Approximation algorithms
+
Limits of computation & undecidability
+
Concluding remarks

ORF 523

Lecture 19

Instructor: Amir Ali Ahmadi, TA: G. Hall,C.Y. Liu, Spring 2017

PRINCETON UNIVERSITY ORFE

Convex relaxations with worst-case guarantees

- One way to cope with NP-hardness is to aim for suboptimal solutions with guaranteed accuracy
- Convex relaxations provide a powerful tool for this task

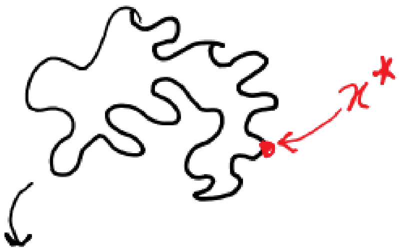
α - approximation algorithm

- Minimization: $f^* \leq \hat{f} \leq \alpha f^*$ $(\alpha > 1)$
- Maximization: $\alpha f^* \leq \hat{f} \leq f^*$ $(0 < \alpha < 1)$

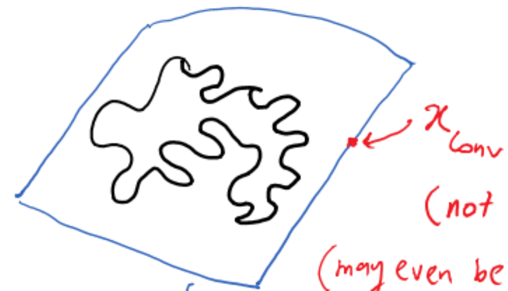
- For randomized algorithms, require this in expectation.

General recipe for convex optimization based approx. algs.

- **Relax**



feasible set of the NP-hard problem



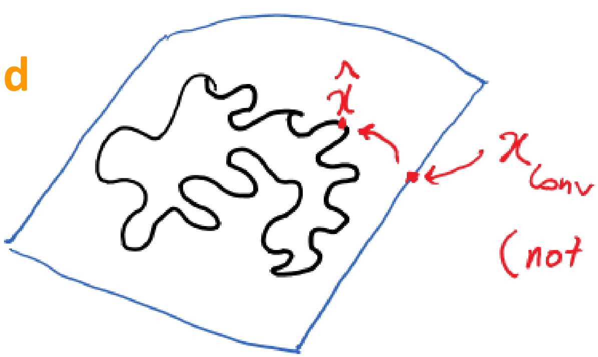
(not feasible)

(may even be in lifted space)

feasible set of the convex relaxation

$f_{conv} := f(x_{conv}) \leq f^* := f(x^*)$ (for a minimization problem)

- **Round**

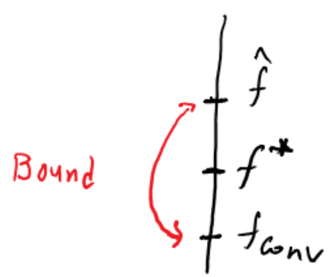


(not feasible)

$\hat{x}$: rounded solution, feasible.

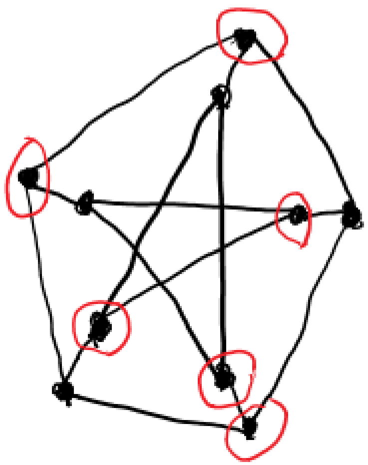
Let $\hat{f} := f(\hat{x})$.

- **Bound**



$\downarrow$ minimization

Vertex Cover



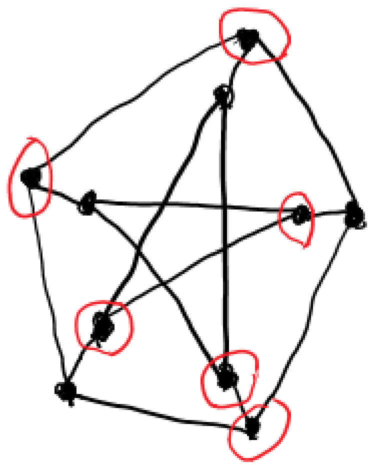
- **Vertex Cover:** A subset of the the vertices that touch all the edges.

- **VERTEX COVER:** Given a graph G(V,E) and an integer k, is there a vertex cover of size smaller than k?

- VERTEX COVER is NP-hard.

$$vc(G) = n - \alpha(G)$$

2-approximation for vertex cover via LP



- Vertex cover as an integer program:

$$f^* := VC(G) = \min_x \sum_{i=1}^n x_i$$

$$x_i + x_j \geq 1 \quad \forall (i,j) \in E$$

$$x_i \in \{0,1\} \quad i = 1, \ldots, n$$

- LP relaxation:

$$f_{LP} := \min \sum_{i=1}^n x_i$$

$$x_i + x_j \geq 1, \quad \text{if } (i,j) \in E$$

$$0 \leq x_i \leq 1 \quad i = 1, \ldots, n$$

Obviously $f_{LP} \leq f^*$. Denote the optimal solution by x_{LP}.

Rounding & Bounding

Rounding: Set $\hat{x}_i = \begin{cases} 1, & \text{if } x_{LP,i} \geq 1/2. \\ 0 & \text{otherwise} \end{cases}$

- $\hat{x}$ gives a valid vertex cover b/c $\forall$ edges, one of the two end nodes in the LP solution must be $\geq 1/2$.
- So $f^* \leq \hat{f} := \sum_i \hat{x}_i$

Bounding:

- $\hat{f} \leq 2 f_{LP}$

  b/c in worst case, we are changing a bunch of "1/2's" to "1's".
- $\Rightarrow \hat{f} \leq 2 f^*$

  b/c $f_{LP} \leq f^*$

Overall: $\boxed{f^* \leq \hat{f} \leq 2 f^*}$

- Best constant approximation ratio known to date.

PRINCETON UNIVERSITY ORFE

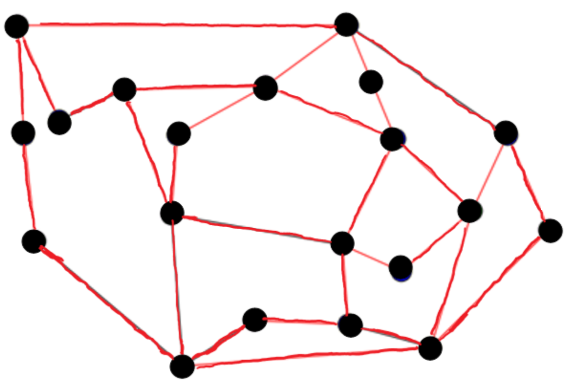
MAXCUT

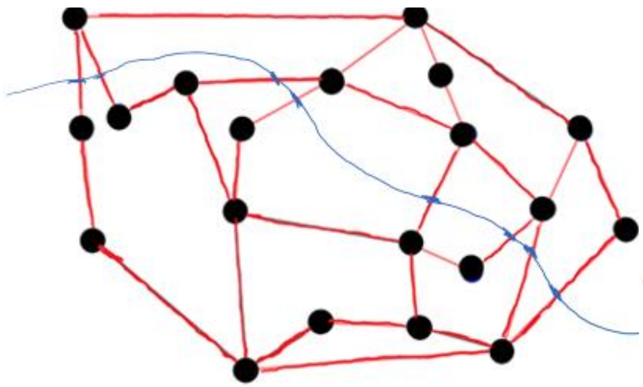
MAXCUT

- **Input:** A graph $G(V, E)$, nonnegative rational numbers w_{ij} on each edge, a rational number k.

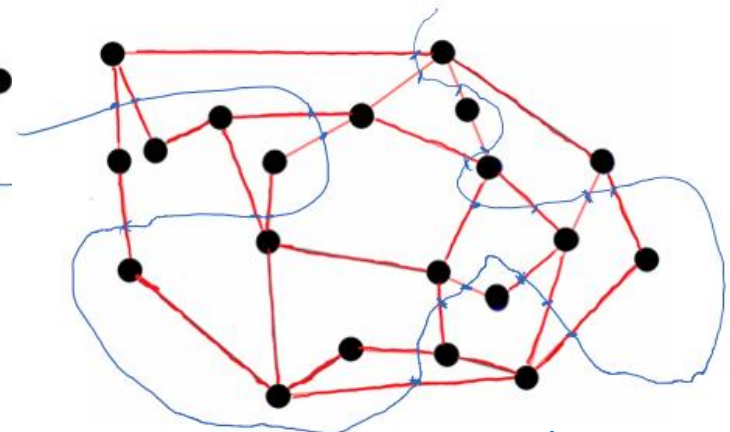
- **Question:** Is there a cut of value $\geq k$?

- Examples with edge costs equal to 1:





- Cut value=8



- Cut value=23 (optimal)

- MAXCUT is NP-complete (e.g., relatively easy reduction from 3SAT)

- Contrast this to MINCUT which can be solved in poly-time by LP

A .878-approximation algorithm for MAXCUT via SDP

- Seminal work of Michel Goemans and David Williamson (1995)
- Before that the best approximation factor was ½
- First use of SDP in approximation algorithms
- Still the best approximation factor to date
- An approximation ratio better than 16/17=.94 implies P=NP (Hastad)
- Under stronger complexity assumptions, .878 is optimal
- No LP-based algorithm is known to match the SDP-based 0.878 bound

The GW SDP relaxation

$$f^* = \max \frac{1}{4}\sum_{i,j} w_{ij}(1 - x_i x_j) = \frac{1}{4}\sum_{i,j} w_{ij} - \frac{1}{4}\underbrace{\begin{bmatrix} \min & \sum_{i,j} w_{ij} x_i x_j \\ \text{s.t.} & x_i^2 = 1 \end{bmatrix}}_{:= f_2^*}$$

$$\text{s.t. } x_i^2 = 1$$

$$Q_{ij} = \begin{cases} 0 & i = j \\ w_{ij} & i \neq j \end{cases} \qquad \text{Then, } f_2^* = \min \; x^T Q x$$

$$\text{s.t. } x_i^2 = 1$$

- It's SDP relaxation: $f_{2SDP} := \min_{X \in S^{n \times n}} \operatorname{Tr}(QX)$

$$X_{ii} = 1$$

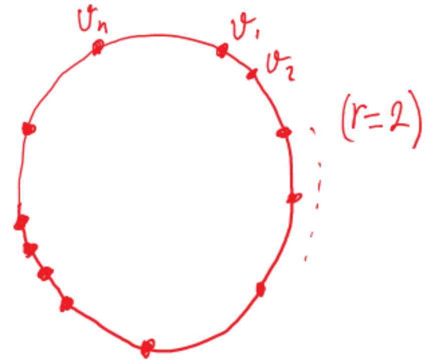
$$X \succeq 0$$

$$f_{2SDP} \leq f_2^*$$

The GW rounding

- If the optimal solution of the SDP is rank-1 $\Rightarrow$ done.
- If not,

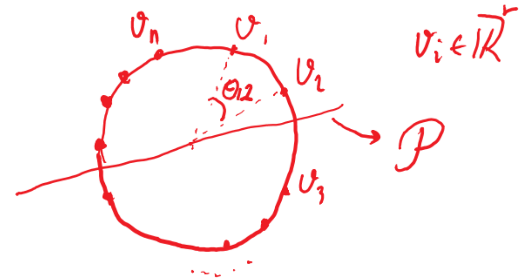
$$\underset{n\times n}{X} = \underset{n\times r}{V^T} \underset{r\times n}{V}, \quad \text{where } r = \text{rank}(X).$$



- Denote the columns of V by $v_i \in \mathbb{R}^r$: $V = [v_1, \ldots, v_n]$
- Observe that $X_{ij} = v_i^T v_j$
- So $\|v_i\| = 1 \ \forall i$ (b/c $X_{ii} = 1$ must hold).
- So we have n points $v_1, \ldots, v_n$ on the unit sphere S^{r-1} in $\mathbb{R}^r$.
- Generate a point $p \in S^{r-1}$ uniformly at random (e.g., p=randn(r,1); p=p/norm(p,2);)
- Set $x_i = \begin{cases} 1 & \text{if } p^T v_i \geq 0 \\ -1 & \text{if } p^T v_i < 0 \end{cases} \qquad i = 1, \ldots, n.$

The GW bound

$$\mathcal{P} := \{x \in \mathbb{R}^r \mid p^T x = 0\}$$



$$\hat{f}_2 = E\Big[\sum_{ij} w_{ij} x_i x_j\Big] = \sum_{i,j} w_{ij} E[x_i x_j]$$

$$\frac{\theta_{ij}}{\pi} = \frac{1}{\pi} \arccos(v_i^T v_j)$$

$$E[x_i x_j] = 1 \cdot \Pr[v_i \,\&\, v_j \text{ on same side of } \mathcal{P}] - 1 \cdot \Pr[v_i \,\&\, v_j \text{ on different sides of } \mathcal{P}]$$

$(i \neq j)$

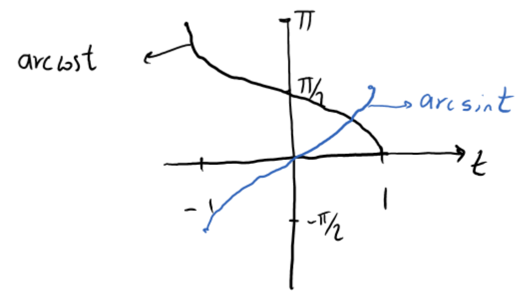
$$= 1 - \frac{\theta_{ij}}{\pi} - \frac{\theta_{ij}}{\pi}$$

$$= 1 - \frac{2}{\pi} \arccos v_i^T v_j$$

Well-defined b/c $X_{ij} \le 1$ (why?)

$$= \frac{2}{\pi} \arcsin v_i^T v_j$$

$\arcsin t + \arccos t = \pi/2$



The GW bound

$$\Rightarrow \boxed{\hat{f}_2 = \frac{2}{\pi} \sum_{i,j} w_{ij} \arcsin X_{ij}}$$

∘ Recall that $f^* = \frac{1}{4} \left(\sum_{i,j} w_{ij} - f_2^* \right)$

∘ Let $\hat{f} := \frac{1}{4} \left(\sum_{i,j} w_{ij} - \hat{f}_2 \right) = \frac{1}{4} \left(\sum_{i,j} w_{ij} - \frac{2}{\pi} \sum_{i,j} w_{ij} \arcsin X_{ij} \right)$

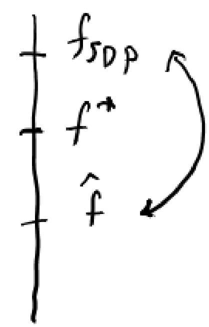
$$= \frac{1}{4} \sum_{i,j} w_{ij} \left[1 - \frac{2}{\pi} \arcsin X_{ij} \right] = \underline{\underline{\frac{1}{4} \cdot \frac{2}{\pi} \sum_{i,j} w_{ij} \arccos X_{ij}}}$$

Relating this to the SDP optimal value

$$\hat{f} = \frac{1}{2\pi} \sum_{i,j} w_{ij} \arccos X_{ij}$$

$$f_{SDP} := \frac{1}{4}\left(\sum_{i,j} w_{ij} - f_{2SDP} \right)$$

$$= \frac{1}{4} \sum_{i,j} w_{ij} - \frac{1}{4} \sum_{i,j} w_{ij} X_{ij} = \underline{\underline{\frac{1}{4} \sum_{i,j} w_{ij} \left(1 - X_{ij}\right)}}$$



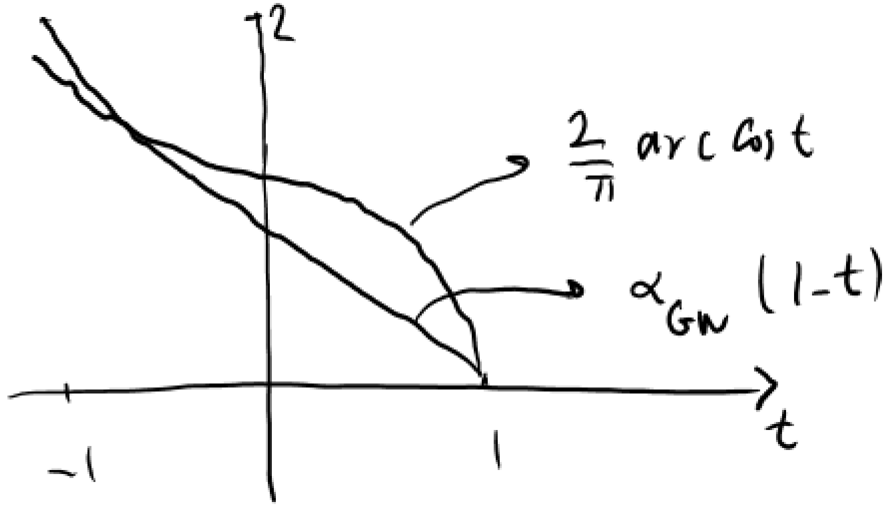
Want to argue: $\alpha f_{SDP} \le \hat{f}$

for α as large as possible.

Need: $\alpha(1-t) \le \frac{2}{\pi} \arccos t \quad \forall t \in [-1,1]$

The final step

Need: $\alpha(1-t) \le \frac{2}{\pi} \arccos t \quad \forall t \in [-1,1]$



Optimal α: $\alpha_{GW} \approx 0.878$

- Bound term by term. You achieve this approximation ratio.

Optimal α: $\alpha_{GW} \approx 0.878$

Sometimes people obtain mathematically significant license plates purely by accident, without making a personal selection. A striking example of this phenomenon is the case of Michel Goemans, who received the following innocuous-looking plate from the Massachusetts Registry of Motor Vehicles when he and his wife purchased a Subaru at the beginning of September 1993:

Two weeks later, Michel got together with his former student David Williamson, and they suddenly realized how to solve a problem that they had been working on for some years: to get good approximations for maximum cut and satisfiability problems by exploiting semidefinite programming. Lo and behold, their new method—which led to a famous, award-winning paper [15]—yielded the approximation factor .878! There it was, right on the license, with `C`, `S`, and `W` standing respectively for cut, satisfiability, and Williamson.

(By D.E. Knuth)

Limits of computation

What theory of NP-completeness established for us

- Recall that all NP-complete problems polynomially reduce to each other.
- If you solve one in polynomial time, you solve ALL in polynomial time.

- Assuming P≠NP, no NP-complete problem can be solved in polynomial time.
- This shows limits of *efficient* computation (under a complexity theoretic assumption)

- **What's coming next:** limits of computation in general (and under no assumptions)

PRINCETON UNIVERSITY ORFE

Matrix mortality

Consider a collection of m $n \times n$ matrices $\{A_1, \ldots, A_m\}$.

We say the collection is mortal if there is a finite product out of the matrices (possibly allowing repetition) that gives the zero matrix.

Example 1:

```
A1 =

     0     0
     0     1

A2 =

     0     1
    -1     0
```

```
>> A1*A2

ans =

     0     0
    -1     0

>> A1*A2*A1*A2

ans =

     0     0
     0     0
```

Example from [W11].

Mortal.

Matrix mortality

Consider a collection of m $n \times n$ matrices $\{A_1, \ldots, A_m\}$.

We say the collection is mortal if there is a finite product out of the matrices (possibly allowing repetition) that gives the zero matrix.

Example 2:

```
A1 =

     1    -2
     3     0

A2 =

     0    -1
    -1     0

A3 =

     1     2
     0    -1
```

Not mortal. (How to prove that?)

- In this case, can just observe that all three matrices have nonzero determinant.
- Determinant of product=product of determinants.

But what if we aren't so lucky?

```
>> A1*A2*A3

ans =

     2     5
     0     3

>> A1*A2*A3*A1*A3

ans =

    17    38
     9    18

>> A2*A2*A3*A1*A3

ans =

     7    16
    -3    -6

>> A2*A2*A1*A3

ans =

     1     4
     3     6

>> ...
```

Matrix mortality

- **MATRIX MORTALITY**

- **Input:** A set of m $n \times n$ matrices with integer entries.
- **Question:** Is there a finite product that equals zero?

Thm. MATRIX MORTALITY is **undecidable** already when

- $n = 3, m = 7,$

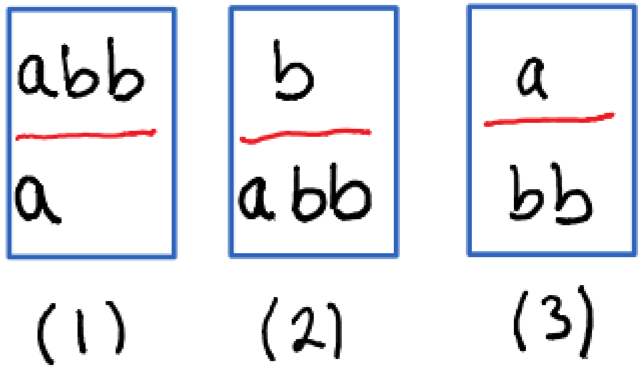
  or

- $n = 21, m = 2.$

- This means that there is no finite time algorithm that can take as input two 21x21 matrices (or seven 3x3 matrices) and always give the correct yes/no answer to the question whether they are mortal.
- This is a definite statement.
  (It doesn't depend on complexity assumptions, like P vs. NP or alike.)

  - How in the world would someone prove something like this?
  - By a reduction from another undecidable problem!

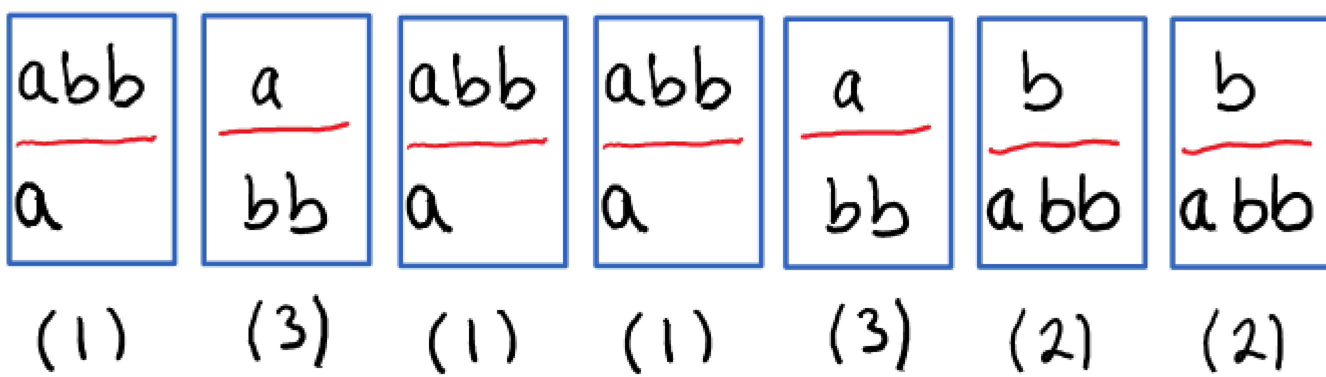
The Post Correspondence Problem (PCP)



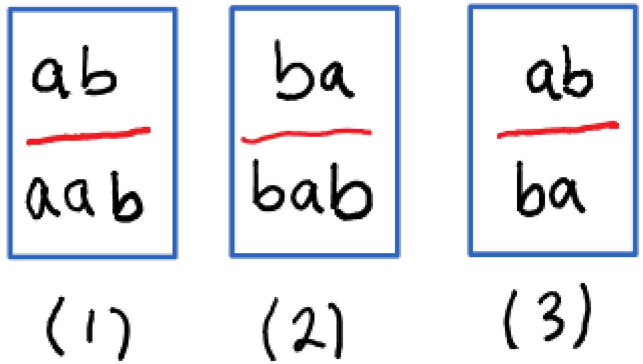
Emil Post
(1897-1954)

Given a set of dominos such as the ones above,
can you put them next to each other (repetitions allowed) in such a way that the top row reads the same as the bottom row?

Answer to this instance is YES:



The Post Correspondence Problem (PCP)



Emil Post
(1897-1954)

What about this instance?

Answer is NO. Why?

There is a length mismatch, unless we only use (3), which is not good enough.

But what if we aren't so lucky?

The Post Correspondence Problem (PCP)

- **PCP**

- **Input:** A finite set of m domino types with letters *a* and *b* written on them.
- **Question:** Can you put them next to each other (repetition allowed) to get the same word in the top and bottom row?

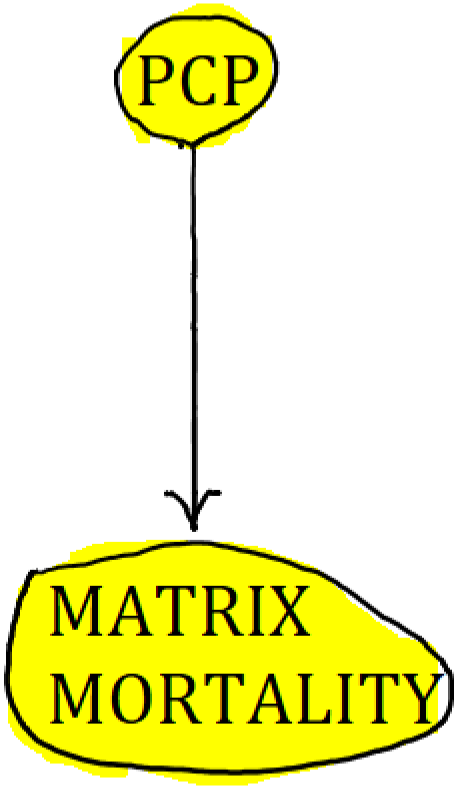
Emil Post
(1897-1954)

Thm. PCP is **undecidable** already when $m = 7$.

- Again, we are ruling out any finite time algorithm.
- PCP is decidable for $m = 2$.
- Status unknown for $2 < m < 7$.

Reductions

- There is a rather simple reduction from PCP to MATRIX MORTALITY; see, e.g., [Wo11].



- This shows that if we could solve MATRIX MORTALITY in finite time, then we could solve PCP in finite time.
- It's impossible to solve PCP in finite time (because of another reduction!)
- Hence, it's impossible to solve MATRIX MORTALITY in finite time.
- Note that these reductions only need to be finite in length (not polynomial in length like before).

Integer roots of polynomial equations

- Can you give me three positive integers x, y, z such that

$$x^2 + y^2 = z^2?$$

- Sure: (3, 4, 5) (5, 12, 13) (8, 15, 17) (7, 24, 25) (20, 21, 29) (12, 35, 37) (9, 40, 41) (28, 45, 53)

And there are infinitely many more…

- How about $x^3 + y^3 = z^3$?
- How about $x^4 + y^4 = z^4$?
- How about $x^5 + y^5 = z^5$?

Fermat's last theorem tells us the answer is NO to all these instances.

Integer roots to polynomial equations

What about integer solutions to $x^3 + y^3 + z^3 = 29$?

YES: (3,1,1)

What about $x^3 + y^3 + z^3 = 30$?

Looped in MATLAB over all $|x, y, z|$ less than 10 million→ no solution!

But answer is YES!! $(-283059965, -2218888517, 2220422932)$

What about $x^3 + y^3 + z^3 = 33$?

No one knows!

Integer roots of polynomial equations

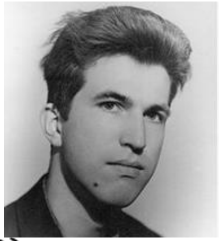
- **POLY INT**

- **Input:** A polynomial p in n variables and of degree d.

- **Question:** Does it have an integer root?

From Logicomix

- **Hilbert's 10th problem (1900):** Is there an algorithm for POLY INT?



- **Matiyasevich (1970)** – building on earlier work by Davis, Putnam, and Robinson:
  No! The problem is undecidable.

- It's undecidable even in fixed degree and dimension (e.g., $d = 4, n = 58$).

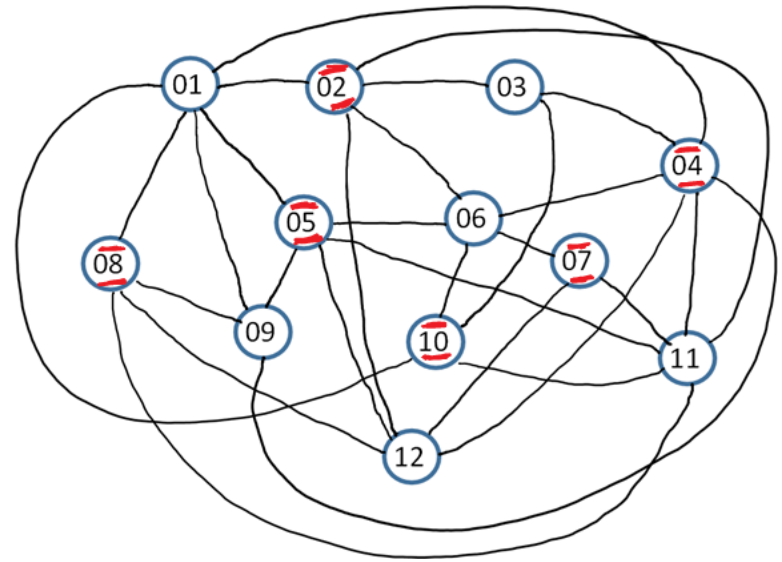
Real/rational roots of polynomial equations

- If instead of integer roots, we were testing existence of **real roots**, then the problem would become decidable.
  - Such finite-time algorithms were developed in the past century (Tarski–Seidenberg)
- If instead we were asking for existence of **rational roots**,
  - We currently don't know if it's decidable!

- Nevertheless, both problems are NP-hard. For example for
  - A set of equations of degree 2
  - A single equation of degree 4.
  - Proof on the next slide.

A simple reduction

- We give a simple reduction from STABLE SET to show that testing existence of a real (or rational or integer) solution to a set of quadratic equations is NP-hard.
- Contrast this to the case of linear equations which is in P.

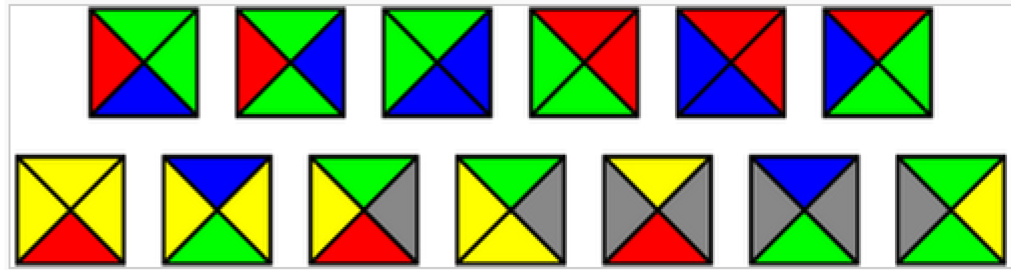


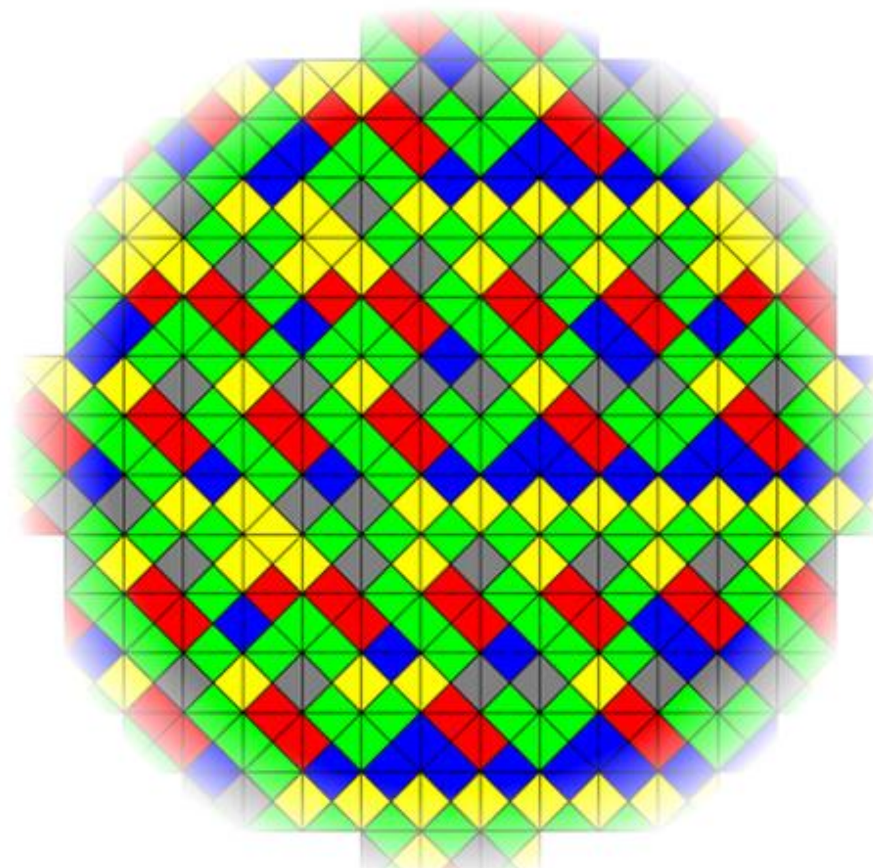
$$\exists \text{ stable set of size } K \iff \exists x \text{ s.t. } \begin{bmatrix} x_1 + \cdots + x_n = K \\ x_i + x_j \leq 1 \quad i,j \in E \\ x_i \in \{0,1\} \end{bmatrix} \iff \exists x, z \text{ s.t. } \begin{bmatrix} (x_1 + \cdots + x_n - K)^2 = 0 \\ 1 - x_i - x_j = z_{ij}^2 \quad i,j \in E \\ x_i(1 - x_i) = 0 \quad i = 1, \ldots, n \end{bmatrix}$$

- How would you go from here to a single equation of degree 4?

Tiling the plane

- Given a finite collection of tile types, can you tile the 2-dimenstional plane such that the colors on all tile borders match.
- Cannot rotate or flip the tiles.
- The answer is YES, for the instance presented.
- But in general, the problem is undecidable.





Stability of matrix pairs

- We say a matrix A is stable if all its eigenvalues are strictly inside the unit circle in the complex plane.
- We say a pair of matrices {A1, A2} is stable if all matrix products out of A1 and A2 are stable.
- Given {A1,A2}, let a* be the largest scalar such that the pair {aA1,aA2} is stable for all a<a*.
- Define r(A1,A2) to be 1/a*.

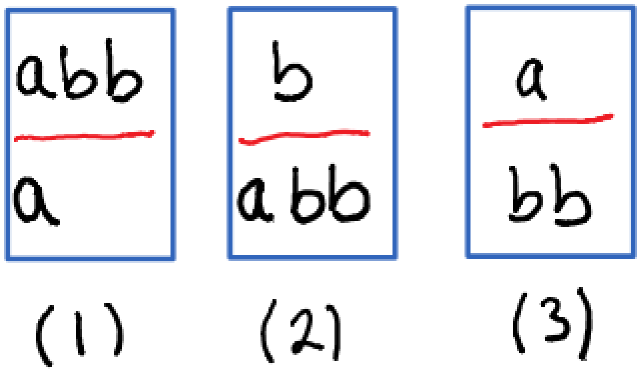
- For a single matrix A, r(A) is the same thing as the spectral radius and can be computed in polynomial time.

- **STABLE MATIRX PAIR:** Given a pair of matrices A1,A2, decide if r(A1,A2)<=1?

- **THM.** STABLE MATRIX PAIR is undecidable already for 47x47 matrices.

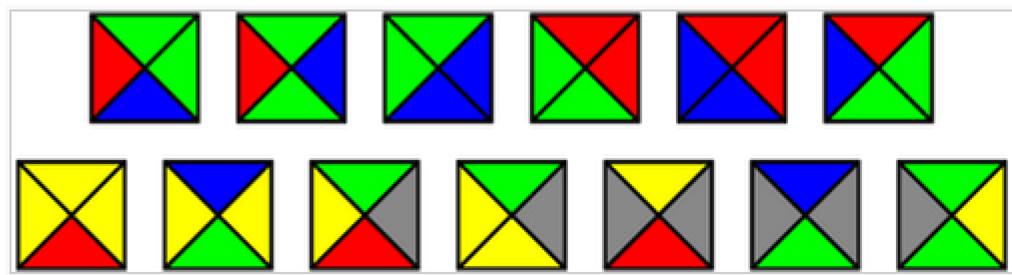
PRINCETON UNIVERSITY ORFE

All undecidability results are proven via reductions



$$A1 = \begin{matrix} 1 & -2 \\ 3 & 0 \end{matrix} \qquad A2 = \begin{matrix} 0 & -1 \\ -1 & 0 \end{matrix} \qquad A3 = \begin{matrix} 1 & 2 \\ 0 & -1 \end{matrix}$$

$$x^3 + y^3 + z^3 = 33?$$



But what about the first undecidable problem?

The halting problem

- **HALTING**

- **Input:** A file containing a computer program p and a file containing an input x to the computer program.

- **Question:** Does p ever terminate (aka halt) when given input x?

An instance of HALTING:

```
function gradient_descent(x)

%gradient descent with exact line search for minimizing a quadratic
%function.
Q=[8 0;0 17];
b=[136;154];
xvec=[];
while norm(Q*x-b,2)>10^-5
    alpha=((Q*x-b)'*(Q*x-b))/((Q*x-b)'*Q*(Q*x-b));
    x=x-alpha*(Q*x-b);
    xvec=[xvec x];
end
```

Program p

x= [3;63];

The halting problem

An instance of HALTING:

```
function gradient_descent(x)

%gradient descent with exact line search for minimizing a quadratic
%function.
Q=[8 0;0 17];
b=[136;154];
xvec=[];
while norm(Q*x-b,2)>10^-5
     alpha=((Q*x-b)'*(Q*x-b))/((Q*x-b)'*Q*(Q*x-b));
     x=x-alpha*(Q*x-b);
     xvec=[xvec x];
end
```

Program p   x= [3;63];

- Both the program p and the input x can be represented with a finite number of bits.
- Can there be a program --- call it **terminates(p,x)** --- that takes p and x as input and always outputs the correct yes/no answer to the question: does p halt on x?
  - We'll show that the answer is no!
  - This will be a proof by contradiction.

The halting problem is undecidable

Proof.

- Suppose there was such a program terminates(p,x).
- We'll use it to create a new program paradox(z):

```
function paradox(z)

1: if terminates(z,z)==1 goto line 1.
```

- The input z to paradox is a computer program.
- As a subroutine, paradox asks terminates to check whether a given computer program z halts when given itself as input. (This is perfectly legal as any program is just a finite number of bits.)
- Note that paradox halts on z if and only if z does not halt when given itself as input.

  - What happens if we run paradox(paradox) ?!
    - If paradox halts on itself, then paradox doesn't halt on itself.
    - If paradox doesn't halt on itself, then paradox halts on itself.
    - This is a contradiction → terminates can't exist.

The halting problem (1936)

Alan Turing
(1912-1954)

Self-reference – a simpler example

Russell's paradox

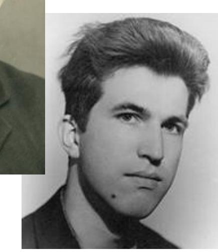
The power of reductions (one last time)

A simple paradox/puzzle:

```
function paradox(z)
1: if terminates(z,z)==1 goto line 1.
```

(lots of nontrivial mathematics,

including the formalization of the notion of an “algorithm”)



A fundamental algorithmic question:

- **POLY INT**
- **Input:** A polynomial p in n variables and degree d.
- **Question:** Does it have an integer root?

A remarkable implication of this…

- Consider the following long-standing open problems in mathematics (among numerous others!):
- Is there an odd perfect number? (an odd number whose proper divisors add up to itself)
- Is every even integer larger than 2 the sum of two primes? (The Goldbach conjecture)

In each case, you can explicitly write down a polynomial of degree 4 in 58 variables, such that if you could decide whether your polynomial has an integer root, then you would be able to solve the open problem.

Proof.

1) Write a code that looks for a counterexample.
2) Code does not halt if and only if the conjecture is true (one instance of the halting problem!)
3) Use the reduction to turn this into an instance of POLY INT.

How to deal with undecidability?

- Well we have only one tool in this class:

Convex optimization!

Stability of matrix pairs

- We say a matrix A is stable if all its eigenvalues are strictly inside the unit circle on the complex plane.
- We say a pair of matrices {A1, A2} is stable if all matrix products out of A1 and A2 are stable.
- Given {A1,A2}, let a* be the largest scalar such that the pair {aA1,aA2} is stable for all a<a*.
- Define r(A1,A2) to be 1/a*.

- For a single matrix A, r(A) is the same thing as the spectral radius and can be computed in polynomial time.

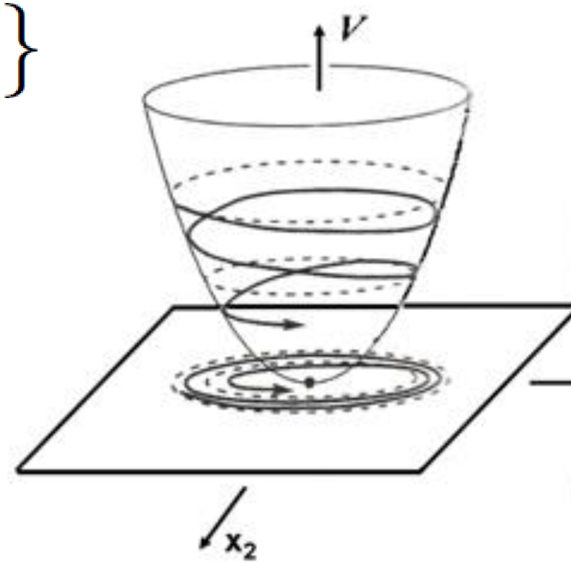
- **STABLE MATIRX PAIR:** Given a pair of matrices A1,A2, decide if r(A1,A2)<=1?

- **THM.** STABLE MATRIX PAIR is undecidable already for 47x47 matrices.

Common Lyapunov function

$$x_{k+1} = A_i x_k$$

$$\mathcal{A} := \{A_1, ..., A_m\}$$



[Chaos, Yorke]

If we can find a function $V(x) : \mathbb{R}^n \to \mathbb{R}$

such that $$V(x) > 0,$$

$$V(A_i x) < V(x), \ \forall i = 1, \ldots, m$$

then, the matrix family is stable.

Such a function always exists! But may be extremely difficult to find!!

Computationally-friendly common Lyapunov functions

$$x_{k+1} = A_i x_k \qquad \mathcal{A} := \{A_1, ..., A_m\}$$

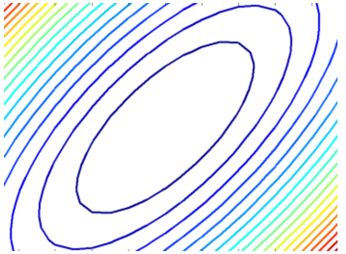
If we can find a function $V(x) : \mathbb{R}^n \to \mathbb{R}$

such that
$$V(x) > 0,$$
$$V(A_i x) < V(x), \ \forall i = 1, \ldots, m$$

then the matrix family is stable.

- Common quadratic Lyapunov function:

$$V(x) = x^T P x$$



$$P \succ 0$$

$$A_i^T P A_i \prec P \quad i=1,\ldots,m$$

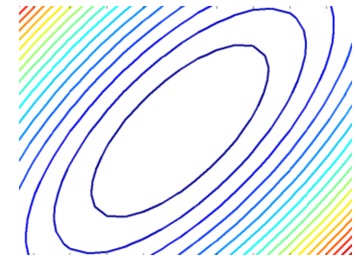
PRINCETON UNIVERSITY ORFE

SDP-based approximation algorithm!

$$V(x) = x^T P x$$

$$P \succ 0$$

$$A_i^T P A_i \prec P \quad i=1,\dots,m$$



- Exact if you have a single matrix (we proved this).
- For more than one matrix: $\beta^* =$ largest β such that SDP feasible for

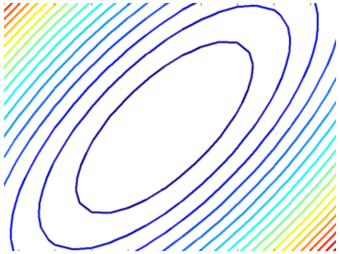
$$\beta\mathcal{A} := \{\beta A_1, \dots, \beta A_m\}.$$

let $\hat{r}(\mathcal{A}) := \dfrac{1}{\beta^*}.$

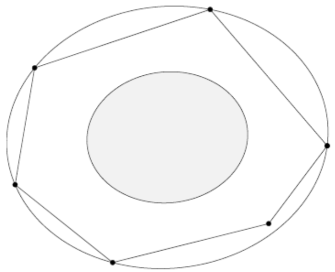
<u>Thm.</u> $\dfrac{1}{\sqrt{n}}\hat{r}(\mathcal{A}) \leq r(\mathcal{A}) \leq \hat{r}(\mathcal{A})$

Proof idea

Thm. $\frac{1}{\sqrt{n}}\hat{r}(\mathcal{A}) \leq r(\mathcal{A}) \leq \hat{r}(\mathcal{A})$



- **Upper bound:**
  - Existence of a quadratic Lyapunov function sufficient for stability
- **Lower bound (due to Blondel and Nesterov):**
  - We know from converse Lyapunov theorems that there always exist a Lyapunov function which is a norm
  - We are approximating the (convex) sublevel sets of this norm by ellipsoids
  - Apply John's ellipsoid theorem (see Section 8.4 of Boyd&Vandenberghe)

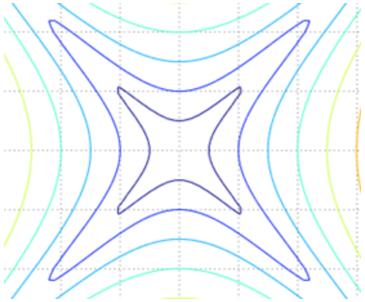


How can we do better than this SDP?

- Why look only for quadratic Lyapunov functions?
- Look for higher order polynomial Lyapunov functions and apply our the **SOS relaxation!**

$$V(x) = c_1 x_1^4 + c_2 x_1 x_2^3 + \cdots + c_{17} x_2 x_3 x_4 x_5 + \cdots + c_{70} x_5^4$$

(w.l.o.g. take V to be homogeneous)



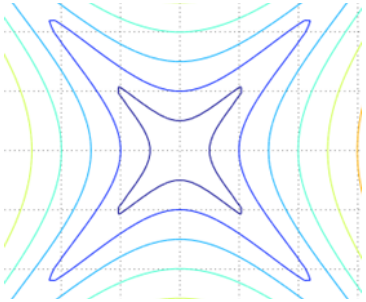
Require $V(x)$ SOS (and $V \neq 0$)

$V(x) - V(A_i x)$ SOS $i = 1, \ldots, m$

Common SOS Lyapunov functions

$V(x) = c_1 x_1^4 + c_2 x_1 x_2^3 + \cdots + c_{17} x_2 x_3 x_4 x_5 + \cdots + c_{70} x_5^4$

(w.l.o.g. take V to be homogeneous)



Require $V(x)$ SOS (and $V \neq 0$)

$V(x) - V(A_i x)$ SOS $i = 1, \ldots, m$

- **Remarks:**
- Since the dynamics $x_{k+1} = A_i x_k$ is homogeneous in x, we can parameterize our polynomial V to be homogeneous.
  - This is just like the quadratic case: we look for $V(x) = x^T P x$, without linear or constant terms.
- Note that the condition $V(x)$ SOS implies that V is nonnegative. To make sure that it is actually positive definite (i.e., $V(x) > 0, \forall x \neq 0$), we can instead impose

$$V(x) - \beta(x_1^2 + \cdots + x_n^2)^d \; SOS,$$

where β is a small constant (say 0.01), and $2d$ is the degree of V.

This condition implies that V is positive on the unit sphere, which by homogeneity implies that V is positive everywhere.

SOS-based approximation algorithm!

$\beta^* =$ largest β such that the SOS program feasible

for $\beta\mathcal{A} := \{\beta A_1, \cdots, \beta A_m\}$.

Let $\hat{r}_{2d}(\mathcal{A}) := \dfrac{1}{\beta^*}$.

<u>Thm.</u> $\dfrac{1}{\sqrt[2d]{n}}\hat{r}_{2d}(\mathcal{A}) \leq r(\mathcal{A}) \leq \hat{r}_{2d}(\mathcal{A})$

SOS-based approximation algorithm!

Comments:

- For 2d=2, this exactly reduces to our previous SDP!
(SOS=nonnegativity for quadratics!)

- We are approximating an undecidable quantity to arbitrary accuracy in polynomial time!!

- In the past couple of decades, approximation algorithms have been actively studied for a multitude of NP-hard problems. There are noticeably fewer studies on approximation algorithms for undecidable problems.

- In particular, the area of integer polynomial optimization seems to be wide open.

Main messages of the course

- Convex optimization is a very powerful tool in computational mathematics.
  - Its power goes much beyond LPs – we saw many examples and applications:
  - In finance (minimum risk portfolio optimization)
  - In machine learning (maximum-margin support vector machines)
  - In combinatorial optimization (bounding NP-hard quantities, clique number, maxcut, vertec cover, etc.)
  - In dynamics and control (finding stabilizing controllers)
  - In information theory (bounding the zero-error capacity of a channel)
  - In approximation algorithms (relax, round, bound)
  - Robust optimization (even robust LP)
- Family of tractable convex programs: LP $\subset$ QP $\subset$ QCQP $\subset$ SOCP $\subset$ SDP
  - SDPs are the broadest in this class and the most powerful
  - We emphasized the power of SDPs in algorithm design over LPs

Main messages of the course

- Which optimization problems are tractable?
  - Convexity is a good rule of thumb.
  - But there are nonconvex problems that are easy (SVD, S-lemma, etc.)
  - And convex problems that are hard (testing matrix copositivity or polynomial nonnegativity).
  - In fact, we showed that every optimization problem can be “written” as a convex problem.
  - Computational complexity theory is essential to answering this question!
- Hardness results
  - Theory of NP-completeness: gives overwhelming evidence for intractability of many optimization problems of interest (no polynomial-time algorithms)
  - Undecidability results rule out finite time algorithms unconditionally
- Dealing with intractable problems
  - Solving special cases exactly
  - Looking for bounds via convex relaxations
  - Approximation algorithms

Main messages of the course

- Sum of squares optimization
  - A very broad and powerful technique that turns any semialgebraic problem into a sequence of semidefinite programs
  - This includes all of NP! But much more
  - It needs absolutely no convexity assumptions!
  - You should think of it anytime you see the inequality sign: $\geq$!!

- Computation, computation, computation
  - Be friends with CVX, YALMIP, and alike.
  - Develop a computational taste in research
  - As Stephen Boyd calls it: Work on "actionable theory", which means "theory which can be implemented as algorithms" (or shows limitations of algorithms)

The take-home assignment

- Tentatively scheduled to go live on Wednesday, May 17, at 9AM.
- Tentatively scheduled to be due on Monday, May 22, at 9 AM in the ORF 523 box in Sherrerd 123.
- Georgina and I will hold office hours before the exam. Time TBA.

- No collaboration allowed.
- Can only use material from this course (notes, psets).
- Please use Piazza for clarification questions (and for clarification questions only)!
- No private questions on Piazza, no emails.
- More time than needed – please keep your answers brief and to the point.
- Please keep an electronic copy of your exam.
- If you've been doing the problem sets and following lecture, you should be OK ☺

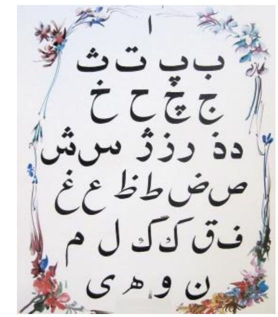
Some open problems that came up in this course

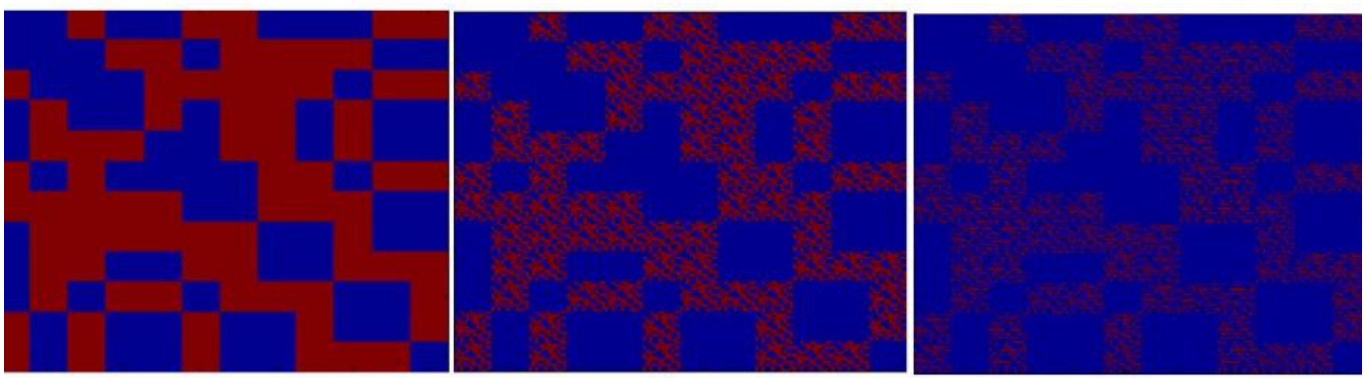
(Many are high-risk (and high-payoff))

1) Compute the Shannon capacity of C7. More generally, give better SDP-based upper bounds on the capacity than Lovasz.

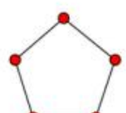
$$\Theta(G) = \lim_{k\to\infty} \sqrt[k]{\alpha(G^k)}$$

(α: size of max stable set)





G $\quad$ G^2 $\quad$ G^3



$\Theta(C_5) = \sqrt{5}$

$\Theta(C_7) = ?$ (open)

Some open problems that came up in this course

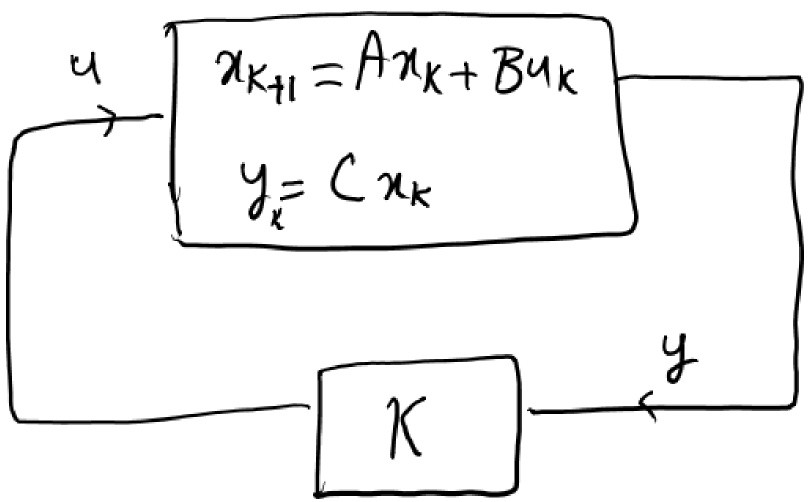
2) Is there a polynomial time algorithm for output feedback stabilization?

Given matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times k}$, $C \in \mathbb{R}^{r\times n}$, does there exist a matrix $K \in \mathbb{R}^{k\times r}$

such that

$$A + BKC$$

is stable?



Some open problems that came up in this course

3) Can you find a local minimum of a quadratic program in polynomial time?

4) Construct a convex, nonnegative polynomial that is not a sum of squares.

5) Can you beat the GW 0.878 algorithm for MAXCUT?

Check your license plate, you never know!

Thank you!
AAA
May 4, 2017

References

- References:

-[Wo11] M.M. Wolf. Lecture notes on undecidability, 2011.

-[Po08] B. Poonen. Undecidability in number theory, *Notices of the American Mathematical Society,* 2008.

-[DPV08] S. Dasgupta, C. Papadimitriou, and U. Vazirani. Algorithms. McGraw Hill, 2008.