Josai Mathematical Monographs
Vol. 11 (2018) pp. 131–143

Semidefinite Programming Approach to Combinatorial Optimization

Hiroshi Miyashita

Abstract. The semidefinite programming is an optimization approach where optimization problems are formulated as optimizing a linear function of matrix variable, subject to finitely many linear equalities or inequalities of this matrix variable, and the positive semidefiniteness for the matrix variable. In this paper, we survey the formulation of semidefinite programming compared with linear programming. Furthermore an application method to maximum cut problem known as Goemans-Williamson algorithm is also examined.

1. Introduction

Linear programming problem is a mathematical model where we minimize or maximize real-valued linear cost function subject to linear equality or inequality constraints on finite-dimensional Euclidean space. Since the invention of the simplex method by George B. Dantzig in 1947, linear programming has been used effectively in various fields such as operations research, economics, and engineering [2]. On the other hand, nonlinear programming has been applied to more complicated problems where nonlinearity affects the quality of the optimization results [1][7]. For example, in computer aided design of integrated circuits, nowadays nonlinear optimization is indispensable to overcome large-scale and high-complexity of the circuits [10].

Recently, optimization methods based on semidefinite programming attract much interest because of its wide domain of applicability, especially combinatorial optimization[8]. Inspite of its ability, most interior-point methods for linear programming can be generalized to semidefinite programs [9].

This paper gives a survey of the formulation of semidefinite programming and its application to maximum cut problem as a typical problem of combinatorial optimization, mainly on the basis of [4][5].

2010 Mathematics Subject Classification. Primary 65K05; Secondary 90C30.
Key Words and Phrases. semidefinite programming, positive semidefinite, relaxation.

2. Semidefinite Programming Formulation

At first we start to show the concept of linear programming. A linear program is the optimization problem that aims to maximize (or minimize) a linear function called objective function in n variables subject to linear equality or inequality constraints. In equality form, a linear program can be formulated as

$$\begin{aligned} &\text{maximize } \boldsymbol{c}^T\boldsymbol{x} \\ &\text{subject to } A\boldsymbol{x} = \boldsymbol{b} \\ &\boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Here, let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a column vector in $\mathbb{R}^n$. This is because n-tuple notation is easy to write compactly. Also $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ is the coefficient vector of linear objective function, and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ is the right-hand side of the equality constraints. The matrix $A \in \mathbb{R}^{m\times n}$ is the constraint matrix whose components are the coefficients of constraint linear inequalities. The bold zero $\boldsymbol{0}$ is the zero vector having all of zero components. From now on, inequality $\boldsymbol{x} \geq \boldsymbol{0}$ indicates that the inequality holds every component. Thus linear programming is summarized as follows: within all those $\boldsymbol{x} \in \mathbb{R}^n$ satisfying the linear equality constraints $A\boldsymbol{x} = \boldsymbol{b}$ and nonnegative constraints $\boldsymbol{x} \geq \boldsymbol{0}$, which are called feasible solutions, we find an $\boldsymbol{x}^*$ with the maximum value $\boldsymbol{c}^T\boldsymbol{x}^*$.

Next, to formulate a semidefinite programming, vector space $\mathbb{R}^n$ is replaced with another vector space

$$S_n = \left\{X \in \mathbb{R}^{n\times n} : x_{ij} = x_{ji},\ 1 \leq i < j \leq n\right\}$$

of symmetric $n \times n$ matrices. The matrix A is also replaced with a linear mapping $A : S_n \to \mathbb{R}^m$

The standard inner product $< \boldsymbol{x}, \boldsymbol{y} >$ on $\mathbb{R}^n$ is replaced with the inner product on S_n

$$X \bullet Y = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}y_{ij}$$

It should be noted that $X \bullet Y$ can be also written $\mathrm{Tr}(X^TY)$. In the semidefinite programming, a symmetric matrix is considered as a variable instead of a vector $\boldsymbol{x} \in \mathbb{R}^n$ in linear programming. Consequently, we replace nonnegative constraint

$\boldsymbol{x} \geq \mathbf{0}$ by

$$X \succeq 0.$$

Here, $X \succeq 0$ denotes positive semidefiniteness of a matrix X.

3. Positive Semidefinite Matrices

A matrix $A \in S_n$ is said to be positive semidefinite if its associated quadratic form $\boldsymbol{x}^T A\boldsymbol{x}$ is nonnegative for all $\boldsymbol{x} \in \mathbb{R}^n$.

Here are several equivalent statements for a symmetric matrix A as follows:

PROPOSITION 3.1. *The following statements are equivalent for a symmetric matrix $A \in S_n$.*
(i) *A is positive semidefinite,*
(ii) *all eigenvalues of A are nonnegative, and*
(iii) *there exists a matrix $B \in \mathbb{R}^{n\times n}$ such that $A = B^T B$.*

The statement (iii) in Proposition 3.1 gives a representation of $A = (a_{ij})$ in a form of $a_{ij} = \boldsymbol{v}_i^T \boldsymbol{v}_j$ for all i, j for some vectors $\boldsymbol{v}_i \in \mathbb{R}^n$.

4. Cholesky Factorization

Given a symmetric positive semidefinite matrix A, a matrix B which satisfies Proposition 3.1 (iii) can be obtained in $O(n^3)$ time by Cholesky factorization [6] as follows: If $A = (\alpha) \in \mathbb{R}^{1\times 1}$, we can set $B = (\sqrt{\alpha})$ since $\alpha \geq 0$ by the nonnegativity of eigenvalues. Otherwise, since A is symmetric, we can denote it in the following manner:

$$A = \begin{pmatrix} \alpha & \boldsymbol{p}^T \\ \boldsymbol{p} & U \end{pmatrix}$$

From the statement (i) in Proposition 3.1, $\alpha = \boldsymbol{e}_1^T A\boldsymbol{e}_1 \geq 0$, where $\boldsymbol{e}_i$ is the i-th unit vector in $\mathbb{R}^n$.

Here we classify succeeding discussion into two cases of $\alpha > 0$ and $\alpha = 0$. If $\alpha > 0$, we can describe A as

$$A = \begin{pmatrix} \sqrt{\alpha} & \mathbf{0}^T \\ \frac{1}{\sqrt{\alpha}}\boldsymbol{p} & I_{n-1} \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & U - \frac{1}{\alpha}\boldsymbol{p}\boldsymbol{p}^T \end{pmatrix} \begin{pmatrix} \sqrt{\alpha} & \frac{1}{\sqrt{\alpha}}\boldsymbol{p}^T \\ \mathbf{0} & I_{n-1} \end{pmatrix}.$$

Thus, the matrix $U - \frac{1}{\alpha}\boldsymbol{p}\boldsymbol{p}^T$ is itself positive semidefinite. Consequently, we can recursively obtain a Cholesky factorization

$$U - \frac{1}{\alpha}\boldsymbol{p}\boldsymbol{p}^T = V^T V.$$

Setting matrix B as

$$B = \begin{pmatrix} \sqrt{\alpha} & \frac{1}{\sqrt{\alpha}}\boldsymbol{p}^T \\ \mathbf{0} & V \end{pmatrix},$$

easy calculation leads to $A = B^T B$. So we obtain a Cholesky factorization.

In the other case of $\alpha = 0$, vector $\boldsymbol{p}$ turns out to be $\mathbf{0}$ because we can choose a vector $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{x}^T A \boldsymbol{x} < 0$ if vector $\boldsymbol{p}$ has nonzero component. Since $\boldsymbol{x}^T A \boldsymbol{x} \geq 0$ for $\boldsymbol{x} = (0, x_2, \ldots, x_n)$, the matrix U is itself positive semidefinite. Thus we can recursively obtain a matrix V such that $U = V^T V$. Choosing the matrix B such that

$$B = \begin{pmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & V \end{pmatrix},$$

we lead to the Cholesky factorization $A = B^T B$.

5. Semidefinite Programming Problem

DEFINITION 5.1. *A semidefinite programming problem subject to constraints of equation system is defined as follows:*

$$\begin{aligned} &\textit{maximize} \sum_{i,j=1}^{n} c_{ij} x_{ij} \\ &\textit{subject to} \sum_{i,j=1}^{n} a_{ijk} x_{ij} = b_k \quad (k = 1, \ldots, m) \\ &X \succeq 0 \end{aligned} \tag{1}$$

where

$$X = (x_{ij})_{i,j=1}^{n} \in S_n$$

i.e., x_{ij} are n^2 variables with symmetry constraints $x_{ij} = x_{ji}$ for all i, j, and c_{ij}, a_{ijk}, b_k are all real coefficients.

As the simpler form than Definition 5.1, the semidefinite program can be described as

$$
\begin{aligned}
\text{maximize} \quad & C \bullet X \\
\text{subject to} \quad & A_1 \bullet X = b_1 \\
& A_2 \bullet X = b_2 \\
& \vdots \\
& A_m \bullet X = b_m \\
& X \succeq 0,
\end{aligned}
\tag{2}
$$

where $C = (c_{ij})_{i,j=1}^{n}$ is the coefficient matrix of the objective function, and $A_k = (a_{ijk})_{i,j=1}^{n}$ $(k = 1, 2, \ldots, m)$. Furthermore we can describe the system of m linear constraints $A_i \bullet X = b_i$ $(i = 1, \ldots, m)$ as a whole much more compactly as

$$A(X) = \boldsymbol{b},$$

where $\boldsymbol{b} = (b_1, \ldots, b_m)$, and $A : S_n \to \mathbb{R}^m$ is a linear mapping. In a similar way to the linear programming case, we call the semidefinite program (2) feasible if there exists some feasible solution, which is defined by $\tilde{X} \in S_n$ such that $A(\tilde{X}) = \boldsymbol{b}$, $\tilde{X} \succeq 0$. The value of a feasible semidefinite program is

$$\sup\{C \bullet X \mid A(X) = \boldsymbol{b},\ X \succeq 0\}.$$

If the value of a feasible semidefinite program is ∞, we call the program unbounded. Otherwise, we call it bounded. An optimal solution is a feasible solution X^* that satisfies $C \bullet X^* \geq C \bullet X$ for all feasible solution X.

6. Maximum Cut Problem (MaxCut)

Given an undirected graph $G = (V, E)$ as the input, we define a cut in the graph as a partition of the vertex set V into two disjoint subsets S and its complement $V \setminus S = \overline{S}$ for $S \subset V$, as shown in Fig 1. We denote the cut by $(S, \overline{S})$. The edge set of the cut $(S, \overline{S})$ denoted by $E(S, \overline{S})$ is defined as a set of edges with one vertex in S and the other in $\overline{S}$ as follows:

$$E(S, \overline{S}) = \left\{ e = \{i, j\} \in E \mid i \in S,\ j \in \overline{S} \right\}$$

The number of edges in $E(S, \overline{S})$ described as $|E(S, \overline{S})|$ is called the size of cut $(S, \overline{S})$. The maximum cut problem (MaxCut) is that of finding the set of vertices $S \subset V$ such that the size of cut $(S, \overline{S})$, i.e., $|E(S, \overline{S})|$ is maximized.

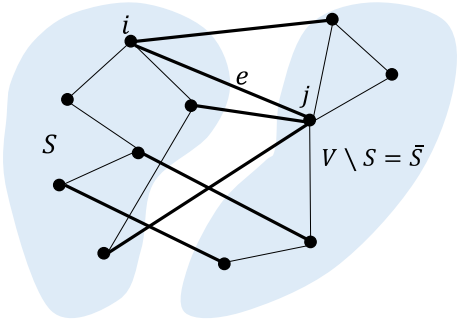

Figure 1. Cut $(S, \overline{S})$

7. The Goemans-Williamson Algorithm

First the MaxCut problem is formulated as a constrained optimization problem. Let $G = (V, E)$ be a given graph, where assume that vertex set $V = \{1, 2, \ldots, n\}$. Next we define variables $z_1, z_2, \ldots, z_n \in \{-1, 1\}$. Any n-tuple $(z_1, z_2, \ldots, z_n) \in \{-1, 1\}^n$ corresponds to a cut $(S, \overline{S})$, where $S = \{i \in V \mid z_i = 1\}$. Conversely, for any cut $(S, \overline{S})$, we can obtain the corresponding n-tuple $(z_1, z_2, \ldots, z_n) \in \{-1, 1\}^n$ by $z_i = 1$ (if $i \in S$), $z_i = -1$ (otherwise) for $i = 1, \ldots, n$. The contribution of edge $\{i, j\}$ to a cut $(S, \overline{S})$ is easily calculated by $(1 - z_i z_j)/2$. Finally we can formulate the MaxCut problem as follows:

$$\begin{aligned} &\text{maximize} \sum_{\{i,j\} \in E} \frac{1 - z_i z_j}{2} \\ &\text{subject to } z_i \in \{-1, 1\} \quad (i = 1, \ldots, n) \end{aligned} \tag{3}$$

The maximum value of this problem denoted by $\mathrm{Opt}(G)$ is the size of a maximum cut. To solve exactly this optimization problem in polynomial time cannot be expected because of NP-completeness of MaxCut problem [3].

8. Semidefinite Programming Relaxation

We replace each real variable z_i with a vector variable $\boldsymbol{u}_i \in S^{n-1} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x}\| = 1\}$. With this replacement, we formulate a semidefinite program whose value can be used as an upper bound for the value $\mathrm{Opt}(G)$ of (3) as follows:

$$\begin{aligned} &\text{maximize} \sum_{\{i,j\}\in E} \frac{1 - \boldsymbol{u}_i^T \boldsymbol{u}_j}{2} \\ &\text{subject to } \boldsymbol{u}_i \in S^{n-1} \ \ (i = 1, 2, \ldots, n). \end{aligned} \tag{4}$$

Since the set $\{-1, 1\}$ can be embedded into S^{n-1} by the mapping $\{-1, 1\} \ni x \mapsto (0, \ldots, 0, x)$, for every solution of (3), we can find a solution of (4) having the same value. This indicates that program (4) is a relaxation of (3). Thus the program (4) has value at least $\mathrm{Opt}(G)$, while this value is finite because $\boldsymbol{u}_i^T \boldsymbol{u}_j \geq -1$ holds for all i, j.

Furthermore, another variable substitution by $x_{ij} = \boldsymbol{u}_i^T \boldsymbol{u}_j$ leads to a semidefinite program:

$$\begin{aligned} &\text{maximize} \sum_{\{i,j\}\in E} \frac{1 - x_{ij}}{2} \\ &\text{subject to } x_{ii} = 1 \ \ (i = 1, 2, \ldots, n) \\ &X \succeq 0 \end{aligned} \tag{5}$$

Proposition 8.1. [4]
Problem (4) is equivalent to semidefinite program (5).

Proof. First assume that $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ is a feasible solution to (4). Let the matrix U to be an $n \times n$ matrix whose columns are $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$. They are unit vectors. Setting $x_{ij} = \boldsymbol{u}_i^T \boldsymbol{u}_j$, we have $X = (x_{ij}) = U^T U$. Such a matrix X is positive semidefinite, and $x_{ii} = 1$ because of $\boldsymbol{u}_i \in S^{n-1}$ for all i. Thus X is a feasible solution to (5) having the same value.

Conversely, every feasible solution X to (5) yields a solution to (4) with the same value. First it should be noted that positive semidefinite matrix X can be represented by $X = U^T U$ as described in Section 4. For a feasible solution X to (5), the columns $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ of matrix U yield a feasible solution to (4) because they are unit vectors by $x_{ii} = 1$. This concludes the proof. □

Accordingly, the semidefinite program (5) has the same finite value $\mathrm{SDP}(G) \geq \mathrm{OPT}(G)$ as (4). So we can find a matrix $X^* \succeq 0$ satisfying $x^*_{ii} = 1$ and

$$\sum_{\{i,j\}\in E} \frac{1 - x^*_{ij}}{2} \geq \mathrm{SDP}(G) - \varepsilon,$$

for every $\varepsilon > 0$ in polynomial time. Also we can obtain a matrix U^* such that $X^* = (U^*)^T U^*$ using a Cholesky factorization of X^* in polynomial time; see Section 4. Finally the columns $\boldsymbol{u}^*_1, \ldots, \boldsymbol{u}^*_n$ of U^* are unit vectors that are optimal solution of (4) satisfying

$$\sum_{\{i,j\}\in E} \frac{1 - \boldsymbol{u}^{*T}_i \boldsymbol{u}_j}{2} \geq \mathrm{SDP}(G) - \varepsilon \geq \mathrm{Opt}(G) - \varepsilon \tag{6}$$

9. Rounding Method

We are actually required to solve problem (3) where n variables $z_i = 1$, or $-1 \in S^0$ $(i = 1, \ldots, n)$. From the solution, a cut $(S, \overline{S})$ can be determined through $S = \{i \in V \mid z_i = 1\}$. We have an almost optimal solution of the relaxed problem (4) which n vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in S^{n-1}$ constitute. So we must map vectors $\boldsymbol{u}_i \in S^{n-1}$ back to S^0 so that the loss of the optimality is as small as possible. Since it is required to classify vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ into 1 or -1, choosing $\boldsymbol{p} \in S^{n-1}$, we define the mapping as follows:

$$\boldsymbol{u} \mapsto \begin{cases} 1 & \text{if } \boldsymbol{p}^T \boldsymbol{u} \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{7}$$

As ilustrated in Fig. 2, $\boldsymbol{p}$ divides S^{n-1} into a closed hemisphere $H = \{\boldsymbol{u} \in S^{n-1} \mid \boldsymbol{p}^T \boldsymbol{u} \geq 0\}$ and its complement $S^{n-1} \setminus H$. Vectors $\boldsymbol{u}_i \in H$ are mapped to 1, while vectors in the complement of H mapped to -1. We choose vector $\boldsymbol{p}$ at random so that it is uniformly distributed on S^{n-1}. This is called randomized rounding. A pair of vectors $\boldsymbol{u}^*_i$ and $\boldsymbol{u}^*_j$ having large value of $(1 - \boldsymbol{u}^{*T}_i \boldsymbol{u}^*_j)/2$ is more likely to yield a cut edge $\{i, j\}$ than a pair of small value so as to maximize the size of the cut. Since this contribution to the cut size depends on the angle between $\boldsymbol{u}^*_i$ and $\boldsymbol{u}^*_j$, the randomized rounding by the vector $\boldsymbol{p}$ should more likely to map pairs with large angles to different values in $\{-1, 1\}$ than pairs with small angles.
The lemma below validates this consideration.

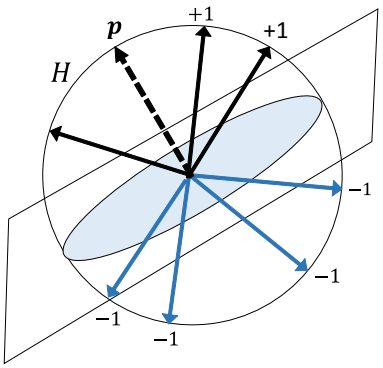

Figure 2. Rounding vectors in S^{n-1} to $\{-1, 1\}$ with a given vector $\boldsymbol{p} \in S^{n-1}$

LEMMA 9.1. *For* $\boldsymbol{u}$,$\boldsymbol{u}' \in S^{n-1}$, *the probability that mapping (7) maps* $\boldsymbol{u}$ *and* $\boldsymbol{u}'$ *to different values in* $\{-1, 1\}$ *is*

$$\frac{1}{\pi} \arccos \boldsymbol{u}^T \boldsymbol{u}'$$

PROOF. Let the angle between the unit vectors $\boldsymbol{u}$ and $\boldsymbol{u}'$ be $\alpha \in [0, \pi]$. So $\cos \alpha = \boldsymbol{u}^T \boldsymbol{u}' \in [-1, 1]$ or equivalently $\alpha = \arccos \boldsymbol{u}^T \boldsymbol{u}'$.

If $\alpha = 0$ or $\alpha = \pi$, then $\boldsymbol{u} = \boldsymbol{u}'$ or $\boldsymbol{u} = -\boldsymbol{u}'$. So the statement is trivially holds. Otherwise, let L be the linear span of $\boldsymbol{u}$ and $\boldsymbol{u}'$, and $\boldsymbol{r}$ be the orthogonal projection of $\boldsymbol{p}$ to that linear space. Thus $\boldsymbol{p}^T \boldsymbol{u} = \boldsymbol{r}^T \boldsymbol{u}$ and $\boldsymbol{p}^T \boldsymbol{u}' = \boldsymbol{r}^T \boldsymbol{u}'$ hold. This indicates that $\boldsymbol{u}$ and $\boldsymbol{u}'$ are mapped to different values if and only if $\boldsymbol{r}$ falls on a "half-open double wedge" W with angle α as shown in Fig 3. Since the distribution of $\boldsymbol{p}$ on S^{n-1} is uniform, the distributuin of $\boldsymbol{r}$ on $[0, 2\pi]$ is uniform. Consequently, the probability of $\boldsymbol{r}$ falling on the double open-wedge is $2\alpha/2\pi = \alpha/\pi = \arccos \boldsymbol{u}^T \boldsymbol{u}'/\pi$. □

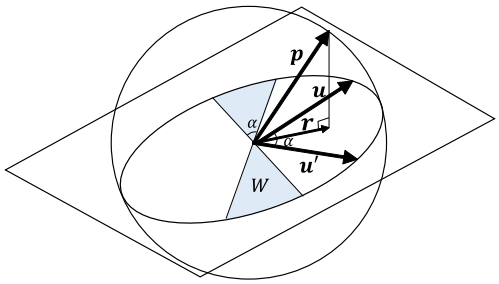

Figure 3. Randomized rounding and half-open double wedge W.

10. Bounding Maximum Cut

Applying the randomized rounding described in Section 9, we can obtain the expectation of the number of edges in the resultant cut as follows:

$$\sum_{\{i,j\}\in E} \frac{\arccos \boldsymbol{u}_i^{*T}\boldsymbol{u}_j^*}{\pi},$$

which is immediately obtained from Lemma 9.1. Although we cannot evaluate to what extent the expectation above itself reaches, we can estimate its degree of approximation using the inequality below, which was previously described as (6).

$$\sum_{\{i,j\}\in E} \frac{1-\boldsymbol{u}_i^{*T}\boldsymbol{u}_j^*}{2} \geq \mathrm{Opt}(G) - \varepsilon$$

The following lemma enables us to compare two terms above.

LEMMA 10.1. *For all* $x \in [-1, 1]$,

$$\frac{\arccos(x)}{\pi} \geq K\frac{1-x}{2}$$

holds for $K = 0.8785672$.

PROOF. For x=1, target inequality holds as right- and left-hand sides of it equal 0. It also holds for $x = -1$. For $x \in (-1, 1)$, we define function f as follows:

$$f(x) = \frac{2\arccos(x)}{\pi(1-x)}$$

For $x \in (-1, 1)$, the derivative calculation of $f(x)$ leads to

$$f'(x) = \frac{2}{\pi} \cdot \frac{\arccos(x) - \sqrt{\frac{1-x}{1+x}}}{(1-x)^2}$$

Here, setting $g(x) = \arccos(x) - \sqrt{\frac{1-x}{1+x}}$, we can obtain $g'(x)$ as

$$g'(x) = -\frac{1}{\sqrt{1-x^2}} \cdot \frac{x}{1+x}$$

Note that $g'(0) = 0$, $g'(x) > 0$ $(-1 < x < 0)$ and $g'(x) < 0$ $(0 < x < 1)$. Thus, at $x = 0$, the function $g(x)$ reaches the maximum $g(0) = \pi/2 - 1 > 0$ as shown in Fig.4. Since $g(1) = 0$, $g(x) \to -\infty$ $(x \to -1+0)$ and $g(0) > 0$, the curve $g(x)$ crosses x-axis at some point $x = x_0$ $(-1 < x_0 < 0)$. Applying Newton method to calculate the cross point, we obtained $x_0 = -0.68915\ldots$, where $f(x_0) = 0.8785672\ldots$. Function $f(x)$ is monotone decreasing on $[-1, x_0]$, while it is monotone increasing on $[x_0, 1]$. Thus, function $f(x)$ has the minimum at $x = x_0$ on $[-1, 1)$ as shown in Fig.5

□

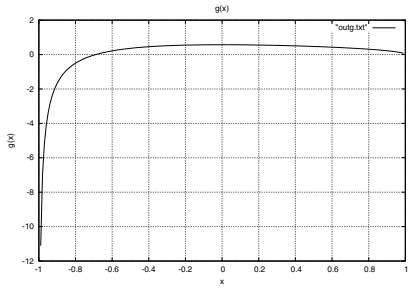

Figure 4. The function $g(x) = \arccos(x) - \sqrt{\frac{1-x}{1+x}}$.

From this lemma, the expected number of cut edges obtained by the algorithm being discussed satisfies

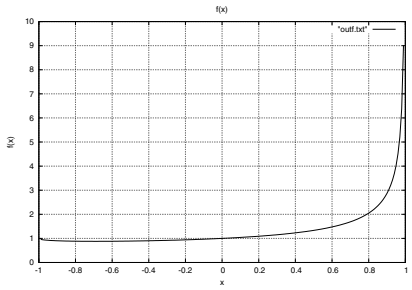

Figure 5. The function $f(x) = \frac{2\arccos(x)}{\pi(1-x)}$.

$$\begin{aligned}\sum_{\{i,j\}\in E} \frac{\arccos \boldsymbol{u}_i^{*T}\boldsymbol{u}_j^*}{2} &\geq 0.8785672 \sum_{\{i,j\}\in E} \frac{1-\boldsymbol{u}_i^{*T}\boldsymbol{u}_j^*}{2} \\ &\geq 0.8785672(\mathrm{Opt}(G)-\varepsilon) \\ &\geq 0.878\mathrm{Opt}(G)\end{aligned}$$

for $0 < \varepsilon \leq 6 \cdot 10^{-4}$.

Finally, the Goemans-Williamson algorithm for approximating the maximum cut of a graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is summarized as follows:

(1) Find an almost optimal solution $\boldsymbol{u}_1^*, \boldsymbol{u}_2^*, \ldots, \boldsymbol{u}_n^*$ of the following problem:

$$\begin{aligned}&\text{maximize} \sum_{\{i,j\}\in E} \frac{1-\boldsymbol{u}_i^T\boldsymbol{u}_j}{2} \\ &\text{subject to } \boldsymbol{u}_i \in S^{n-1} \quad (i = 1, 2, \ldots, n).\end{aligned}$$

in polynomial time using semidefinite programming and Cholesky factorization. The solution satisfies

$$\sum_{\{i,j\}\in E} \frac{1-\boldsymbol{u}_i^{*T}\boldsymbol{u}_j^*}{2} \geq \mathrm{SDP}(G) - 6\cdot 10^{-4} \geq \mathrm{Opt}(G) - 6\cdot 10^{-4}.$$

(2) Let $\boldsymbol{p} \in S^{n-1}$ be chosen uniformly at random on S^{n-1}. The cut $(S, \overline{S})$ is

determined by

$$S = \{i \in \{1, 2, \ldots, n\} \mid \boldsymbol{p}^T \boldsymbol{u}_i^* \geq 0\}$$

11. Conclusions

This paper reviewed semidefinite programming approach to combinatorial optimization, which centered not on solving algorithm but on its formulation and application methods to combinatorial optimization problems. As an example, we adopted the Goemans-Williamson maximum cut algorithm. This algorithm helps us take a survey of application methods of semidefinite programming. To solve a variety of combinatorial optimization problems, an effective formulation of them as semidefinite programing is required more than ever.

References

[1] D.P. Bertsekas, Nonlinear programming, Second Edition, Athena Scientific, 1999.
[2] V. Chvátal, Linear programming, W.H. Freeman and Company, 1983.
[3] M.R. Garey and D.S. Johnson, Computers and intractability, A guide to the theory of NP-completeness, W.H. Freeman and Company, 1979.
[4] B. Gärtner and J. Matoušek, Approximation algorithms and semidefinite programming, Springer, 2012.
[5] M.X. Goemans and D.P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, Journal of the ACM, **42**, (1995), 1115–1145.
[6] G.H. Golub and C.F. Van Loan, Matrix computation, 4th Edition, The Johns Hopkins University Press, 2013.
[7] D.G. Luenberger, Linear and nonlinear programming, Second Edition, Kluwer Academic Publishers, 2003.
[8] L. Tunçel, Polyhedral and semidefinite programming methods in combinatorial optimization, The Fields Institute Monographs, 27, 2010.
[9] L. Vandenberghe and S. Boyd, Semidefinite programming, Siam Review, (1995), 49-95.
[10] L.-T. Wang, Y.-W. Chang, and K.-T. Cheng, Electronic design automation, synthesis, verification, and test, Elsevier, 2009.

Hiroshi Miyashita

Department of Information Engineering, Faculty of Environmental Engineering, The University of Kitakyushu
1-1, Hibikino, Kitakyushu, 808-0135, Japan
E-mail: miyash1225@gmail.com