26 Semidefinite Programming

In the previous chapters of Part II of this book we have shown how linear programs provide a systematic way of placing a good upper bound on OPT (assuming a minimization problem), for numerous **NP**-hard problems. As stated earlier, this is a key step in the design of an approximation algorithm for an **NP**-hard problem. It is natural, then, to ask if there are other widely applicable ways of doing this.

In this chapter we provide another class of relaxations, called *vector programs*. These serve as relaxations for several **NP**-hard problems, in particular, for problems that can be expressed as *strict quadratic programs* (see Section 26.1 for a definition). Vector programs are equivalent to a powerful and well-studied generalization of linear programs, called semidefinite programs. Semidefinite programs, and consequently vector programs, can be solved within an additive error of ε, for any $\varepsilon > 0$, in time polynomial in n and $\log(1/\varepsilon)$, using the ellipsoid algorithm (see Section 26.3).

We will illustrate the use of vector programs by deriving a 0.87856 factor algorithm for the following problem (see Exercises 2.1 and 16.6 for a factor 1/2 algorithm).

Problem 26.1 (Maximum cut (MAX-CUT)) Given an undirected graph $G = (V, E)$, with edge weights $w : E \to \mathbf{Q}^+$, find a partition $(S, \overline{S})$ of V so as to maximize the total weight of edges in this cut, i.e., edges that have one endpoint in S and one endpoint in $\overline{S}$.

26.1 Strict quadratic programs and vector programs

A *quadratic program* is the problem of optimizing (minimizing or maximizing) a quadratic function of integer valued variables, subject to quadratic constraints on these variables. If each monomial in the objective function, as well as in each of the constraints, is of degree 0 (i.e., is a constant) or 2, then we will say that this is a *strict quadratic program*.

Let us give a strict quadratic program for MAX-CUT. Let y_i be an indicator variable for vertex v_i which will be constrained to be either $+1$ or -1. The partition $(S, \overline{S})$ will be defined as follows. $S = \{v_i \mid y_i = 1\}$ and $\overline{S} = \{v_i \mid y_i = -1\}$. If v_i and v_j are on opposite sides of this partition,

then $y_i y_j = -1$, and edge (v_i, v_j) contributes w_{ij} to the objective function. On the other hand, if they are on the same side, then $y_i y_j = 1$, and edge (v_i, v_j) makes no contribution. Hence, an optimal solution to this program is a maximum cut in G.

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j) && (26.1) \\ \text{subject to} \quad & y_i^2 = 1, && v_i \in V \\ & y_i \in \mathbf{Z}, && v_i \in V \end{aligned}$$

We will relax this program to a vector program. A *vector program* is defined over n *vector variables* in $\mathbf{R}^n$, say $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\boldsymbol{v}_i \cdot \boldsymbol{v}_j, 1 \le i \le j \le n$, subject to linear constraints on these inner products. Thus, a vector program can be thought of as being obtained from a linear program by replacing each variable with an inner product of a pair of these vectors.

A strict quadratic program over n integer variables defines a vector program over n vector variables in $\mathbf{R}^n$ as follows. Establish a correspondence between the n integer variables and the n vector variables, and replace each degree 2 term with the corresponding inner product. For instance, the term $y_i y_j$ in (26.1) is replaced with $\boldsymbol{v}_i \cdot \boldsymbol{v}_j$. In this manner, we obtain the following vector program for MAX-CUT.

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - \boldsymbol{v}_i \cdot \boldsymbol{v}_j) && (26.2) \\ \text{subject to} \quad & \boldsymbol{v}_i \cdot \boldsymbol{v}_i = 1, && v_i \in V \\ & \boldsymbol{v}_i \in \mathbf{R}^n, && v_i \in V \end{aligned}$$

Because of the constraint $\boldsymbol{v}_i \cdot \boldsymbol{v}_i = 1$, the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ are constrained to lie on the n-dimensional sphere, S_{n-1}. Any feasible solution to (26.1) yields a solution to (26.2) having the same objective function value, by assigning the vector $(y_i, 0, \ldots, 0)$ to $\boldsymbol{v}_i$. (Notice that under this assignment, $\boldsymbol{v}_i \cdot \boldsymbol{v}_j$ is simply $y_i y_j$.) Therefore, the vector program (26.2) is a relaxation of the strict quadratic program (26.1). Clearly, this holds in general as well; the vector program corresponding to a strict quadratic program is a relaxation of the quadratic program.

Interestingly enough, vector programs are approximable to any desired degree of accuracy in polynomial time, and thus relaxation (26.2) provides an upper bound on OPT for MAX-CUT. To show this, we need to recall some interesting and powerful properties of positive semidefinite matrices.

Remark 26.2 Vector programs do not always come about as relaxations of strict quadratic programs. Exercise 26.13 gives an **NP**-hard problem that has vector program relaxation; however, we do not know of a strict quadratic program for it.

26.2 Properties of positive semidefinite matrices

Let $\boldsymbol{A}$ be a real, symmetric $n \times n$ matrix. Then $\boldsymbol{A}$ has real eigenvalues and has n linearly independent eigenvectors (even if the eigenvalues are not distinct). We will say that $\boldsymbol{A}$ is *positive semidefinite* if

$$\forall \boldsymbol{x} \in \mathbf{R}^n,\ \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \geq 0.$$

We will use the following two equivalent conditions crucially. We provide a proof sketch for completeness.

Theorem 26.3 *Let $\boldsymbol{A}$ be a real symmetric $n \times n$ matrix. Then, the following are equivalent:*

1. $\forall \boldsymbol{x} \in \mathbf{R}^n,\ \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \geq 0$.
2. *All eigenvalues of $\boldsymbol{A}$ are nonnegative.*
3. *There is an $n \times n$ real matrix $\boldsymbol{W}$, such that $\boldsymbol{A} = \boldsymbol{W}^T \boldsymbol{W}$.*

Proof: $(1 \Rightarrow 2)$: Let λ be an eigenvalue of $\boldsymbol{A}$, and let $\boldsymbol{v}$ be a corresponding eigenvector. Therefore, $\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}$. Pre-multiplying by $\boldsymbol{v}^T$ we get $\boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v} = \lambda \boldsymbol{v}^T \boldsymbol{v}$. Now, by (1), $\boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v} \geq 0$. Therefore, $\lambda \boldsymbol{v}^T \boldsymbol{v} \geq 0$. Since $\boldsymbol{v}^T \boldsymbol{v} > 0$, $\lambda \geq 0$.

$(2 \Rightarrow 3)$: Let $\lambda_1, \ldots, \lambda_n$ be the n eigenvalues of $\boldsymbol{A}$, and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ be the corresponding complete collection of orthonormal eigenvectors. Let $\mathbf{Q}$ be the matrix whose columns are $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$, and $\boldsymbol{\Lambda}$ be the diagonal matrix with entries $\lambda_1, \ldots, \lambda_n$. Since for each i, $\boldsymbol{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, we have $\boldsymbol{A}\mathbf{Q} = \mathbf{Q}\boldsymbol{\Lambda}$. Since $\mathbf{Q}$ is orthogonal, i.e., $\mathbf{Q}\mathbf{Q}^T = I$, we get that $\mathbf{Q}^T = \mathbf{Q}^{-1}$. Therefore,

$$\boldsymbol{A} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T.$$

Let $\boldsymbol{D}$ be the diagonal matrix whose diagonal entries are the positive square roots of $\lambda_1, \ldots, \lambda_n$ (by (2), $\lambda_1, \ldots, \lambda_n$ are nonnegative, and thus their square roots are real). Then, $\boldsymbol{\Lambda} = \boldsymbol{D}\boldsymbol{D}^T$. Substituting, we get

$$\boldsymbol{A} = \mathbf{Q}\boldsymbol{D}\boldsymbol{D}^T\mathbf{Q}^T = (\mathbf{Q}\boldsymbol{D})(\mathbf{Q}\boldsymbol{D})^T.$$

Now, (3) follows by letting $\boldsymbol{W} = (\mathbf{Q}\boldsymbol{D})^T$.

$(3 \Rightarrow 1)$: For any

$$\boldsymbol{x} \in \mathbf{R}^n,\ \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{W}^T \boldsymbol{W} \boldsymbol{x} = (\boldsymbol{W}\boldsymbol{x})^T (\boldsymbol{W}\boldsymbol{x}) \geq 0. \qquad \square$$

Using Cholesky decomposition (see Section 26.7), a real symmetric matrix can be decomposed, in polynomial time, as $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$, where $\boldsymbol{\Lambda}$ is a diagonal matrix whose diagonal entries are the eigenvalues of $\boldsymbol{A}$. Now $\boldsymbol{A}$ is positive semidefinite iff all the entries of $\boldsymbol{\Lambda}$ are nonnegative, thus giving a polynomial time test for positive semidefiniteness. The decomposition $\boldsymbol{W}\boldsymbol{W}^T$ is not polynomial time computable because in general it may contain irrational entries. However, it can be approximated to any desired degree by approximating the square roots of the entries of $\boldsymbol{\Lambda}$. In the rest of this chapter we will assume that we have an exact decomposition, since the inaccuracy resulting from an approximate decomposition can be absorbed into the approximation factor (see Exercise 26.6).

It is easy to see that the sum of two $n \times n$ positive semidefinite matrices is also positive semidefinite (e.g., using characterization (1) of Theorem 26.3). This is also true of any convex combination of such matrices.

26.3 The semidefinite programming problem

Let $\boldsymbol{Y}$ be an $n \times n$ matrix of real valued variables whose (i, j)th entry is y_{ij}. The problem of maximizing a linear function of the y_{ij}'s, subject to linear constraints on them, and the additional constraint that $\boldsymbol{Y}$ be symmetric and positive semidefinite, is called the *semidefinite programming problem*.

Let us introduce some notation to state this formally. Denote by $\mathbf{R}^{n\times n}$ the space of $n \times n$ real matrices. Recall that the *trace* of a matrix $\boldsymbol{A} \in \mathbf{R}^{n\times n}$ is the sum of its diagonal entries and is denoted by $\mathrm{tr}(\boldsymbol{A})$. The Frobenius inner product of matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbf{R}^{n\times n}$, denoted $\boldsymbol{A} \bullet \boldsymbol{B}$, is defined to be

$$\boldsymbol{A} \bullet \boldsymbol{B} = \mathrm{tr}(\boldsymbol{A}^T\boldsymbol{B}) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij},$$

where a_{ij} and b_{ij} are the (i, j)th entries of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. Let M_n denote the cone of symmetric $n \times n$ real matrices. For $\boldsymbol{A} \in M_n$, $\boldsymbol{A} \succeq 0$ denotes the fact that matrix $\boldsymbol{A}$ is positive semidefinite.

Let $\boldsymbol{C}, \boldsymbol{D}_1, \ldots, \boldsymbol{D}_k \in M_n$ and $d_1, \ldots d_k \in \mathbf{R}$. Following is a statement of the general semidefinite programming problem. Let us denote it by $\mathcal{S}$.

$$\begin{aligned} &\text{maximize} && \boldsymbol{C} \bullet \boldsymbol{Y} && (26.3)\\ &\text{subject to} && \boldsymbol{D}_i \bullet \boldsymbol{Y} = d_i, \quad 1 \le i \le k\\ &&& \boldsymbol{Y} \succeq 0,\\ &&& \boldsymbol{Y} \in M_n. \end{aligned}$$

Observe that if $\boldsymbol{C}, \boldsymbol{D}_1, \ldots, \boldsymbol{D}_k$ are all diagonal matrices, this is simply a linear programming problem. As in the case of linear programs, it is easy to

see that allowing linear inequalities, in addition to equalities, does not make the problem more general.

Let us call a matrix in $\mathbf{R}^{n\times n}$ satisfying all the constraints of $\mathcal{S}$ a *feasible solution*. Since a convex combination of positive semidefinite matrices is positive semidefinite, it is easy to see that the set of feasible solutions is *convex*, i.e., if $\boldsymbol{A} \in \mathbf{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbf{R}^{n\times n}$ are feasible solutions then so is any convex combination of these solutions.

Let $\boldsymbol{A} \in \mathbf{R}^{n\times n}$ be an infeasible point. Let $\boldsymbol{C} \in \mathbf{R}^{n\times n}$. A hyperplane $\boldsymbol{C} \bullet \boldsymbol{Y} \leq b$ is called a *separating hyperplane for* $\boldsymbol{A}$ if all feasible points satisfy it and point $\boldsymbol{A}$ does not satisfy it. In the next theorem we show how to find a separating hyperplane in polynomial time. As a consequence, for any $\varepsilon > 0$, semidefinite programs can be solved within an additive error of ε, in time polynomial in n and $\log(1/\varepsilon)$, using the ellipsoid algorithm (see Section 26.7 for more efficient methods).

Theorem 26.4 *Let $\mathcal{S}$ be a semidefinite programming problem, and $\boldsymbol{A}$ be a point in $\mathbf{R}^{n\times n}$. We can determine, in polynomial time, whether $\boldsymbol{A}$ is feasible for $\mathcal{S}$ and, if it is not, find a separating hyperplane.*

Proof: Testing for feasibility involves ensuring that $\boldsymbol{A}$ is symmetric and positive semidefinite and that it satisfies all the linear constraints. By remarks made in Section 26.2, this can be done in polynomial time. If $\boldsymbol{A}$ is infeasible, a separating hyperplane is obtained as follows.

- If $\boldsymbol{A}$ is not symmetric, $a_{ij} > a_{ji}$ for some i, j. Then $y_{ij} \leq y_{ji}$ is a separating hyperplane.
- If $\boldsymbol{A}$ is not positive semidefinite, then it has a negative eigenvalue, say λ. Let $\boldsymbol{v}$ be the corresponding eigenvector. Now $(\boldsymbol{v}\boldsymbol{v}^T) \bullet \boldsymbol{Y} = \boldsymbol{v}^T\boldsymbol{Y}\boldsymbol{v} \geq 0$ is a separating hyperplane.
- If any of the linear constraints is violated, it directly yields a separating hyperplane.

□

Next, let us show that vector programs are equivalent to semidefinite programs, thereby showing that the former can be solved efficiently to any desired degree of accuracy. Let $\mathcal{V}$ be a vector program on n n-dimensional vector variables $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$. Define the corresponding semidefinite program, $\mathcal{S}$, over n^2 variables $y_{ij}, 1 \leq i, j \leq n$, as follows. Replace each inner product $\boldsymbol{v}_i \cdot \boldsymbol{v}_j$ occurring in $\mathcal{V}$ by the variable y_{ij}. The objective function and constraints are now linear in the y_{ij}'s. Additionally, require that matrix $\boldsymbol{Y}$, whose (i, j)th entry is y_{ij}, be symmetric and positive semidefinite.

Lemma 26.5 *Vector program $\mathcal{V}$ is equivalent to semidefinite program $\mathcal{S}$.*

Proof: We will show that corresponding to each feasible solution to $\mathcal{V}$, there is a feasible solution to $\mathcal{S}$ of the same objective function value, and vice

versa. Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ be a feasible solution to $\mathcal{V}$. Let W be the matrix whose columns are $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$. Then, it is easy to see that $\boldsymbol{A} = \boldsymbol{W}^T\boldsymbol{W}$ is a feasible solution to $\mathcal{S}$ having the same objective function value.

For the other direction, let $\boldsymbol{A}$ be a feasible solution to $\mathcal{S}$. By Theorem 26.3, there is an $n \times n$ matrix $\boldsymbol{W}$ such that $\boldsymbol{A} = \boldsymbol{W}^T\boldsymbol{W}$. Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ be the columns of $\boldsymbol{W}$. Then, it is easy to see that $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ is a feasible solution to $\mathcal{V}$ having the same objective function value. □

Finally, we give the semidefinite programming relaxation to MAX-CUT that is equivalent to vector program 26.2.

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j) && && (26.4) \\ \text{subject to} \quad & y_i^2 = 1, && v_i \in V \\ & \boldsymbol{Y} \succeq 0, \\ & \boldsymbol{Y} \in M_n. \end{aligned}$$

26.4 Randomized rounding algorithm

We now present the algorithm for MAX-CUT. For convenience, let us assume that we have an optimal solution to the vector program (26.2). The slight inaccuracy in solving it can be absorbed into the approximation factor (see Exercise 26.6). Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ be an optimal solution, and let OPT_v denote its objective function value. These vectors lie on the n-dimensional unit sphere S_{n-1}. We need to obtain a cut $(S, \overline{S})$ whose weight is a large fraction of OPT_v.

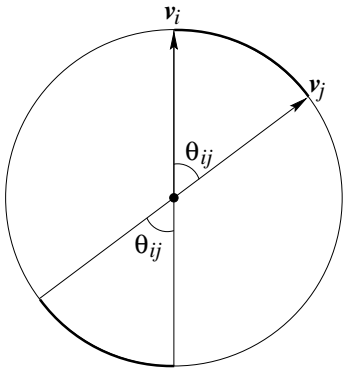
Let θ_{ij} denote the angle between vectors $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$. The contribution of this pair of vectors to OPT_v is

$$\frac{w_{ij}}{2}(1 - \cos\theta_{ij}).$$

Clearly, the closer θ_{ij} is to π, the larger this contribution will be. In turn, we would like vertices v_i and v_j to be separated if θ_{ij} is large. The following method accomplishes precisely this. Pick $\boldsymbol{r}$ to be a uniformly distributed vector on the unit sphere S_{n-1}, and let $S = \{v_i \mid \boldsymbol{a}_i \cdot \boldsymbol{r} \ge 0\}$.

Lemma 26.6 $\quad \mathbf{Pr}[v_i \textit{ and } v_j \textit{ are separated}] = \dfrac{\theta_{ij}}{\pi}.$

Proof: Project $\boldsymbol{r}$ onto the plane containing $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. Now, vertices v_i and v_j will be separated iff the projection lies in one of the two arcs of angle θ_{ij} shown below.



Since $\boldsymbol{r}$ has been picked from a spherically symmetric distribution, its projection will be a random direction on this plane. The lemma follows. □

The next lemma shows how to generate vectors that are uniformly distributed on the unit sphere S_{n-1}.

Lemma 26.7 *Let $x_1, \ldots, x_n$ be picked independently from the normal distribution with mean 0 and unit standard deviation. Let $d = (x_1^2 + \ldots + x_n^2)^{1/2}$. Then, $(x_1/d, \ldots, x_n/d)$ is a random vector on the unit sphere S_{n-1}.*

Proof: Consider the vector $\boldsymbol{r} = (x_1, \ldots, x_n)$. The distribution function for $\boldsymbol{r}$ has density

$$f(y_1, \ldots, y_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-y_i^2/2} = \frac{1}{(2\pi)^{n/2}} e^{-\frac{1}{2}\sum_i y_i^2}.$$

Notice that the density function depends only on the distance of the point from the origin. Therefore, the distribution of $\boldsymbol{r}$ is spherically symmetric. Hence, dividing by the length of $\boldsymbol{r}$, i.e., d, we get a random vector on S_{n-1}. □

The algorithm is summarized below.

Algorithm 26.8 (MAX-CUT)

1. Solve vector program (26.2). Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ be an optimal solution.
2. Pick $\boldsymbol{r}$ to be a uniformly distributed vector on the unit sphere S_{n-1}.
3. Let $S = \{v_i \mid \boldsymbol{a}_i \cdot \boldsymbol{r} \geq 0\}$.

Let W be the random variable denoting the weight of edges in the cut picked by Algorithm 26.8, and let

$$\alpha = \frac{2}{\pi} \min_{0 \leq \theta \leq \pi} \frac{\theta}{1 - \cos\theta}.$$

One can show that $\alpha > 0.87856$ (see Exercise 26.3).

Lemma 26.9 $\mathbf{E}[W] \geq \alpha \cdot \mathrm{OPT}_v$.

Proof: By the definition of α we have that for any θ, $0 \leq \theta \leq \pi$,

$$\frac{\theta}{\pi} \geq \alpha \left(\frac{1 - \cos\theta}{2} \right). \tag{26.5}$$

Using this and Lemma 26.6, we get

$$\mathbf{E}[W] = \sum_{1 \leq i < j \leq n} w_{ij} \mathbf{Pr}[\boldsymbol{v}_i \text{ and } \boldsymbol{v}_j \text{ are separated}]$$

$$= \sum_{1 \leq i < j \leq n} w_{ij} \frac{\theta_{ij}}{\pi} \geq \alpha \cdot \sum_{1 \leq i < j \leq n} \frac{1}{2} w_{ij}(1 - \cos\theta_{ij}) = \alpha \cdot \mathrm{OPT}_v.$$

□

Let us define the *integrality gap* for relaxation (26.2) to be

$$\inf_I \frac{\mathrm{OPT}(I)}{\mathrm{OPT}_v(I)},$$

where the infimum is over all instances I of MAX-CUT.

Corollary 26.10 *The integrality gap for relaxation (26.2) is at least* $\alpha > 0.87856$.

Theorem 26.11 *There is a randomized approximation algorithm for MAX-CUT achieving an approximation factor of* 0.87856.

Proof: Let us first obtain a "high probability" statement using the bound on expectation established in Lemma 26.9. Let T denote the sum of weights of all edges in G, and define a so that $\mathbf{E}[W] = aT$. Let

$$p = \mathbf{Pr}[W < (1 - \varepsilon)aT],$$

where $\varepsilon > 0$ is a constant. Since the random variable W is always bounded by T, we get

$$aT \leq p(1 - \varepsilon)aT + (1 - p)T.$$

Therefore,

$$p \leq \frac{1 - a}{1 - a + a\varepsilon}.$$

Now,

$$T \geq \mathbf{E}[W] = aT \geq \alpha \cdot \mathrm{OPT}_v \geq \alpha \cdot \mathrm{OPT} \geq \frac{\alpha T}{2},$$

where the last inequality follows from the fact that $\mathrm{OPT} \geq T/2$ (see Exercise 2.1). Therefore, $1 \geq a \geq \alpha/2$. Using this upper and lower bound on a, we get

$$p \leq 1 - \frac{\varepsilon\alpha/2}{1 + \varepsilon - \alpha/2} \leq 1 - c,$$

where

$$c = \frac{\varepsilon\alpha/2}{1 + \varepsilon - \alpha/2}.$$

Run Algorithm 26.8 $1/c$ times, and output the heaviest cut found in these runs. Let W' be the weight of this cut. Then,

$$\mathbf{Pr}[W' \geq (1-\varepsilon)aT] \geq 1 - (1-c)^{1/c} \geq 1 - \frac{1}{e}.$$

Since $aT \geq \alpha \cdot \mathrm{OPT} > 0.87856\ \mathrm{OPT}$, we can pick a value of $\varepsilon > 0$ so that $(1-\varepsilon)aT \geq 0.87856\ \mathrm{OPT}$. □

Example 26.12 The following example shows that the bound on the integrality gap of relaxation (26.2) given in Corollary 26.10 is almost tight. Consider a graph which is a 5-cycle $v_1, v_2, v_3, v_4, v_5, v_1$. Then, an optimal solution to relaxation (26.2) is to place the five vectors in a 2-dimensional subspace within which they are given by $\boldsymbol{v}_i = (\cos(\frac{4i\pi}{5}), \sin(\frac{4i\pi}{5}))$, for $1 \leq i \leq 5$ (see Exercise 26.5). The cost of this solution is $\mathrm{OPT}_v = \frac{5}{2}(1 + \cos\frac{\pi}{5}) = \frac{25+5\sqrt{5}}{8}$. Since $\mathrm{OPT} = 4$ for this graph, the integrality gap for this example is $\frac{32}{25+5\sqrt{5}} = 0.88445....$ □

26.5 Improving the guarantee for MAX-2SAT

MAX-2SAT is the restriction of MAX-SAT (Problem 16.1) to formulae in which each clause contains at most two literals. In Chapter 16 we obtained a factor 3/4 algorithm for this problem using randomization, followed by the method of conditional expectation. We will give an improved algorithm using semidefinite programming.

The key new idea needed is a way of converting the obvious quadratic program (see Exercise 26.8) for this problem into a strict quadratic program. We will accomplish this as follows. Corresponding to each Boolean variable

x_i, introduce variable y_i which is constrained to be either $+1$ or -1, for $1 \leq i \leq n$. In addition, introduce another variable, say y_0, which is also constrained to be $+1$ or -1. Let us impose the convention that Boolean variable x_i is true if $y_i = y_0$ and false otherwise. Under this convention we can write the value of a clause in terms of the y_i's, where the *value*, $v(C)$, of clause C is defined to be 1 if C is satisfied and 0 otherwise. Thus, for clauses containing only one literal,

$$v(x_i) = \frac{1 + y_0 y_i}{2} \text{ and } v(\overline{x_i}) = \frac{1 - y_0 y_i}{2}.$$

Consider a clause containing 2 literals, e.g., $(x_i \vee x_j)$. Its value is

$$\begin{aligned} v(x_i \vee x_j) &= 1 - v(\overline{x_i})v(\overline{x_j}) = 1 - \frac{1 - y_0 y_i}{2}\frac{1 - y_0 y_j}{2} \\ &= \frac{1}{4}\left(3 + y_0 y_i + y_0 y_j - y_0^2 y_i y_j\right) \\ &= \frac{1 + y_0 y_i}{4} + \frac{1 + y_0 y_j}{4} + \frac{1 - y_i y_j}{4}. \end{aligned}$$

Observe that in this derivation we have used the fact that $y_0^2 = 1$. In all the remaining cases as well, it is easy to check that the value of a 2 literal clause consists of a linear combination of terms of the form $(1 + y_i y_j)$ or $(1 - y_i y_j)$. Therefore, a MAX-2SAT instance can be written as the following strict quadratic program, where the a_{ij}'s and b_{ij}'s are computed by collecting terms appropriately.

$$\begin{aligned} \text{maximize} \quad & \sum_{0 \leq i < j \leq n} a_{ij}(1 + y_i y_j) + b_{ij}(1 - y_i y_j) && && (26.6) \\ \text{subject to} \quad & y_i^2 = 1, && 0 \leq i \leq n \\ & y_i \in \mathbf{Z}, && 0 \leq i \leq n \end{aligned}$$

Following is the vector program relaxation for (26.6), where vector variable $\boldsymbol{v}_i$ corresponds to y_i.

$$\begin{aligned} \text{maximize} \quad & \sum_{0 \leq i < j \leq n} a_{ij}(1 + \boldsymbol{v}_i \cdot \boldsymbol{v}_j) + b_{ij}(1 - \boldsymbol{v}_i \cdot \boldsymbol{v}_j) && && (26.7) \\ \text{subject to} \quad & \boldsymbol{v}_i \cdot \boldsymbol{v}_i = 1, && 0 \leq i \leq n \\ & \boldsymbol{v}_i \in \mathbf{R}^{n+1}, && 0 \leq i \leq n \end{aligned}$$

The algorithm is similar to that for MAX-CUT. We solve vector program (26.7). Let $\boldsymbol{a}_0, \ldots, \boldsymbol{a}_n$ be an optimal solution. Pick a vector $\boldsymbol{r}$ uniformly distributed on the unit sphere in $(n + 1)$ dimensions, S_n, and let $y_i = 1$ iff

$\boldsymbol{r} \cdot \boldsymbol{a}_i \geq 0$, for $0 \leq i \leq n$. This gives a truth assignment for the Boolean variables. Let W be the random variable denoting the weight of this truth assignment.

Lemma 26.13 $\mathbf{E}[W] \geq \alpha \cdot \mathrm{OPT}_v$.

Proof:

$$\mathbf{E}[W] = 2 \sum_{0 \leq i < j \leq n} a_{ij} \mathbf{Pr}[y_i = y_j] + b_{ij} \mathbf{Pr}[y_i \neq y_j].$$

Let θ_{ij} denote the angle between $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$. By inequality (26.5),

$$\mathbf{Pr}[y_i \neq y_j] = \frac{\theta_{ij}}{\pi} \geq \frac{\alpha}{2}(1 - \cos\theta_{ij}).$$

By Exercise 26.4,

$$\mathbf{Pr}[y_i = y_j] = 1 - \frac{\theta_{ij}}{\pi} \geq \frac{\alpha}{2}(1 + \cos\theta_{ij}).$$

Therefore,

$$\mathbf{E}[W] \geq \alpha \cdot \sum_{0 \leq i < j \leq n} a_{ij}(1 + \cos\theta_{ij}) + b_{ij}(1 - \cos\theta_{ij}) = \alpha \cdot \mathrm{OPT}_v.$$

□

26.6 Exercises

26.1 Is matrix $\boldsymbol{W}$ in Theorem 26.3 unique (up to multiplication by -1)? **Hint:** Consider the matrix $\mathbf{Q}\boldsymbol{D}\mathbf{Q}^T$.

26.2 Let $\boldsymbol{B}$ be obtained from matrix $\boldsymbol{A}$ by throwing away a set of columns *and* the corresponding set of rows. We will say that $\boldsymbol{B}$ is a *principal submatrix* of $\boldsymbol{A}$. Show that the following is another equivalent condition for a real symmetric matrix to be positive semidefinite: that all of its principal submatrices have nonnegative determinants. (See Theorem 26.3 for other conditions.)

26.3 Show, using elementary calculus, that $\alpha > 0.87856$.

26.4 Show that for any ϕ, $0 \leq \phi \leq \pi$,

$$1 - \frac{\phi}{\pi} \geq \frac{\alpha}{2}(1 + \cos\phi).$$

Hint: Substitute $\theta = \pi - \phi$ in inequality (26.5).

26.5 Show that for a 5-cycle, the solution given in Example 26.12 is indeed an optimal solution to the vector program relaxation for MAX-CUT.

26.6 Show that the inaccuracies resulting from the fact we do not have an optimal solution to the vector program (26.2) and that matrix $\boldsymbol{A}$ is not exactly decomposed as $\boldsymbol{W}\boldsymbol{W}^T$ (see end of Section 26.2) can be absorbed into the approximation factor for MAX-CUT.
Hint: Use the idea behind the proof of Theorem 26.11 and the fact that the solution to program (26.2) lies in the range $[T/2, T]$, where T is the sum of weights of all edges in G.

26.7 Theorem 26.11 shows how to obtain a "high probability" statement from Lemma 26.9. Obtain a similar statement for MAX-2SAT, using Lemma 26.13, thereby obtaining a 0.87856 factor algorithm for MAX-2SAT.

26.8 Give a quadratic program for MAX-2SAT.

26.9 (Linial, London, and Rabinovich [190]) Let G be the complete undirected graph on n vertices, V, and let w be a function assigning nonnegative weights to the edges of G. The object is to find an optimal distortion ℓ_2^2-embedding of the vertices of G. Let vertex i be mapped to $\boldsymbol{v}_i \in \mathbf{R}^n$ by such an embedding. The embedding should satisfy:

1. no edge is overstretched, i.e., for $1 \leq i < j \leq n$, $||\boldsymbol{v}_i - \boldsymbol{v}_j||^2 \leq w_(ij)$, and
2. the maximum shrinkage is minimized, i.e.,

$$\text{maximize} \quad \min_{(i,j):w_{ij}\neq 0} (||\boldsymbol{v}_i - \boldsymbol{v}_j||^2 / w_{ij}).$$

Give a vector program for finding such an optimal embedding and give the equivalent semidefinite program.
Hint: The vector program is:

$$\begin{array}{lll} \text{minimize} & c & \qquad (26.8) \\ \text{subject to} & \boldsymbol{v}_i \cdot \boldsymbol{v}_i + \boldsymbol{v}_j \cdot \boldsymbol{v}_j - 2\boldsymbol{v}_i \cdot \boldsymbol{v}_j \leq w_{ij}, & 1 \leq i < j \leq n \\ & \boldsymbol{v}_i \cdot \boldsymbol{v}_i + \boldsymbol{v}_j \cdot \boldsymbol{v}_j - 2\boldsymbol{v}_i \cdot \boldsymbol{v}_j \geq c w_{ij}, & 1 \leq i < j \leq n \\ & \boldsymbol{v}_i \in \mathbf{R}^n, & 1 \leq i \leq n \end{array}$$

26.10 (Knuth [174]) Give an efficient algorithm for sampling from the normal distribution with mean 0 and unit standard deviation, given a source of unbiased random bits.

26.11 Give a strict quadratic program for the MAX k-CUT and maximum directed cut problems, Problems 2.14 and 2.15 stated in Exercises 2.3 and 2.4. Give a vector program relaxation and an equivalent semidefinite program as well.

26.12 (Goemans and Williamson [106]) Consider MAX-CUT with the additional constraint that specified pairs of vertices be on the same/opposite sides of the cut. Formally, we are specified two sets of pairs of vertices, S_1 and S_2. The pairs in S_1 need to be separated, and those in S_2 need to be on the same side of the cut sought. Under these constraints, the problem is to find a maximum weight cut. Assume that the constraints provided by S_1 and S_2 are not inconsistent. Give a strict quadratic program and vector program relaxation for this problem. Show how Algorithm 26.8 can be adapted to this problem so as to maintain the same approximation factor.

26.13 (Karger, Motwani, and Sudan [158]) Let $G = (V, E)$ be an undirected graph. Consider a vector program with n n-dimensional vectors corresponding to the vertices of G, and constraints that the vectors lie on the unit sphere, S_{n-1}, and that for each edge $(i, j) \in G$,

$$\boldsymbol{v}_i \cdot \boldsymbol{v}_j \leq -\frac{1}{k-1}.$$

Show that this vector program is a relaxation of the k-coloring problem, i.e., if G is k-colorable, then this vector program has a feasible solution.
Hint: Consider the following k vectors in $\mathbf{R}^n$. Each vector has 0 in the last $n-k$ positions. Vector i has $-\sqrt{\frac{k-1}{k}}$ in the ith position and $1/\sqrt{k(k-1)}$ in the remaining positions.

26.14 (Chor and Sudan [43]) Consider the following problem:

Problem 26.14 (Betweenness) We are given a set $S = \{x_1, x_2, \ldots, x_n\}$ of n items and a set T of m triplets $T \subseteq S \times S \times S$. Each triplet consists of three distinct items. A total ordering (permutation) of S, $x_{\pi_1} < x_{\pi_2} < \ldots < x_{\pi_n}$ *satisfies* a triplet $(x_i, x_j, x_k) \in T$ if x_j occurs between x_i and x_k in the ordering, i.e., if either $x_i < x_j < x_k$ holds or $x_k < x_j < x_i$ holds. The problem is to find a total ordering that maximizes the number of satisfied triplets.

1. Show that a random ordering (i.e., a permutation chosen uniformly at random among all possible permutations) will satisfy in expectation one third of all triplets in T.
2. Use the method of conditional expectation to derandomize the above algorithm, thereby obtaining a factor 1/3 approximation algorithm. What upper bound on OPT is this algorithm using? Give an example showing that with this upper bound a better algorithm is not possible.

3. The rest of the exercise develops an algorithm based on semidefinite programming. The ideas can be illustrated more simply by assuming that the instance is satisfiable, i.e., that all m triplets can be satisfied simultaneously. Note that checking for this condition is **NP**-hard, so the restriction of the betweenness problem to such instances is not an **NP**-optimization problem (see Exercise 1.9). Show that an instance is satisfiable iff the following strict quadratic program in variables $p_i \in \mathbf{R}$, $i = 1, \ldots, n$, has a solution:

$$\begin{aligned} (p_i - p_j)^2 &\geq 1 \quad \text{for all } i, j, \\ (p_i - p_j)(p_k - p_j) &\leq 0 \quad \text{for all } (x_i, x_j, x_k) \in T. \end{aligned}$$

4. Obtain the vector programming relaxation of this strict quadratic program as well as the equivalent semidefinite program.
5. Give an instance where the above semidefinite program is satisfiable but the instance itself is not satisfiable.
6. Let us assume that $n \times n$ matrix Y is a feasible solution to the above semidefinite program, and let $v_i \in \mathbf{R}^n$ for $i = 1, \cdots, n$ be vectors such that $Y_{ij} = v_i^T v_j$. Now select r uniformly at random on the unit sphere S_{n-1}. Consider the random ordering obtained by sorting $r^T v_i$. Show that, in expectation, this random ordering satisfies at least half of the constraints in T.
   Hint: What is the probability that a single triplet is satisfied? What is the angle between $v_i - v_j$ and $v_k - v_j$?

26.7 Notes

The results of this chapter are based on the seminal work of Goemans and Williamson [106] that introduced the use of semidefinite programs in approximation algorithms. Experimental results reported in their paper show that Algorithm 26.8 performs much better on typical instances than the worst case guarantee. Mahajan and Ramesh [200] give a derandomization of Algorithm 26.8, as well as the MAX-2SAT algorithm, using the method of conditional expectation. Karloff [161] provides a family of tight examples for Algorithm 26.8, for which the expected weight of the cut produced is arbitrarily close to $\alpha \cdot \mathrm{OPT}_v$. Feige and Schechtman [84] strengthen this to showing that there are graphs such that even the best hyperplane (rather than a random one, as prescribed in Algorithm 26.8) gives a cut of weight only $\alpha \cdot \mathrm{OPT}_v$. They also show that the integrality gap of the semidefinite relaxation (26.2) for MAX-CUT is α.

For efficient algorithms, using interior point methods, for approximating semidefinite programs, see Alizadeh [4], Nesterov and Nemirovskii [214] and Overton [215]. For a duality theory for semidefinite programs, see Wolkowitz [260] and Vandeberghe and Boyd [250].

Lovász and Schrijver [196] use semidefinite programming to provide an automatic way of strengthening any convex relaxation (having a convex feasible region) of a 0/1 integer program. They also show that if the original relaxation can be optimized in polynomial time, then so can the strengthened relaxation (however, in order to guarantee polynomial running time, this process can be applied only a constant number of times).

Feige and Goemans [81] improve the approximation factor for MAX-2SAT to 0.931. They also give a 0.859 factor for the maximum directed cut problem (see Exercise 26.11). For semidefinite-programming-based algorithms for the MAX k-CUT problem see Frieze and Jerrum [90]. Karger, Motwani, and Sudan [158] use the relaxation in Exercise 26.13 to obtain an $O(n^{1-3/(k+1)} \log^{1/2} n)$ coloring for k-colorable graphs.