Approximation Algorithms for Maximization Problems arising in Graph Partitioning

Thesis for the M.Sc. Degree

by

Michael Langberg

Under the Supervision of
Prof. Uriel Feige
Department of Applied Mathematics
The Weizmann Institute of Science

Submitted to the Feinberg Graduate School of
the Weizmann Institute of Science
Rehovot 76100, Israel

December 3, 1998

Abstract

Given a graph $G = (V, E)$, a weight function $w : E \rightarrow R^+$ and a parameter k we examine a family of maximization problems arising naturally when considering a subset $U \subseteq V$ of size exactly k. Specifically we consider the problem of finding a subset $U \subseteq V$ of size k that maximizes :

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

We present a number of approximation algorithms based on linear and semidefinite programming, and obtain approximation ratios higher than those previously published.

1 Introduction

1.1 Optimization problems and approximation algorithms

Due to the fact that no *NP-Hard* problem can be solved in polynomial time (unless *P=NP*) we are motivated when dealing with such problems to explore the possibility of solving them polynomially in a partial manner. This tradeoff between computational efficiency and the quality of the solution obtained is the essence of *approximation algorithms*. In our work we concentrate on finding *approximation algorithms* for *NP-Hard optimization* problems.

An optimization problem A is a problem in which for each instance x of A we define a set of feasible solutions $F(x)$ and a cost function $c : F(x) \rightarrow R^+$. When solving an instance x of an optimization problem A we aim to find the optimal solution s^* that satisfies

$$c(s^*) = \max_{s \in F(x)} c(s) \quad , \quad c(s^*) = \min_{s \in F(x)} c(s)$$

when A is a maximization or minimization problem respectively.

We say that an algorithm M for a maximization problem A is an r-approximation algorithm, or that M has an approximation ratio of r, if for **every** instance x of A, the ratio between the cost $c(s)$ of the solution s given by M and the cost $c(s^*)$ of the optimal solution s^* of x is at least r. Note that r is not necessarily a constant and may be, for example, a function of the *size* of an instance x. Furthermore, r is always less than or equal to one, and the closer it is to one the better. A similar definition is given when A is a minimization problem.

Over the past decade, much effort has been invested in the study of approximation algorithms for *NP-Hard* problems. While many of these problems share the ability to be solved exhaustively in exponential time, it turns out that they behave in very diverse and intriguing ways when considering approximability. For instance there are *NP-Hard* problems for which there exists polynomial r-approximation algorithms for any constant r less than one, an example to this phenomenon is the well known Knapsack problem. On the other hand, unless *(P=NP)* there is no r-approximation algorithm for the Max-Clique problem for any constant $r > 0$. There are also many problems with less extreme results such as the Max-3SAT problem. Karloff and Zwick [KZ97] have shown the existence of a polynomial approximation algorithm based on semidefinite programming which obtains an approximation ratio of 7/8 on the Max-3SAT problem. This result is of exceptional interest due to the result stated in [Hås97] that it is *NP-Hard* to approximate Max-3SAT beyond the ratio of $7/8 + \varepsilon$ for any constant $\varepsilon > 0$.

One may hope that despite this diverse behavior, there exists some generic algorithmic technique which can be used when trying to approximate *NP-Hard* problems. This is far from being true, as it is usually the case that approximation algorithms for *NP-Hard* optimization problems are designed specially to *fit* a specific problem. Various techniques have been used in the construction of approximation algorithms. In our work we mention algorithms based on combinatorial observations, random algorithms, and algorithms based on linear and semidefinite programming. Following we present the four *NP-Hard* maximization problems analyzed in this thesis.

1.2 Four maximization problems

Given a graph $G = (V, E)$ with $V = \{v_1 \dots v_n\}$, a weight function $w : E \to R^+$, and a parameter k we consider the maximization problem of finding a subset $U \subseteq V$ of size k incident to the maximal weight of edges. We denote this problem as the Max-k-Vertex Cover (Max-k-VC) problem and often refer to the edges incident to the vertex set U as the edges *covered* by U. The Max-k-VC problem has not been dealt with directly in the literature and to the best of our knowledge is mentioned for the first and only time in [Pet94], where it is shown that for some parameter k it is *NP-Hard* to approximate the Max-k-VC problem beyond some fixed constant strictly less than one.

A generalization of Max-k-VC 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-k-VC problem can be viewed as a special case of the Max-k-Coverage problem. A greedy algorithm for the Max-k-Coverage problem is presented in [Hoc95]. This algorithm, that at each stage picks a set that covers the maximal weight of uncovered elements until exactly k sets are picked, yields an approximation ratio of $(1 - 1/e)$. Namely, if the weight covered by the optimal k sets is Z^*, then the k sets obtained by this greedy algorithm cover the weight of at least $(1 - 1/e)Z^*$.

The Max-k-VC problem can also be viewed as a variant of the Max-2SAT problem. A 2SAT formula is a conjunctive normal form boolean formula (usually denoted as a SAT formula) in which each clause contains the disjunction of exactly 2 literals. Given a 2SAT formula Φ and a weight function w associating a positive weight to each clause of Φ, Max-2SAT is the problem of finding an assignment to Φ that maximizes the weight of clauses satisfied. Given an instance of the Max-k-VC problem we build a corresponding 2SAT formula Φ by associating with each vertex $v_i \in V$ a variable x_i and with each edge $e_{ij} = (v_i, v_j)$ of weight w_{ij} a clause $(x_i \vee x_j)$ of identical weight (we denote the value $w(e_{ij})$ by w_{ij}). This correspondence is shown in Figure 1. Note that every literal in Φ appears in a positive manner, hence trivially assigning a *true* value to each of the variables $\{x_1 \dots x_n\}$ in Φ we obtain an assignment that satisfies all clauses of Φ. Thus we consider a variant of the Max-2SAT problem in which we add the restriction that exactly k

variables are assigned a *true* value. We denote this problem as the Max-k-2SAT problem. Given an instance of the Max-k-VC problem $(G = (V, E),\ k,\ w)$ a subset $U \subseteq V$ of size k corresponds to the assignment $\langle \{x_i = true \mid v_i \in U\}\ ,\ \{x_i = false \mid v_i \notin U\}\rangle$ of the formula Φ described above. The weight of clauses satisfied by this corresponding *valid* assignment is exactly the weight of edges covered by U.

Although the Max-k-2SAT problem has not been studied directly, many similar variants have been explored. Goemans and Williamson [GW94], Yannakakis [Yan94] study the general Max-SAT problem in which we seek to find an assignment that maximizes the number of clauses satisfied in a general SAT formula and obtain an approximation ratio of 3/4. This ratio is slightly improved in [GW95, Asa97] using approximation algorithms based on semidefinite programming, and flow techniques respectively. An approximation algorithm for the Max-2SAT problem is also presented in [GW95] yielding the ratio of 0.8785. This ratio was later improved in [FG95] to 0.931. Recently [Svi98] has studied the Max-k-SAT problem of finding an assignment with at most k *true* variables that maximizes the weight of clauses satisfied in a general SAT formula, and achieve an approximation ratio of $(1 - 1/e)$ which is the best possible unless *P=NP* (the latter shown in [Fei94]).

One cannot discuss the Max-k-VC problem without mentioning its close resemblance to the Min-Vertex Cover problem of finding the minimum sized vertex set in G that covers the entire edge set E. Much research has been done regarding the Min-VC problem, as we do not intend to bring a full review on this research we content on mentioning that an approximation algorithm based on linear programming achieves an approximation ratio of 2 (*i.e.* given a graph $G = (V, E)$ we find a subset U that *covers* all edges in E such that U is at most twice the size of the optimal vertex cover), while it is not known if one can beat this ratio by any constant factor.

There are three other maximization problems related to covering edges when a set of k vertices is 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-k-VC problem, one can seek to maximize :

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

Note that the problem Max-k-Cut usually denotes the problem of partitioning a given graph $G = (V, E)$ into k subgraphs such that the weight of edges with endpoints from different subgraphs is maximized. In this work the graph is always cut into two parts, and k denotes the number of vertices in one of these parts. These four problems are illustrated in Figure 1 and all have corresponding presentations that can be stated in terms of boolean formulas as in the case of Max-k-VC described above.

1.3 Previous work and current results

Following we give a brief review on the work previously done regarding the above four maximization problems and state the results obtained in this thesis. In general, for each problem we analyze two approximation algorithms, the first based on linear programming and the second

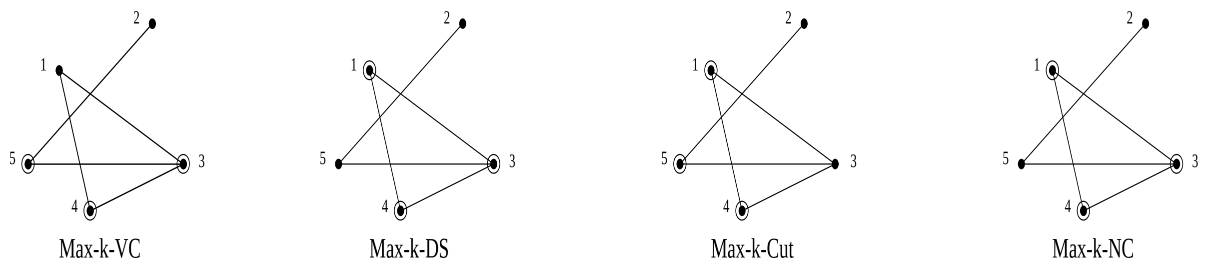

Figure 1: An example of the optimal solution for our four maximization problems with $k = 3$. The boolean formulas corresponding to the above problems are : $(x_1 \vee x_4) \wedge (x_3 \vee x_4) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_5) \wedge (x_3 \vee x_5)$ for Max - k - VC. $(x_1 \wedge x_4) \wedge (x_3 \wedge x_4) \wedge (x_1 \wedge x_3) \wedge (x_2 \wedge x_5) \wedge (x_3 \wedge x_5)$ for Max - k - DS. $(x_1 \oplus x_4) \wedge (x_3 \oplus x_4) \wedge (x_1 \oplus x_3) \wedge (x_2 \oplus x_5) \wedge (x_3 \oplus x_5)$ for Max - k - Cut. $\neg(x_1 \oplus x_4) \wedge \neg(x_3 \oplus x_4) \wedge \neg(x_1 \oplus x_3) \wedge \neg(x_2 \oplus x_5) \wedge \neg(x_3 \oplus x_5)$ for Max - k - NC. Note that all edges are of unit weight.

based on semidefinite programming. Due to the similar nature of our problems, the techniques used in order to analyze these approximation algorithms are identical in all four cases. We show, for various values of k, that the use of semidefinite programming enables us to achieve approximation algorithms with approximation ratios strictly higher than those achieved using linear programming.

Max-k-Vertex Cover : As mentioned above, much work has been done regarding problems similar to the Max-k-VC problem, but little is known about Max-k-VC itself. In this thesis we start by analyzing a known greedy algorithm for the Max-k-Coverage problem presented in [Hoc95] to achieve an approximation ratio of $\alpha_{Greedy} = \max(1 - 1/e, 1 - (1 - k/n)^2)$ on the Max-k-VC problem for all values of k. In addition we prove that this ratio is tight by showing a family of graphs on which the greedy algorithm does no better than stated above. We improve this approximation ratio using linear programming to achieve the ratio of $\alpha_{LP} = 3/4(1 - \varepsilon)$ for all k and any constant $\varepsilon > 0$. Using ideas from [GW95, FG95, FJ97] based on semidefinite programming, this approximation ratio is further improved to the ratio of at least $3/4 + \varepsilon$ for all k and some universal constant $\varepsilon > 0$. In particular for $k \geq n/2$ we achieve an approximation ratio of 0.8. Finally using ideas from [NT75] we obtain an approximation ratio of 0.8 when for a given graph $G = (V, E)$ the size of k is greater than or equal to the size of the minimum vertex cover in G.

Max-k-Dense Subgraph : The Max-k-DS problem has been explored thoroughly during the last past years. Asahiro et al. [AITT95] analyze the greedy algorithm for the Max-k-DS problem that at each stage removes a vertex with the least weight of incoming edges until only k vertices are left, 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 ratio of at least $\max(k/2n, n^{-1/3+\varepsilon})$ for all k and some 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-k-DS problem beyond a ratio of r for some constant $r > 0$. It might even be the case that the existence of such an algorithm is possible only if P=NP.

Using techniques based on linear programming we review a slight variation to an approximation algorithm introduced by Goemans and achieve the approximation ratio of $\alpha_{LP} = \frac{k}{n}(1-\varepsilon)$ for all values of k and any constant $\varepsilon > 0$. 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-k-DS within a ratio better than k/n. In Section 4.2 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-k-Cut : 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 in [GW95]. Frieze an Jerrum use this analysis in [FJ97] when studying the Max-k-Cut 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 we show that for all values of k and any constant $\varepsilon > 0$ an approximation ratio of $\alpha_{LP} = 1/2(1-\varepsilon)$ can be achieved. We improve this ratio using ideas from [GW95, FJ97] based on semidefinite programming to achieve the ratio of at least $1/2 + \varepsilon$ for all k and some universal constant $\varepsilon > 0$. In particular for $n/2$ 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-k-Not Cut : Little is known about the Max-k-NC 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-k-DS problem, the existence of a polynomial approximation algorithm for the Min-Bisection problem with a constant approximation ratio remains open.

We show that the straightforward use of linear programming on the Max-k-NC 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 $k = n/2$.

A few remarks are in place. Define a BPP algorithm as one that runs in random polynomial time and succeeds with overwhelming probability. We show that most of the above algorithms are BPP algorithms. Thus our results all hold with extremely high probability only. In addition, it is the case that for two of our problem (Max-k-VC, Max-k-Cut) we were able to beat the approximation ratio achieved using linear programming by the use of semidefinite programming for all values of k. On the remaining problems our algorithms based on semidefinite programming dominate the ones based on linear programming only for values of k close to $n/2$.

We discuss this point later. Finally, we have been informed that independently of our work, Ageev and Sviridenko [AS98] studied the Max-k-VC and Max-k-Cut problems. They analyze an approximation algorithm based on linear programming and prove approximation ratios of 3/4 and 1/2 respectively, thus avoiding the arbitrarily small additive error of ε that we loose in our analysis. The algorithm used in [AS98] differs from the one we suggest in its *rounding* technique (a shifting argument rather than *randomized rounding*). Semidefinite relaxations for Max-k-VC and Max-k-Cut are not studied in [AS98].

1.4 Structure

In Section 2 we present the tools and techniques used throughout our work. In particular we give a review on linear and semidefinite programming and their use in approximation algorithms. In Sections 3 through 6 we concentrate on the four maximization problems on which this thesis is built, 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 Section 7 we summarize the new and old results stated throughout our work, and discuss problems that still remain open. Finally in the Appendix we present most of the computations done in our work.

2 Preliminaries

Many approximation algorithms for *NP-Hard* optimization problems have been developed using linear and semidefinite programming. In this section we give a brief review on these algorithmic tools concentrating on specific methods used throughout our work.

2.1 Linear programming

Linear programming is the problem of maximizing (or minimizing) a linear objective function subject to a set of linear constraints, usually presented as

(LP)	Maximize	$c^T x$
	subject to:	$Ax \geq b$

where A is a $k \times n$ matrix, the vectors x and c are in R^n and b is in R^k. The use of only one type of inequality (*i.e.* $\geq$) suffices due to the simple fact that by negating a constraint of the form $a^T x \leq b$ we are able to receive the designed format.

The constraints given by $Ax \geq b$ describe a convex polytope in R^n containing all *feasible* points of the linear program *(LP)*. A feasible point in R^n that maximizes the linear objective function $c^T x$ is said to be the optimal solution of *(LP)* and is denoted throughout this work as x^*. Using the ellipsoid algorithm, one can solve a given linear program in polynomial time [Kha79]. This fact enables us to use linear programming in the design of algorithms without worrying about super polynomial time complexity. Since 1979 other polynomial time algorithms solving linear programming have been developed based on interior-point methods. Karmarkar [Kar84] was the first to suggest such an algorithm.

It happens to be the case that many *NP-Hard* optimization problems can be phrased as linear programs that are restricted to integer valued variables. These programs are denoted as *integer programs* (IP). For example, given an instance $G = (V, E)$, k, and $w : E \to R^+$ of the Max-k-VC problem discussed in Section 3, we define the corresponding integer program as :

(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)	$\left.\begin{array}{l} x_i,\ z_{ij} \in [0,1] \\ x_i,\ z_{ij} \text{ are integral} \end{array}\right\}$	for $1 \leq i \leq n$ and $e_{ij} \in E$

Recall that the Max-k-VC problem is the problem of finding a subset $U \subseteq V$ of size k covering the maximal weight of edges. By choosing the vertex v_i to be in the cover U if and only if $x_i = 1$, the above integer program corresponds to the Max-k-VC problem on G and the value of the objective function on the optimal integer solution is exactly the weight of edges covered by the optimal subset $U \subseteq V$.

It is shown in [GJ79] that solving such an integer program is *NP-Hard* even when the variables $\{x_i\}_{i=1}^{n}$ are restricted to the value of 0 or 1 as in *(IP-VC)*. Thus in order to approximately solve a given integer program, under the restriction of polynomial time, the following general approach is often used. Rephrase the given integer program *(IP)* by dropping the integer constraints, thus obtaining a *relaxed* linear program *(LP)*, solve the relaxed program to obtain an optimal *fractional* solution x^*, and finally *round* the fractional solution x^* into a feasible integer solution $\bar{x}$ of *IP*. Note that when dealing with maximization problems the optimal value of the relaxed program *(LP)* denoted as Z^*_{LP}, is no less than the optimal value of the original program *(IP)* denoted as Z^*_{IP}, because every feasible point in *(IP)* is a feasible point of *(LP)*. When dealing with minimization problems we obtain the opposite inequality of $Z^*_{LP} \leq Z^*_{IP}$. As our work mainly deals with maximization problems we restrict our discussion from this point on to these problems only.

It is obvious that an approximation algorithm A based on the above general scheme is determined exclusively by the relaxation *(LP)* and the rounding technique chosen for A. A rounding technique that guarantees $c(\bar{x}) \geq rc(x^*)$, where c is the objective function and $r \in (0, 1]$, implies an approximation ratio of at least r for the above algorithm A :

$$c(\bar{x}) \geq rc(x^*) = rZ^*_{LP} \geq rZ^*_{IP}.$$

Although algorithm A may have an approximation ratio higher than r, it is often the case that the only provable bound on the approximation ratio of A is that achieved by the rounding technique. Two common rounding techniques used are the *threshold* and *randomized rounding* techniques. In the threshold technique we fix some constant between 0 and 1 as our threshold and round a fractional variable x_i to $\lfloor x_i \rfloor$ or $\lceil x_i \rceil$ depending on whether $x_i - \lfloor x_i \rfloor$ is above or below our threshold. The latter rounding technique introduced in [RT87] is used in integer programs which are restricted to the values of 0 and 1 (as in *(IP-VC)*). When using the randomized rounding technique we associate with each variable x_i a probability p_i (for instance $p_i = x_i$ or $p_i = \sqrt{x_i}$), and independently set x_i to be of value 1 with probability p_i and 0 otherwise. If the number of variables x_i set to one is restricted, as in *(IP-VC)*, this rounding technique might result with an integer solution that is not feasible (*i.e.* the number of variables x_i set to one does not satisfy the restriction). In such cases an additional step is added to the initial rounding in order to ensure the feasibility of the resulting integer solution. Throughout our work we use this extended randomized rounding technique on all four maximization problems.

As for the relaxation *(LP)*, in the description of the above approximation scheme the trivial relaxation obtained by dropping the integrality constraints of *(IP)* was considered. Other linear program relaxations may be considered without harming our approximation scheme as long as the relation $Z^*_{LP} \geq Z^*_{IP}$ is preserved. For example, one may obtain a relaxation *(LP)* for the integer program *(IP)* by dropping the integrality constraints and adding additional constraints that hold for integer valued variables. An important property of relaxation *(LP)* is its *integrality gap*. The integrality gap of a relaxation *(LP)* is the ratio between the optimal value Z^*_{LP} of *(LP)* and the optimal value Z^*_{IP} of the original integer program. In general the integrality gap is always larger than one, and the smaller it is the *closer* the relaxation *(LP)* is to the original program *(IP)*.

Let A be an approximation algorithm for an integral program *(IP)* based on the relaxation *(LP)* and rounding technique ρ. We have seen that if $c(\bar{x}) \geq rc(x^*)$, algorithm A has an approximation ratio of at least r. Let α be the integrality gap of relaxation *(LP)*. It is not hard to see that the factor r is less than or equal to $1/\alpha$. For if it is known that the optimal valued fractional solution is α times the optimal valued integral solution, it cannot be the case, given an optimal fractional solution x^* on which the objective function yields the value Z^*_{LP}, that the rounded integer solution $\bar{x}$ will be of value more than $1/\alpha Z^*_{LP} = Z^*_{IP}$. Thus when analyzing such algorithms an effort is made to show that the approximation ratio achieved by ρ (*i.e.* r) is of value $1/\alpha$. In such cases it is clear that our analysis regarding ρ is tight. The integrality gap on one hand and the ratio achieved by the rounding technique on the other are shown in Figure 2. Note that the delicate relation between the two may lead to an approximation algorithm with an approximation ratio strictly larger than that achieved by the rounding procedure.

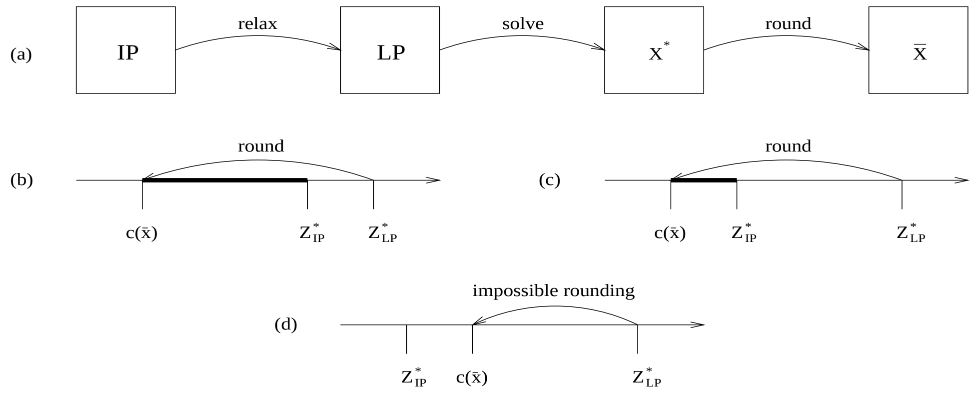

Figure 2: Let Z^*_{IP}, Z^*_{LP} denote the value of the optimal integer and fractional solution to *(IP)* and *(LP)* respectively, and $c(\bar{x})$ denote the value of the solution obtained by the approximation scheme. *(a) :* A schematic review of the approximation scheme. *(b), (c),* and *(d) :* The relation between the values Z^*_{IP}, Z^*_{LP}, and $c(\bar{x})$. In *(b)* the small integrality gap yields a relatively small approximation ratio. In *(c)* a larger integrality gap yields a better approximation ratio. Finally in *(d)* we show an impossible state in which the value of the rounded integer solution $c(\bar{x})$ is super-optimal. This illustrates the fact that the approximation ratio achieved by rounding cannot be larger than $1/\alpha$ for the integrality gap $\alpha = Z^*_{LP}/Z^*_{IP}$. Note that the emphasized line fragment between $c(\bar{x})$ and Z^*_{IP} marks the true approximation ratio in the above cases.

Despite the fact that we do not intend to bring a full survey on the use of linear programming in approximation algorithms, one cannot review such a subject without mentioning the *duality theory* of linear programming. Given a linear program *(LP)* for which we are to maximize the objective function, a dual linear program *(DLP)* can be technically obtained where *(DLP)* is a minimization problem. This primal-dual pair has the property that the value of every feasible solution y to the dual *(DLP)* is greater than or equal to the value of any feasible solution x to the primal program *(LP)*. Furthermore, the value of the optimal solution x^* of *(LP)*, if one exists, is equal to the value of the optimal solution y^* of *(DLP)*. This and various other facts regarding the relationship between a primal and dual linear program are often used in the design of approximation algorithms. As we do not use this important tool in our work, the above discussion is not extended.

2.2 Semidefinite programming

We say than an $n \times n$ symmetric matrix X is *positive semidefinite* if $y^T X y \geq 0$ for every vector $y \in R^n$. It can be seen that a symmetric matrix X is positive semidefinite if and only if there exists an $m \times n$ matrix B for some $m \leq n$ such that $X = B^T B$. 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.

(SDP)	Maximize	$c^T \hat{\mathbf{x}}$
	subject to:	$A\hat{\mathbf{x}} \geq b$
		X is symmetric positive semidefinite.

Where A is a $k \times n^2$ matrix, $\hat{\mathbf{x}} = (x_{11}, x_{12}, \ldots, x_{nn})$ and c are in $R^{(n^2)}$, b is a vector in R^k and $X = \{x_{ij}\}_{i,j=1}^n$ is an $n \times n$ matrix.

Using the decomposition $X = B^T B$ one can correspond a symmetric positive semidefinite $n \times n$ matrix X with n vectors $\{v_1 \ldots v_n\}$ in R^m for some $m \leq n$. This is achieved by corresponding the vector v_i with the i'th column of B, thus obtaining that every entry x_{ij} of X is equal to the inner product $\langle v_i, v_j \rangle$. Hence *(SDP)* can be reformulated using the above equivalence as :

(SDP2)	Maximize	$c^T \hat{\mathbf{x}}$	
	subject to:	$A\hat{\mathbf{x}} \geq b$	
		$v_i \in R^n$	for $1 \leq i \leq n$

Where A, b, c are defined as before, and $\hat{\mathbf{x}}$ is the vector of inner products $(v_1 v_1, v_1 v_2, \ldots, v_n v_n)$ in $R^{(n^2)}$ (we denote the inner product $\langle v_i, v_j \rangle$ by $v_i v_j$). The solution of *(SDP2)* is an optimal set of vectors $\{v_1^* \ldots v_n^*\}$ that maximize the given objective function. We use this formulation of semidefinite programming throughout our work.

Using the ellipsoid method or other recently developed interior point methods, one can solve a semidefinite program formulated as *(SDP)* up to an arbitrarily small additive error in polynomial time. In addition using the incomplete Cholesky decomposition [GV83], one can polynomially obtain an $m \times n$ matrix B from a given symmetric positive semidefinite matrix X such that $X = B^T B$. Thus *(SDP2)* can be solved in polynomial time (up to an arbitrarily small additive error). We now describe how this fact can be used in approximation algorithms for *NP-Hard* problems.

Many *NP-Hard* problems can be phrased as $\{-1, 1\}$ quadratic programs (*i.e.* the problem of maximizing a quadratic objective function subject to a set of quadratic constraints and the constraint that all variables receive the value -1 or 1). For example given an instance $G = (V, E)$, k, and $w : E \rightarrow R^+$ of the Max-k-VC problem discussed in Section 3, we define the corresponding $\{-1, 1\}$ quadratic program as :

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

Notice that for each edge $e_{ij} = (v_i, v_j)$ 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-k-VC problem on G.

A $\{-1, 1\}$ quadratic program *(QIP)* can be viewed as the restricted semidefinite program obtained by replacing each variable x_i by the inner product $v_i v_0$, replacing each pair $x_i x_j$ by the inner product $v_i v_j$, and by adding the restriction that the vectors $\{v_0 \dots v_n\}$ all be of unit length in the one dimensional space R^1. Note that we have added a *new* vector v_0 which as seen below determines the *positive* direction of the vectors $\{v_0 \dots v_n\}$. Due to the fact that the only one dimensional vectors of unit length are the vectors 1 and -1, the correspondence between *(QIP)* and our restricted semidefinite program is obtained by setting each variable x_i to 1 if the vector v_i is equal to the vector v_0 and to -1 otherwise. The restricted semidefinite program corresponding to *(QI-VC)* is given below.

(SDP-VC)	Maximize	$\sum_{e_{ij} \in E} w_{ij} \left(\frac{3+v_iv_0+v_jv_0-v_iv_j}{4} \right)$	
	subject to:		
	(1)	$\sum_{i=1}^{n} v_i v_0 = 2k - n$	
	(2)	$v_i v_i = 1$	for $0 \leq i \leq n$
	(3)	$v_i \in R^1$	for $0 \leq i \leq n$

As it is *NP-Hard* to solve such a restricted program in polynomial time a relaxation of the above is considered by allowing the vectors $\{v_0 \dots v_n\}$ to be of arbitrary dimension (*i.e.* we drop constraint *(3)*). The resulting problem is a semidefinite program solvable (up to an arbitrarily small additive error) in polynomial time, thus an approximation scheme analogous to the one presented in our previous discussion on linear programming is suggested.

Given a $\{-1, 1\}$ quadratic program, phrase the corresponding semidefinite relaxation, solve this relaxation obtaining an optimal vector solution $\{v_0^* \dots v_n^*\}$ and finally round this solution into a valid integral solution to the original quadratic program. As before an approximation algorithm A based on the above scheme is determined by the semidefinite relaxation and the rounding technique chosen for A. Furthermore, our previous discussion regarding the possibility of adding additional constraints to the trivial relaxation described above also holds in this case as long as the additional constraints are satisfied by one dimensional unit vectors. The *integrality gap* of a semidefinite relaxation and its relation to the approximation ratio achieved by a given rounding technique are also as in the case of linear programming. Hence the discussion from the previous section is relevant also to the present semidefinite case.

Goemans and Williamson [GW95] introduce the *random hyperplane* rounding technique for semidefinite programming. Given an optimal vector solution $\{v_0^* \dots v_n^*\}$ to a semidefinite program, we round these vectors into the value -1 or 1 by choosing a random hyperplane passing through the origin and fixing the variable x_i, corresponding to the vector v_i^*, to be 1 if the vectors v_0^* and v_i^* are on the same side of the hyperplane and -1 otherwise. The random hyperplane technique and variations of it are used throughout our work. This technique is illustrated in Figure 3.

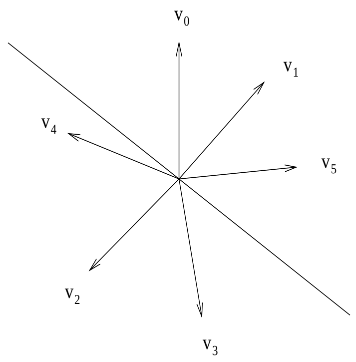

Figure 3: The *random hyperplane* rounding technique applied to a vector configuration $\{v_0 \dots v_5\}$. The resulting values to the variables $x_1 \dots x_5$ are 1,-1,-1,-1,1 respectively.

3 Max-k-Vertex Cover

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

One of the first approximation algorithms that come in mind for the Max-k-VC problem is the simple algorithm that uniformly picks a random subset $U \subseteq V$ of size k. In such a process the expected weight of edges covered by the resulting set U is $\left(1 - (1 - k/n)^2\right) w(V)$ where $w(V)$ is the weight of the entire edge set E. As $Opt(G)$ is at most $w(V)$, this simple algorithm yields an expected approximation ratio of $\alpha_{Random} = 1 - (1 - k/n)^2$.

In this section we state and analyze a number of approximation algorithms for the Max-k-VC problem than improve the trivial approximation ratio stated above. In general, each algorithm presented improves the approximation ratio of previously presented ones for various values of k. As most of our work is on the positive side, we start by showing that Max-k-VC isn't an *easy* problem.

3.1 Hardness of approximation

The Max-k-VC problem is clearly *NP-Hard* which can be seen by a reduction from the Min-VC problem. We show that it is *NP-Hard* to approximate Max-k-VC beyond some fixed constant less than one. This is done in three steps. First we show, for a specific k, that it is *NP-Hard* to distinguish between a graph for which Max-k-VC is the size of the full edge set m, and a graph for which Max-k-VC is at most $(1 - \varepsilon)m$ for some constant $\varepsilon > 0$. Thus for this k it is *NP-Hard* to approximate the Max-k-VC problem beyond a factor of $(1 - \varepsilon)$. Afterwards we present two technical lemmas that give us the full result for all $cn > k > n^{\varepsilon}$, where $c < 1$ and $\varepsilon > 0$ are constants.

Lemma 3.1 *For some $\varepsilon > 0$, it is NP-Hard to distinguish between a graph for which Max-k-VC is the size of the full edge set m, and a graph for which Max-k-VC is at most $(1 - \varepsilon)m$ when $k = \frac{44}{87}n$.*

Proof : Define Max-E3-SAT to be the problem of finding a truth assignment that maximizes the number of clauses satisfied in a SAT formula Φ where each clause in Φ is of length exactly three, and Max-3-SAT(29) to be the problem of finding a truth assignment that maximizes the number of clauses satisfied in a SAT formula where each clause is of length at most three and the number of occurrences of each variable is bounded by 29.

We start our proof from the following result stated in [Hås97] :
For any $\varepsilon > 0$, it is NP-Hard to distinguish satisfiable E3-SAT formulas from E3-SAT formulas for which only a fraction $7/8 + \varepsilon$ of the clauses can be satisfied.

Using a reduction, analyzed in [PY91, Hoc95], from Max-E3-SAT to Max-3-SAT(29), and another reduction, shown in [PY91], from Max-3-SAT(29) to Max-k-VC with $k = \frac{44}{87}n$ we conclude the assertion proposed above. Schematically we are to show

$$\begin{array}{lclcl} \text{Max-E3-SAT}(\Phi) = 1 & \stackrel{\varphi}{\Longrightarrow} & \text{Max-3-SAT(29)}(\varphi(\Phi)) = 1 & \stackrel{\eta}{\Longrightarrow} & \text{Max-}k\text{-VC}(\eta\varphi(\Phi)) = 1 \\ \text{Max-E3-SAT}(\Phi) < 1-\varepsilon & \stackrel{\varphi}{\Longrightarrow} & \text{Max-3-SAT(29)}(\varphi(\Phi)) < 1-\frac{\varepsilon}{43} & \stackrel{\eta}{\Longrightarrow} & \text{Max-}k\text{-VC}(\eta\varphi(\Phi)) < 1-\frac{\varepsilon}{1305} \end{array}$$

Let Φ be an instance of the Max-E3-SAT problem with n variables and m clauses. Using the reduction from [Hoc95] we construct an instance, $\varphi(\Phi)$, of Max-3-SAT(29). Basically $\varphi(\Phi)$ is obtained using special 14-regular *expander* graphs $H_n = (V_n, E_n)$ in which the size of the vertex set V_n is n. These graphs have the property that for every subset $S \subseteq V_n$ of vertices the number of edges between S and its compliment $V_n \setminus S$ is at least $\min(|S|, |V_n \setminus S|)$. An outline of the construction is given below.

For each variable x_i in Φ let m_i denote the number of times x_i appears in Φ. Define m_i new variables $x_i^1 \dots x_i^{m_i}$, and replace the j'th occurrence of x_i by x_i^j. Next to ensure that the optimal assignment in $\varphi(\Phi)$ assigns the same value to $x_i^1 \dots x_i^{m_i}$, add the new clauses $(x_i^k \vee \neg x_i^l)$ and $(x_i^l \vee \neg x_i^k)$ for each edge (k,l) of H_{m_i}. Due to the fact that all clauses in the original formula Φ are of length exactly three, we conclude that the resulting formula $\varphi(\Phi)$ has $N = 3m$ variables and $M = 43m$ clauses, where $42m$ clauses are of length two and m of length three. As seen in [Hoc95], if Φ is satisfiable then $\varphi(\Phi)$ is also satisfiable, and if no assignment is able to satisfy more than $(1-\varepsilon)m$ causes of Φ then in $\varphi(\Phi)$ no assignment is able to satisfy more than $(1-\frac{\varepsilon}{43})M$ clauses. Thus we conclude the first part of our proof.

We now define the reduction η from $\varphi(\Phi)$ into a bounded degree graph $G = (V, E)$ as in [PY91]. The vertex set V has one vertex for every occurrence of a literal in $\varphi(\Phi)$ and thus is of size $87m$. The edge set E includes one edge between each two vertices corresponding to a literal and its negation and one edge between two vertices corresponding to literals in the same clause. Note that the size of the largest independent set in G is equal to the number of clauses satisfied in an optimal assignment to $\varphi(\Phi)$. Set k to be $\frac{44}{87}|V| = 44m$. If $\varphi(\Phi)$ is satisfiable then the largest independent set in G is of size exactly $M = 43m$, thus the smallest vertex cover in G is of size $|V| - 43m = 44m = k$. We conclude that in this case the value of Max-k-VC, for $k = 44m$, is the size of the full edge set E. On the other hand, if no assignment can satisfy more than $(1-\delta)M$ clauses of $\varphi(\Phi)$ then the smallest vertex cover in G is of size at least $|V| - (1-\delta)M = 44m + 43\delta m$. Thus the maximal cover obtained by a vertex set of size k is at most $|E| - 43\delta m$ (otherwise we could add a single vertex to the set for every edge not in the cover and obtain a vertex cover less than $44m + 43\delta m$). The degree of G is bounded by 30, hence we conclude that Max-k-VC is at most

$$|E| - 43\delta m = |E| - \frac{43}{87}\delta|V| \le |E|\left(1 - \frac{43}{87 \cdot 15}\delta\right)$$

$\blacksquare$

Lemma 3.2 *For some constant $\varepsilon > 0$ and every constant $1 > \delta > 0$, it is NP-Hard to distinguish between a graph for which Max-*k*-VC is the size of the full edge set m, and a graph for which Max-*k*-VC is at most $(1-\varepsilon)m$ when $n^\delta \le k \le \frac{44}{87}n$.*

Proof : Adding an independent set of corresponding size gives us the reduction needed from the Max-k-VC problem with $k = \frac{44}{87}n$.

∎

Lemma 3.3 *For some constant $\varepsilon > 0$, it is NP-Hard to distinguish between a graph for which Max-*k*-VC is the size of the full edge set m, and a graph for which Max-*k*-VC is at most $(1-\varepsilon)m$ when $k = \frac{\alpha+c}{\alpha+2}n$, α is an integer, and $c = \frac{43}{87}$.*

Proof : Given an instance $G = (V, E)$, for the Max-cn-VC problem with $c = \frac{43}{87}$, of constant maximal degree Δ and minimal degree one (as in Lemma 3.1) we define a new graph $\hat{G} = (\hat{V}, \hat{E})$ as

$$G + K^{(1)}_{2\alpha\Delta} \times IS^{(1)}_{2\Delta} + K^{(2)}_{2\alpha\Delta} \times IS^{(2)}_{2\Delta} + \ldots + K^{(\frac{n}{2\Delta})}_{2\alpha\Delta} \times IS^{(\frac{n}{2\Delta})}_{2\Delta}$$

where $K_{2\alpha\Delta}$ is a clique of size $2\alpha\Delta$, $IS_{2\Delta}$ is an independent set of size 2Δ, and in the graph $H_1 \times H_2$ every vertex in H_1 is connected to each vertex in H_2 , *i.e.* $\hat{G}$ is the collection of $\frac{n}{2\Delta}$ identical disjoint graphs $K_{2\alpha\Delta} \times IS_{2\Delta}$ and the original graph G. We assume w.l.o.g. that 2Δ and $1/c$ divide n.

Denoting the size of V by n, we have that $|\hat{V}| = N = n(\alpha + 2)$ and $k = \frac{\alpha+c}{\alpha+2}N = \alpha n + cn$. Note that $\hat{G} \setminus G$ has a full vertex cover of size exactly αn. We claim that the maximal coverage in $\hat{G}$ obtained by a vertex set of size k may be achieved when taking exactly αn vertices from the subgraph $\hat{G} \setminus G$. Taking more than αn vertices from $\hat{G} \setminus G$ is wasteful because of the fact that $\hat{G} \setminus G$ has a full cover of size αn. On the other hand, choosing a subset $\hat{U} \subseteq \hat{V}$ with less than αn vertices from $\hat{G} \setminus G$ leaves us with a subgraph $K^{(i)}_{2\alpha\Delta} \times IS^{(i)}_{2\Delta}$, for some i, in which at most $2\alpha\Delta - 1$ vertices are in $\hat{U}$. It is not hard to see that in this case there exists some vertex v_1 in $K^{(i)}_{2\alpha\Delta}$ such that v_1 is not in $\hat{U}$ and adding v_1 to $\hat{U}$ will increase the weight covered by $\hat{U}$ by at least Δ. As the maximal degree in G is Δ, removing any vertex $v_2 \in \hat{U}$ from the subgraph $G \subseteq \hat{G}$ will decrease the weight covered by $\hat{U}$ by at most Δ (there exists such a vertex because $k = \alpha n + cn$). Thus replacing v_1 by v_2 and continuing until there are αn vertices in $\hat{U}$ from $\hat{G} \setminus G$, we achieve the desired result. Denoting the weight of edges covered by the optimal vertex set of size cn in G by $Opt(G)$, the weight covered by the optimal vertex set of size k in $\hat{G}$ by $Opt(\hat{G})$ and the total weight of edges in $\hat{G} \setminus G$ by $w(\hat{G} \setminus G)$ we conclude

$$Opt(\hat{G}) = Opt(G) + w(\hat{G} \setminus G).$$

Thus if the value of $Opt(G)$ is exactly $|E|$ then the value of $Opt(\hat{G})$ is also exactly $|\hat{E}|$. Furthermore, due to the fact that $|E| \geq n/2$, and $|\hat{E}| \leq \frac{n\Delta}{2}(1 + 2\alpha^2 + 4\alpha)$ we conclude that if the value of $Opt(G)$ is at most $(1-\varepsilon)|E|$ then the value of $Opt(\hat{G})$ is at most

$$(1-\varepsilon)|E| + w(\hat{G} \setminus G) = |\hat{E}| - \varepsilon|E| \leq |\hat{E}| \left(1 - \frac{\varepsilon}{\Delta(1 + 2\alpha^2 + 4\alpha)}\right)$$

∎

Using the above lemmas we conclude the following theorem.

Theorem 3.4 *For any constant $\varepsilon > 0$, $c \in (0, 1)$, and $cn > k > n^{\varepsilon}$, it is NP-Hard to approximate the Max-*k*-VC problem beyond some constant factor less than one.*

3.2 The greedy approach

The first algorithm we present is a deterministic, greedy algorithm usually used on the Max-k-Coverage problem. Given a set system $\mathcal{S}$ and a parameter k, the Max-k-Coverage problem is the problem of finding k sets such that the total weight of elements in their union is maximized. The following greedy algorithm analyzed in [Hoc95] for the Max-k-Coverage problem can be applied to the Max-k-VC problem by associating with each vertex v_i in a given graph $G = (V, E)$, the set of edges incident with v_i. We extend the analysis presented in [Hoc95] and show that this greedy algorithm has an approximation ratio of $\alpha_{Greedy} \geq \max(1 - 1/e, 1 - (1 - k/n)^2)$. We also show that there are infinitely many graphs on which the greedy approach yields an approximation ratio no better than $(1 + \varepsilon)\alpha_{Greedy}$ for any constant $\varepsilon > 0$, thus implying that our extended analysis is tight.

3.2.1 Greedy algorithm

$U \leftarrow \phi$
for $i = 1 \ldots k$ do
  select $v \in V \setminus U$ that maximizes the cover gained by $U \cup \{v\}$
  $U \leftarrow U \cup \{v\}$
end.

Proposition 3.5 [Hoc95] *The above algorithm yields an approximation ratio of* $\alpha_{Greedy} > (1 - 1/e)$.

Proof : Let $u_1 \ldots u_k$ be the vertices picked by the greedy algorithm and w_i be the weight *added* to the cover at stage i. At stage θ of the greedy process at least $Opt(G) - \sum_{i=1}^{\theta-1} w_i$ of the total weight not covered by the first $(\theta - 1)$ picked vertices can be covered by the vertices of the optimal cover that have not been chosen yet. From the fact that there are at most k such vertices we conclude the existence of a vertex u_θ that covers the weight of at least $\frac{1}{k}\left(Opt(G) - \sum_{i=1}^{\theta-1} w_i\right)$ new edges. The greedy algorithm picks the vertex with the highest coverage, hence for every $\theta = 1 \ldots k$ we have $w_\theta \geq \frac{1}{k}\left(Opt(G) - \sum_{i=1}^{\theta-1} w_i\right)$. Using this analysis we prove by induction that $\sum_{i=1}^{\theta} w_i \geq \left(1 - (1 - 1/k)^\theta\right) Opt(G)$:

$$\begin{aligned}\sum_{i=1}^{\theta+1} w_i &= \sum_{i=1}^{\theta} w_i + w_{\theta+1} \geq \sum_{i=1}^{\theta} w_i + \frac{Opt(G) - \sum_{i=1}^{\theta} w_i}{k} = (1 - 1/k)\sum_{i=1}^{\theta} w_i + \frac{Opt(G)}{k} \geq \\ &\geq (1 - 1/k)\left(1 - (1 - 1/k)^\theta\right) Opt(G) + \frac{Opt(G)}{k} = \left(1 - (1 - 1/k)^{\theta+1}\right) Opt(G).\end{aligned}$$

We conclude that $W_{Greedy} = \sum_{i=1}^{k} w_i \geq \left(1 - (1 - 1/k)^k\right) Opt(G) > (1 - 1/e)Opt(G)$.

■

Proposition 3.6 *The above algorithm yields an approximation ratio of* $\alpha_{Greedy} \geq 1 - (1 - k/n)^2$.

Proof : 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. Using the notations from the previous proposition we conclude that 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 $W_{Greedy} = \sum_{i=1}^{k} w_i \geq \left(1 - (1 - k/n)^2\right) Opt(G)$.

■

3.2.2 'Hard' graphs

In order to prove tightness of the previous analysis, we present two families of graphs. The first family will be used as instances of the Max-k-VC problem with k one third the size of the given graphs vertex set. For such k we have that $\alpha_{Greedy} \geq \max(1 - 1/e, 1 - (1 - k/n)^2) = 1 - 1/e$. Thus we show that the greedy algorithm, on these graphs, yields an approximation ratio no better than $(1 + \varepsilon)(1 - 1/e)$ for any constant $\varepsilon > 0$. Analogously we use the second family, with k one half the size of the vertex set of the given graphs, to prove that the greedy algorithm on these graphs yields an approximation ratio no better than $(1 + \varepsilon)(1 - (1 - k/n)^2)$.

Let the three-partite graph $G_m = (A_m, B_m, C_m; E_m)$ and $k = m^m$ be an instance of the Max-k-VC problem where each of the three vertex sets are of size k (*i.e.* the size of the full vertex set is $3k$). The vertex set A_m is divided into m sets of vertices each of size $\frac{k}{m}$ and of degree $k(1 - 1/m)^i$ for $i = 0 \dots m - 1$ (note that $k(1 - 1/m)^i$ is an integer). Thus there are $\sum_{i=0}^{m-1} \frac{k^2}{m}(1 - 1/m)^i = k^2\left(1 - (1 - 1/m)^m\right) = (1 + \varepsilon_m)k^2(1 - 1/e)$ edges leaving A_m, all of which enter the vertex set B_m (which is also of size k). The edges between A_m and B_m are spread out in an incremental manner (*i.e.* denoting the vertices of A_m, in decreasing order of degree, as $\{a_1 \dots a_k\}$ and the vertices of B_m as $\{b_1 \dots b_k\}$, we have that the *first* edge leaving a_1 is connected to b_1, the *second* edge leaving a_1 is connected to b_2 and so on. Furthermore, if the last edge connected to a_i is b_j we fix the first edge connected to a_{i+1} as b_{j+1} and continue as before). This way all vertices of B_m have degree exactly $k\,(1-(1-1/m)^m)$. We now wish to turn the degree of all k vertices in B_m to be exactly k, this is achieved by adding edges from B_m to C_m such that the degree of each vertex in C_m is exactly $k(1-1/m)^m$ (recall that C_m is of size k). An example of the graph G_2 obtained by fixing m to be equal to two is given in Figure 4.

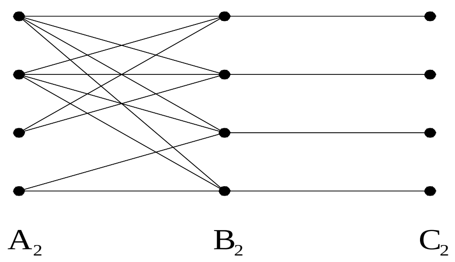

Figure 4: The graph G_2 on which the greedy algorithm yields the subset A_2 of size 4 and weight 12, instead of the optimal subset B_2 of weight 16.

Obviously, the set B_m of size k is an optimal (and full) cover of weight k^2, but the following argument proves that the greedy algorithm can pick the k vertices of A_m as a cover with total weight $k^2\,(1-(1-1/m)^m)$. Let $u_1 \dots u_k$ be the vertices picked by the greedy algorithm, and $a_1 \dots a_k$ be the vertices of A_m in decreasing order of degree. We claim by induction that at stage i, u_i can be picked to be equal to a_i given u_j was picked to be a_j for all $j < i$. The claim in clearly true for $i = 1$. Let $a_i \in A_m$ be the vertex, of degree Δ, picked at stage i. If $a_{i+1} \in A_m$ is of equal degree then a_{i+1} can be picked by the greedy algorithm for the current stage. Otherwise all vertices of degree $\geq \Delta$ in A_m have been picked, meaning that $i = \frac{k}{m}\theta$, and $\Delta = k(1-1/m)^{\theta-1}$ for some $\theta \in \{1 \dots m-1\}$. Furthermore, we know that the maximal degree of unpicked vertices in A_m is $k(1-1/m)^{\theta}$ and that up to now the algorithm has covered $\sum_{i=0}^{\theta-1} \frac{k^2}{m}(1-1/m)^i$ edges. Thus the degree of all vertices in B_m is exactly

$$k - \sum_{i=0}^{\theta-1} \frac{k}{m}(1-1/m)^i = k - k\left(1-(1-1/m)^{\theta}\right) = k(1-1/m)^{\theta}$$

We conclude that picking the vertex a_{i+1} of degree $k(1-1/m)^{\theta}$ from A_m is a valid step for the greedy algorithm. Thus we have established the following proposition.

Proposition 3.7 *There are infinitely many graphs on which the greedy approach yields an approximation ratio no better than* $(1+\varepsilon)\alpha_{greedy} = (1+\varepsilon)(1-1/e)$ *for any constant* $\varepsilon > 0$.

For our second family we define the bipartite graph $H_m = (A_m, B_m; E_m)$ and $k = 2m$ as an instance of the Max-k-VC problem. The vertex sets $A_m = \{a_1 \dots a_{2m}\}$ and $B_m = \{b_1 \dots b_{2m}\}$ are of equal size $2m$, and $E_m = \{\ (a_i, b_j) \mid 1 \leq i \neq j \leq 2m\}$, *i.e.* H_m is a complete bipartite graph missing a perfect matching, and k is of size half the vertex set of H_m. It can be seen that choosing the vertices $\{a_1, b_1, a_2, b_2, \ldots, a_m, b_m\}$ in this order is a valid execution of the greedy algorithm. This execution yields a vertex set of size $2m$ and weight $3m^2 - m$ while the optimal vertex set of size $2m$ covers the weight of $4m^2 - 2m$. Thus we have proven :

Proposition 3.8 *There are infinitely many graphs on which the greedy approach yields an approximation ratio no better than* $(1+\varepsilon)\alpha_{greedy} = (1+\varepsilon)(1-(1-k/n)^2)$ *for any constant* $\varepsilon > 0$.

3.3 An approximation algorithm based on linear programming

Random approximation algorithms based on linear programming have been studied by Goemans and Williamson regarding the Max-2SAT problem in [GW94]. In general, in order to solve an instance of the Max-2SAT problem using linear programming we first phrase and solve a linear relaxation of our problem to obtain an optimal fractional solution. Afterwards we convert the fractional solution into a valid integer solution by some *rounding* technique. The algorithm suggested by [GW94] for the Max-2SAT problem yields an expected approximation ratio of $\frac{3}{4}$. Despite the close nature of the Max-2SAT and Max-k-VC problems, this algorithm cannot be used 'as is' for the Max-k-VC problem due to the fact that the rounding technique used in [GW94] cannot ensure a vertex set of size exactly k. Thus we suggest a slight variant to the algorithm presented in [GW94] to achieve the following theorem. Recall that a *BPP* algorithm is one that runs in random polynomial time and succeeds with overwhelming probability.

Theorem 3.9 *There exists a BPP algorithm based on linear programming that approximates the Max-*k*-VC problem within the ratio of* $\alpha_{LP} \geq \frac{3}{4}(1-\varepsilon)$ *for every* k *and constant* $\varepsilon > 0$.

Proof : 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-k-VC 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 to receive values in the interval $[0,1]$. We denote the relaxed linear program by *(LP-VC)*. Clearly $Z^*_{LP-VC} \geq Z^*_{IP-VC} = Opt(G)$. Consider the following algorithm.

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

1. Solve the relaxation *(LP-VC)*, corresponding to a given graph $G = (V, E)$, to obtain an optimal fractional covering (x^*, z^*).

2. Independently for each vertex $v_i \in V$, set v_i to be in the cover with probability x_i^*. Denote by U the set of vertices picked by this rounding process.

3. If $|U| < k$ arbitrarily add $k - |U|$ vertices to U to obtain a set U_k of size exactly k. If $|U| \geq k$ use the *fixing* lemma stated below to obtain a subset $U_k \subseteq U$ of size k with weight at least $\frac{k}{|U|} w(U)$.

4. Independently repeat steps *(2)* and *(3)* above a polynomial number of times and return the set U_k with the maximal cover.

We now claim that with high probability this random algorithm can achieve the approximation ratio of $\frac{3}{4}(1-\varepsilon)$ stated in Theorem 3.9. This is done in three steps. First we show that the expected weight and size of the set U obtained after the rounding process in step *(2)* fit our demands (*i.e.* the expected size of U is k and the expected weight of U is at least $3/4Opt(G)$). Afterwards we see that with some non-negligible probability the actual size and weight of the set U are very close to these expected values. Finally we see how small disturbances in the size of U can be *fixed* without reducing by much the weight covered by U.

Lemma 3.10 [GW94] *After step (2) of algorithm A_{LP-VC}, we obtain a set U of expected weight at least $\frac{3}{4} Z^*_{LP-VC}$ and expected size exactly k.*

Proof : Relying on the fact that $\frac{1}{2}(a_1 + a_2) \geq \sqrt{a_1 a_2}$, for any nonnegative numbers a_1 and a_2, we conclude that for the optimal fractional solution (x^*, z^*) :

$$1 - (1 - x^*_i)(1 - x^*_j) \geq 1 - \left(\frac{2 - (x^*_i + x^*_j)}{2} \right)^2 \geq 1 - \left(1 - \frac{z^*_{ij}}{2} \right)^2.$$

Since the function $f(z) = 1 - (1 - z/2)^2$ is convex in $[0,1]$ and $f(0) = 0$, $f(1) = 3/4$ we derive that $1 - (1 - x^*_i)(1 - x^*_j) \geq \frac{3}{4} z^*_{ij}$. Note that this inequality is tight only when both x^*_i and x^*_j are equal to $1/2$. This fact will be used later in further algorithms.

We conclude that after step *(2)* of algorithm A_{LP-VC} the expected weight of edges covered by U is at least $\frac{3}{4} Opt(G)$:

$$\begin{aligned} E[w(U)] &= \sum_{e_{ij} \in E} w_{ij} \Pr(e_{ij} \text{ is covered by } U) = \sum_{e_{ij} \in E} w_{ij} \left(1 - (1 - x^*_i)(1 - x^*_j) \right) \geq \\ &\geq \frac{3}{4} \sum_{e_{ij} \in E} w_{ij} z^*_{ij} = \frac{3}{4} Z^*_{LP} \geq \frac{3}{4} Opt(G). \end{aligned}$$

Furthermore, by constraint *(2)* of relaxation *(LP-VC)* we have that

$$E[|U|] = \sum_{i=1}^{n} \Pr(v_i \in U) = \sum_{i=1}^{n} x^*_i = k.$$

■

Using the *chernoff* bound stated below, we conclude that the set U received after rounding the optimal fractional solution (x^*, z^*) in step *(2)* of algorithm A_{LP-VC} is of size at most $k + k^{\frac{2}{3}}$ with high probability (depending on the size of k).

Fact 3.11 (Chernoff Inequality [HR90]) *Let* $x_1 \ldots x_n \in [0,1]$ *such that* $\sum_{i=1}^n x_i = k$, $X_1 \ldots X_n$ *be independent 0-1 random variables with* $\Pr(X_i = 1) = x_i$ *for all* $i = 1 \ldots n$, *and* $S = \sum_{i=1}^n X_i$. *Then for any* $\varepsilon > 0$

$$\Pr(S \geq (1+\varepsilon)k) \leq e^{\frac{-\varepsilon^2 k}{3}} \quad , \quad \Pr(S \leq (1-\varepsilon)k) \leq e^{\frac{-\varepsilon^2 k}{2}}$$

■

Furthermore, due to the fact that the weight of edges covered by U is bounded by $w(V)$ it must be the case for any constant $\varepsilon > 0$ that with some non-negligible probability our set U is of weight at least $(1-\varepsilon)E[w(U)]$. The following lemma justifies this intuition.

Lemma 3.12 *For any constant* $\varepsilon > 0$ *there exists a constant* M_ε *such that if* $k > M_\varepsilon$ *step (2) of algorithm* A_{LP-VC} *yields a set* U *of size less than* $k + k^{\frac{2}{3}}$ *and of weight at least* $(1-\varepsilon)E[w(U)]$ *with probability larger than* $\frac{\varepsilon}{10}$.

Proof : We start by proving a weaker claim for $k > (6\log n)^3$. From the *chernoff* inequality, for k as above, we have that $\Pr(|U| > k + k^{\frac{2}{3}}) < \frac{1}{n^2}$. Let p be the probability that $w(U)$ is smaller than $(1-\varepsilon)E[w(U)]$, from basic probability theory we conclude that

$$E[w(U)] \leq p(1-\varepsilon)E[w(U)] + (1-p)w(V).$$

Furthermore, combining results from Lemma 3.10 and Proposition 3.6 we have that

$$E[w(U)] \geq 3/4Opt(G) \geq 3/4(1-(1-k/n)^2)w(V).$$

Hence

$$\begin{aligned} p &\leq \frac{w(V) - E[w(U)]}{w(V) - (1-\varepsilon)E[w(U)]} = 1 - \frac{\varepsilon E[w(U)]}{w(V) - (1-\varepsilon)E[w(U)]} \leq 1 - \frac{\varepsilon E[w(U)]}{w(V)} \leq \\ &\leq 1 - \frac{3\varepsilon}{4}\left(\frac{2k}{n} - \frac{k^2}{n^2}\right) \leq 1 - \frac{3\varepsilon}{4n}. \end{aligned}$$

Thus with probability at least $\frac{\varepsilon}{2n}$ we obtain a set U with the desired properties. Lemma 3.12 is proven in a similar manner. Full proof is given in Lemma A.1 of the appendix.

■

A set U with properties as stated in Lemma 3.12 will be a feasible solution to the Max-k-VC problem only if $|U| = k$. If $|U| < k$, we fix the size of U by adding $k - |U|$ arbitrary vertices to U (thus the weight $w(U)$ does not decrease). Otherwise we need to remove $|U| - k$ vertices from U such that the weight $w(U)$ is reduced by as little as possible. The following lemma shows that this is possible.

Lemma 3.13 (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* $W\frac{k}{|U|}$.

Proof : By summing up the weight covered by $U - \{u_i\}$ for all $u_i \in U$, we have that each edge covered by U is counted at least $|U| - 1$ times. Thus we conclude that

$$\sum_{u_i \in U} w(U - \{u_i\}) \geq (|U| - 1)w(U)$$

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

■

We now conclude that by independently performing the above process, of randomly picking the vertices of U and then fixing the size of U, a polynomial number of times and returning the set U_k with the heaviest cover we obtain the stated approximation ratio with extremely high probability when k is larger that some constant depending on ε. For k less than the above constant an exhaustive algorithm suffices.

■

3.3.1 Integrality gap

We now show that our analysis is tight, up to an arbitrarily small factor, by studying the integrality gap of relaxation *(LP-VC)*.

Proposition 3.14 *For any $k \leq n/2$ and constant $\varepsilon > 0$ there are instances of the Max-*k*-VC problem on which the integrality gap of relaxation* (LP-VC) *is* $\frac{4}{3}(1 - \varepsilon)$.

Proof : For any $k \leq \frac{n}{2}$ consider the graph $K_{2k} + IS_{n-2k}$ with edges of unit weight (where K_{2k}, IS_{n-2k} denote a clique of size $2k$ and an independent set of size $n - 2k$ respectively). The optimal cover of size k is half of the graph K_{2k}, hence the weight of this cover is $\frac{3k^2}{2} - \frac{k}{2}$. On the other hand, setting $x_i = 0.5$ for all v_i corresponding to vertices of K_{2k} yields a valid fractional solution to *(LP-VC)* of value $2k^2 - k$. Thus we establish a gap of $\frac{4}{3}(1 - O(k^{-1}))$.

■

For $k > \frac{n}{2}$ our previous analysis of the greedy algorithm guarantees that the optimal vertex cover is of size at least $\left(1 - (1 - k/n)^2\right) w(V)$ while any valid fractional solution to *(LP-VC)* is of value at most $w(V)$, hence the gap is smaller.

3.4 Approximation algorithms based on semidefinite programming

Goemans and Williamson [GW95] have studied a semidefinite relaxation to the Max-2SAT problem. Their ideas were improved 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-k-VC. The following theorem is proven.

Theorem 3.15 *For the following values of α_{SDP} and k, there exists a BPP algorithm based on semidefinite programming that approximates the Max-*k*-VC problem within the ratio of α_{SDP}.*

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

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

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

Proof of Theorem 3.15(a) : Given an instance $G = (V, E)$, w and k of the Max-k-VC problem, consider the following quadratic integer program.

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

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-k-VC 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.

3.4.1 The semidefinite relaxation

Consider the following relaxation for the Max-k-VC problem, that is solvable using *semidefinite programming* :

$$
\text{(SDP-VC)} \qquad \begin{array}{lll}
\text{Maximize} & \sum_{e_{ij} \in E} w_{ij} u(i,j) & \\
\text{subject to:} & & \\
(1) & \sum_{i=1}^{n} v_i v_0 = 2k - n & \\
(2) & v_i \in S_n & \text{for } 0 \leq i \leq n \\
(3) & \left. \begin{array}{ll} v_k v_i + v_k v_j + v_i v_j & \geq -1 \\ v_i v_j - v_k v_j - v_k v_i & \geq -1 \end{array} \right\} & \forall i, j, k \in \{0 \ldots n\}
\end{array}
$$

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

$$
u(i,j) = \frac{3 + v_i v_0 + v_j v_0 - v_i v_j}{4}
$$

Without constraint *(3)* relaxation *(SDP-VC)* is obtained from the quadratic integer program *(QI-VC)* by replacing each variable x_i with the inner product $v_i v_0$ and each pair $x_i x_j$ with the inner product $v_i v_j$. In order to assure that the full relaxation *(SDP-VC)*, including constraint *(3)*, still corresponds to the Max-k-VC problem we show that given an instance of Max-k-VC 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 3.15*(b)*.

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].

3.4.2 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 $f(\theta) = \theta + 0.8067(\frac{\pi}{2}(1 - \cos(\theta)) - \theta)$. 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 $\{v_i \mid \mathrm{sgn}(\tilde{v}_i \cdot r) = \mathrm{sgn}(\tilde{v}_0 \cdot r)\}$.

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.

$$\Pr(v_i \notin U) \geq \alpha_1 \left(\frac{1 - v_i v_0}{2} \right), \quad \Pr(e_{ij} \text{ is covered by } U) \geq \alpha_2 u(i,j)$$

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

3.4.3 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. We give a precise definition of *best* later.

4. If $|\tilde{U}| > k$, fix the size of $\tilde{U}$ to be exactly k by the *fixing* lemma (Lemma 3.13). Otherwise, add $k-|\tilde{U}|$ vertices to $\tilde{U}$ using the greedy algorithm from Section 3.2. Denote the resulting set by U_k.

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

3.4.4 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 3.16 *The expected value of Z is at least $\alpha_2 + \alpha_1\theta_k$ with α_1, and α_2 as defined earlier.*

Proof : By linearity of expectation it is suffice 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

$$\begin{aligned} E[|U|] &= \sum_{i=1}^n \Pr(v_i \in U) = \sum_{i=1}^n (1 - \Pr(v_i \notin U)) \leq n - \alpha_1\left(\sum_{i=1}^n \frac{1-v_iv_0}{2}\right) = \\ &= n\left(1-\frac{\alpha_1}{2}\right) + \frac{\alpha_1}{2}\sum_{i=1}^n v_iv_0 = n\left(1-\frac{\alpha_1}{2}\right) + \frac{\alpha_1}{2}(2k-n) = n(1-\alpha_1) + \alpha_1 k. \end{aligned}$$

Which gives us the desired result.

■

As in section 3.3, we prove 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 $(1-\varepsilon)E[Z]$. From analysis of the greedy algorithm (Section 3.2) we know 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

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

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.

Lemma 3.17 *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 : Let $k = \frac{n}{c}$ and $\tilde{U}$ 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.* $\mu \geq 1/c$) applying the *fixing* lemma it follows 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

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

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 3.2. 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 A.2 of the appendix). For $c = 1$ (*i.e.* $k = n$) our algorithm trivially returns the optimal vertex set V.

■■

We now show how the above result 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-k-VC 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 $\frac{1+v_iv_0}{2}$ 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 $v_iv_0 \in [-1, 1]$ we have that $x_i \in [0, 1]$. We also conclude using constraint *(1)* of relaxation *(SDP-VC)* that $\sum_{i=1}^n x_i = \sum_{i=1}^n \frac{1+v_iv_0}{2} = k$. Finally by constraint *(3)* of *(SDP-VC)* we have that $u(i,j) = \frac{3+v_iv_0+v_jv_0-v_iv_j}{4} \leq \min(1, x_i + x_j) = z_{ij}$. These three observations show us that the fractional solution (x, z) is indeed a valid one for *(LP-VC)* of value at least Z^*. Consider the following rounding procedure. Independently set each vertex v_i to be in the cover U with probability x_i, and then *fix* the size of U to be exactly k. Using the discussion above and the analysis of algorithm A_{LP-VC} from the previous section, we conclude that executing steps *(2-4)* of algorithm A_{LP-VC} (*i.e.* repeating this rounding procedure polynomially many times) we obtain a set U_k of size exactly k that covers the weight of at least

$$(1-\varepsilon)\frac{3}{4}\sum_{e_{ij}\in E} w_{ij} z_{ij} \geq (1-\varepsilon)\frac{3}{4} Z^*$$

A closer look at the analysis of algorithm A_{LP-VC} in Lemma 3.10 from the previous section, shows us that an approximation ratio greater than 3/4 can be achieved if for each edge e_{ij} one of the values x_i or x_j from the above fractional solution (x, z) differ from 1/2. 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-k-VC 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 the probabilities x^* computed above and the rounding technique described in steps *(2-4)* of algorithm A_{LP-VC}, we achieve an approximation ratio strictly greater than $3/4$ on the set of *good* edges and an approximation ratio of at least $3/4(1-\varepsilon)$ 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 3.15. 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 *(b)* of Theorem 3.15 stated below (detailed proof of the above is given in Lemma A.3 of the appendix).

Theorem 3.15(b) *There exists a BPP algorithm based on semidefinite programming that approximates the Max-*k*-VC problem within the ratio of $\alpha \geq 3/4 + \varepsilon$ for all k and some universal constant $\varepsilon > 0$.*

The third part of Theorem 3.15 deals with the special case of Max-k-VC in which k is greater than or equal to the minimum vertex cover of a given graph G.

Theorem 3.15(c) *Given a graph G, there exists a BPP algorithm based on semidefinite programming that approximates the Max-*k*-VC 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)	$x_i \in [0, 1]$	for $1 \leq i \leq n$

Nemhauser and Trotter state in [NT75] that 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 $1 \leq i \leq n$. 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-k-VC 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 3.15*(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).

■

3.4.5 Integrality gap

In contrast to the case in the previous section, we were not able to prove the tightness of our analysis by studying the integrality gap of *(SDP-VC)*. Following are a few results that show the existence of a gap strictly larger than one, and the importance of constraint *(3)* in our relaxation. Note that we have shown in Theorem 3.15*(b)* that the integrality gap of our relaxation is at most 4/3 (or to be precise a bit below 4/3).

Proposition 3.18 *There are instances of the Max-*k*-VC problem with $k = n/2$ on which the integrality gap of relaxation* (SDP-VC) *is* $\frac{65}{64}$.

Proof : Consider the graph $K_5 + K_5$ (where K_5 is a clique of size 5). The optimal vertex cover with 5 vertices is of value 16, while the value of our semidefinite relaxation on the following symmetric vector configuration is 16.25. For both of the subgraphs K_5 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 $-\frac{1}{4}$.

■

Proposition 3.19 *For any $k \leq n/2$ there are instances of the Max-*k*-VC problem on which the integrality gap of relaxation* (SDP-VC) *without constraint* (3) *is* $\frac{3}{2} - \frac{k}{n}$.

Proof : Consider the complete bipartite graph $IS_{\frac{n}{2}} \times IS_{\frac{n}{2}}$ (where $IS_{\frac{n}{2}}$ is an independent set of size $\frac{n}{2}$). 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 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 $\frac{1+w_1v_0}{2} = \frac{1+w_2v_0}{2} = \frac{k}{n}$ (see Figure 5). With this configuration the angle between w_1 and w_2 is two times the angle between w_1 and v_0 (modulo 2π), 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 $\frac{3}{4}nk - \frac{k^2}{2}$.

■

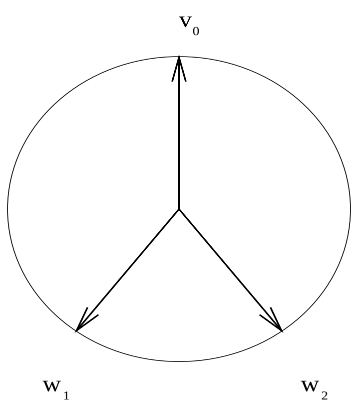

Figure 5: The above vector configuration in which $\frac{1+w_1v_0}{2} = \frac{1+w_2v_0}{2} = \frac{k}{n}$.

4 Max-k-Dense Subgraph

Little is known about the Max-k-DS problem. Clearly it is *NP-Hard* which can be seen by a reduction from the Max-Clique problem, but it is not known whether or not there exists some constant $\varepsilon > 0$ such that approximating the Max-k-DS problem beyond a factor of $(1-\varepsilon)$ is *NP-Hard*. Asahiro et al. [AITT95] analyze the greedy algorithm for the Max-k-DS problem that at each stage removes a vertex with the least weight of incoming edges until only k vertices are left 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 ratio of at least $\max(k/2n, n^{-1/3+\varepsilon})$ for all k and some constant $\varepsilon > 0$. It can also be seen that the simple algorithm of uniformly picking a random subset of k vertices has an expected approximation ratio of $\frac{k(k-1)}{n(n-1)}$.

The approximation ratios presented in [KP93, AITT95, FKP98] are less than $\delta\frac{k}{n}$ for some $\delta \in (0,1)$ when k is linear in n. In this section we present two additional approximation algorithms for the Max-k-DS problem. The first, based on linear programming, is a variation to an approximation algorithm suggested by Goemans achieving an approximation ratio arbitrarily close to k/n for all values of 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 for various values of k. We show a fault in their analysis and present an improved algorithm that fulfills their original claim.

As before we denote an instance of the Max-k-DS 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 $U = \{u_1 \ldots u_k\}$, and $w(U)$ to be the total weight of edges in the subgraph induced by U.

4.1 An approximation algorithm based on linear programming

Following is an integer program for the Max-k-DS problem.

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

By choosing the vertex v_i to be in the set U if and only if $x_i = 1$, the integer program *(IP-DS)* corresponds to the Max-k-DS problem on a given graph G and the optimal integer solution Z^*_{IP-DS} is equal to $Opt(G)$. As before we relax this program by allowing x_i to be in the range [0,1] to achieve the linear relaxation *(LP-DS)*. The following is a variation on an algorithm suggested by Goemans.

Algorithm A_{LP-DS} :

1. Solve the relaxation *(LP-DS)*, corresponding to a given graph $G = (V, E)$, to obtain an optimal fractional solution (x^*, z^*).
2. Independently for each vertex $v_i \in V$, set v_i to be in the cover with probability $\sqrt{x_i^*}$. Denote by U the set of vertices picked by this rounding process.
3. If $|U| < k$ arbitrarily add $k - |U|$ vertices to U to obtain a set U_k of size k, otherwise use the *fixing* lemma stated below to obtain a subset $U_k \subseteq U$.
4. Independently repeat steps *(2)* and *(3)* above a polynomial number of times and return the set U with the *heaviest* induced subgraph.

Note that there are two main differences between this algorithm and algorithm A_{LP-VC} presented in the previous section. The values $\{x_1^* \dots x_n^*\}$ obtained by solving relaxation *(LP-DS)* are not used directly as probabilities, but are changed first to achieve

$$\Pr(e_{ij} \text{ is in the subgraph induced by } U) = \sqrt{x_i^*}\sqrt{x_j^*} \geq z_{ij}$$

By the Cauchy Schwartz inequality we conclude that the expected size of the set U received from the rounding, in step *(2)* above, is at most $\sqrt{nk}$ (instead of being exactly k as in the previous section). Furthermore, *fixing* the size of U in step *(3)* above may cost us a quadratic factor instead of the linear factor paid when *fixing* the size of a set in the Max-k-VC problem. Changing the analysis given in Section 3.3 slightly to adapt to these differences, and using the fact that for $k < n^{1/3}$ picking a subset U with at least one edge yields an algorithm with an approximation ratio of k/n, we conclude the following proposition.

Proposition 4.1 (Goemans) *There exists a BPP algorithm based on linear programming that approximates the Max-*k*-DS problem within the ratio of* $\alpha_{LP} \geq \frac{k}{n}(1 - \varepsilon)$ *for every* k *and constant* $\varepsilon > 0$.

We now state the *fixing* lemma from [SW98] for the Max-k-DS problem.

Lemma 4.2 (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 $w(U)\frac{k(k-1)}{|U|(|U|-1)} \geq w(U)\frac{k^2}{|U|^2}(1 - 1/k)$. ■

We show that the above analysis is tight by considering the complete graph on n vertices K_n. An optimal subset of size k has weight $\frac{k^2}{2} - \frac{k}{2}$. On the other hand, setting $x_i = \frac{k}{n}$ for all i yields a valid fractional solution to *(LP-DS)* of value $\left(\frac{n^2}{2} - \frac{n}{2}\right)\frac{k}{n}$. Thus we establish an integrality gap of $\frac{n-1}{k-1} \geq \frac{n}{k}$.

■

4.2 An approximation algorithm based on semidefinite programming

Using ideas from [GW95, FJ97, FS97, SW98] we analyze a semidefinite relaxation to the Max-k-DS 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 4.3 *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-k-DS problem can be achieved by a BPP algorithm based on semidefinite programming.*

Proof : Consider the following quadratic integer program.

(QI-DS)

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

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)

$$\begin{array}{lll} \text{Maximize} & \sum_{e_{ij}\in E} w_{ij}\left(\frac{1+v_iv_0+v_jv_0+v_iv_j}{4}\right) & \\ \text{subject to:} & & \\ (1) & \sum_{i=1}^{n} v_iv_0 = 2k-n & \\ (2) & v_i \in S_n & \text{for } 0 \leq i \leq n \end{array}$$

As in Section 3.4, *(SDP-DS1)* is obtained from the quadratic integer program *(QI-DS)* by replacing each variable x_i with the inner product v_iv_0 and each pair x_ix_j with v_iv_j. For reasons that will be discussed shortly, we define the following *improved* relaxation.

(SDP-DS2)

$$\begin{array}{lll} \text{Maximize} & \sum_{e_{ij}\in E} w_{ij}\left(\frac{1+v_iv_0+v_jv_0+v_iv_j}{4}\right) & \\ \text{subject to:} & & \\ (1) & \sum_{j=1}^{n} v_iv_j = v_iv_0(2k-n) & \forall i \in \{0\ldots n\} \\ (2) & v_i \in S_n & \text{for } 0 \leq i \leq n \\ (3) & \left.\begin{array}{ll} v_kv_i + v_kv_j + v_iv_j & \geq -1 \\ v_iv_j - v_kv_j - v_kv_i & \geq -1 \end{array}\right\} & \forall i,j,k \in \{0\ldots n\} \end{array}$$

Notice that there are two differences 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 $i \in \{0 \dots n\}$. 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]. The second difference between the two relaxations is the additional constraint *(3)* of *(SDP-DS2)*. This constraint is not used directly in the analysis of our upcoming algorithm, but we suspect that its inclusion improves the integrality gap of our relaxation.

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 $(1, 0, \dots, 0)$, the vector v_i with the vector $(1, 0, \dots, 0)$ if the vertex v_i is in U and $(-1, 0, \dots, 0)$ otherwise. With these values for $\{v_0, \dots v_n\}$ 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 $\{v_0^* \dots v_n^*\}$. 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 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 set U obtained by the above rounding procedure cannot guarantee these properties, but [GW95] have shown that

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

where $\alpha = 0.8785$ and $\beta = 0.796$. 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 $\{v_0^*, \dots v_n^*\}$ 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 *best* solution found, where *best* is defined as in Section 3.4.4.

4. If $|\tilde{U}| > k$, fix the size of $\tilde{U}$ to be exactly k by the *fixing* lemma. Otherwise, *add* $k - |\tilde{U}|$ vertices to $\tilde{U}$. This *addition* is not arbitrary and will be discussed later. Denote the resulting set by U_k.

Let Z^*_{SDP} be the optimal value and $\{v_0^* \dots v_n^*\}$ 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 3.4.4 and guarantee that the expected value Z is at least some normalized sum of the expected weight and size of our subset U. It can be seen that relaying on the expected weight and size of U alone, the same line of analysis used in Section 3.4.4 will not beat the approximation ratio of $\frac{k}{n}$. Thus we consider the second moment of $|U|$ and add the last component $|U|(2k - |U|)$ of Z, which is maximal when $|U|$ is exactly k. We now analyze the expected value of Z.

Lemma 4.4 *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 the following bounds regarding the *random hyperplane* rounding technique shown in [GW95].

$$\Pr(v_i \in U) \ge \alpha\left(\frac{1 + v_i v_0}{2}\right) \quad , \quad \Pr\left(v_i \text{ and } v_j \text{ are } \textit{separated}\right) \ge \alpha\left(\frac{1 - v_i v_j}{2}\right)$$

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 3.16) 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

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

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

$$\sum_{i,j} v_i v_j = \sum_i \sum_j v_i v_j = (2k - n)\sum_i v_i v_0 = (2k-n)^2.$$

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 $\varepsilon > 0$, repeating our rounding scheme a polynomial number of times we obtain a subset U such that its corresponding Z is of value at least $(1-\varepsilon)E[Z]$. Repeating Lemma 3.17 from the previous section with small changes, we prove our theorem (detailed analysis for the case $k = n/2$ is given in Lemma B.1 of the appendix).

■

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 examples contradict this claim and show 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 $\{v_1 \ldots v_n\}$ are equal to a single vector w such that w is perpendicular to v_0 (see Figure 6(a)). 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 4.4.

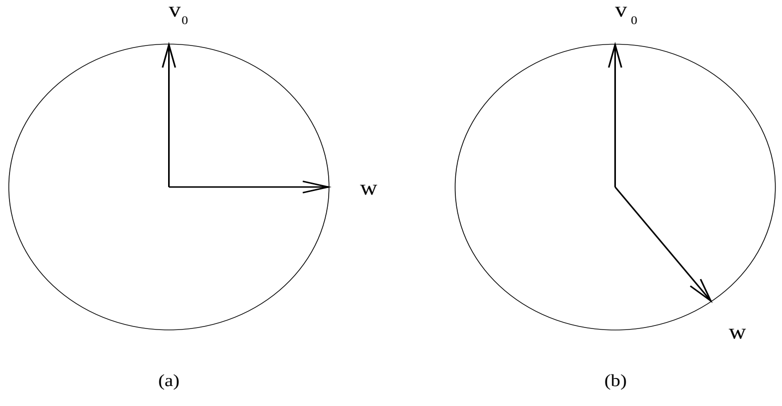

Figure 6: An integrality gap of value n/k is achieved for relaxation *(SDP-DS1)* on the complete graph K_n using the above vector configurations for *(a)* $k = n/2$ and *(b)* general k. In both cases $v_0 w = \frac{2k}{n} - 1$.

For general k, by setting the vector w such that the inner product $v_0 w$ is exactly $\frac{2k}{n} - 1$ (as in Figure 6(b)) 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]. The fault lies in the proof of 'Theorem 1' and results from the mistaken assumption that the subset U obtained by step *(2)* of algorithm A_{SDP-DS} is of size greater than αk **and** has a corresponding variable Z of value at least $(1-\varepsilon)E[Z]$ with non-negligible probability. Although these two events individually happen with non-negligible probability, there are instances of the Max-k-DS problem in which these events are disjoint.

Before analyzing the integrality gap of our original relaxation *(SDP-DS2)*, note that in Section 3.4, we were able to take advantage of the improved approximation ratio for the Max-k-VC problem using semidefinite programming, in order to beat the ratio of 3/4 achieved by linear programming for all values of k. This was done by using the fact that our semidefinite algorithm was better than the algorithm based on linear programming A_{LP-VC} in the case in which A_{LP-VC} was at its worst. In the current Max-k-DS problem, algorithm A_{LP-DS} is at its worst when all values of the fractional solution are equal (*i.e.* for all i, $x_i = k/n$). Unfortunately, we have not been able to show that running algorithm A_{SDP-DS} on a corresponding vector configuration will yield an approximation ratio better than k/n.

4.2.1 Integrality gap

We now show that the integrality gap of relaxation *(SDP-DS2)* is strictly larger than one and bounded by approximately n/k. In addition, we analyze the gap when constraint *(3)* is removed from our relaxation.

Proposition 4.5 *The integrality gap of relaxation* (SDP-DS2) *is at most* $\frac{n}{k}(1+\varepsilon)$ *for any constant* $\varepsilon > 0$.

Proof : Given an instance $G = (V, E)$, w and k of the Max-k-DS problem we use the standard technique described in the proof of Theorem 3.15*(b)* in order to show that every vector solution to the semidefinite relaxation *(SDP-DS2)* of value Z_{SDP} can be converted into a fractional solution to our linear relaxation *(LP-DS)*, from Section 4.1, of value at least Z_{SDP}. Let $\{v_0 \dots v_n\}$ be a valid vector solution to *(SDP-DS2)*, for $i \in \{1 \dots n\}$ set x_i to be $\frac{1+v_i v_0}{2}$ thus constructing a valid fractional solution to *(LP-DS)*. Due to constraint *(3)* of *(SDP-DS2)* we have that $\frac{1+v_i v_0 + v_j v_0 + v_i v_j}{4} \leq \min(x_i \ , \ x_j)$. We conclude that the optimal value of our semidefinite relaxation is at most the optimal value of our linear relaxation. Previous analysis shows that there exists a vertex set of size k with an induced subgraph of weight at least $\frac{k}{n}(1-\varepsilon)$ times the optimal value of our linear relaxation for any constant $\varepsilon > 0$. Thus our proposition is proven.

■

Proposition 4.6 *There exist instances on which the integrality gap of relaxation* (SDP-DS2) *is* $\frac{10}{9}$ *for any* $k \leq \frac{n}{2}$.

Proof : Consider the graph $5 * K_{\frac{2}{5}k} + IS_{n-2k}$ consisting of five cliques each of size $\frac{2}{5}k$ and an independent set of size $n-2k$. 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 $\{v_0 \ldots v_n\}$. 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, $\{\hat{v}_1 \ldots \hat{v}_5\}$, 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.

■

Proposition 4.7 *There exist instances on which the integrality gap of relaxation* (SDP-DS2) *without constraint* (3) *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 constraint *(3)* of our relaxation. For the second result, consider the graph $K_{\frac{n}{2}} + K_{\frac{n}{2}}$ consisting of two equally sized cliques. The optimal subset with k vertices is of weight $\frac{k^2}{2} - \frac{k}{2}$. On the other hand, a valid vector configuration to the new relaxation 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 $\frac{1+w_1v_0}{2} = \frac{1+w_2v_0}{2} = \frac{k}{n}$ (as in Figure 5). 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 $\frac{nk}{4}$.

■

5 Max-k-Cut

Our third problem is the Max-k-Cut problem. Similar to the previous sections we show that it is *NP-Hard* to approximate this problem beyond some constant factor less than one, and we present two approximation algorithms based on linear and semidefinite programming. We denote an instance of the Max-k-Cut 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. Note that the value $w(U)$ is equal to the value $w(V \setminus U)$ hence we only consider the case of $k \leq n/2$.

5.1 Hardness of approximation

Theorem 5.1 *For any constant $\varepsilon > 0$ and $n/2 \geq k > n^\varepsilon$, it is NP-Hard to approximate the Max-*k*-Cut problem beyond some constant factor less that one.*

Proof : Given a graph $G = (V, E)$ and a weight function $w : E \to R^+$, Max-Cut is the problem of finding a subset $U \subseteq V$ of arbitrary size such that the weight of edges cut by the partition $(U, V \setminus U)$ is maximal. In [PY91] it is shown that Max-Cut is *Max-SNP* complete and thus *NP-Hard* to approximate beyond some constant factor less than one. Let $G = (V, E)$, w be an instance of the Max-Cut problem with $|V| = n$. Consider the graph $\hat{G} = G + IS_N$ where IS_N is an independent set of size N and $N \geq n$. For any k between $\frac{n}{2}$ and $\frac{N+n}{2}$, it is not hard to see that if the Max-k-Cut problem on $\hat{G}$ can be approximated beyond any constant factor less than one in polynomial time then so can the Max-Cut problem on G. As N can be polynomial in n, we conclude the assertion stated in our theorem.

■

5.2 An approximation algorithm based on linear programming

Theorem 5.2 *There exists a BPP algorithm based on linear programming that approximates the Max-*k*-Cut problem within the ratio of $\alpha_{LP} \geq \frac{1}{2}(1 - \varepsilon)$ for every k and constant $\varepsilon > 0$.*

Proof : Consider the following integer program for the Max-k-Cut problem.

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

By choosing the vertex v_i to be in the vertex set U if and only if $x_i = 1$, the integer program *(IP-Cut)* corresponds to the Max-k-Cut problem on a given graph G and the optimal integer solution Z^*_{IP-Cut} is equal to $Opt(G)$. We relax this program by allowing x_i to be in the range

[0,1] to achieve the linear relaxation *(LP-Cut)*. Solving the linear relaxation *(LP-Cut)* to obtain an optimal fractional solution (x^*, z^*) and independently picking a vertex v_i to be in the set U with probability x_i^*, it is not hard to see that

$$Pr(e_{ij} \text{ is cut}) = x_i^*(1 - x_j^*) + x_j^*(1 - x_i^*) \geq \frac{1}{2} z_{ij}^*$$

Thus we obtain a subset U of V with expected size k and expected weight $\frac{1}{2} Z_{LP}^*$. Using the algorithmic techniques and analysis described in the previous sections with a corresponding *fixing* lemma we obtain the approximation ratio stated above.

Note that in contrast to the previous problems, when dealing with Max-k-Cut, we must also use the *fixing* lemma when our rounding technique results with a set U of size smaller than k, for adding arbitrary vertices to U in this case may reduce the weight of U.

Lemma 5.3 (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.

■■

In order to prove that the analysis of the rounding technique described above is tight, up to an arbitrarily small factor, consider the graph $K_{2k} + IS_{n-2k}$ (we assume $k \leq n/2$). The optimal partition $(U, V \setminus U)$ with U of size k is achieved by setting U to be half the graph K_{2k}, hence the optimal cut is of value k^2. On the other hand, setting x_i to be 0.5 for all x_i corresponding to the vertices in K_{2k} yields a fractional solution of value $2k^2 - k$. Thus we establish an integrality gap of $2 - 1/k$.

5.3 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-k-Cut 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 5.4 *There exists a BPP algorithm based on semidefinite programming that approximates the Max-*k*-Cut 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 $\varepsilon > 0$.*

Proof : Consider the following quadratic integer program.

(QI-Cut)	Maximize	$\sum_{e_{ij}\in E} w_{ij}\left(\frac{1-x_ix_j}{2}\right)$	
	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	$\sum_{e_{ij}\in E} w_{ij}\left(\frac{1-v_iv_j}{2}\right)$	
	subject to:		
	(1)	$\sum_{j=1}^{n} v_iv_j = v_iv_0(2k-n)$	$\forall i \in \{0\dots n\}$
	(2)	$v_i \in S_n$	for $0 \le i \le n$
	(3)	$\left.\begin{array}{ll} v_kv_i+v_kv_j+v_iv_j & \ge -1 \\ v_iv_j-v_kv_j-v_kv_i & \ge -1 \end{array}\right\}$	$\forall i,j,k \in \{0\dots n\}$

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

With a few slight changes we are able to use algorithm A_{SDP-DS} from the previous section and the analysis given in Theorem 4.3 to achieve our assertion (we sketch the proof in Lemma C.1 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 5.5 *There exists a BPP algorithm based on semidefinite programming that approximates the Max-*k*-Cut 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 5.6 *There exists a BPP algorithm that approximates the Max-*k*-Cut 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 5.5) : Consider the following algorithm based on ideas introduced in Theorem 3.15*(b)*.

Given a valid vector configuration $\{v_0 \dots v_n\}$ 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 $\frac{1+v_iv_0}{2}$ for all $i = 1\dots n$ and z_{ij} to be

$\min(x_i + x_j, 2 - x_i - x_j)$ for all $e_{ij} \in E$. As in Theorem 3.15*(b)* we obtain $x_i \in [0,1]$, $\sum_{i=1}^n x_i = k$ and $\frac{1-v_i v_j}{2} \leq z_{ij}$ 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 described in Section 3.3 (repeating the process of independently setting each vertex v_i to be in the subset U with probability x_i, and then *fixing* the size of U, polynomially many times) we obtain a set U_k of size exactly k with weight at least

$$(1-\varepsilon)\frac{1}{2}\sum_{e_{ij}\in E} w_{ij} z_{ij} \geq (1-\varepsilon)\frac{1}{2}Z^*$$

As in Lemma 3.10 it can be seen that if x_i or x_j differ from $1/2$ then the probability that e_{ij} is cut in the above rounding technique 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-k-Cut 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^* 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 $\{v_1^* \ldots v_n^*\}$ using the probabilities x^* computed above and the rounding technique described above, 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 $0.8785 Z_h^*$, 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 $k \leq \frac{n}{3.1}$, 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 *fix* the size of U_h using our *fixing lemma* (Lemma 5.3) 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 3.15*(b)*.

■

Regarding the integrality gap of relaxation *(SDP-Cut)*, we have just seen that from any valid vector solution to relaxation *(SDP-Cut)* of value Z^* we can derive a valid partition $(U, V \setminus U)$ of weight greater than $0.5Z^*$. Thus we conclude that the integrality gap of relaxation *(SDP-Cut)* is at most 2 (or to be precise a bit below 2).

On the negative side, 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 4.6. 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, $\{\hat{v}_1 \ldots \hat{v}_5\}$, 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 $1 \leq i \neq j \leq 5$.

Similarly a gap of $\max\left(\frac{9}{8}, 2 - \frac{2k}{n}\right)$ can be achieved when ignoring constraint *(3)* of *(SDP-Cut)* on 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 Figure 5. These results are proven using techniques described earlier in Propositions 3.19, 4.6, and 4.7.

6 Max-k-Not Cut

Our last problem is the Max-k-NC problem. It can be seen that the Max-k-NC 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-k-NC when $k = n/2$.

Let $G = (V, E)$, w and k be an instance of the Max-k-NC 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-k-Cut 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 $1 - \frac{2k(n-k)}{n(n-1)}$. Hence in contrast to the problems dealt with earlier, as a function of k we see that this ratio increases as k decreases. An algorithm based on the straightforward linear relaxation to the Max-k-NC problem shown below, yields an approximation ratio no better than the above for all k. This is due to the fact that setting x_i to be exactly k/n yields an optimal fractional solution.

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

Using semidefinite programming we improve the approximation ratio of $1 - \frac{2k(n-k)}{n(n-1)}$ when k is close to $n/2$. Consider the following quadratic integer program.

(QI-NC)	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-NC)*. 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-NC)* by the following semidefinite relaxation similar to the relaxations presented earlier.

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

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

We are now able to use a slight variant of algorithm A_{SDP-DS} from Section 4 and the analysis given in Theorem 4.3 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 D.1 of the appendix).

The only difference worth mentioning between the current problem and the ones described earlier is the *fixing* lemma for the Max-k-NC problem (based on ideas from the previous *fixing* lemma for the Max-k-Cut problem).

Lemma 6.1 (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*

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

Proof : Assume w.l.o.g. that the set U is of size greater than k and weight W, and we would like to *fix* the size of U to be exactly k (the other three possibilities are analogous). Uniformly picking a random subset $U_k \subseteq U$ of size k we have that for an edge e_{ij} *not* in the cut $(U, V \setminus U)$

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

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 $\gamma_1 W$.

■

As for the integrality gap of relaxation *(SDP-NC)*, it can be shown to be at most 2 using a proof similar to that of Proposition 4.5. Furthermore, we achieve a gap of $\frac{10}{9}$ for $k = n/2$ by considering the graph $5 * K_{\frac{2k}{5}}$ as in Proposition 4.6.

7 Summary and concluding remarks

In this thesis 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. Following we state the approximation ratios achieved on our four maximization problems using the algorithmic techniques discussed in our work.

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

Recall that using an algorithm based on linear programming, [AS98] have recently proven approximation ratios of exactly $3/4$ and $1/2$ on the Max-k-VC and Max-k-Cut problems respectively.

Many questions remain open. We discuss some of them.

Is the approximation ratio achieved using semidefinite programming strictly higher than that achieved using linear programming for all values of k? In the Max-k-VC and Max-k-Cut problems our algorithms based on semidefinite programming achieve approximation ratios strictly above those using linear programming for all values of k. To be precise the approximation ratios achieved using semidefinite programming were higher than those achieved using linear programming by some universal constant $\varepsilon > 0$. We were not able to obtain similar results in the case of the remaining two problems, Max-k-DS and Max-k-NC.

How *good* are the rounding techniques suggested in our algorithms based on semidefinite programming? When analyzing the quality of a rounding technique to a given semidefinite relaxation, one aims to reduce the gap between the ratio $Z^*/c(\bar{x})$ and the integrality gap Z^*/Opt.

Where Z^* is the optimal value of our relaxation, $c(\bar{x})$ is the value of the solution returned by our rounding technique, and *Opt* is the value of the optimal solution to our problem. In the analysis of all four of our problems, we have left a large unresolved gap between these two ratios. This gap can result from the analysis of our rounding technique, the actual quality of our rounding technique, the integrality gap of our relaxation, or any combination of the above. As many related problems, such as Max-Cut and Max-SAT, may profit by progress made in any of these directions, new results are welcome.

Acknowledgments

I would like to express my deep gratitude to my advisor Prof. Uriel Feige for his guidance, helpful suggestions, and for always being available.

Appendix

A Max-k-VC

Lemma A.1 (Lemma 3.12) *For any constant $\varepsilon > 0$ there exists a constant M_ε such that if $k > M_\varepsilon$ step (2) of algorithm A_{LP-VC} yields a set U of size less than $k + k^{\frac{2}{3}}$ and of weight at least $(1-\varepsilon)E[w(U)]$ with probability larger than $\frac{\varepsilon}{10}$.*

Proof : Using notations from Lemma 3.12 we conclude from basic probability theory that

$$E[w(U)] = \Pr(|U| < 2k)E[w(U) \mid |U| < 2k] + \sum_{i=2}^{\lfloor n/k \rfloor} \Pr(ik \leq |U| < (i+1)k)E[w(U) \mid ik \leq |U| < (i+1)k]$$

From the *fixing* lemma (Lemma 3.13) we conclude that any subset U of size less than ik covers the weight of at most $iOpt(G)$ edges. Otherwise *fixing* the size of U to be k would yield a subset $U_k \subseteq U$ with super-optimal weight. Hence

$$E[w(U) \mid ik \leq |U| < (i+1)k] \leq (i+1)Opt(G).$$

Thus we conclude using the *chernoff* inequality (Fact 3.11) that for any constant $\delta > 0$

$$\begin{aligned}
\sum_{i=2}^{\lfloor n/k \rfloor} \Pr(ik \leq |U| < (i+1)k)E[w(U) \mid ik \leq |U| < (i+1)] &\leq Opt(G) \sum_{i=2}^{\lfloor n/k \rfloor} (i+1)\Pr(ik \leq |U| < (i+1)k) \\
&\leq Opt(G) \left(3e^{-k/3} + \sum_{i=2}^{\lfloor n/k \rfloor - 1} e^{-i^2 k/3} \right) \\
&\leq \delta Opt(G)
\end{aligned}$$

Where the second inequality is due to the fact that we are dealing with a telescopic sum and the last inequality is achieved for k greater than some constant M_δ. Hence using the fact that $E[w(U)] \geq 3/4Opt(G)$ (Lemma 3.10) we conclude for such k

$$E_{2k} = E[w(U) \mid |U| < 2k] \geq E[w(U)] - \delta Opt(G) \geq (1-2\delta)E[w(U)].$$

Thus for any constant $\varepsilon > 0$, $\delta = \varepsilon/4$, and k large enough we obtain

$$\begin{aligned}
\Pr(w(U) < (1-\varepsilon)E[w(U)]) &\leq \Pr(w(U) < (1-\varepsilon/2)E_{2k}) \\
&= \Pr(w(U) < (1-\varepsilon/2)E_{2k} \wedge |U| < 2k) + \\
&\quad\ \Pr(w(U) < (1-\varepsilon/2)E_{2k} \wedge |U| \geq 2k) \\
&\leq \Pr(w(U) < (1-\varepsilon/2)E_{2k} \mid |U| < 2k) + e^{-k/3}
\end{aligned}$$

As stated above when $|U| < 2k$ we have that $w(U) < 2Opt(G)$, and when δ is small enough $E_{2k} \geq (1-2\delta)E[w(U)] \geq 1/2Opt(G)$. This enables us, for large enough k, to use the same technique used in the proof of the weaker lemma (Lemma 3.12) to obtain

$$\Pr(w(U) < (1-\varepsilon/2)E_{2k} \mid |U| < 2k) \leq 1 - \frac{\varepsilon E_{2k}}{4Opt(G)} \leq 1 - \varepsilon/8.$$

We also have for k larger than some constant that

$$\Pr(|U| \geq k + k^{2/3}) < \frac{\varepsilon}{80} \quad , \quad e^{-k/3} < \frac{\varepsilon}{80}$$

Thus we conclude, for appropriate k, that our rounding scheme yields a set U with size less than $k + k^{2/3}$ and weight at least $(1-\varepsilon)E[w(U)]$ with probability above $\varepsilon/10$.

■

Lemma A.2 (Lemma 3.17) *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 3.4 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 3.17 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 A.3 (Theorem 3.15*(b)*) *There exists a BPP algorithm based on semidefinite programming that approximates the Max-*k*-VC 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-k-VC 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 the proof of Lemma 3.10 that if e_{ij} is a *good* edge then the probability that this edge is covered, by independently choosing a vertex v_i with probability x_i^*, is of value at least $0.754z_{ij}$. Thus running steps *(2-4)* of algorithm A_{LP-VC} on the valid fractional solution (x^*, z^*) we obtain with high probability a set U_k of size exactly k that (up to a factor of $(1-\varepsilon)$) covers the weight of

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

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-$\hat{k}$-VC 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 3.17 when $c \in [1.75, 2.3]$ and $\theta = 0.85$). Clearly $\lceil \hat{k} \rceil$ is less than 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 $0.7625 W_{bad} \geq 0.754 Z^*$.

■

B Max-k-DS

Lemma B.1 (Theorem 4.3) *With high probability, algorithm A_{SDP-DS} approximates the Max-*k*-DS problem within the ratio of $\alpha_{SDP} > 0.517$ when $k = n/2$.*

Proof : Using notations from Theorem 4.3 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.

Omitting the dependence on arbitrarily small disturbances 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 4.2) 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.

■

C Max-k-Cut

Lemma C.1 (Lemma 5.4) *There exists a BPP algorithm based on semidefinite programming that approximates the Max-*k*-Cut 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 4.2, 5.3 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 4.2

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

Finally fix the size of $\tilde{U}$ to be exactly k by the *fixing* lemma (Lemma 5.3).

From analysis given in Section 4.2 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 $\varepsilon > 0$. 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

$$E[Z] \;\; = \;\; \alpha + \theta_1\alpha + \theta_2(\alpha(1 - 1/c)^2 - 1 + 2/c)$$

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

It is not hard to see, relying on the analysis in Section 4.2 and the *fixing* lemma (Lemma 5.3), 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 7 schematically shows the values of c and $|U| = \mu n$ for which each of the functions above are considered.

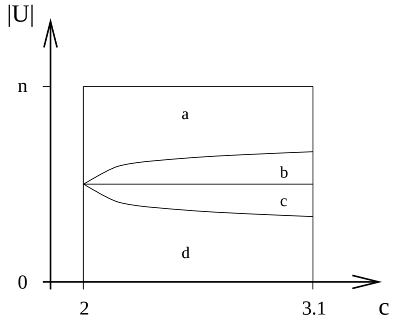

Figure 7: The range relevant to each of the functions f_a, f_b, f_c and f_d when $\mu \in [0,1]$ and $c \in [2, 3.1]$.

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 equal to 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.

■

D Max-k-Not Cut

Lemma D.1 *There exists a BPP algorithm based on semidefinite programming that approximates the Max*-k-*NC problem within the ratio of* $\alpha_{SDP} \geq 0.54$ *when* $k = n/2$.

Sketch of Proof : Using notations from Sections 4.2, 6 define the following algorithm A_{SDP-NC}. Solve the relaxation *(SDP-NC)* 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 6.1).

From analysis given in Section 4.2 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 $\varepsilon > 0$. 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-NC} 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 equal to 0.5417. When k is close to $n/2$ similar analysis guarantees an approximation ratio slightly above 0.5.

∎

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.*

[AS98] A.A. Ageev, M.I. Sviridenko. Approximation Algorithms for Maximum Coverage and Max Cut with Cardinality Constraints. *Manuscript - 1998.*

[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. *Manuscript - 1998.*

[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.*

[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.*

[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.*

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