Semidefinite Programming

Kalavasis Alvertos - Skiadopoulos Athinagoras

National Technical University of Athens

kalabasis.al@gmail.com, athinagoras.sk@gmail.com

June 7, 2018

Overview

General Framework

- SDP is another problem class for convex optimization.
- SDP is closely related to LP. All linear programs can be expressed as SDP's.
- Convex Programs $\supset$ SDP $\equiv$ Vector Programs $\supset$ LP $\supset$ ILP.
- SDP has applications in fields like convex constrained optimization, combinatorial optimization (graph problems) and control theory. It can be used both in proofs and in the design of approximation algorithms for NP-hard problems using the schema of SDP relaxation and Rounding.
- Quadratic approximations are better than linear approximations.

Linear Programming Review

The standard form of an LP program is :

$$LP : min \sum_{i=1}^{n} c_i x_i = min \vec{c}^{\,T} \bullet \vec{x}$$
$$s.t. : \vec{a_j} \bullet \vec{x} = \vec{b_j}, \quad j \in \{1, 2, ..., k\} = [k]$$
$$\vec{x} \in \mathbb{R}^n_{\geq 0}$$

As we can see the objective function is linear subject to linear (in)equality constraints generating a convex polytope as feasible region.

Convex Programming Review

Let $\mathbb{X}$ be a real-valued vector space and let $\mathcal{S} \subset \mathbb{X}$ be a convex set. We consider a set of convex functions $f_i : \mathcal{S} \to \mathbb{R}, i \in \{0\} \cup [k]$,
i.e., satisfy :

$$f_i(\alpha \mathbf{x} + \beta \mathbf{y}) \leq \alpha f_i(\mathbf{x}) + \beta f_i(\mathbf{y}) \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{S}, \alpha, \beta \in \mathbb{R}_{\geq 0} \quad \textit{with } \alpha + \beta = 1$$

Then a convex optimization problem is one of the form :

$$\begin{gathered} CP : min \;\; f_0(\mathbf{x}) \\ s.t. : f_i(\mathbf{x}) \leq \mathbf{b_i}, \quad i \in [k] \\ \mathbf{x} \in \mathcal{S} \end{gathered}$$

- Why Convex ? Because local minimum must be global minimum.
- So LP is a special case of Convex Programming (CP). In CP the objective function and the constraints have to be convex. In LP,they are both convex and concave, and so they are linear.

Connection between LP, SDP and CP

So, we have seen that $LP \subset CP$. In the middle of these two problem classifications, the SDP rises.

$$LP \subset SDP \subset CP$$

- As we will see, the SDP looks remarkably similar to LP. In SDP, we have a linear objective function too and the crucial difference is found in the constraints.
- SDP is a generalization of LP and so we can think that we broaden our workspace from variables to vectors relaxing our constraints as we will see. We relax a LP to a Vector program.
- In LP, our vector $\mathbf{x} = (x_1, ..., x_n)$ must lie in $R^n_{\geq 0}$ and so it must consist of n non-negative variables.
- In SDP, we replace the nonnegativity constraints on the n real variables $x_1, ..., x_n$ of LP with the positive semidefiniteness constraints on n vector variables that consist an array $X = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$.

From LP space to SDP space

- It is helpful to think our matrix variable X in SDP as a vector of n^2 variables s.t. $(x_{11}, x_{12}, ..., x_{ij}, ..., x_{nn}) \in \mathbb{R}^{n^2}$.
- The constraint in LP that $\mathbf{x}$ lies in $\mathbb{R}^n_{\geq 0}$ is now replaced with the constraint that X lies in the cone of positive semidefinite matrices $\mathcal{S}^n_+$, i.e., the set of all symmetric positive semidefinite matrices of dimension $n \times n$.
- So the fact that "$\mathbf{x} \geq \mathbf{0}$" (meaning that each of the n variables must be nonnegative), we can think that in SDP, we have the constraint "$X \succeq 0$" stating that X must be pos. semidefinite (each of the n eigenvalues of X must be nonnegative).
- So we broaden our space from the nonnegative orthant $R^n_{\geq 0}$ to the $\mathcal{S}^n_+$ by replacing nonnegativity of variables to nonnegativity of eigenvalues. The space remains convex since the cone $\mathcal{S}^n_+$ is a proper cone (i.e. closed, convex, pointed and solid).
- Now we have to think of a linear objective function of the matrix X.

Linear Algebra Overview

Definition (Positive Semidefinite Matrix (PSD))

A symmetric $n \times n$ matrix A is called PSD if $\mathbf{x}^T A\mathbf{x} \geq 0 \ \ \forall \mathbf{x} \in \mathbb{R}^n$. The set of PSD matrices is denoted by $\mathcal{S}_+^n$.

Theorem (PSD equivalences)

The following statements are equivalent :

- *The symmetric matrix A is PSD ($A \succeq 0$).*
- *All eigenvalues of A are nonnegative.*
- *There exists U s.t. $A = UU^T$ (Cholesky decomposition).*

Theorem (Cone)

Let C be a subset of a vector space $\mathbb{V}$. We say that C is a (linear) cone if $\forall \mathbf{x} \in C$ and $\alpha > 0$, the product $\alpha \mathbf{x} \in C$.

The cone $\mathcal{S}_+^n = \{A \in Sym(\mathbb{R}_{n \times n}) : A \succeq 0\}$ is a closed convex cone in $\mathbb{R}^{n^2}$.

Linear Algebra Overview

Let $\mathbb{S}^n = \{X_{n \times n} : X^T = X, x_{ij} \in \mathbb{R}\}$ the set of $n \times n$ symmetric matrices. We now make $\mathbb{S}^n$ an inner-product space $(\mathbb{S}^n, <, >)$ by defining :

$$< A, B >=< A, B >_{\mathbb{S}^n} := tr(A^T B) = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} B_{ij}$$

We can think of the inner product $<, >_{\mathbb{S}^n}$ as a matrix inner product by compressing the columns of A and the rows of B and getting the normal vector inner product.
Working with that inner product space, we can define an SDP by expressing the linear objective function of X as $< C, X >$. Note that if X is a symmetric matrix, there is no loss of generality in assuming that the matrix C is also symmetric.

- Let $X \in \mathbb{S}^n$. Then $X \succeq 0$ iff $< A, X > \geq 0, \forall A \succeq 0$.

SDP Definition

Let $X \in \mathbb{S}^n$. Then a SDP is an optimization of the following form :

$$\begin{gathered} SDP : min < C, X > \\ s.t. :< A_i, X >= b_i, \quad i \in [k] \\ X \succeq 0 \end{gathered}$$

- The matrix variable has to lie in the closed convex cone $\mathcal{S}^n_+$.
- The data of the SDP consists of the symmetric matrix C and the finite collection $\{(A_i, b_i)\}$ that defines the k linear equations.
- $C, A_i \in \mathbb{S}^n, \forall i \in [k]$.

LP is a special case of SDP

Consider a LP problem :

$$LP : min \;\; \mathbf{c}^T \bullet \mathbf{x}$$
$$s.t. : \mathbf{a_i} \bullet \mathbf{x} = \mathbf{b_i}, \quad i \in [k]$$
$$\mathbf{x} \in \mathbb{R}^n_{\geq 0}$$

Define : $A_i = \begin{bmatrix} a_{i1} & & \\ & \ddots & \\ & & a_{in} \end{bmatrix} = Diag(a_{i1}, ..., a_{in}), i \in [k]$ and $C = Diag(c_1, ..., c_n)$, then the LP can be written as a SDP problem of the form :

$$SDP_{LP} : min < C, X >$$
$$s.t. :< A_i, X >= b_i, \quad i \in [k]$$
$$X_{ij} = 0, \quad i \in [n], j > i$$
$$X (= Diag(x_1, ..., x_n)) \succeq 0$$

SDP Duality

The dual problem of SDP is :

$$SDD : max \sum_{i=1}^{k} b_i y_i$$
$$s.t. : \sum_{i=1}^{k} A_i y_i + S = C$$
$$S \succeq 0$$

or equivalently : $S \succeq 0 \Rightarrow C - \sum_{i=1}^{k} A_i y_i \succeq 0.$
$S, A_i, C \in \mathbb{S}^n$

Feasible solutions on SDP

Definition (Feasible solution)

A matrix in $\mathbb{R}_{n\times n}$ satisfying all the constraints of SDP is a feasible solution of SDP.

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 $A, B \in \mathbb{R}_{n\times n}$ are feasible solutions then so is any convex combination of these solutions.

SDP Weak Duality

Theorem (Weak Duality of SDP)

Given a feasible solution X of SDP and a feasible solution (y,S) of SDD, the duality gap is $< C, X > - \sum_{i=1}^{k} b_i y_i =< S, X > \geq 0$. *If* $< S, X >= 0$, *then* X *and* (y, S) *are each optimal solutions of SDP and SDD, respectively, and furthermore,* $SX = 0$

On weak and strong duality

Consider the SDP $p^* = min_{X \in \mathcal{S}_+^n} < C, X > s.t. < A_i, X >= b_i, i \in [m]$ and its dual $d^* = max_{\mathbf{y}} \mathbf{y}^T \mathbf{b} s.t. \sum_{i=1}^{m} y_i A_i \succeq C$. The following holds :
1)Duality is symmetric, in the sense that the dual of the dual is the primal.
2)Weak duality always holds : $< C, X > \geq \sum_{i=1}^{k} b_i y_i$.
3)If the primal (resp. dual) problem is bounded below (resp. above), and strictly feasible, then $p^* = d^*$ and the dual (resp. primal) is attained.
4)If both problems are strictly feasible, then $p^* = d^*$ and both problems are attained.

Properties of LP that do not extend to SDP

- There may be a finite or infinite duality gap. The SDP and/or SDD may or may not attain their optima. Both SDP and SDD will attain their common optimum if both programs have feasible solutions in the interior of the semidefinite cone.
- There is no direct analog of a "basic feasible solution" for SDP. There is no finite algorithm for solving SDP.
- Contrarily to linear optimization problems, SDPs can fail to have a zero duality gap, even when they are feasible. Consider the example :
$p^* = min_{\mathbf{x}} x_2$ s.t. $\begin{pmatrix} x_2+1 & 0 & 0 \\ 0 & x_1 & x_2 \\ 0 & x_2 & 0 \end{pmatrix} \succeq 0$. We have $p^* = 0$.
The dual is
$d^* = max_{Y \in \mathcal{S}^3_+}(-Y_{11}) : Y \succeq 0, Y_{22} = 0, 1 - Y_{11} - 2Y_{23} = 0$ Any dual feasible Y satisfies $Y_{23} = 0$ (since $Y_{22} = 0$), thus $Y_{11} = -1 = d^*$.

Applications of SDP

- We have already seen a framework for creating approximation algorithms :
  $ILP \rightarrow LP$ and then Rounding (f.e. Set Cover and $x_i > \frac{1}{f}$)
- By formulating an optimization problem in terms of SDP, we can create approx. algorithms in the same way. For example, MAXCUT is NP-hard. We formulate this problem as ILP/IQP and we approximate using the scheme :
  $QP \rightarrow SDP(\equiv VP)$ and then Randomized Rounding.
- QP : Quadratic Programming, VP : Vector Programming.
- A QP problem is defined as :

$$QP : min \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{c}^T \mathbf{x}$$
$$s.t. : A\mathbf{x} \leq \mathbf{b}$$
$$Q \in \mathbb{S}^n$$

MAXCUT

Definition (MAXCUT on unweighted G)

Given the undirected unweighted graph $G = (V, E)$, find $max_{\mathcal{U} \subseteq V} |E(\mathcal{U}, V \setminus \mathcal{U})|$.

We can define the ILP version of MAXCUT as following :

Let $X_u = \mathbb{1}[u \in \mathcal{U}] = \begin{cases} 1, & if \;\; u \in \mathcal{U} \\ 0, & otherwise \end{cases}$. Then :

$$ILP : \frac{1}{2} max \sum_{(u,v) \in E} M_{uv}$$

$$s.t. : M_{uv} \leq X_u + X_v, \quad \forall u, v$$

$$M_{uv} \leq (1 - X_u) + (1 - X_v), \quad \forall u, v$$

$$M_{uv}, X_u \in \{0, 1\} \forall u, v$$

M_{uv} captures whether edge (u,v) is in cut.

MAXCUT

Definition (MAXCUT on weighted G)

Given the undirected graph $G = (V, E)$, with edge weights $w : E \to Q^+$, find a partition $(S, \bar{S})$ of V, so as to maximize the total weight of edges in this cut.

MAXCUT is NP-hard. We want to find an approximation algorithm for MAXCUT.

$$MAXCUT\_QP : max \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - y_i y_j)$$

$$s.t. : y_i \in \{-1, +1\}, \quad i \in [n]$$

This Quadratic Program ($y_i^2 = 1, y_i \in \mathbb{Z}$) models the MAXCUT. Consider the cut $U = \{i : y_i = -1\}$ and $\bar{U} = \{i : y_i = +1\}$. If an edge (i, j) is in this cut, then $y_i y_j = -1$, otherwise $+1$. So the objective function gives the total weight of the cut. Goal : Find $U, \bar{U}$ that maximize the total weight.

MAXCUT_QP Relaxation

In order to obtain a relaxation, we will allow the variables to be in a higher dimension space. We want to relax the constraints $y_i^2 = 1, y_i \in \mathbb{Z}$. As we saw in SDP, we go from real-valued variables to real vector variables. We call this a Vector Program. So :

$$MAXCUT\_VP : max \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - \mathbf{v}_i \mathbf{v}_j)$$

$$s.t. : \mathbf{v}_i \bullet \mathbf{v}_i = +1, \quad i \in [n]$$

$$\mathbf{v}_i \in \mathbb{R}_n, \quad i \in [n]$$

This program is a relaxation since we can take any feasible solution y of MAXCUT_QP and generate a solution of MAXCUT_VP by setting $\mathbf{v}_i = (y_i, 0, 0, ..., 0)$. We can solve this problem in Poly time. We would like to round the solution to obtain a near-optimal cut.

VP equivalent to SDP

Lemma

Vector program VP is equivalent to semidefinite program SDP.

Proof

We will show that corresponding to each feasible solution to VP, there is a feasible solution to SDP of the same objective function value, and vice versa.

$\Rightarrow$ Let $\{\mathbf{a}_i\}_{i=1}^n$ be a SOL(VP). Construct the matrix $U = (\mathbf{a}_1|...|\mathbf{a}_n)$ whose columns are the n vectors. Then the matrix $A = U^T U$ is a SOL(SDP) with the same objective function value.

$\Leftarrow$ Let matrix A be a SOL(SDP). Then $A \succeq 0$ and, by Cholesky decomposition, $A = U^T U$. Let $\{\mathbf{a}_i\}_{i=1}^n$ be the columns of U. These columns are a SOL(VP) with the same objective function value.

MAXCUT_SDP

So, a SDP relaxation for MAXCUT is :

$$MAXCUT\_SDP : max \frac{1}{2} \sum_{1 \leq i \leq j \leq n} w_{ij}(1 - y_i y_j)$$
$$s.t. : y_i^2 = 1, \quad i \in [n], v_i \in V$$
$$Y \succeq 0, Y \in \mathbb{S}^n$$

Semidefinite programs, and consequently vector programs, can be solved within an additive error of ϵ, for any $\epsilon > 0$, in time polynomial in n and $log(1/\epsilon)$, using the ellipsoid algorithm (later). Now, we have to use a rounding algorithm for getting a MAXCUT approximation.

Randomized rounding algorithm for MAXCUT

- Let us assume that we have OPT(MAXCUT_VP). Let $\{\mathbf{a}_i\}_{i=1}^{n}$ be an optimal solution with value OPT_{VP}.
- These vectors lie on the $n-$ dimensional unit sphere S_{n-1}. Let θ_{ij} be the angle between vectors $\mathbf{a}_i$ and $\mathbf{a}_j$.
- This pair's contribution to OPT_{VP} is $\frac{w_{ij}}{2}(1 - cos\theta_{ij})$.
- The closer the angle is to π, the larger the contribution will be. So, we want the graph vertices v_i and v_j to be separated in the cut $(S, \bar{S})$ if θ_{ij} is large.

Idea : Pick $\mathbf{r}$ to be a uniformly distributed vector on S_{n-1} and let $S = \{v_i : \mathbf{a}_i \bullet \mathbf{r} \geq 0\}$.

Randomized rounding algorithm for MAXCUT

Lemma (Cutting Probability)

$\mathbb{P}[v_i \in S, v_j \in \bar{S}] = \frac{\theta_{ij}}{\pi}$.

Proof

Project $\mathbf{r}$ onto the plane containing $\mathbf{a}_i$ and $\mathbf{a}_j$. If the projection lies in one of the two arcs of angle θ_{ij}, and the two vertices will be separated and since, $\mathbf{r}$ is picked randomly from the S_{n-1}, its projection will be random too.

How can we pick a random vector from the sphere S_{n-1}?
Pick independently $x_1, ..., x_n$ from the distribution $\mathcal{N}(0, 1)$. Then, if we normalize each x_i with the vector's $\mathbf{x} = (x_1, ..., x_n)$ $\mathcal{L}_2$ norm, we get a random unit vector on S_{n-1}.

- $\mathcal{L}_2(\mathbf{x}) = \sqrt{\sum x_i^2}$.

Randomized rounding algorithm for MAXCUT

MAXCUT - Algorithm

1. Solve VP/SDP. Let $\{\mathbf{a}_i\}_{i=1}^n$ be an OPT(VP) solution.
2. Pick $\mathbf{r}$ to be a uniformly distributed vector on S_{n-1}.
3. Let $S = \{v_i : \mathbf{a}_i \bullet \mathbf{r} \geq 0\}$.
4. Output $Weight(E(S, \bar{S})) = W$.

- The output W is a random variable that corresponds the weight of edges in the cut.
- Define the Goemans-Williamson constant :

$$\alpha_{GW} = \frac{2}{\pi} min_{0 \leq \theta \leq \pi} \frac{\theta}{1 - cos\theta} > 0.87856.$$

Randomized rounding algorithm for MAXCUT

Lemma

$\mathbb{E}[W] \geq \alpha_{GW} OPT_{VP}$.

Proof

- $\forall \theta \in [0, \pi]$, $\frac{\theta}{\pi} \geq \alpha_{GW} \frac{1-cos\theta}{2}$. (From definition of the constant)
- $\mathbb{E}[W] = \sum_{1 \leq i \leq j \leq n} w_{ij} \mathbb{P}[v_i, v_j \textit{ separated}]$.
- From Lemma (Cutting Probability) :
  $\mathbb{E}[W] = \sum_{1 \leq i \leq j \leq n} w_{ij} \frac{\theta_{ij}}{\pi} \geq \alpha_{GW} \sum w_{ij} \frac{1-cos\theta}{2} = \alpha_{GW} OPT_{VP}$.

Theorem (0.87856 - approximation for MAXCUT)

There is a randomized approximation algorithm for MAXCUT achieving an approximation factor of 0.87856.

Randomized rounding algorithm for MAXCUT

Proof of Thm(0.87856 - approximation for MAXCUT)

- Let $T = \sum_{e \in E} w(e)$ and define $\beta : \mathbb{E}[W] = \beta T$.
- Let $p = \mathbb{P}[W < (1-\epsilon)\beta T]$, for some constant $\epsilon > 0$.
- Since the random variable W is upper bounded by T, $\mathbb{E}[W] = \beta T \leq p(1-\epsilon)\beta T + (1-p)T$.
- So, $p \leq \frac{1-\beta}{1-\beta+\beta\epsilon}$.
- By the previous lemma and because of VP relaxation, $T \geq \beta T = \mathbb{E}[W] \geq \alpha_{GW} OPT_{VP} \geq \alpha_{GW} OPT \geq \frac{\alpha T}{2}$.
- So, $\frac{\alpha_{GW}}{2} \leq \beta \leq 1$ and we get $p \leq 1 - \frac{\epsilon\alpha_{GW}/2}{1+\epsilon-\alpha_{GW}/2} = 1 - c$.
- Run our MAXCUT rounding algorithm $1/c$ times, and output the max weight cut founded in these runs. Then :
- $\mathbb{P}[maxW \geq (1-\epsilon)\beta T] \geq 1 - (1-c)^{1/c} \geq 1 - \frac{1}{e}$.
- Since $\beta T \geq \alpha_{GW} OPT > 0.87856 OPT$, we can choose $\epsilon > 0$ s.t. $(1-\epsilon)\beta T \geq 0.87856 OPT$.

Other applications of SDP

- Eigenvalue optimization :
  Given symmetric $n \times n$ matrices $B, \{A_i\}_{i=1}^{k}$, choose weights $w_1, ..., w_k$ to create a new matrix $S := B - \sum_{i=1}^{k} w_i A_i$ s.t. : $\lambda_{max}(S) - \lambda_{min}(S)$ is minimized.
- 3-colorabilility : We want to color a 3-colorable graph.
  We can color it with $\mathcal{O}(\sqrt{n})$ colors. But, can we do better ?
  We will define $\tilde{\mathcal{O}}$ as follows : A function $g(n) = \tilde{\mathcal{O}}(f(n))$ if there exists some constant $c \geq 0$ and some n_0 such that for all $n \geq n_0$, $g(n) = \mathcal{O}(f(n) log^c n)$.
  By using semidefinite programming, we can obtain an algorithm that uses $\tilde{\mathcal{O}}(n^{0.387})$ colors. The best known algorithm does not use much fewer than $\tilde{\mathcal{O}}(n^{0.387})$ colors
- Satisfiability : There is a polynomial time approximation algorithm for MAX-3-SAT with an approximation ratio of $7/8$.

Ellipsoid algorithm for SDP

- The problem considered by Ellipsoid algorithm is :
  Given a bounded convex set $\mathcal{K} \in \mathbb{R}^n$, find $x \in \mathcal{K}$.
- The algorithm works as follows :

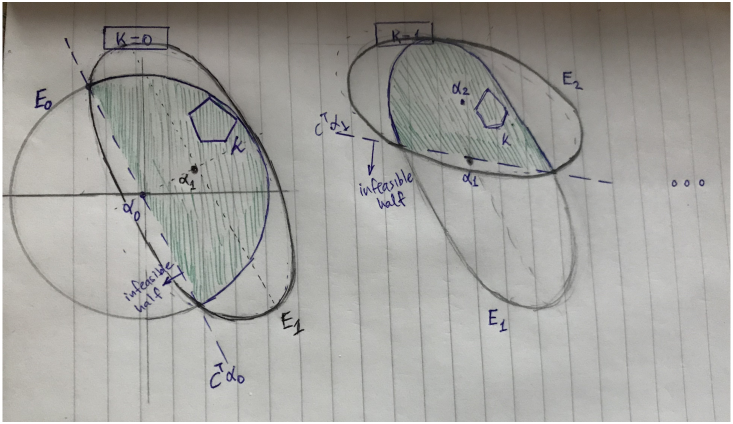
```
1) Let $E_0$ be an ellipsoid containing $\mathcal{K}$
2) While center $a_k$ of $E_k$ is not in $\mathcal{K}$ :
    2a)Let $c^T x \leq c^T a_k$ be s.t. $\{x : c^T x \leq c^T a_k\} \supseteq \mathcal{K}$
    (to chop off infeasible half-ellipsoid).
    2b)Let $E_{k+1}$ be the minimum volume ellipsoid that contains the
    feasible half-ellipsoid $E_k \cap \{x : c^T x \leq c^T a_k\}$
    2c) $k++$;
    2d) Jump to (2)
```

- $\frac{Vol(E_{k+1})}{Vol(E_k)} < e^{-\frac{1}{2(n+1)}}$. (the ellipsoids constructed shrink in volume)

Definition (Ellipsoid)

Given a center α and matrix $A \succ 0$, the ellipsoid $E(\alpha, A)$ is defined as $\{x \in \mathbb{R}^n : (x - \alpha)^T A^{-1}(x - \alpha) \leq 1\}$.

Ellipsoid algorithm for SDP



Ellipsoid algorithm for SDP

As we saw ellipsoid algorithm, finds a feasible solution. How do we connect feasibility to optimization ? We will focus on the most important situation in combinatorial optimization when we are given a set $S \subseteq \{0,1\}^n$ and $\mathcal{K} = convex(S)$.

From Feasibility to Optimization

Let $\mathbf{c}^T\mathbf{x}$ be the objective function we want to minimize over the convex region $\mathcal{K}, \mathbf{c} \in \mathbb{R}^n$. Assume that $\mathbf{c} \in \mathbb{Z}^n$. Instead of optimizing, we can check the non-emptiness of $\mathcal{K}' = \mathcal{K} \cap \{\mathbf{x} : \mathbf{c}^T\mathbf{x} \leq l + \frac{1}{2}\}$ for $l \in \mathbb{Z}$ and our optimum value corresponds to $min l$. Since $S \subseteq \{0,1\}^n$, $-nmax_i c_i \leq l \leq nmax_i c_i$. To find l use binary search with total cost $(\mathcal{O}(log(n) + log(max_i c_i)))$.

References

- Approximation Algorithms by Vijay V. Vazirani
- The Design of Approximation Algorithms by David P. Williamson
- Lecture notes by L. Lovasz *Semidefinite programs and combinatorial optimization*. Microsoft Research, Redmond, WA 98052.

$\mathcal{T}he\ \mathcal{E}nd$