (\theta\alpha_1 + \alpha_2)\right)}\right)$$

Clearly $\alpha(c)$ is a decreasing function of c, hence we conclude for any $c \in (1,2]$

$$\alpha(c) \geq \alpha(2) = \min(0.9 \ , \ 0.802 \ , \ 0.802) = 0.802$$

■

Lemma B.3 (Theorem 2.2*(c))* *There exists a BPP algorithm based on semidefinite programming that approximates the Max-VC$_k$ problem within the ratio of $\alpha \geq 3/4 + \varepsilon$ for all k and some universal constant $\varepsilon > 0$.*

Proof : Let $G = (V, E)$, w and k be an instance of the Max-VC$_k$ problem. Let Z^* be the optimal value and $\{v_0^* \dots v_n^*\}$ be the optimal vector solution obtained by solving the semidefinite relaxation *(SDP-VC)*. Finally let (x^*, z^*) be the corresponding valid fractional solution to *(LP-VC)* derived from our optimal vector solution, and E_{bad} be the set of edges e_{ij} for which both end point values x_i^* and x_j^* are in the range $[0.435, 0.565]$. Denote by W_{bad} the total weight that these *bad* edges contribute to the value Z^*. Assume that W_{bad} is less than $0.99Z^*$ (*i.e.* the *good* edges contribute some non-negligible portion of Z^*). It can be shown directly from analysis given in [GW94], that if e_{ij} is a *good* edge then $1 - (1 - x_i^*)(1 - x_j^*)$ is of value at least $0.754 z_{ij}$. Thus using [AS99]'s rounding technique on the valid fractional solution (x^*, z^*), we obtain with high probability a set U_k of size exactly k that covers the weight of

$$0.754 \sum_{e_{good}} w_{ij} z_{ij} + 0.75 \sum_{e_{bad}} w_{ij} z_{ij} \geq 0.75 \sum_{e_{ij} \in E} w_{ij} z_{ij} + 0.004 \sum_{e_{good}} w_{ij} z_{ij} \geq 0.75004 Z^*.$$

On the other hand, assume that W_{bad} is at least $0.99Z^*$ and consider the subgraph $H = (V_h, E_h)$ of G induced by the *bad* edges. The vector configuration induced by this subgraph is a valid vector configuration for the Max-VC$_{\hat{k}}$ problem with $\hat{k} = \sum_{v_i \in V_h} \frac{1 + v_i^* v_0^*}{2}$. Note that for all v_i in H we have established $\frac{1 + v_i^* v_0^*}{2} = x_i^* \geq 0.435$, hence we conclude that $\hat{k}$ is at least $0.435|V_h|$. Using algorithm A_{SDP-VC} we round the vectors v_i^* corresponding to vertices in H to obtain a subset $U \subseteq V_h$ with size exactly $\lceil \hat{k} \rceil$ that covers the weight of at least $0.7625 W_{bad}$ (the ratio of 0.7625 is derived using Lemma 2.4 when $c \in [1.75, 2.3]$ and $\theta = 0.85$). Clearly $\lceil \hat{k} \rceil$ is less than or equal to k, thus adding $k - \lceil \hat{k} \rceil$ arbitrary vertices to U we obtain the desired subset $U_k \subseteq V$ covering the weight of at least $0.7625 W_{bad} \geq 0.754 Z^*$.

■

C Max-DS$_k$

Lemma C.1 ([SW98] Fixing lemma) *Given a set U of size $|U| > k$ and weight W we can efficiently find a subset $U_k \subseteq U$ of size k and weight at least $W \frac{k^2}{|U|^2}(1 - 1/k)$.*

Proof : By summing up the weight of edges in the subgraph induced by $U - \{u_i\}$ for all $u_i \in U$, we have that each edge in the subgraph induced by U is counted exactly $|U| - 2$ times. Thus we conclude that

$$\sum_{u_i \in U} w(U - \{u_i\}) = (|U| - 2)w(U)$$

It follows that there exists a vertex $u_i \in U$ such that $w(U - \{u_i\}) \geq w(U)\frac{|U|-2}{|U|}$. Removing this vertex and applying a similar argument inductively, we obtain a subset U_k of size k with weight at least $w(U)\frac{k(k-1)}{|U|(|U|-1)} \geq w(U)\frac{k^2}{|U|^2}(1 - 1/k)$. ∎

Lemma C.2 (Theorem 3.1) *With high probability, algorithm A_{SDP-DS} approximates the problem Max-DS$_k$ within the ratio of $\alpha_{SDP} > 0.517$ when $k = n/2$.*

Proof : Using notations from Theorem 3.1 we have seen that Z is bounded and thus with high probability after step *(3)* of algorithm A_{SDP-DS} we are left with a set $\tilde{U}$ that has a corresponding Z of value at least $(1-\varepsilon)E[Z]$. Let $w(\tilde{U}) = \lambda Z^*_{SDP}$ and $|\tilde{U}| = \mu n$ for some $\lambda \geq 0$ and $\mu \in [0, 1]$. Under these conditions we conclude that for $k = n/2$ (*i.e.* c is equal to 2)

$$\lambda + \theta_1(1-\mu)\frac{c}{c-1} + \theta_2\mu(2/c - \mu) = Z \geq (1-\varepsilon)E[Z] \geq (1-\varepsilon)(\beta + \theta_1\alpha + \theta_2(\alpha(1-1/c)^2 - 1 + 2/c))$$

Hence

$$\lambda \geq (1-\varepsilon)\left(\beta + \theta_1\alpha + \frac{\theta_2\alpha}{4}\right) - 2\theta_1(1-\mu) - \theta_2\mu(1-\mu)$$

Define the following functions :

$$\begin{aligned}
E[Z] &= \beta + \theta_1\alpha + \frac{\theta_2\alpha}{4} \\
\lambda(\mu) &= E[Z] - 2\theta_1(1-\mu) - \theta_2\mu(1-\mu) \\
p(\mu) &= \frac{\frac{1}{2} - \mu}{1 - \mu} \\
f^+(\mu) &= \frac{1}{4\mu^2}\lambda(\mu) \\
f^-(\mu) &= \lambda(\mu)(1 - p(\mu)^2) + p(\mu)^2
\end{aligned}$$

for fixed β, α, θ_1, and θ_2.

Neglecting terms that can be made arbitrarily small such as $(1-\varepsilon)$ and $(1-2/n)$ (the latter arising from the *fixing* lemma) we conclude that if the set $\tilde{U}$ is of size greater than or equal to k, we *fix* the size of $\tilde{U}$ (using Lemma C.1) to obtain a subset U_k of size exactly k and weight at least $\alpha_+ Z^*_{SDP}$ where

$$\alpha_+ \geq \min_{\mu \in [1/2, 1]} f^+(\mu)$$

On the other hand, if the size of $\tilde{U}$ is less than k we must add $k - |\tilde{U}|$ vertices to $\tilde{U}$ to obtain a subset U_k of size exactly k. Let $p = \frac{\frac{1}{c} - \mu}{1-\mu}$. Adding a vertex v_i to $\tilde{U}$ with probability p, independently for every $v_i \notin \tilde{U}$ gives us a new set $\hat{U}$. In this process an edge that is not in the subgraph induced by $\tilde{U}$ will be in the subgraph induced by $\hat{U}$ with probability at least p^2, hence the expected weight of edges in the subgraph induced by $\hat{U}$ is at least $w(\tilde{U}) + (w(V) - w(\tilde{U}))p^2$. The same result (up to an arbitrarily small error) can also be achieved by picking at random a vertex set of size exactly $k - |\tilde{U}|$ from $V \setminus \tilde{U}$, where each vertex set has identical probability. Using the method of conditional expectation over this sample space, we are able to obtain a set U_k of size exactly k and weight at least $\alpha_- Z^*_{SDP}$ where

$$\alpha_- \geq \min_{\mu \in [0,1/2)} f^-(\mu)$$

Fixing θ_1 to be 0.65, and θ_2 to be 3.87 we achieve, through basic analysis, that $\min_{\mu\in[0,1/2)} f^-(\mu)$ is obtained at $\mu^* = 0.3677$ and $\min_{\mu\in[1/2,1]} f^+(\mu)$ is obtained at $\mu^* = 0.7135$. We conclude that α^+ and α_- are at least 0.517. Thus implying an approximation ratio of at least 0.517 for algorithm A_{SDP-DS} when $k = n/2$.

When k is close to $n/2$ similar analysis also guarantees that algorithm A_{SDP-DS} yields an approximation ratio slightly above k/n.

∎

D Max-Cut$_k$

Lemma D.1 (Fixing lemma) *Given a set U of weight W such that $|U| \neq k$ we can efficiently find a set U_k of size k and weight at least γW where $\gamma = \max(\gamma_1, \gamma_2)$ and*

$$\gamma_1 = \begin{cases} \frac{k}{|U|} & |U| \geq k \\ \frac{n-k}{n-|U|} & |U| \leq k \end{cases}, \quad \gamma_2 = \begin{cases} \frac{n-k}{|U|} & |U| \geq n-k \\ \frac{k}{n-|U|} & |U| \leq n-k \end{cases}$$

Sketch of proof : Given a partition $(U, V \setminus U)$ we consider the following three cases. The size of U is less than k, the size of U is between k and $n-k$, and the size of U is larger than $n-k$. In the first case we *fix* the size of U to be k by greedily adding vertices to U. In the second case we use the fact that $w(U)$ is equal to $w(V \setminus U)$ and *fix* the size of U to be k or $n-k$ depending on which causes the least damage. Finally in the last case we *fix* U to be of size $n-k$. Detailed proof of the *fixing* lemma is omitted.

∎

Lemma D.2 (Lemma 4.2) *There exists a BPP algorithm based on semidefinite programming that approximates the Max-Cut$_k$ problem within the ratio of $\alpha_{SDP} \geq \frac{1}{2} + \varepsilon$ for $k \in [\frac{n}{3.1}, \frac{n}{2}]$ and some universal constant $\varepsilon > 0$.*

Sketch of Proof : Using notations from Sections 3.1, 4.2 define the following algorithm $A_{SDP-Cut}$. Solve the relaxation *(SDP-Cut)* obtaining an optimal set $\{v_0^*, \dots v_n^*\}$ of vectors in S_n. Round the optimal set of vectors using the *random hyperplane* rounding technique to obtain a subset U of V. Repeat the above rounding procedure a polynomial number of times and set $\tilde{U}$ to be the subset U with the highest corresponding variable Z, where Z is defined as in Section 3.1

$$Z = \frac{w(U)}{Z^*_{SDP}} + \theta_1 \frac{n-|U|}{n-k} + \theta_2 \frac{|U|(2k-|U|)}{n^2}.$$

Finally fix the size of $\tilde{U}$ to be exactly k by the *fixing* lemma (Lemma D.1).

From analysis given in Section 3.1 we have for $\alpha = 0.8785$ that $E[Z] = \alpha + \theta_1\alpha + \theta_2(\alpha(1 - 1/c)^2 - 1 + 2/c)$ and that with high probability the subset $\tilde{U}$ has a corresponding variable Z of value at least $(1-\varepsilon)E[Z]$ for any constant $\varepsilon > 0$. Let $w(\tilde{U}) = \lambda Z^*_{SDP}$ and $|\tilde{U}| = \mu n$ for some $\lambda \geq 0$ and $\mu \in [0,1]$. Define the following functions

$$\begin{aligned}
E[Z] &= \alpha + \theta_1\alpha + \theta_2(\alpha(1-1/c)^2 - 1 + 2/c) \\
\lambda(\mu) &= E[Z] - \theta_1(1-\mu)\frac{c}{c-1} - \theta_2\mu(2/c - \mu) \\
f_a(\mu) &= \frac{c-1}{c\mu}\lambda(\mu) \quad , \quad f_b(\mu) = \frac{1}{c(1-\mu)}\lambda(\mu) \\
f_c(\mu) &= \frac{1}{c\mu}\lambda(\mu) \quad , \quad f_d(\mu) = \frac{c-1}{c(1-\mu)}\lambda(\mu)
\end{aligned}$$

It is not hard to see, relying on the analysis in Section 3.1 and the *fixing* lemma (Lemma D.1), that algorithm $A_{SDP-Cut}$ yields an approximation ratio of $\alpha = \min(\alpha_a, \alpha_b, \alpha_c, \alpha_d)$ where

$$\alpha_a = \min_{\mu \in [1-1/c, 1]} f_a(\mu) \quad , \quad \alpha_b = \min_{\mu \in [1/2, 1-1/c]} f_b(\mu) \quad , \alpha_c = \min_{\mu \in [1/c, 1/2]} f_c(\mu) \quad , \alpha_d = \min_{\mu \in [0, 1/c]} f_d(\mu).$$

Figure 2 schematically shows the values of c and $|U| = \mu n$ for which each of the functions above are considered.

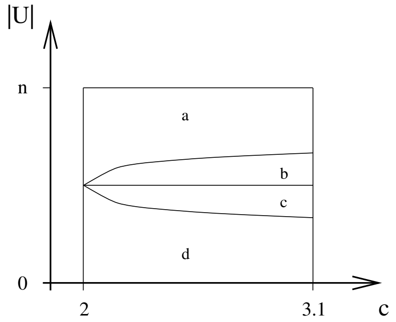

Figure 2: The range relevant to each of the functions f_a, f_b, f_c and f_d when $\mu \in [0,1]$ and $c \in [2, 3.1]$.

Fixing θ_1 to be 0.05 and θ_2 to be 1.3, we conclude using basic calculus that the resulting approximation ratio α is at least 0.55. Note that for specific values of c one can pick different values of θ_1 and θ_2 that yield higher approximation ratios. For example, fixing $\theta_1 = 0$ and $\theta_2 = 3.75$ an approximation ratio of 0.6514 is achieved for $k = n/2$. This slightly improves the ratio of 0.6511 achieved by [FJ97]. In their analysis, Frieze and Jerrum do not introduce the parameters θ_1 and θ_2 used above. Instead they define the random variable Z as

$$Z = \frac{w(U)}{Z^*_{SDP}} + \frac{4|U|(2k - |U|)}{n^2}.$$

The freedom obtained by adding the parameters θ_1 and θ_2 are the cause of our slight improvement. For $k = n/3$ by fixing $\theta_1 = \theta_2 = 0$ a ratio of 0.5856 is achieved.

∎

E Max-Uncut$_k$

Lemma E.1 (Fixing lemma) *Given a set U of weight W such that $|U| \neq k$ we can efficiently find a set U_k of size k and weight at least γW where $\gamma = \max(\gamma_1, \gamma_2)$ and*

$$\gamma_1 = \begin{cases} \dfrac{\binom{|U|-2}{k-2} + \binom{|U|-2}{k}}{\binom{|U|}{k}} & |U| \geq k \\[2ex] \dfrac{\binom{n-|U|-2}{n-k-2} + \binom{n-|U|-2}{n-k}}{\binom{n-|U|}{n-k}} & |U| \leq k \end{cases}, \quad \gamma_2 = \begin{cases} \dfrac{\binom{|U|-2}{n-k-2} + \binom{|U|-2}{n-k}}{\binom{|U|}{n-k}} & |U| \geq n-k \\[2ex] \dfrac{\binom{n-|U|-2}{k-2} + \binom{n-|U|-2}{k}}{\binom{n-|U|}{k}} & |U| \leq n-k \end{cases}$$

Proof : Our proof follows the line of proof given in Lemma D.1. Assume that the set U is of size greater than k and weight W, and that we would like to *fix* the size of U to be exactly k (the other possibilities have similar analysis). Uniformly picking a random subset $U_k \subseteq U$ of size k we have that for an edge e_{ij} *not* in the cut $(U, V \setminus U)$

$$\Pr(e_{ij} \text{ is } \textit{not} \text{ in the new cut } (U_k, V \setminus U_k)) \geq \frac{\binom{|U|-2}{k-2} + \binom{|U|-2}{k}}{\binom{|U|}{k}}.$$

Thus using the method of conditional expectation we are able to obtain a set $U_k \subseteq U$ of size exactly k and weight at least $\gamma_1 W$.

∎

Lemma E.2 *There exists a BPP algorithm based on semidefinite programming that approximates the Max-UC$_k$ problem within the ratio of $\alpha_{SDP} \geq 0.54$ when $k = n/2$.*

Sketch of Proof : Using notations from Sections 3.1, 5 define the following algorithm A_{SDP-UC}. Solve the relaxation *(SDP-UC)* obtaining an optimal set $\{v_0^*, \ldots v_n^*\}$ of vectors in S_n. Round the optimal set of vectors using the *random hyperplane* rounding technique to obtain a subset U of V. Repeat the above rounding procedure a polynomial number of times and set $\tilde{U}$ to be the subset U with the highest corresponding variable Z, where Z is defined as

$$Z = \frac{w(U)}{Z^*_{SDP}} + \theta_1 \frac{|U|(2k - |U|)}{n^2}.$$

Finally fix the size of $\tilde{U}$ to be exactly k by the *fixing* lemma (Lemma E.1).

From analysis given in Section 3.1 we have for $\alpha = 0.8785$ and $c = 2$ that $E[Z] = \alpha + \theta_1\alpha/4$ and that with high probability the subset $\tilde{U}$ has a corresponding variable Z of value at least $(1-\varepsilon)E[Z]$ for any constant $\varepsilon > 0$. Let $w(\tilde{U}) = \lambda Z^*_{SDP}$ and $|\tilde{U}| = \mu n$ for some $\lambda \geq 0$ and $\mu \in [0,1]$. Define the following functions

$$\begin{aligned} E[Z] &= \alpha + \frac{\theta_1 \alpha}{4} \\ \lambda(\mu) &= E[Z] - \theta_1 \mu(1-\mu) \end{aligned}$$

$$f_a(\mu) = \frac{\binom{|U|-2}{k-2} + \binom{|U|-2}{k}}{\binom{|U|}{k}} \lambda(\mu) \quad , \quad f_b(\mu) = \frac{\binom{n-|U|-2}{k-2} + \binom{n-|U|-2}{k}}{\binom{n-|U|}{k}} \lambda(\mu)$$

It is not hard to see that algorithm A_{SDP-UC} yields an approximation ratio of $\alpha = \min(\alpha_a, \alpha_b)$ where

$$\alpha_a = \min_{\mu \in [1/2,1]} f_a(\mu) \quad , \quad \alpha_b = \min_{\mu \in [0,1/2]} f_b(\mu)$$

Fixing θ_1 to be 4.4, we conclude using basic calculus that the resulting approximation ratio is at least 0.5417.

■