Approximation Algorithms

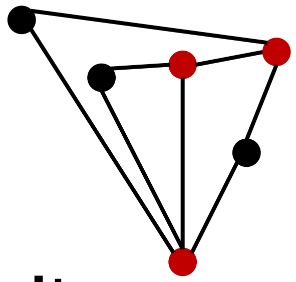
Book: Design of Approximation Algorithms by Williamson and Shmoys
https://www.designofapproxalgs.com/

Approximation Algorithms

- There are many optimization problems which are NP hard
- We don't expect to be able to solve them in polynomial time
- What's the best next thing we can do?
  - Average case algorithms (work well for "most" inputs)
  - Heuristic algorithms (seem to work well but we're not always sure why)
  - Approximation algorithms
- **Approximation algorithms** find a solution which isn't optimal, but we can bound how far it is from the optimum
  - Even though we don't know the optimum!

Vertex Cover

- Let $G = (V, E)$ be an undirected graph. A **vertex cover** in G is a set of vertices $S \subseteq V$ such that every edge touches a vertex in S
- Minimum vertex cover problem: find a minimal set with this property
- This is an NP hard problem
- We want to find, efficiently, a vertex cover which isn't necessarily minimal, but also not too large

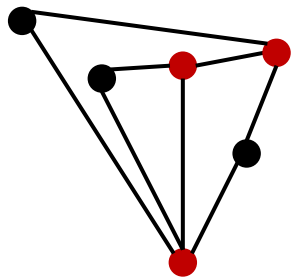


הגדרה לכיסוי בקודקודים:
כל קשת נוגעת
לפחות בקודקוד אחד

Vertex Cover

While $E \neq \emptyset$:

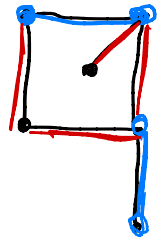
1. Pick an arbitrary edge (u, v)
2. Add **both** u and v to the cover
3. Delete all edges incident to either u or v



Claim 1: this algorithm returns a vertex cover.

Proof: Let (u, v) be an edge in G. Since at the end of the algorithm $E = \emptyset$, either u or v were added.

Vertex Cover



Claim 2: let OPT denote the minimal size vertex cover in G and ALG denote the size of the set returned by the algorithm. Then $\mathrm{OPT} \leq \mathrm{ALG} \leq 2\mathrm{OPT}$.

Proof:

Suppose the algorithm makes k iterations. Then $\mathrm{ALG} = 2k$. The set of vertices that the algorithm returns in a matching. Covering just those edges requires k vertices, hence $\mathrm{OPT} \geq k$.

This is a **2-approximation** algorithm.

Approximation Algorithms

Definition (min version): Let A be a minimization problem.
An α-approximation algorithm for A is an algorithm that for every input produces a solution which is **at most** α times the optimum.

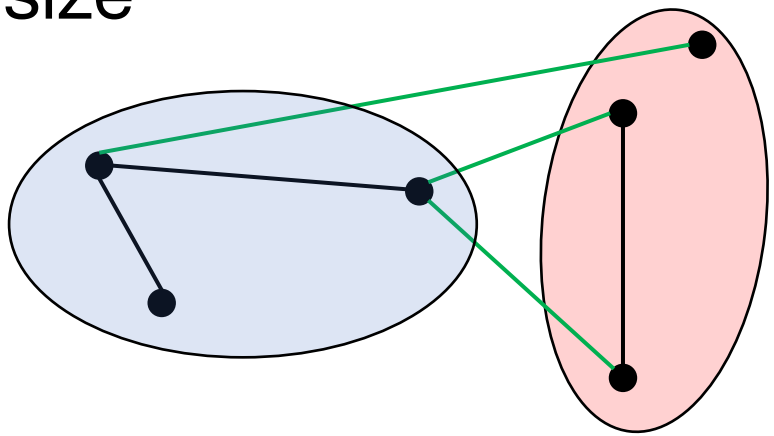
$$\mathrm{ALG} \leq \alpha \cdot \mathrm{OPT} \quad (\alpha \geq 1)$$

Definition (max version): Let A be a maximization problem.
An α-approximation algorithm for A is an algorithm that for every input produces a solution which is **at least** α times the optimum.

$$\alpha \cdot \mathrm{OPT} \leq \mathrm{ALG} \quad (\alpha \leq 1)$$

Max Cut

- Recall: A **cut** in G is a partition of the vertices into two disjoint non-empty sets $V = A \cup B$
- The **size** of the cut is the number of edges crossing the cut: i.e., edges (u, v) such that $u \in A$ and $v \in B$
- **Max Cut**: find a cut with maximal size
- NP hard problem



Approximating Max Cut

We start by showing a lower bound on the optimum.

Claim: in every graph G with m edges, there's a cut whose size is at least $\frac{m}{2}$.

Proof: produce a **random** cut (A, B): put every vertex v in A with probability $\frac{1}{2}$ and in B with probability $\frac{1}{2}$.

For every edge (u, v), let $X_{uv} = \begin{cases} 1 & \text{if } (u,v) \text{ crosses the cut} \\ 0 & \text{otherwise} \end{cases}$

Approximating Max Cut

$$X_{uv} = \begin{cases} 1 & \text{if } (u,v) \text{ crosses the cut} \\ 0 & \text{otherwise} \end{cases}$$

Then $\mathbb{E}[X_{uv}] = \frac{1}{2}$. Therefore, $\mathbb{E}_{A,B}\left[\sum_{(u,v)\in E} X_{u,v}\right] = \frac{m}{2}$.

$\mathbb{E}_{A,B}\left[\sum_{(u,v)\in E} X_{u,v}\right]$ is the expected value of size of a **random cut**.

Therefore, there must be a cut whose size is at least $\frac{m}{2}$.

This powerful idea is known as **the probabilistic method**.

Approximating Max Cut

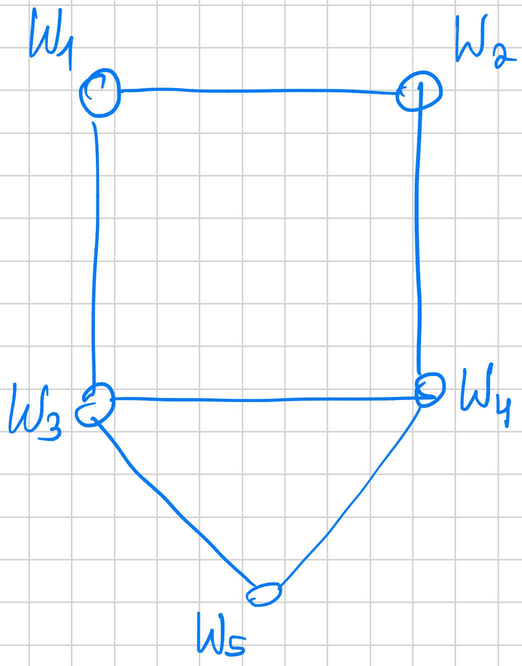
- Q: How to find a cut whose value is $\geq \frac{1}{2}$ the optimum?
- A: Pick one at random!
  - It works, in expectation: $\mathrm{OPT} \leq m$ and $\mathrm{ALG} \geq \frac{m}{2}$ so $\frac{1}{2} \cdot \mathrm{OPT} \leq \mathrm{ALG}$
- This is a randomized approximation algorithm which achieves approximation factor $\frac{1}{2}$ in expectation.
- It's possible to tweak it into a **deterministic** algorithm with the same approximation ratio.

Method of Conditional Expectations

- Let $w_1, \ldots, w_n$ be the vertices of G.
- Suppose we have already decided on the assignments of $w_1, \ldots, w_i$ to either A to B
- Can we compute $\mathbb{E}\left[\sum X_{u,v}\right]$ conditioned on this choice?
- Yes. For every (u, v):
  - If both already assigned, $\mathbb{E}\left[X_{u,v}\right]$ is known and either 0 (if both in the same set) or 1 (if in different set)
  - If at least one not assigned, $\mathbb{E}\left[X_{u,v}\right] = \frac{1}{2}$ as before.

Method of Conditional Expectations

- In the beginning, $\mathbb{E}\left[\sum X_{u,v}\right] = \frac{m}{2}$
- Therefore either $\mathbb{E}\left[\sum X_{u,v} \mid w_1 \in A\right] \geq \frac{m}{2}$ or $\mathbb{E}\left[\sum X_{u,v} \mid w_1 \in B\right] \geq \frac{m}{2}$
- We just argued we can compute these conditional expectations
- Pick w_1 to maintain the conditional expectation above $\frac{m}{2}$
- Invariant: $\mathbb{E}\left[\sum X_{u,v} \mid w_1, \ldots, w_i \text{ decided}\right] \geq \frac{m}{2}$
- Pick w_{i+1} to maintain the conditional expectation above $\frac{m}{2}$
- At the end of the process, we get a cut of size $\geq \frac{m}{2}$



$$\mathbb{E}\left[\sum_{(u,v)\in E} X_{uv}\right] = \frac{7}{2}$$

$$X_{uv} = \begin{cases} 1 & \text{חוצה את החתך } (u,v) \\ 0 & \text{אחרת} \end{cases}$$

שלב ראשון

$$\mathbb{E}\left[\sum_{(u,v)\in E} X_{uv} \,\middle|\, w_1 \in A\right] = \frac{7}{2}$$

$$\mathbb{E}\left[\sum_{(u,v)\in E} X_{uv} \,\middle|\, w_1 \in B\right] = 2$$

נבחר שרירותי ש-w_1 יהיה ב-A ⟸

שלב שני

$$\mathbb{E}\left[\sum_{(u,v)\in E} X_{uv} \,\middle|\, \begin{matrix} w_1 \in A \\ w_2 \in A \end{matrix}\right] = \frac{6}{2} + 0$$

$$\mathbb{E}\left[\sum_{(u,v)\in E} X_{uv} \,\middle|\, \begin{matrix} w_1 \in B \\ w_2 \in B \end{matrix}\right] = \frac{6}{2} + 1 = 4$$

נבחר ש-w_2 יהיה ב-B ⟸

Max 3SAT

Given a 3-CNF formula on n variables and m clauses:

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_4 \vee \overline{x_5}) \wedge \cdots \wedge (\overline{x_{13}} \vee \overline{x_{10}} \vee x_5)$$

Find an assignment to $x_1, \ldots, x_n$ which maximizes the number of satisfied clauses.

Clearly, an NP hard problem.

How many clauses does a random assignment satisfy?

Max 3SAT

- For each clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$:
  - 7 satisfying assignments
  - 1 unsatisfying assignment
- $\mathbb{E}[C \text{ satisfied}] = \frac{7}{8}$
- A random assignment satisfies $\frac{7}{8}$ fraction of the clauses
- To find it deterministically: can use conditional expectations
- A $\frac{7}{8}$ approximation algorithm.
- Can we do better?

Max 3SAT

Theorem: For every $\varepsilon > 0$, if there's a polynomial time $\left(\frac{7}{8} + \varepsilon\right)$ approximation algorithm for Max-3SAT, then P=NP.

Proof: not in this class.

There are many similar **hardness of approximation** results.

Can we do better than $\frac{1}{2}$ for Max-Cut?

We'll soon get back to this question.

LP Relaxations

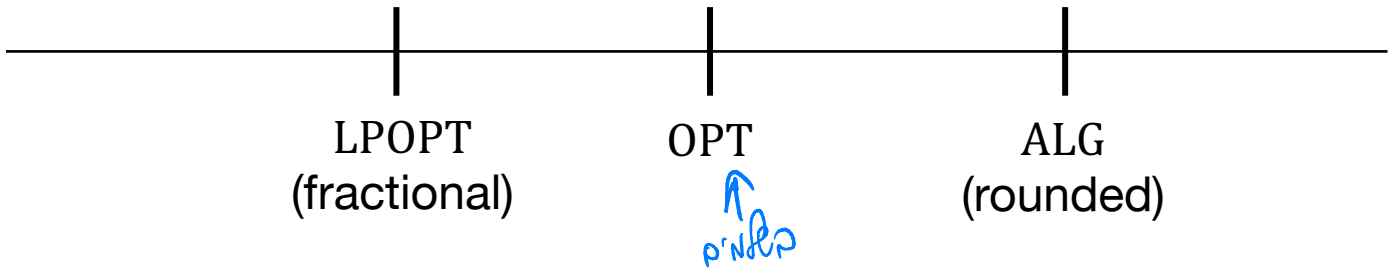
When we talked about Integer Programming, we mentioned the following idea:

1. Express optimization problem as an Integer Program
2. Relax the IP to get an LP
3. Solve the LP and get a fractional solution
4. Round the fractional solution

We want approximation guarantees on the rounded solution.

LP Relaxations

For a minimization problem:



- We want to prove $\frac{\text{ALG}}{\text{OPT}} \leq \alpha$
- It's often easier to prove $\frac{\text{ALG}}{\text{LPOPT}} \leq \alpha$
  (since $\frac{\text{ALG}}{\text{OPT}}$ can only be smaller)

Example: Vertex Cover

1. Express as an integer program:

   Minimize $\sum_{v \in V} x_v$

   Subject to $x_v \in \{0,1\}$

   $x_u + x_v \geq 1$ for every edge (u, v)

2. Relax to LP by requiring $0 \leq x_v \leq 1$
3. Solve LP and get a fractional solution x^*
4. Round. How?

Rounding Scheme for VC

$x_1^* = 0.2 \Rightarrow 0$

$x_2^* = 0.7 \Rightarrow 1$

$x_3^* = 0.3 \Rightarrow 0$

Simple rule:

"If $x_v^* \geq \frac{1}{2}$ set $x_v = 1$, otherwise $x_v = 0$"

(The vertex cover is the set of vertices such that $x_v = 1$)

Claim 1: x corresponds a vertex cover.

Proof: for every edge, $x_v^* + x_u^* \geq 1$, hence either x_v^* or $x_u^* \geq \frac{1}{2}$. $\Rightarrow$ לפחות 1 מהם
1 - לפחות
כיסוי שלם

This means that the edge (u, v) is covered.

Claim 2: $\mathrm{ALG} \leq 2\mathrm{OPT}$. 2-approximation

Bounding the Approximation Ratio

Claim 2: $\mathrm{ALG} \leq 2\mathrm{OPT}$.

Proof: Recall: it's easier to show $\mathrm{ALG} \leq 2\mathrm{LPOPT}$.

For every v, $x_v \leq 2x_v^*$.

$$\mathrm{ALG} = \sum_{v \in V} x_v \leq \sum_{v \in V} 2x_v^* = 2\mathrm{LPOPT}$$

Proof completed.

Rounding schemes

1. Round to nearest integer
   - Like we just saw
2. Randomized rounding:
   - If $0 \leq x_i \leq 1$ we can think of them as probabilities
   - And set $x_i = 1$ with probability x_i
   - Will see an example in HW
3. More complicated rules
   - Will see an example soon

Max Cut

We'd like to do apply the same framework for Max Cut.

We first try to express Max Cut as an IP.

Variables: x_{uv} for every edge, y_v for every vertex.

Maximize $\sum_{(u,v)\in E} x_{u,v}$

Subject to $x_{u,v} \in \{0,1\}, y_v \in \{0,1\}$

$$x_{u,v} = y_v \overset{\text{xor}}{\oplus} y_u \qquad x_{u,v} = 1 \text{ iff } y_v \neq y_u$$

Not a linear constraint!

Recall that for min-cut this was possible (dual of max flow).
Weird.

Max Cut Integer Program

First idea: switch to $y_v \in \{-1,1\}$ and require $x_{u,v} = y_v \cdot y_u$

Now $x_{u,v} = -1$ if $y_v \neq y_v$ and 1 otherwise

Maximize $\sum_{(u,v)\in E} \frac{(1-x_{uv})}{2}$ ← linear constraint

Subject to $y_v \in \{-1,1\}$ ← integer constraint

$x_{u,v} = y_u \cdot y_v$ ← non linear constraint

Still not a linear constraint!

Semidefinite Programming

We will show we can relax the Max-Cut IP into an optimization problem from a more general class than linear programs:
Semidefinite programs

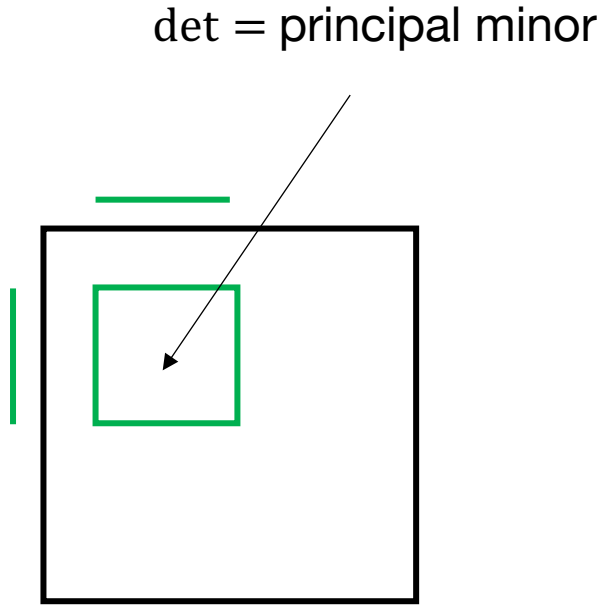
Definition: A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is **positive semidefinite** (PSD) if for every $v \in \mathbb{R}^n$, $v^T A v \geq 0$.

Notation: $A \succcurlyeq 0$.

PSD Matrices

The following are equivalent definitions for PSD matrices:

1. For every $v \in \mathbb{R}^n$, $v^T A v \geq 0$ (previous definition)
2. All eigenvalues of A are non-negative
3. There exists $B \in \mathbb{R}^{m \times n}$ such that $A = B^T B$
4. All principal minors of A are non-negative



Most of these proofs are simple.

Semidefinite Programming

Semidefinite Programming (SDP): Optimization of a linear function (in the entries of a matrix X) subject to linear constraints (on the entries of X) **and a constraint that X is PSD**.

Example: $X = \begin{pmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{pmatrix}$

Maximize $x_{11} - x_{12} + 2x_{22}$

Subject to $3x_{11} + x_{12} = 4$

$X \succcurlyeq 0$

Equivalent to:

$x_{11} \geq 0, x_{22} \geq 0,$

$x_{11}x_{22} - x_{12}^2 \geq 0$

So in effect we added non-linear constraints!

Semidefinite Programming

In general form: convenient (for notational purposes) to think of X as a vector $x \in \mathbb{R}^{\frac{n(n+1)}{2}}$ (upper triangular part).

Minimize $c^T x$

Subject to $Ax = b$,

$\text{“}x \succcurlyeq 0\text{”}$

Here x is thought of as a vector

Here x is thought of as a matrix

We’ll see other formulations which make is less confusing.

Semidefinite Constraint

PSD:
1. $v^T X v \geq 0$
2. All eigenvalues are positive
3. $X = B^T B$
4. Principal minors non-negative

- Using def 4, requiring $X \succcurlyeq 0$ is like adding exponentially many constraints of degree $\leq n$
- Using def 1, requiring $X \succcurlyeq 0$ is like adding infinitely many linear constraints: $\sum_{i,j} v_i v_j X_{i,j} \geq 0$
- Nevertheless: recall that when we analyzed the ellipsoid algorithm, we said that it works as long as we can efficiently find a **separating hyperplane**
- (there's also the issue of the volume upper and lower bounds – let's assume this is not an issue)

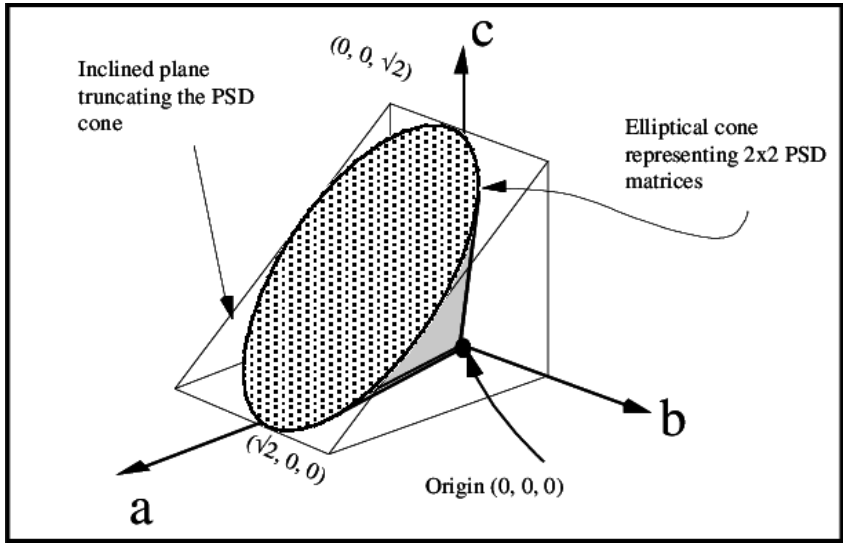
Separating Hyperplanes for PSD

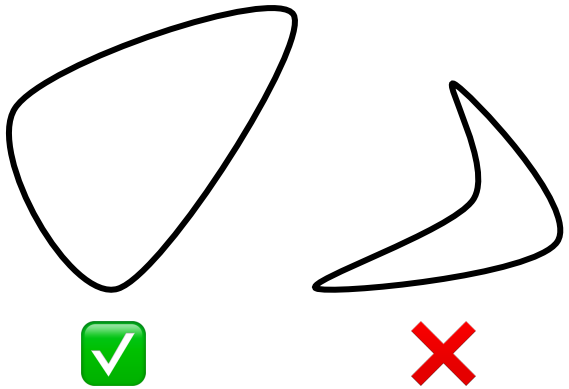
- Let P be the set "$Ax = b, x \geqslant 0$"
- We're given a point $y \notin P$
- If $Ay \neq b$, we know what to do (proceed as in the LP case)
- If y isn't PSD:
  - Let Y be the symmetric matrix corresponding to y
  - Y isn't PSD so it has a negative eigenvalue λ with eigenvector v.
  - $v^T Y v = v^T \lambda v = \lambda \|v\|^2 < 0$
  - Whereas $v^T X v \geq 0$ for all PSD X
  - v is the separating hyperplane!
  - Can be found efficiently

PSD:
1. $v^T X v \geq 0$
2. All eigenvalues are positive
3. $X = B^T B$
4. Principal minors non-negative

Optimization over PSD Matrices

- **Conclusion**: we can efficiently optimize over PSD matrices using the ellipsoid algorithm
  - Assuming reasonable upper and lower bounds on the volume of the set
- This set is non-linear but is it **convex**
- There's a rich area of **convex optimization**





SDP: Vector formulation

PSD:
1. $v^T X v \geq 0$
2. All eigenvalues are positive
3. $X = B^T B$
4. Principal minors non-negative

Equivalent formulation for SDP (using Def 3):

Minimize $c^T x$

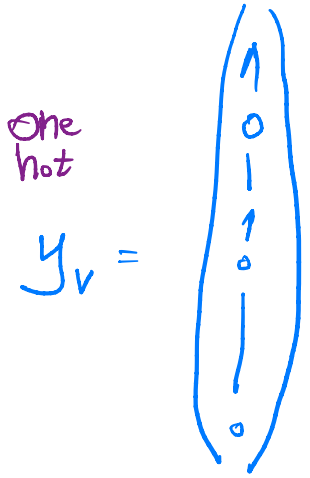
Subject to $Ax = b$,

$$\exists u_1, \ldots, u_n \in \mathbb{R}^n,\ x_{i,j} = u_i^T u_j$$

The u_i's are the columns of B

They're not given as part of the input (only c, A, b)

Max Cut IP



Recall the Max Cut Integer Program:

Maximize $\sum_{(u,v)\in E} \frac{(1-x_{uv})}{2}$

Subject to $y_v \in \{-1,1\}$

$$x_{u,v} = y_u \cdot y_v$$

We'll relax it to an **SDP** by allowing y_v's to range over unit vectors

Max Cut SDP

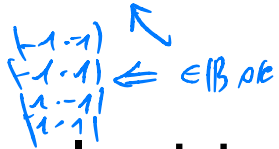
SDP Relaxation:

Maximize $\sum_{(u,v)\in E} \frac{(1-x_{uv})}{2}$

Subject to $\exists y_1, \ldots, y_n \in \mathbb{R}^n$ such that

$$x_{u,v} = y_u^T \cdot y_v$$

$$x_{v,v} = \|y_v\|^2 = 1$$



The conditions are equivalent to: "x is PSD with 1 on diagonal"

Max Cut SDP

Maximize $\sum_{(u,v)\in E} \frac{(1-x_{uv})}{2}$

Subject to $\exists y_1, \ldots, y_n \in \mathbb{R}^n$ s.t.

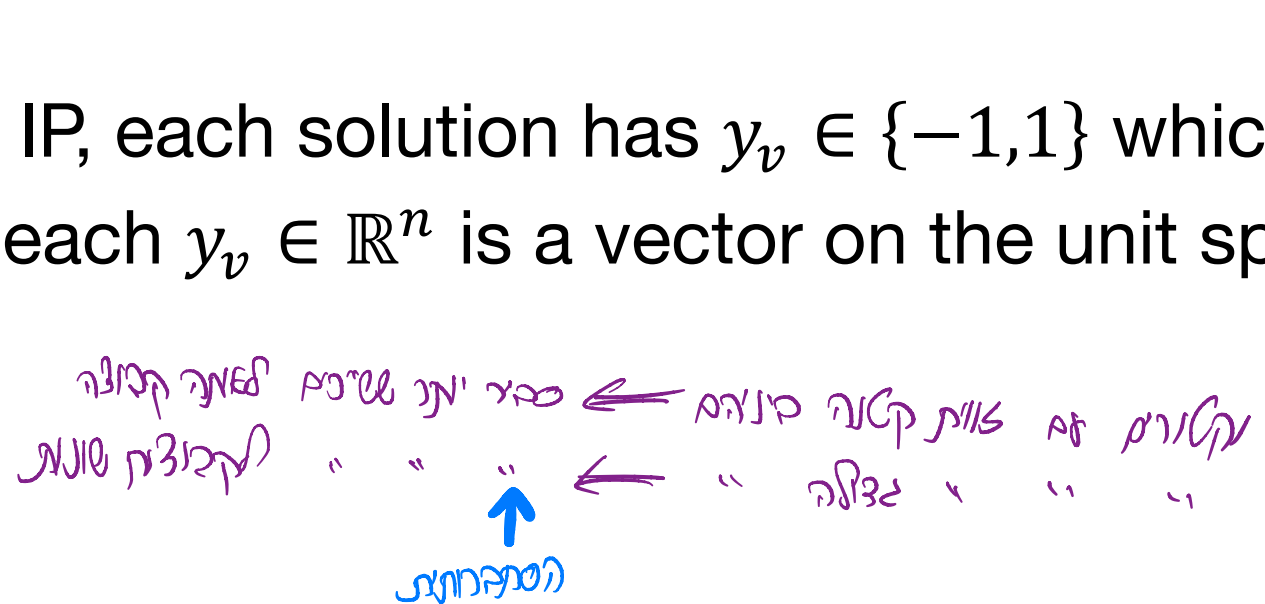
$$x_{u,v} = y_u^T \cdot y_v$$

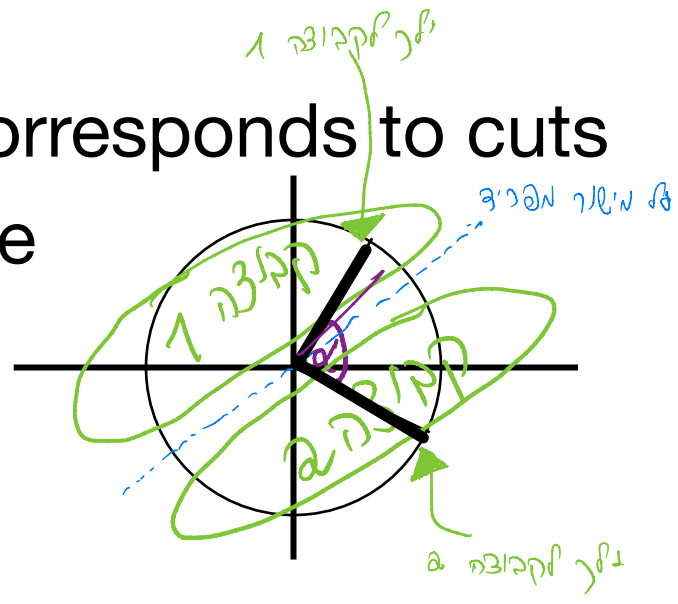
$$x_{v,v} = \|y_v\|^2 = 1$$

1. Write IP
2. Relax to SDP
3. Solve SDP with Ellipsoid (← We are here)
4. Round and argue about approximation ratio

In the IP, each solution has $y_v \in \{-1,1\}$ which corresponds to cuts

Now, each $y_v \in \mathbb{R}^n$ is a vector on the unit sphere





Max Cut Rounding

Maximize $\sum_{u,v} \frac{(1-x_{uv})}{2}$

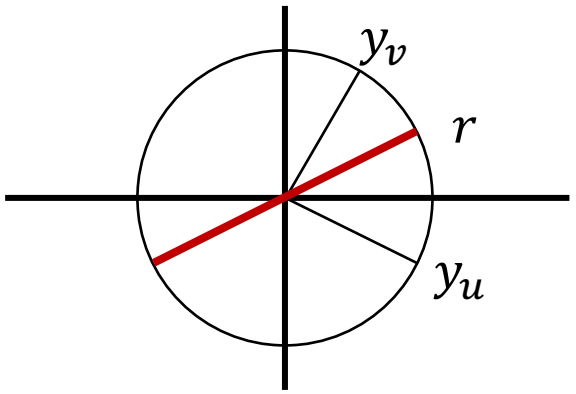
Subject to $\exists y_1, \ldots, y_n \in \mathbb{R}^n$ s.t.

$$x_{u,v} = y_u^T \cdot y_v$$

$$x_{v,v} = \|y_v\|^2 = 1$$

Goemans-Williamson Algorithm:

1. Pick a **random** hyperplane through the origin
   - Represented by a vector $r \in \mathbb{R}^n$
2. "Round" according to **which side** it falls
   - $A = \{v : \langle r, y_v \rangle \geq 0\}$
   - $B = \{v : \langle r, y_v \rangle < 0\}$
3. Return the cut (A, B)



Max Cut Rounding

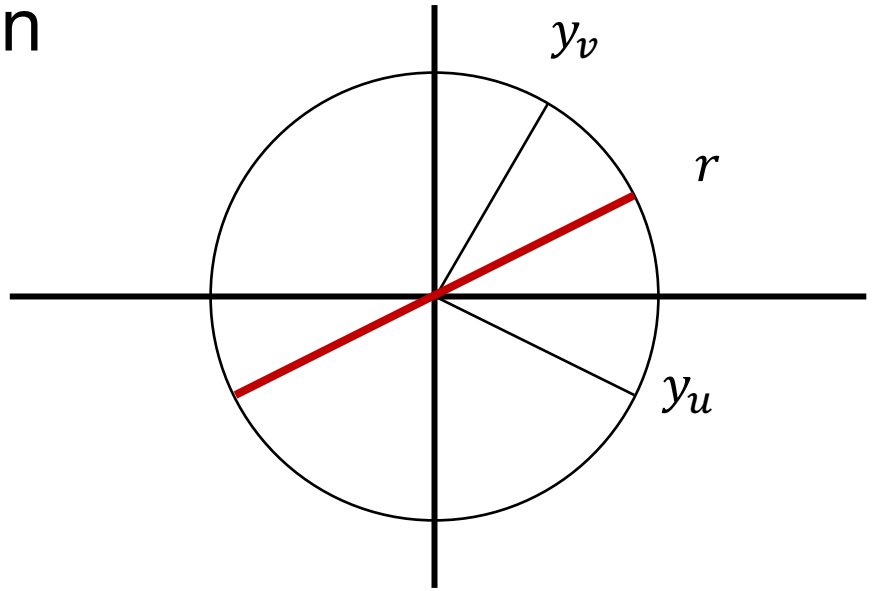
Let (u, v) be an edge in G

Claim: the probability that (u, v) in the cut is $\frac{1}{\pi}\angle(y_u, y_v)$

$\angle(y_u, y_v) =$ the angle between y_u and y_v

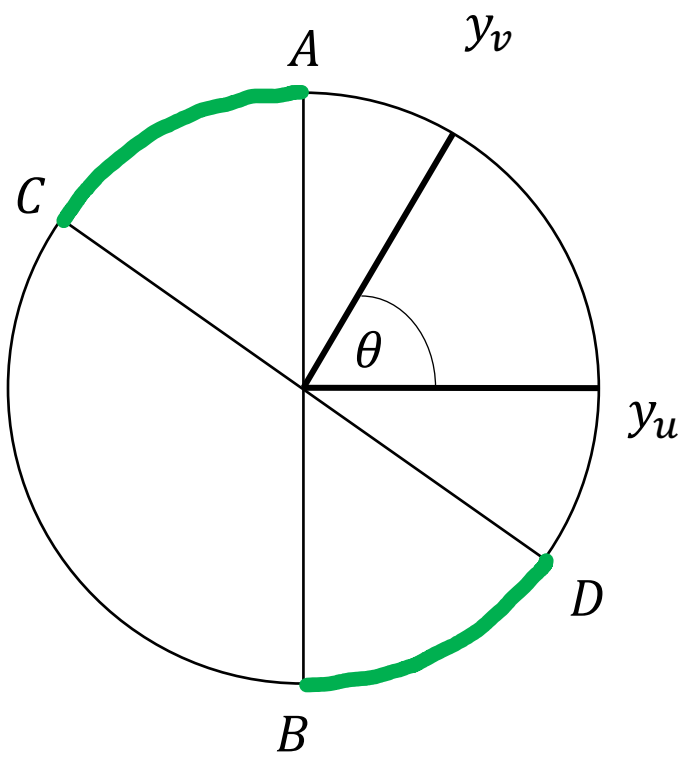
(u, v) crosses the cut if y_u and y_v are in opposite sides of the hyperplane

By looking at $\text{span}\{y_u, y_v\}$,
enough to argue about 2-dim case.



Proof by Picture

Claim: the probability that (u, v) in the cut is $\frac{1}{\pi}\angle(y_u, y_v)$



- $\langle r, y_u \rangle < 0$ iff r to the left of AB
- $\langle r, y_v \rangle < 0$ iff r southwest of CD

(u, v) in the cut iff r in **green zone**

This happens with probability $\frac{2\theta}{2\pi} = \frac{\theta}{\pi}$

Bounding the ratio

Maximize $\sum_{(u,v)\in E} \frac{(1-x_{uv})}{2}$

Subject to $\exists y_1, \ldots, y_n \in \mathbb{R}^n$ s.t.

$$x_{u,v} = y_u^T \cdot y_v$$

$$x_{v,v} = \|y_v\|^2 = 1$$

- Let SDPOPT be the optimal solution of the SDP
- And ALG the expected value of the rounded solution
- We want to show: $\mathrm{ALG} \geq \alpha \cdot \mathrm{OPT}$ $(\alpha \leq 1)$
- We'll show: $\mathrm{ALG} \geq \alpha \cdot \mathrm{SDPOPT}$
- $\mathrm{SDPOPT} = \sum_{(u,v)\in E} \frac{1-x_{uv}}{2} = \sum_{(u,v)\in E} \frac{1-\langle y_v, y_u \rangle}{2} = \sum_{(u,v)\in E} \frac{1-\cos\theta_{u,v}}{2}$
- (u, v) is in the cut with probability $\frac{\theta_{u,v}}{\pi}$
- $\mathbb{E}[\mathrm{ALG}] = \sum_{(u,v)\in E} \Pr[(u,v) \text{ in cut}] = \sum_{(u,v)\in E} \frac{\theta_{u,v}}{\pi}$

$$\theta_{u,v} = \angle(y_u, y_v)$$

Bounding the ratio

Maximize $\sum_{(u,v)\in E} \frac{(1-x_{uv})}{2}$

Subject to $\exists y_1, \ldots, y_n \in \mathbb{R}^n$ s.t.

$$x_{u,v} = y_u^T \cdot y_v$$

$$x_{v,v} = \|y_v\|^2 = 1$$

- We'll show: $\mathrm{ALG} \geq \alpha \cdot \mathrm{SDPOPT}$
- $\mathrm{SDPOPT} = \sum_{(u,v)\in E} \frac{1-(x^*)_{uv}}{2} = \sum_{(u,v)\in E} \frac{1-\langle y_v, y_u\rangle}{2} = \sum_{(u,v)\in E} \frac{1-\cos\theta_{u,v}}{2}$
- $\mathbb{E}[\mathrm{ALG}] = \sum_{(u,v)\in E} \Pr[(u,v) \text{ in cut}] = \sum_{(u,v)\in E} \frac{\theta_{u,v}}{\pi}$ $\qquad \theta_{u,v} = \angle(y_u, y_v)$
- We want pick α such that we always have:

$$\frac{\theta}{\pi} \geq \alpha \cdot \frac{1-\cos\theta}{2} \quad \Leftarrow \quad \frac{2\theta}{\pi} \cdot \frac{1}{1-\cos\theta} \geq \alpha$$

- In other words, we'll pick $\alpha = \frac{2}{\pi} \min_{\theta} \frac{\theta}{1-\cos\theta}$

שתק"ס: α כמה גדול

$\forall \theta \in [0,\pi]$: $\frac{\theta}{\pi} \geq \alpha \cdot \frac{1-\cos\theta}{2}$

An α approximation

minimize 2/pi * (x/(1-cos x)) between 0 and pi

NATURAL LANGUAGE | MATH INPUT | EXTENDED KEYBOAR

Input interpretation

minimize	function	$\frac{2}{\pi} \times \frac{x}{1-\cos(x)}$
	domain	$0 \le x \le \pi$

Global minimum | More digits

$$\min\left\{\frac{2x}{\pi(1-\cos(x))} \,\middle|\, 0 \le x \le \pi\right\} \approx 0.878567 \text{ at } x \approx 2.33112$$

$$\alpha = 0.8785 \ldots$$

$$\text{ALG} \ge \alpha \cdot \text{OPT}$$

Much better than $\frac{1}{2}$

לפחות 87.8% מגודל החתך המקסימלי
בתוחלת !

Max Cut

- Can we do better?
- **Theorem**: Assuming “unique games conjecture” (something slightly stronger than P≠NP) can’t improve beyond $\alpha = 0.878$ ...!
- The Goemans Williamson Algorithm with its weird approximation parameter is optimal unless some other problem is unexpectedly easy

Shortest Vector in a Lattice

Definition: a lattice in $\mathbb{R}^n$ a set of integral linear combinations of linear independent vectors $b_1, \ldots, b_n \in \mathbb{R}^n$

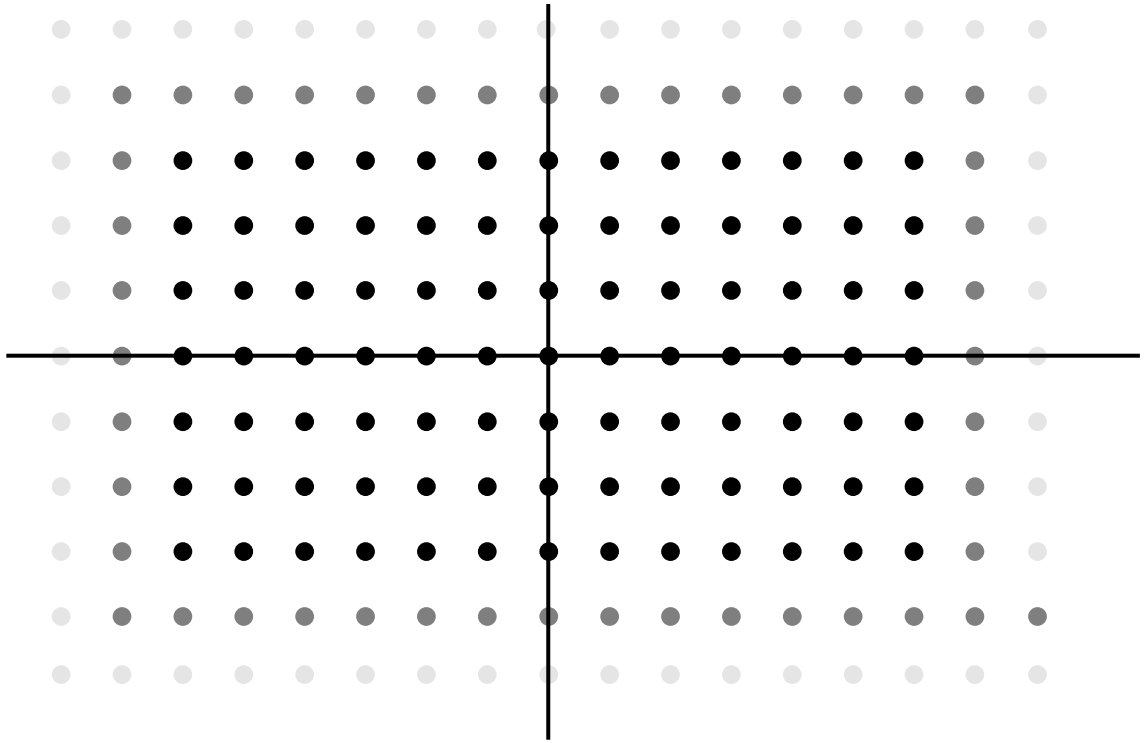
$$L = \left\{ \sum_{i=1}^{n} \lambda_i b_i : \lambda_i \in \mathbb{Z}, 1 \leq i \leq n \right\}$$

$b_1, \ldots, b_n$ is called a **basis** for L

Notation: $L = L(b_1, \ldots, b_n)$

Example

Two dimensional lattice with $b_1 = (1,0), b_2 = (0,1)$:



Lattices are very useful!

- Integer programming and optimization
- Number theory
- Cryptography
- Complexity

Shortest Vector Problem

- Given a basis for a lattice $(b_1, \ldots, b_n)$, find the **shortest** non-zero vector $v \in L(b_1, \ldots, b_n)$
  - **Shortest** = smallest norm $(\neq \vec{0})$
- Very important problem in cryptography
- NP hard! Even to approximate well…
  - That's why it seems useful for crypto
- We'll see a polynomial time approximation algorithm for some approximation ratio

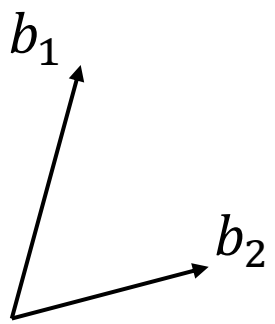
"אפילו עם מחשבים קוונטים לא ניתן לפתור ביעילות"

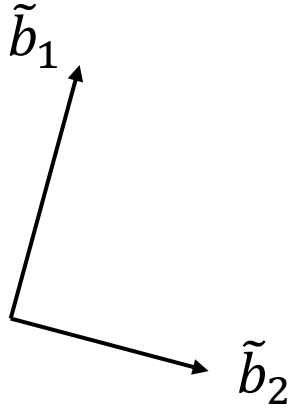
Lenstra-Lenstra-Lovász (LLL) Algorithm

- The LLL algorithm approximates the shortest vector in a lattice
- It has many uses:
  - Factoring polynomials over rational numbers
  - Solving IP in a fixed dimension
  - Breaking poorly designed cryptographic systems
  - Finding integer relations between real numbers

Gram-Schmidt Orthogonalization

Given a basis $b = (b_1, \ldots, b_n)$, the **Gram-Schmidt Orthogonalization** process finds an **orthogonal** basis for the same subspace:





Gram-Schmidt Orthogonalization

- Set $\tilde{b}_1 = b_1$
- We subtract a multiple of $\tilde{b}_1$ from b_2: $\tilde{b}_2 = b_2 - \mu_{2,1}\tilde{b}_1$
- We require that $\langle \tilde{b}_1, \tilde{b}_2 \rangle = 0$:

$$\langle \tilde{b}_1, \tilde{b}_2 \rangle = \langle \tilde{b}_1, b_2 - \mu_{2,1}\tilde{b}_1 \rangle = \langle \tilde{b}_1, b_2 \rangle - \mu_{2,1}\langle \tilde{b}_1, \tilde{b}_1 \rangle$$

- Corollary: we should set $\mu_{2,1} = \frac{\langle \tilde{b}_1, b_2 \rangle}{\langle \tilde{b}_1, \tilde{b}_1 \rangle}$
- Now $\tilde{b}_3 = b_3 - \mu_{3,1}\tilde{b}_1 - \mu_{3,2}\tilde{b}_2$ should be orthogonal to $\tilde{b}_1, \tilde{b}_2$ which gives the values of $\mu_{3,1}, \mu_{3,2}$

Gram-Schmidt Orthogonalization

- Set $\tilde{b}_1 = b_1$
- $\tilde{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j}\, \tilde{b}_j$, where $\mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$
- Since we add multiples of vector bases to one another, the subspace doesn't change
- By induction $\langle \tilde{b}_i, \tilde{b}_j \rangle = 0$ at the end of the process

Note that this operation doesn't preserve the **lattice** spanned by $b_1, \ldots, b_n$!

Lower bound on shortest vector

Claim: Let $b_1, \ldots, b_n$ and let v be ~~the~~ shortest non-zero vector in $L(b_1, \ldots, b_n)$. Then $\|v\| \geq \min_{1 \leq i \leq n} \|\tilde{b}_i\|$.

Proof: write $v = \sum_{i=1}^{n} \lambda_i b_i$

and suppose k is largest s.t. $\lambda_k \neq 0$.

Substitute $b_i = \tilde{b}_i + \sum_{j=1}^{i-1} \mu_{i,j} \tilde{b}_j$, we get

$$v = \sum_{i=1}^{k} \tilde{\lambda}_i \tilde{b}_i$$

for some $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_k$, where $\tilde{\lambda}_k = \lambda_k$

Lower bound on shortest vector

$$v = \sum_{i=1}^{k} \tilde{\lambda}_i \tilde{b}_i$$

Now compute

$$\|v\|^2 = \langle v, v \rangle = \left\langle \sum_{i=1}^{k} \tilde{\lambda}_i \tilde{b}_i \, , \sum_{i=1}^{k} \tilde{\lambda}_i \tilde{b}_i \right\rangle$$

$$= \sum_{i=1}^{k} \tilde{\lambda}_i^2 \langle \tilde{b}_i, \tilde{b}_i \rangle \geq \tilde{\lambda}_k^2 \left\| \tilde{b}_k \right\|^2 = \lambda_k^2 \left\| \tilde{b}_k \right\|^2 \geq \left\| \tilde{b}_k \right\|^2$$

(since $\lambda_k \in \mathbb{Z}$ is nonzero)

Reduced Basis

Definition: a basis $(b_1, \ldots, b_n)$ is called **reduced** if:

1. $|\mu_{i,j}| \leq \frac{1}{2}$ for every $i > j$, $1 \leq i, j \leq n$

2. $\left\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\right\|^2 \geq \frac{3}{4}\left\|\tilde{b}_i\right\|^2$ for every $1 \leq i \leq n$

Recall: $\mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$

- We'll explain the definition
- Then we'll show that if we can find a reduced basis, we can approximate the shortest vector
- Then we'll show how to find a reduced basis (this is the main part)

"אַבְרָהָם סוֹכֵן"

Reduced Basis

$(b_1, \ldots, b_n)$ is **reduced** if:

1. $|\mu_{i,j}| \leq \frac{1}{2}$
2. $\left\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\right\|^2 \geq \frac{3}{4}\left\|\tilde{b}_i\right\|^2$

$$\mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$$

First property means that the basis is "almost orthogonal" (if it were orthogonal we'd have $\mu_{i,j} = 0$)

Second property: note that

$$\left\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\right\|^2 = \left\|\tilde{b}_{i+1}\right\|^2 + \mu_{i+1,i}^2\left\|\tilde{b}_i\right\|^2$$

Since $\mu_{i+1,i}^2 \leq \frac{1}{4}$, we get $\left\|\tilde{b}_{i+1}\right\|^2 \geq \frac{1}{2}\left\|\tilde{b}_i\right\|^2$

So $\tilde{b}_{i+1}$ isn't much shorter than $\tilde{b}_i$.

Short Vector in Reduced Basis

$(b_1, \dots, b_n)$ is **reduced** if:
1. $|\mu_{i,j}| \leq \frac{1}{2}$
2. $\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\|^2 \geq \frac{3}{4}\|\tilde{b}_i\|^2$

is a reduced basis

Lemma: suppose $(b_1, \dots, b_n)$ and let v be a shortest non-zero vector in $L(b_1, \dots, b_n)$. Then $\|b_1\| \leq 2^{\frac{n-1}{2}}\|v\|$.

Proof: recall that $\|v\| \geq \min_i \|\tilde{b}_i\|$. By property 2:

$$\|\tilde{b}_n\|^2 \geq \frac{1}{2}\|\tilde{b}_{n-1}\|^2 \geq \frac{1}{4}\|\tilde{b}_{n-2}\|^2 \cdots \geq \frac{1}{2^{n-1}}\|\tilde{b}_1\|^2 = \frac{1}{2^{n-1}}\|b_1\|^2$$

~ b_1 ~ $\tilde{b}_1$

It follows that $\|b_1\|^2 \leq 2^{i-1}\|\tilde{b}_i\|^2$ and

$$\|b_1\|^2 \leq 2^{n-1}\min_i \|\tilde{b}_i\|^2 \leq 2^{n-1}\|v\|^2$$

Finding Reduced Basis: LLL Algorithm

Input: lattice basis $b_1, \ldots, b_n \in \mathbb{Z}^n$

Output: reduced basis for $L(b_1, \ldots, b_n)$

1. Start: compute $\tilde{b}_1, \ldots, \tilde{b}_n$
2. Reduction step:
   For $i = 2$ to n
      For $j = i - 1$ down to 1:
         $b_i \leftarrow b_i - c_{i,j} b_j$, where $c_{i,j}$=closest integer to $\mu_{i,j}$
3. Swap step:
   If $\exists i$ such that $\left\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\right\|^2 < \frac{3}{4}\left\|\tilde{b}_i\right\|^2$:
      Swap b_i and b_{i+1} and go to start.
4. Output $b_1, \ldots b_n$

$(b_1, \ldots, b_n)$ is **reduced** if:

1. $\left|\mu_{i,j}\right| \leq \frac{1}{2}$
2. $\left\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\right\|^2 \geq \frac{3}{4}\left\|\tilde{b}_i\right\|^2$

$\mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$

Finding Reduced Basis: LLL Algorithm

$(b_1, \ldots, b_n)$ is **reduced** if:

1. $|\mu_{i,j}| \leq \frac{1}{2}$
2. $\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\|^2 \geq \frac{3}{4}\|\tilde{b}_i\|^2$

$\mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$

First observations:

1. The algorithm outputs a basis for the same lattice as the input (since we only swap basis elements, or add integral multiples of one basis element to another)
2. When (if) the algorithm terminates, the swap step takes care of property (2) in the definition of reduced basis
3. We need to show that the reduction step takes care of property (1), and that the total number of steps is polynomial.

Reduction step

$(b_1, \ldots, b_n)$ is **reduced** if:

1. $|\mu_{i,j}| \leq \frac{1}{2}$
2. $\left\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\right\|^2 \geq \frac{3}{4}\left\|\tilde{b}_i\right\|^2$

$\mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$

For $i = 2$ to n

For $j = i - 1$ down to 1:

$b_i \leftarrow b_i - c_{i,j} b_j$, where $c_{i,j}$=closest integer to $\mu_{i,j}$

The reduction step doesn't change $\tilde{b}_1, \ldots, \tilde{b}_n$:

The i-th step of the process projects b_i onto $\tilde{b}_1, \ldots, \tilde{b}_{i-1}$ and subtracts it from b_i so adding multiplies of $\tilde{b}_1, \ldots, \tilde{b}_{i-1}$ doesn't make a difference.

Reduction step

Consider the Gram-Schmidt process with b_i vs. the same process with $b_i = b_i - \sum_{j=1}^{i-1} c_{i,j} b_j$:

- $\tilde{b}_1, \ldots, \tilde{b}_{i-1}$ computed as before, orthogonal and span the same subspace as $b_1, \ldots, b_{i-1}$
- Now $b_i = b_i - \sum_{j=1}^{i-1} \tilde{c}_{i,j} \tilde{b}_j$, so $\langle b_i, \tilde{b}_j \rangle = \langle b_i, \tilde{b}_j \rangle - \sum_{j=1}^{i-1} \tilde{c}_{i,j} \langle \tilde{b}_j, \tilde{b}_j \rangle$ and

$$\tilde{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \tilde{b}_j = b_i - \sum_{j=1}^{i-1} \tilde{c}_{i,j} \tilde{b}_j - \sum_{j=1}^{i-1} \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} \tilde{b}_j =$$

$$b_i - \sum_{j=1}^{i-1} \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} \tilde{b}_j = \tilde{b}_i$$

Reduction step

$(b_1, \dots, b_n)$ is **reduced** if:

1. $|\mu_{i,j}| \leq \frac{1}{2}$
2. $\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\|^2 \geq \frac{3}{4}\|\tilde{b}_i\|^2$

$\mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$

Replacing b_i with $b_i = b_i - c_{i,j}b_j$ where $c_{i,j}$=closest integer to $\mu_{i,j}$, we have

$$|\mu_{i,j}| = \left|\frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}\right| = \left|\frac{\langle b_i - c_{i,j}b_j, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}\right|$$

$$= \left|\frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} - c_{i,j}\frac{\langle b_j, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}\right| = |\mu_{i,j} - c_{i,j}| \leq \frac{1}{2}$$

since $\langle b_j, \tilde{b}_j \rangle = \langle \tilde{b}_j, \tilde{b}_j \rangle$ by orthogonality.

Corollary: at the end of the reduction process, property (1) in the definition of reduced basis holds

Swap step

$(b_1, \ldots, b_n)$ is **reduced** if:

1. $|\mu_{i,j}| \leq \frac{1}{2}$
2. $\|\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i\|^2 \geq \frac{3}{4}\|\tilde{b}_i\|^2$

- How does the swap step affects the Gram-Schmidt basis?
- Let $a_1, \ldots, a_n$ be the basis obtained by swapping b_i and b_{i+1}
- $\tilde{a}_j = \tilde{b}_j$ unless $j = i$ or $i + 1$
- $\tilde{a}_i = a_i - \sum_{j=1}^{i-1} \frac{\langle a_i, \tilde{a}_j \rangle}{\langle \tilde{a}_j, \tilde{a}_j \rangle} \tilde{a}_j = b_{i+1} - \sum_{j=1}^{i-1} \frac{\langle b_{i+1}, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} \tilde{b}_j = b_{i+1} - \sum_{j=1}^{i-1} \mu_{i+1,j}\, \tilde{b}_j$
- $\tilde{b}_{i+1} = b_{i+1} - \sum_{j=1}^{i} \mu_{i+1,j} \tilde{b}_j$
- Corollary: $\tilde{a}_i = \tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i$
- If we swapped, $\|\tilde{a}_i\|^2 < \frac{3}{4}\|\tilde{b}_i\|^2$
- $\tilde{a}_{i+1}$: we won't care about it as much.

Running time

Left to show: total number of steps in LLL is polynomial

Define $D = \|\tilde{b}_1\|^{2n} \|\tilde{b}_2\|^{2(n-1)} \cdot \|\tilde{b}_n\|^2 = \prod_{i=1}^{n} \left(\prod_{j=1}^{i} \|\tilde{b}_j\|^2 \right)$

We'll show:

1. D is an integer.
2. D isn't too large to begin with
3. D drops by a factor of $\frac{3}{4}$ after each swap.

D depends on the Gram-Schmidt basis, and we showed how each reduction/swap affects the basis.

Determinant of a Lattice

Definition: Let L be a lattice spanned by $b_1, \ldots, b_n \in \mathbb{Z}^n$. The determinant of L, $\det L$, is the absolute value of the determinant of the matrix B whose columns are $b_1, \ldots, b_n$.

Determinant doesn't depend on the bases (every other basis $b'_1, \ldots, b'_n$ is obtained by multiplying B by a matrix whose determinant is ± 1).

Claim: If $b_1, \ldots, b_n$ is a basis, $\det L = \|\tilde{b}_1\| \|\tilde{b}_2\| \cdots \|\tilde{b}_n\|$.

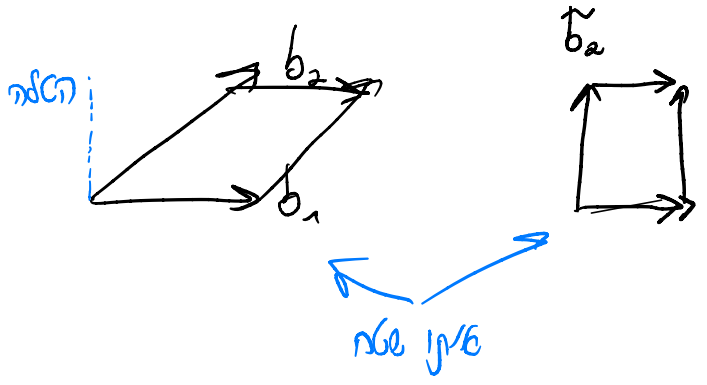
Determinant of a Lattice

Claim: If $b_1, \ldots, b_n$ is a basis, $\det L = \|\tilde{b}_1\|\|\tilde{b}_2\| \cdots \|\tilde{b}_n\|$.

Proof:

$$\begin{pmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \tilde{b}_1 & \tilde{b}_2 & \cdots & \tilde{b}_n \\ | & | & & | \end{pmatrix} \cdot \begin{pmatrix} 1 & * & * & * \\ 0 & 1 & * & * \\ \vdots & 0 & \ddots & * \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

and $\tilde{b}_1, \ldots, \tilde{b}_n$ orthogonal.



Running time

$$D = \|\tilde{b}_1\|^{2n} \|\tilde{b}_2\|^{2(n-1)} \cdot \|\tilde{b}_n\|^2$$

We showed how both steps affect the Gram-Schmidt basis:

- Reduction doesn't change it
- Swapping does: new $\tilde{a}_i$ is $\tilde{b}_{i+1} + \mu_{i+1,i}\tilde{b}_i$ and $\|\tilde{a}_i\|^2 < \frac{3}{4}\|\tilde{b}_i\|^2$
- Since determinant doesn't change,

$$\|\tilde{a}_1\|^2 \|\tilde{a}_2\|^2 \cdots \|\tilde{a}_n\|^2 = \|\tilde{b}_1\|^2 \|\tilde{b}_2\|^2 \cdots \|\tilde{b}_n\|^2 = (\det L)^2$$

- Therefore $\|\tilde{a}_i\|^2 \|\tilde{a}_{i+1}\|^2 = \|\tilde{b}_i\|^2 \|\tilde{b}_{i+1}\|^2$, and

$$\|\tilde{a}_i\|^{2(n-i+1)} \|\tilde{a}_{i+1}\|^{2(n-i)} \leq \frac{3}{4}\|\tilde{b}_i\|^{2(n-i+1)} \|\tilde{b}_{i+1}\|^{2(n-i)}$$

- D drops by a factor of $\frac{3}{4}$

Running time

$$D = \|\tilde{b}_1\|^{2n} \|\tilde{b}_2\|^{2(n-1)} \cdot \|\tilde{b}_n\|^2$$

As before, for every i

$$B_i := \begin{pmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_i \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \tilde{b}_1 & \tilde{b}_2 & \cdots & \tilde{b}_i \\ | & | & & | \end{pmatrix} \cdot \begin{pmatrix} 1 & * & * & * \\ 0 & 1 & * & * \\ \vdots & 0 & \ddots & * \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Therefore $\det\left(B_i^T B_i\right) = \prod_{j=1}^{i} \|\tilde{b}_i\|^2$

Note that $D = \prod_{i=1}^{n} \det\left(B_i^T B_i\right)$, so D is always an integer

In the beginning, on input $b_1, \ldots, b_n$, we have

$$D = \|\tilde{b}_1\|^{2n} \|\tilde{b}_2\|^{2(n-1)} \cdot \|\tilde{b}_n\|^2 \leq \|b_1\|^{2n} \|b_2\|^{2(n-1)} \cdot \|b_n\|^2$$
$$\leq \left(\|b_1\|^2 \cdots \|b_n\|\right)^{2n}$$

Running time

$$D = \left\|\tilde{b}_1\right\|^{2n} \left\|\tilde{b}_2\right\|^{2(n-1)} \cdot \left\|\tilde{b}_n\right\|^2$$

Claim: For every i, $\left\|\tilde{b}_i\right\| \leq \|b_i\|$.

Proof: $\left\langle \tilde{b}_i, \tilde{b}_i \right\rangle = \left\langle b_i - \sum_{j=1}^{i-1} \mu_{ij} \tilde{b}_j \, , b_i - \sum_{j=1}^{i-1} \mu_{ij} \tilde{b}_j \right\rangle = \langle b_i, b_i \rangle$
$- 2\left\langle b_i, \sum_{j=1}^{i-1} \mu_{i,j} \, \tilde{b}_j \right\rangle + \sum_{j=1}^{i-1} \mu_{ij}^2 \langle \tilde{b}_j, \tilde{b}_j \rangle$

Now use $\mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$ to get

$$\left\langle \tilde{b}_i, \tilde{b}_i \right\rangle = \langle b_i, b_i \rangle - \sum_{i=1}^{j-1} \frac{\left\langle b_i, \tilde{b}_j \right\rangle^2}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$$

In the beginning, on input $b_1, \ldots, b_n$, we have

$$D = \left\|\tilde{b}_1\right\|^{2n} \left\|\tilde{b}_2\right\|^{2(n-1)} \cdot \left\|\tilde{b}_n\right\|^2 \leq \|b_1\|^{2n} \|b_2\|^{2(n-1)} \cdot \|b_n\|^2$$
$$\leq \left(\|b_1\|^2 \cdots \|b_n\|^2\right)^{2n}$$

Running time

$$D = \|\tilde{b}_1\|^{2n} \|\tilde{b}_2\|^{2(n-1)} \cdot \|\tilde{b}_n\|^2$$

Summary:

1. $D \leq (\|b_1\|^2 \cdots \|b_n\|^2)^{2n}$ in the beginning
2. $D \geq 1$ on each step
3. After every swap, D drops by a factor of $\frac{3}{4}$

Corollary: total number of iterations is at most

$$O\big(n(\log\|b_1\| + \cdots + \log\|b_n\|)\big)$$

polynomial in the input length!

+זמן ריצה של גרם-שמידט
+עוד כל מני דברים
⇐ עדיין פולינומי.

Summary

- Approximation algorithms exist for many NP-hard problems
- Sometimes we can't do anything beyond picking a solution at random
- But sometimes we can!
- For some problems: A $(1 + \varepsilon)$-approximation algorithm for every $\varepsilon > 0$