Semi-Definite Programming (SDP)

$$\max \sum_{ij} C_{ij} X_{ij} = \text{Trace}(C \cdot X)$$

$$\text{s.t.} \quad \text{Trace}(A_i \cdot X) = b_i \qquad i \in I$$

$$X \succeq 0 \qquad (X \text{ is a } n \times n \text{ matrix})$$

Thm: Grötschel-Lovász-Schriver (GLS)

Optimization / test feasibility over a convex set $S \subseteq \mathbb{R}^d$ is equivalent to separation over S.

(i.e. we need a separation oracle).

Let $S = \{\Lambda \in \mathbb{R}^{n \times n} \mid \Lambda \succeq 0\} \subseteq \mathbb{R}^{n \times n}$

Separation oracle for S

- given a matrix Q, decide if Q belongs to S

  OR Find a separating hyperplane.

→ find all eigenvalues of Q

→ if $\lambda_i \geq 0 \ \forall i$, then $Q \in S$

→ otherwise $\exists\, y \in \mathbb{R}^n$ s.t.

$$Qy = \lambda y, \quad \lambda < 0$$

$$\Rightarrow y^T Q y = y^T \lambda y = \lambda \|y\|_2^2 < 0$$

→ Separating hyperplane $y^T \Lambda y = 0$

i.e. $y^T \Lambda y \geq 0$ is valid for S.

Therefore, GLS $\Rightarrow$ SDP can be solved (By Ellipsoid Method)

(but we can also do this with interior point methods)

Application in Combinatorial Optimization

Stable Set Problem (NP-Complete)

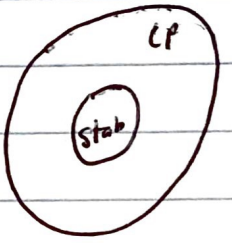
Find a maximum size stable set of vertices where no two vertices are connected by an edge

Prop: S is a stable set $\Leftrightarrow$ V/S is a vertex cover

$$\begin{aligned} \max \quad & \sum x_i \\ \text{s.t.} \quad & x_i + x_j \le 1 \qquad \forall (i,j) \in E \\ & x_i \in \{0,1\} \end{aligned}$$

$$LP(G) = \{x \mid x_i + x_j \le 1 \quad \forall (i,j) \in E,\ 0 \le x_i \le 1\}$$

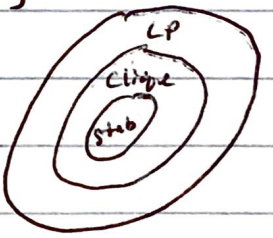
$$Stab(G) = Conv(\{x \mid x_i + x_j \le 1,\ x_i \in \{0,1\}\})$$



Defn: A clique is a set of vertices s.t. $i,j \in E$ forall $i,j \in C$ (the clique)

Observation: For every clique C, at most 1 vertex from C belongs to a stable set.

$$Clique(G) = \{x \mid \sum_{i \in C} x_i \le 1,\ 0 \le x_i \le 1\}$$



Recall that we can write binary variables using quadratic constraints

- one formulation is

$$x_i^2 - x_i = 0$$
$$x_i \cdot x_j = 0 \qquad \forall (i,j) \in E$$

Application in Combinatorial Optimization

Stable Set Problem (NP-Complete)

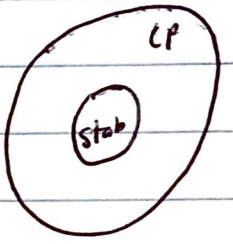
Find a maximum size stable set of vertices where no two vertices are connected by an edge

Prop: S is a stable set $\Leftrightarrow$ V/S is a vertex cover

$$\begin{aligned} \text{Max} \quad & \sum x_i \\ \text{s.t.} \quad & x_i + x_j \leq 1 \qquad \forall (i,j) \in E \\ & x_i \in \{0,1\} \end{aligned}$$

$$LP(G) = \{x \mid x_i + x_j \leq 1 \quad \forall (i,j) \in E, \; 0 \leq x_i \leq 1\}$$

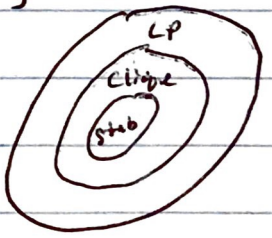
$$Stab(G) = Conv(\{x \mid x_i + x_j \leq 1, \; x_i \in \{0,1\}\})$$



Defn: A clique is a set of vertices s.t. $i,j \in E$ forall $i,j \in C$ (the clique)

Observation: For every clique C, at most 1 vertex from C belongs to a stable set.

$$Clique(G) = \{x \mid \sum_{i \in C} x_i \leq 1, \; 0 \leq x_i \leq 1\}$$



Recall that we can write binary variables using quadratic constraints

- one formulation is

$$x_i^2 - x_i = 0$$
$$x_i \cdot x_j = 0 \qquad \forall (i,j) \in E$$

★ Introduce "linearizing variables" y_{ij}

$y_{ij} = x_i x_j$, $y_{jj} = x_j^2 = x_j$, $y_{0j} = y_{j0} = x_j$, $y_{00} = 1$

$\Rightarrow$ $Y = \begin{pmatrix} 1 \\ x \end{pmatrix} (1 \; x^T)$

★ Y is a positive semidefinite matrix

$\Rightarrow$ Relaxation: $Y \succeq 0$

i.e. need not be that $Y = (\;)(\;)$ for rank 1 matrices.

$\Rightarrow$ cannot recover x from the constraint.

$$TH(G) = \{x \mid Y \succeq 0,\ y_{ij} = 0 \ \forall (i,j) \in E,\ y_{0j} = y_{j0} = y_{jj} = x_j,\ y_{00} = 1,\ 0 \le x_j \le 1\}$$

"Theta body" from Lovász

Theorem

$$Stab(G) \subseteq TH(G) \subseteq Clique(G)$$

proof: Clearly $Stab \subseteq TH(G)$ by construction.

WTS. $TH(G) \subseteq Clique(G)$.

Assume $x \in TH(G)$ $\Rightarrow$ $\exists\ Y \succeq 0$ in $TH(G)$ definition.

$\Rightarrow$ $v^T Y v \ge 0$ $\forall$ $v \in \mathbb{R}^{n+1}$

Take any clique C

set $v_i = \begin{cases} 1 & \text{if } i = 0 \\ -1 & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases}$

Then $0 \le v^T Y v = \sum_{i,j} y_{ij} v_i v_j$

$= \sum_{i,j \in C} y_{ij} \underbrace{v_i v_j}_{(-1)(-1)} + \sum_{i=1}^{n} y_{i0} \underbrace{v_i v_0}_{1} + \sum_{j=1}^{n} y_{0j} \underbrace{v_0 v_j}_{1} + y_{00} \underbrace{v_0^2}_{1}$

$= 0 + \sum_{j=1}^{n} y_{jj} - \sum_{i \in C} x_i - \sum_{i \in C} x_i + 1$

(★ $y_{ij} = 0$ if $(i,j) \in E$)

$\Rightarrow$ $\sum_{i \in C} x_i \le 1$ ☺ i.e. $x \in Clique(G)$ □

★ Introduce "linearizing variables" y_{ij}

$y_{ij} = x_i x_j$, $y_{jj} = x_j^2 = x_j$, $y_{0j} = y_{j0} = x_j$, $y_{00} = 1$

$\Rightarrow$ $Y = \begin{pmatrix} 1 \\ x \end{pmatrix} (1 \; x^T)$

★ Y is a positive semidefinite matrix

$\Rightarrow$ **Relaxation:** $Y \succeq 0$

i.e. need not be that $Y = (\;)(\;)$ for rank 1 matrices.

$\Rightarrow$ cannot recover x from the constraint.

$$TH(G) = \{x \mid Y \succeq 0,\ y_{ij} = 0 \ \forall (i,j) \in E,\ y_{0j} = y_{j0} = y_{jj} = x_j,\ y_{00} = 1,\ 0 \le x_j \le 1\}$$

"Theta body" from Lovász

Theorem

$$Stab(G) \subseteq TH(G) \subseteq Clique(G)$$

proof: Clearly $Stab \subseteq TH(G)$ by construction.

WTS. $TH(G) \subseteq Clique(G)$.

Assume $x \in TH(G)$ $\Rightarrow$ $\exists\ Y \succeq 0$ in $TH(G)$ definition.

$\Rightarrow$ $v^T Y v \ge 0$ $\forall$ $v \in \mathbb{R}^{n+1}$

Take any clique C

set $v_i = \begin{cases} 1 & \text{if } i = 0 \\ -1 & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases}$

Then $0 \le v^T Y v = \sum_{i,j} y_{ij} v_i v_j$

$= \sum_{i,j \in C} y_{ij} \underbrace{v_i v_j}_{(-1)(-1)} + \sum_{i=1}^{n} y_{i0} \underbrace{v_i v_0}_{1} + \sum_{j=1}^{n} y_{0j} \underbrace{v_0 v_j}_{1} + y_{00} \underbrace{v_0^2}_{1}$

$= 0 + \sum_{j=1}^{n} y_{jj} - \sum_{i \in C} x_i - \sum_{i \in C} x_i + 1$

(★ $y_{ij} = 0$ if $(i,j) \in E$)

$\Rightarrow$ $\sum_{i \in C} x_i \le 1$ ☺ i.e. $x \in Clique(G)$ □

Max-Cut

Given a graph $G=(V,E)$ and weights on the edges, find a partition of $V = S \sqcup V/S$ such that the total weight of the edges crossing the partition is maximized.

Goemans-Williamson Approximate Algorithm (SDP)

There is an efficient (polytime) algorithm which gets within 82% of the optimal max cut.

- ∄ any algorithm better than this unless $P=NP$

Formulation: $X_i = \begin{cases} 1 & \text{if } i \in S \\ -1 & \text{if } i \notin S \end{cases}$, $i \in V$.

$X_i \in \{\pm 1\} \iff X_i^2 = 1$

$X_i \neq X_j \iff \frac{1 - X_i X_j}{2} = 1$

$$\Rightarrow \max \sum w_{ij} \left(\frac{1 - X_i X_j}{2}\right)$$
$$\text{s.t. } X_i^2 = 1$$

Max-Cut

Given a graph $G = (V, E)$ and weights on the edges, find a partition of $V = S \sqcup V/S$ such that the total weight of the edges crossing the partition is maximized.

Goemans-Williamson Approximate Algorithm (SDP)

There is an efficient (polytime) algorithm which gets within 82% of the optimal max cut.

- ∄ any algorithm better than this unless $P = NP$

Formulation: $x_i = \begin{cases} 1 & \text{if } i \in S \\ -1 & \text{if } i \notin S \end{cases}$, $i \in V$.

$x_i \in \{\pm 1\} \iff x_i^2 = 1$

$x_i \neq x_j \iff \frac{1 - x_i x_j}{2} = 1$

$$\Rightarrow \max \sum w_{ij} \left(\frac{1 - x_i x_j}{2}\right)$$
$$\text{s.t. } x_i^2 = 1$$

SOS Theory

Defn: f is convex if $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$

$\forall\ x, y \in \mathbb{R}^n,\ \lambda \in [0,1]$

Prop: if f is twice-differentiable, then

$$f \text{ is convex} \iff H_f = \left[\left(\frac{\partial f}{\partial x_i \partial x_j}\right)\right] \succeq 0 \qquad \forall\ x \in \mathbb{R}^n$$

"the Hessian is PSD for all $x \in \mathbb{R}^n$"

Polynomial Matrices $P(x) = [p_{ij}]$ $(m \times m)$

p_{ij}: polynomial in x

Defn: $P(x)$ is a PSD matrix if $P(x)$ is PSD $\forall\ x \in \mathbb{R}^n$

Fact: $P(x)$ is PSD $\iff$ $y^T P(x) y \geq 0$ as an element of $\mathbb{R}[x, y]$

$y \in \mathbb{R}^m$

Defn: $P(x)$ is an SOS matrix if $\exists$ a polynomial matrix $M(x)$

such that $P(x) = M^T(x) M(x)$

Fact: $P(x)$ is SOS $\Rightarrow$ $P(x)$ is PSD

Fact: $P(x)$ is an SOS matrix $\iff$ $y^T P(x) y$ is a sum of squares polynomial.

Example PSD-Matrix not SOS-matrix

$$A(x) = \begin{bmatrix} x_1^2 + 2x_2^2 & -x_1x_2 & -x_1x_3 \\ -x_1x_2 & x_2^2 + 2x_3^2 & -x_2x_3 \\ -x_1x_3 & -x_2x_3 & x_3^2 + 2x_1^2 \end{bmatrix}$$

$A(x)$ is PSD, but $A(x)$ is not SOS

★ $A(x)$ is not a Hessian matrix

since $\frac{\partial}{\partial x_3}\left(\frac{\partial^2 p}{\partial x_1^2}\right) \neq \frac{\partial}{\partial x_1}\left(\frac{\partial p^2}{\partial x_1 \partial x_3}\right)$

Note: if $n=1$, then PSD = SOS

Deciding complexity of a polynomial function:

Defn: $P(x)$ is SOS-convex if $H(x)$ is an SOS-matrix

Clearly SOS-convex $\Rightarrow$ convex

Theorem (Amir Ali & Pablo Parrilo)

$\exists$ a polynomial that is convex, but not SOS-convex.

Note: Deciding $H(x) = \text{SOS}$ is easy since we just show that $y^T H(x) y$ is a SOS polynomial

(use SDP to solve)

Time Complexity n = size of input

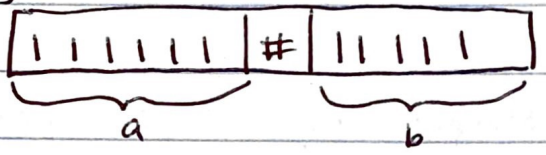
$T(n)$ = # of steps/transitions the turing machine makes

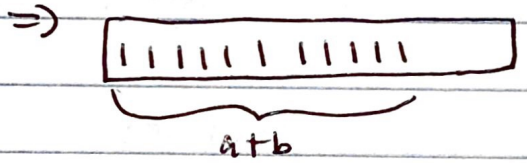
Space complexity

$S(n)$ = amount of storage required

Example: important to consider your input!!!

Adding: $a+b$





"unary encoding"

$T_1(n) = c_1 n$

length $a+b$.

versus:

a base 2 b base 2

a+b base 2

"binary encoding"

$T_2(n) = c_2 n$

only $\log(a) + \log(b)$ length

★ both linear time algorithms, but T_2 is exponentially better than T_1.

Defn: A problem is a set of strings on Σ

An instance is a particular string from the problem

Defn: The class of polynomial time problems P is the set of problems for which there exists a turing machine with polynomial time complexity.

Reduction: Problem B reduces to problem A if for all instances of B there exists a map

$R: B \to A$, $R(b) \in A$, s.t. R is polytime.

Defn: A and B are polynomially equivalent if A reduces to B and B reduces to A.

Example: Vertex Cover

- Optimization asks for the minimum vertex cover
- Feasibility asks for a vertex cover of size $\leq k$

Stable Set

- optimization asks for the maximum stable set
- feasibility asks for stable set of size $\geq n-k$

All of these are polynomially equivalent

Defn: A non-deterministic polynomial time (NP) turing machine has a map

$$f: Q \times \Sigma \to 2^Q \times \Sigma \cup \{e\}$$

it can carry out parallel computations.

Alternative Defn: A problem is NP if there exists a polynomial time algorithm to check a given certificate.

Clearly $P \subseteq NP$. Probably $P \neq NP$.