Limits of Computation

+

Course Recap

ORF 363/COS 323

Instructor: Amir Ali Ahmadi

Reminder: NP-hard and NP-complete problems

Definition.

▪A decision problem is said to be NP-hard if every problem in NP reduces to it via a polynomial-time reduction.
(roughly means "harder than all problems in NP.")

Definition.

▪A decision problem is said to be NP-complete if

(i)It is NP-hard

(ii)It is in NP.

(roughly means "the hardest problems in NP.")

Remarks.

▪NP-hardness is shown by a reduction from a problem that's already known to be NP-hard.

▪Membership in NP is shown by presenting an easily checkable certificate of the YES answer.

▪NP-hard problems may not be in NP (or may not be known to be in NP as is often the case.)

The complexity class NP

- ADDITION
- MULTIPLICATION
- LINEQ
- LP
- MAXFLOW
- MINCUT
- MATRIXPOS
- SHORTEST PATH
- SDP $_\epsilon$
- PRIMES
- ZEROSUMNASH
- PENONPAPER,…

- TSP
- MAXCUT
- STABLE SET
- SAT
- 3SAT
- PARTITION
- KNAPSACK
- IP
- COLORING
- VERTEXCOVER
- 3DMATCHING
- SUDOKU,…

$\in$**P**

NP-complete

Reductions

■A reduction from a decision problem A to a decision problem B is

   ■**a "general recipe" (aka an algorithm)** for taking any instance of A and explicitly producing an instance of B, such that

   ■the answer to the instance of A is YES if and only if the answer to the produced instance of B is YES.

■This enables us to answer A by answering B.

■Using reductions for showing NP-hardness:

   ■ If A is known to be hard, then B must also be hard.
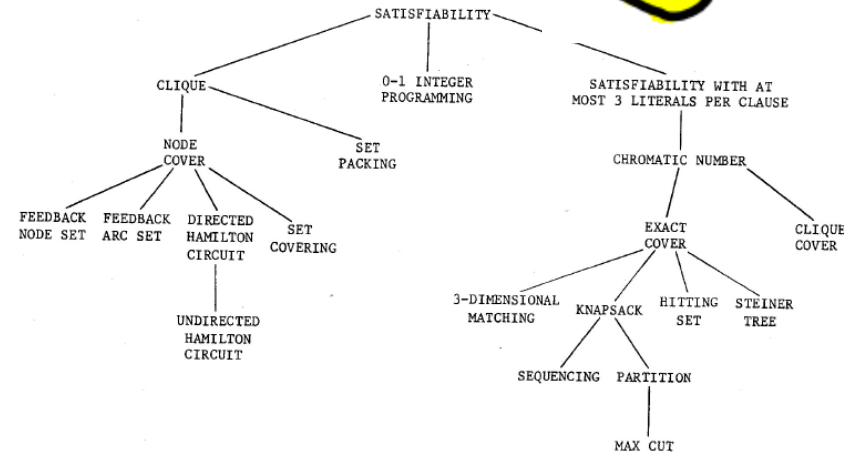

FIGURE 1 - Complete Problems

P versus NP

▪All NP-complete problems 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)

▪**Today:** limits of computation in general
(and under no assumptions)

Matrix mortality

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

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 =              A2 =

     0      0           0      1
     0      1          -1      0
```

```
>> A1*A2

ans =

          0        0
         -1        0


>> A1*A2*A1*A2

ans =

          0        0
          0        0
```

Example from [W11].

Mortal.

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 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

>> ...
```

PRINCETON UNIVERSITY ORFE

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 anything like that.)

  - How in the world would someone prove something like this?

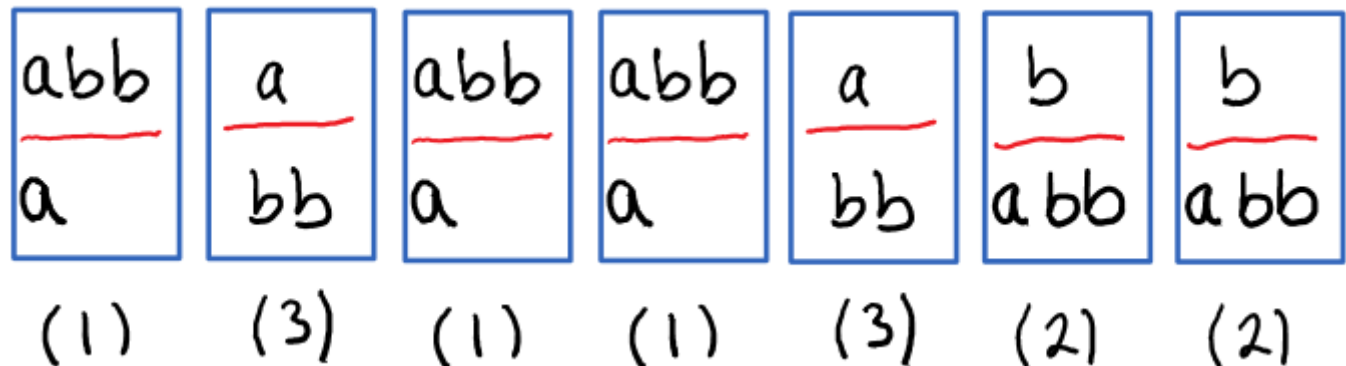 - By a reduction from another undecidable problem!
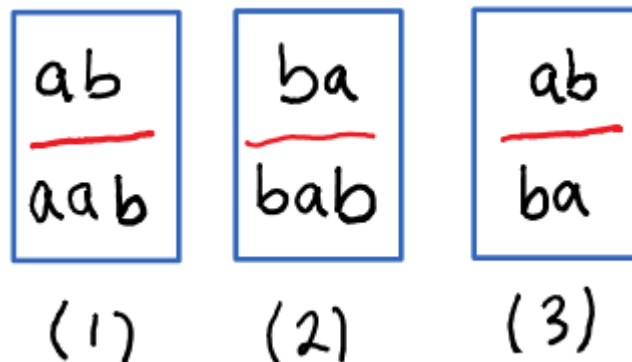
The Post Correspondence Problem (PCP)
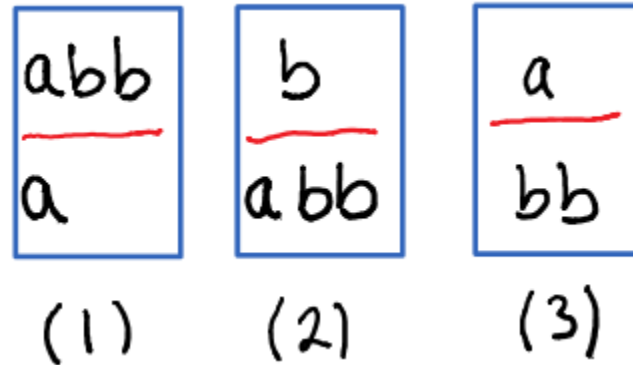
Emil Post
(1897-1954)

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:

PRINCETON UNIVERSITY   ORFE

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).
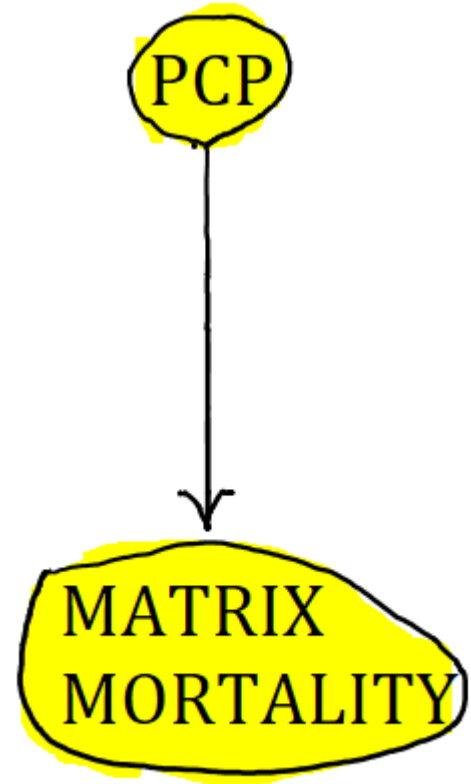
Integer roots of polynomial equations

- 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 $x^4 + y^4 = z^4?$

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

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

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

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

No one knows!

Source: [Po08]

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?

- **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.

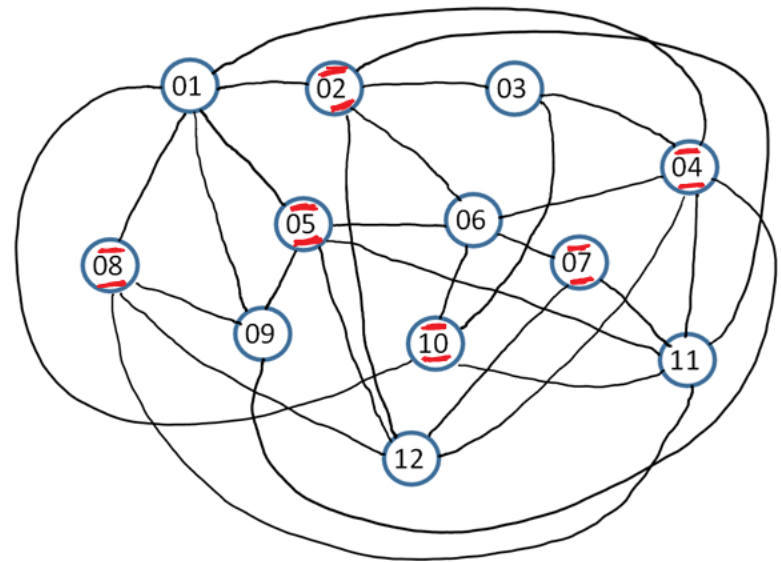- It's undecidable even in fixed degree and dimension (e.g., $d = 4, n = 58$).
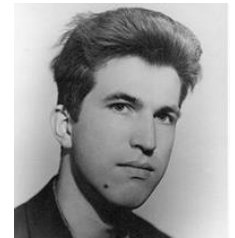
Real/rational roots of polynomial equations

- 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.

$\exists$ Stable set of size K $\iff$

$\exists x$ s.t.
$$\begin{bmatrix} x_1 + \cdots + x_n = K \\ x_i + x_j \leq 1 \quad i,j \in E \\ x_i \in \{0,1\} \end{bmatrix}$$

$\iff$

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

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

PRINCETON UNIVERSITY ORFE

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.
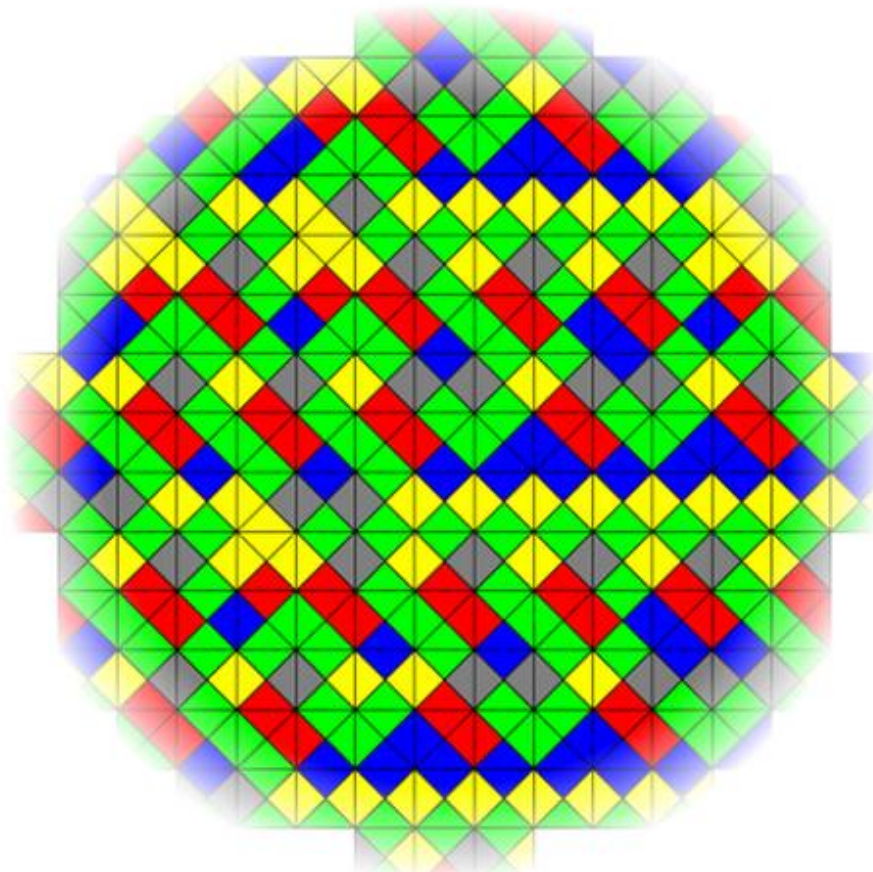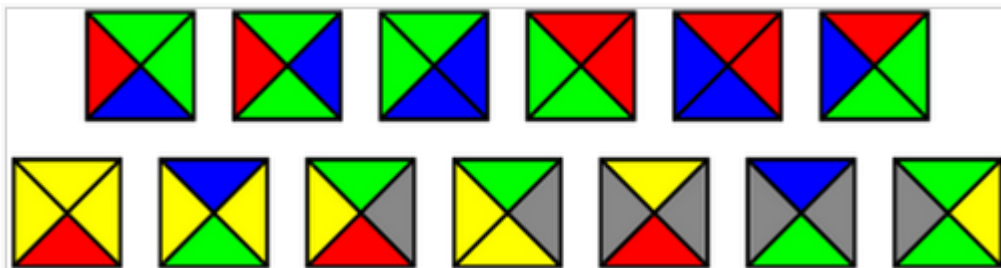
All undecidability results are proven via reductions

abb / a (1)   b / abb (2)   a / bb (3)

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

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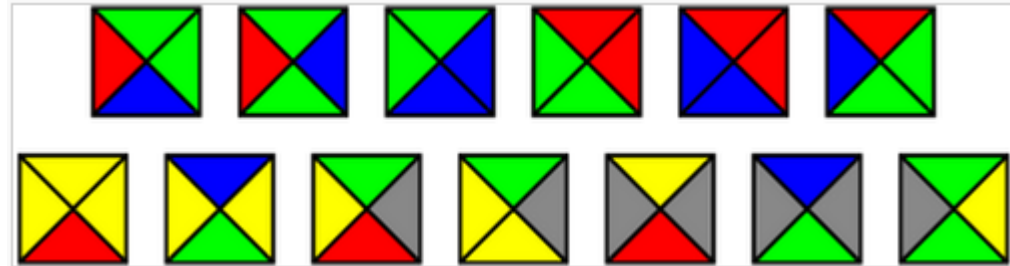
A3 =
$$\begin{matrix} 1 & 2 \\ 0 & -1 \end{matrix}$$

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

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:

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

Program P $x = [3;63];$

The halting problem

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

$\rightarrow$ 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.

PRINCETON UNIVERSITY ORFE

21

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.

22

Typical 1ˢᵗ time reaction to the proof of the halting problem
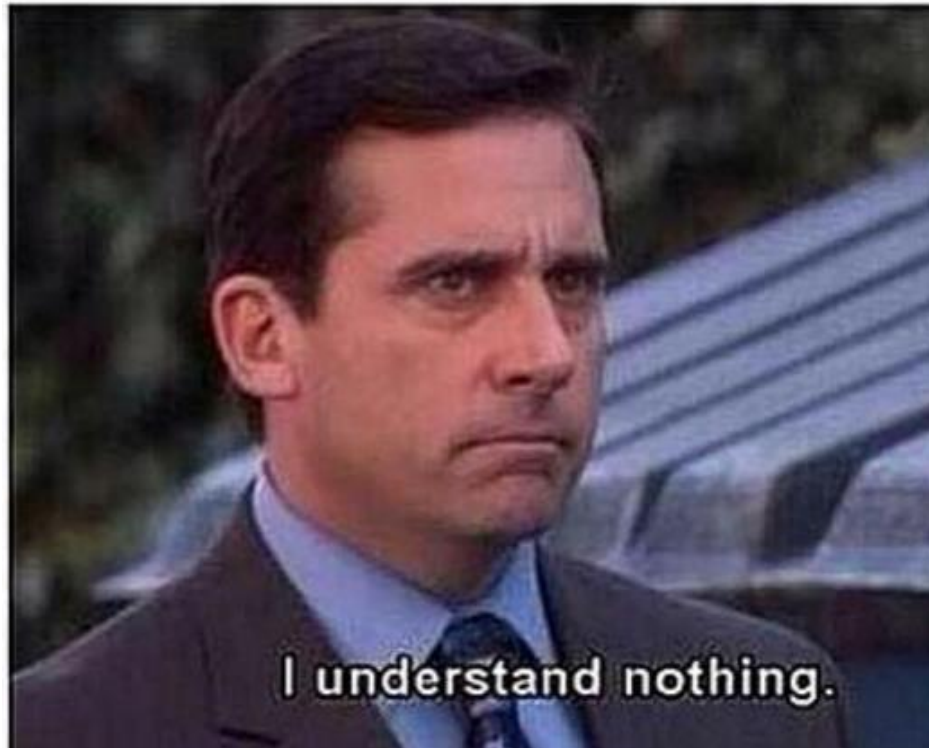
The halting problem (1936)

**Alan Turing
(1912-1954)**

A simpler story to tell strangers at a bar…

(aka 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?

PRINCETON UNIVERSITY · ORFE

A remarkable implication of this…

Take your favorite long-standing open problem in mathematics:
e.g.,

- Is there an odd perfect number? (an odd number whose proper divisors add up to itself?)

- Is every even integer >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, you would have solved 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 into an instance of POLY INT.

A look back at ORF 363/COS 323

OBJECTS IN MIRROR ARE CLOSER THAN THEY APPEAR

Topics we covered in optimization

- Optimality conditions for unconstrained optimization

- Convex analysis
    - Convex sets and functions
    - Optimality conditions for constrained convex problems
    - Convexity detection and convexity-preserving rules

- Modeling a problem as a convex program
    - Solving it in CVX

- Algorithms for convex unconstrained optimization

- Algorithms for constrained linear optimization

- Semidefinite programming

- Convex relaxations for non-convex and combinatorial optimization

- Theory of NP-completeness

- Undecidability

PRINCETON UNIVERSITY ORFE
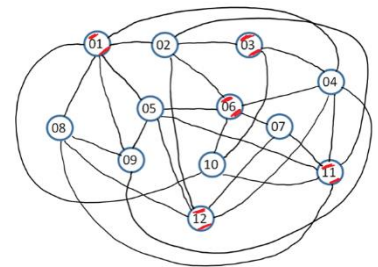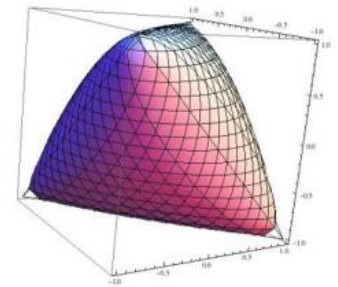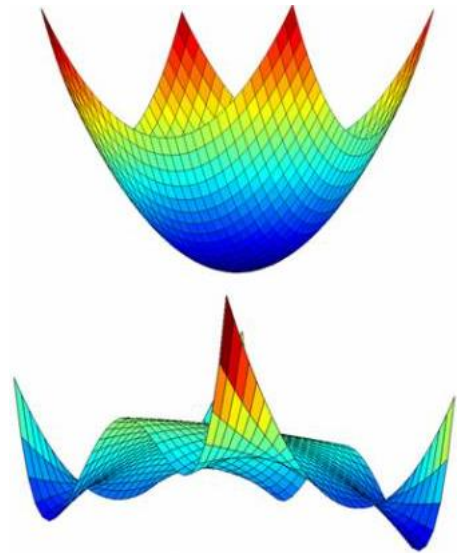
Topics we covered in numerical computing

- Least squares

  – Optimality conditions and normal equations

- Singular value decomposition

- Solving linear systems

- Conjugate gradient methods

- Root finding

  – Bisection, the secant method

  – The Newton method

- Nonlinear least squares

  – The Gauss-Newton method

- Iterative descent algorithms

  – Convergence rates of gradient descent and Newton
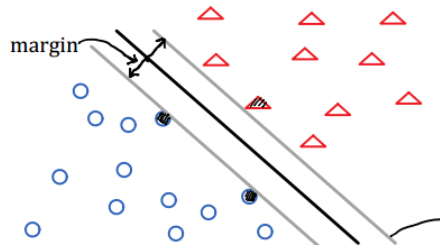
  – Condition number

- Approximation and fitting
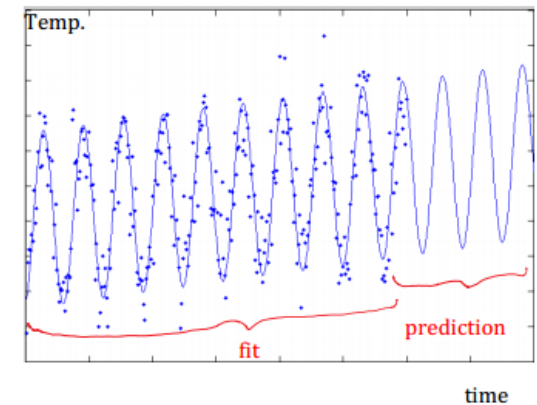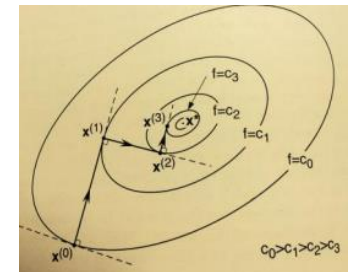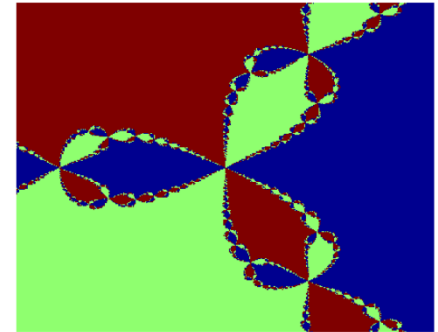
Applications of these tools are ubiquitous…
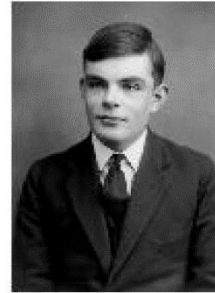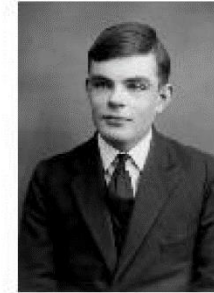
Support vector machines

Hillary vs. Bernie

Scheduling

Image compression

Optimal facility location
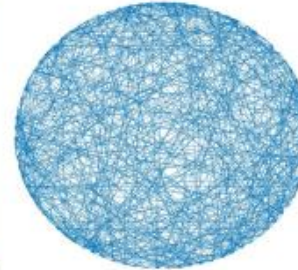
Event planning
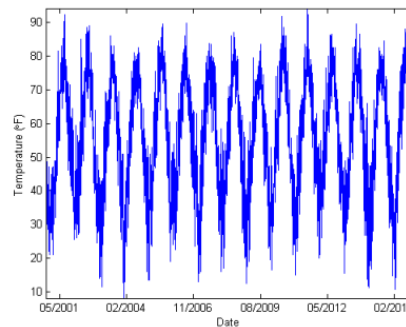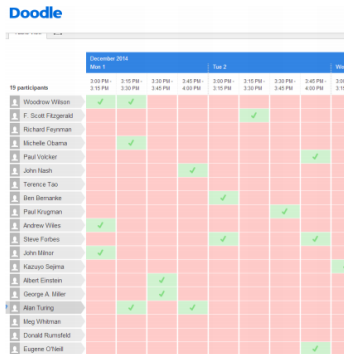
Leontief input-output economy
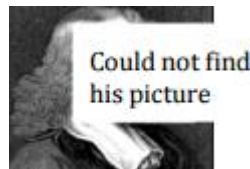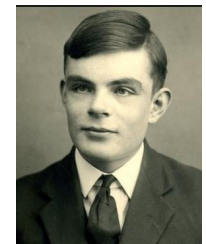
Minimum intensity radiation therapy

…

We met lots of mathematicians!

Who is who?

And on what topic did they show up in this class?

How to check if an optimization problem is easy?

- Check if it's convex!

- The functional form of convexity meant:
    - Objective a convex function (if you are minimizing)
    - Constraints: "Convex≤Concave", "Affine==0".

- If it is, then (most of the time) CVX can already solve it for you up to a reasonably large size.

- There are occasional exceptions:

- Nonconvex problems can be easy:
    - Singular value decomposition (best rank r approximation to a given matrix)
    - One can argue that there is "hidden convexity" (e.g., the dual is an SDP)

- Convex problems can be hard:
    - Optimizing over the set of nonnegative polynomials or copositive matrices
    - Not quite in functional form, but they can be made as such.

- Checking convexity may not be easy

- But the calculus of convex functions and convexity-preserving rules often suffice.

PRINCETON UNIVERSITY ORFE

# How to check if an optimization problem is easy (formally)?

- Can you reduce it to a problem in P?

- If yes, then it's often easy

  – Unless the polynomial in the running time has high degree or large constants—often rare

  – Unless your input size is massive --- not so rare these days in the era of "big data"
  (we almost finished the course without using the term "big data"….damn.)

- Can you show it's NP-hard?

- You must reduce a different NP-hard problem to it.

  – If you succeed, an exact efficient algorithm is out of the picture (unless P=NP)

- NP-hard problems still routinely solved in practice.

- Workarounds: heuristics, solving special cased exactly, convex relaxations.

- Convex optimization is often a powerful tool for approximating non-convex and NP-hard problems.

- We saw many examples in recent weeks; e.g., LP and SDP relaxations.

Slide from lecture 1: Course objectives

- The skills I hope you acquire:

- Ability to view your own field through the lens of optimization and computation

  - To help you, we'll draw applications from operations research, statistics, economics, machine learning, engineering, …

- Learn about several topics in scientific computing

- More mathematical maturity and ability for rigorous reasoning

  - There will be some proofs in lecture. Easier ones on homework.

- Enhance your coding abilities (nothing too fancy, simple MATLAB)

  - There will be a MATLAB component on every homework and on the take-home final.

- Ability to recognize hard and easy optimization problems

- Ability to use optimization software

  - Understand the algorithms behind the software for some easier subclass of problems.

An example: Jacob Eisenberg's work

- The "real strike zone" in major league baseball!
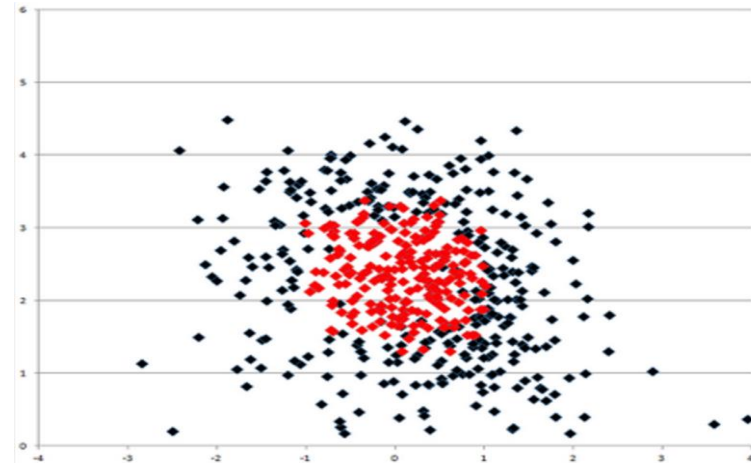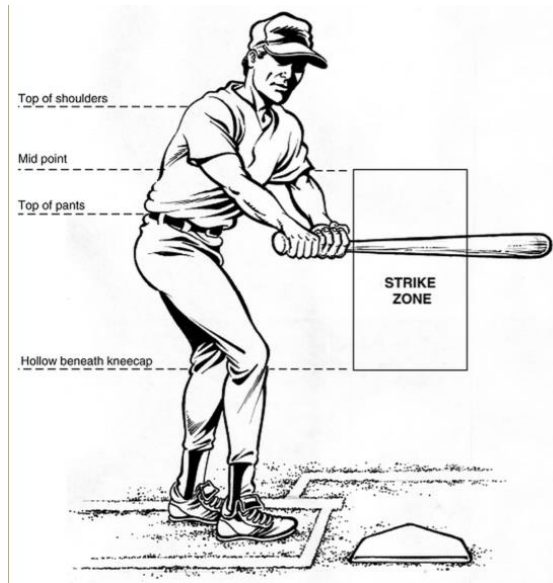
Robust minimum-volume ellipsoids obtained from *semidefinite programming*
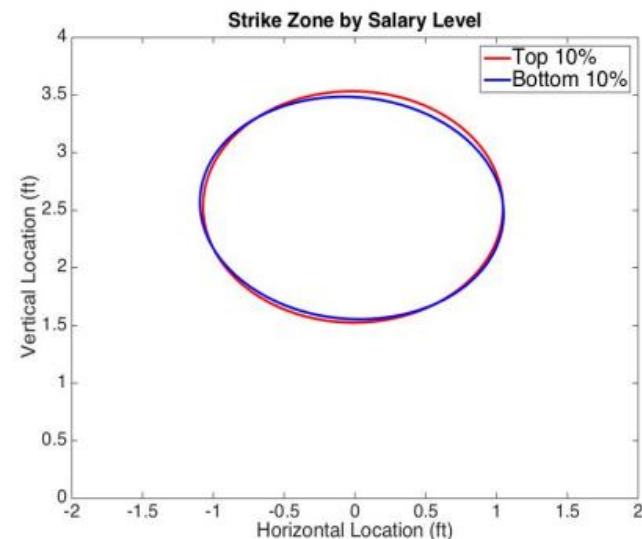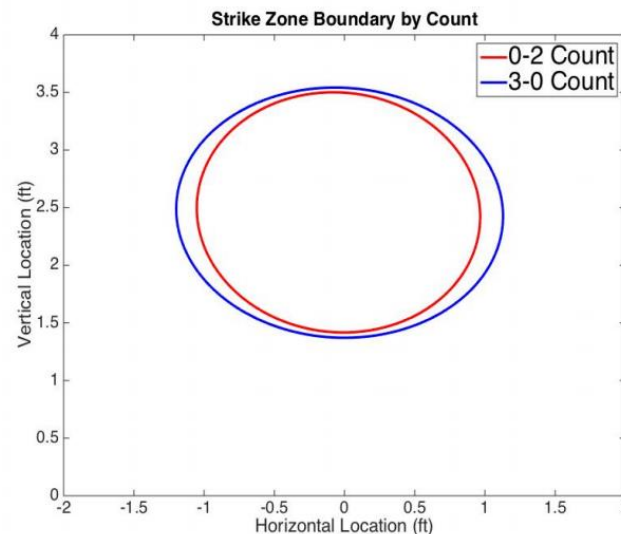
The final exam!

- Take-home exam. No collaboration allowed. Can only ask *clarification questions* as *public questions* on Piazza. Can use my notes and reference books of the course. Can only use "Google" for problems you may encounter with MATLAB (although even that should not be needed).

- To be given **out on Wednesday, January 17, 10AM**.

- Will be **due Friday, January 19, 10AM** (in my office→Sherrerd 329, or in Sherrerd 123, box for ORF 363).

  - Must keep an electronic copy of your exam as a single PDF file.

- **The WTF extension formula:** if w/t/f respectively denote the number of additional exams you have on Wed/Thu/Fri, then the number of additional hours you get to complete the exam equals:

$$\sum_{\{i=w,t,f\}} 24 \min\{i, 1\} + 5 \max\{i - 1, 0\}$$

  Send w/t/f to Jing with course numbers, and CC all TAs: jingy@p..., gh4@p..., bkhadir@p..., cdibek@p..., jeffz@p..., auysal@p...

- Don't forget pset 8 is due Friday, January 12, 1 PM in Sherrerd 123, box for ORF 363.

What to study for the final?

- All the lecture notes I post. I plan to post a single PDF file with everything in ~2 weeks.

- Psets 1-8.

- If you need extra reading, the last page of the notes points you to certain sections of the book for additional reading.

  - [DPV] Chapters 7 and 8 are particularly good. (You can skip 7.2 and 7.5.)

- Be comfortable with MATLAB and CVX. If off campus, make sure your MATLAB runs.

Some good news

- Undecidability from today's lecture won't be on the final.

- Our guest lecture won't be on the final.

- Theory of NP-completeness won't be on the final (but it is on HW 8).

- The TAs and I will hold office hours during reading period, and up to the day of the day of the exam. Regular schedule (see syllabus, or slides of lecture 1).

- In addition, each TA will have a 2-hour review session going over psets, etc.

  - Jing (pset 1&2) - Monday Jan 8, 1-3 PM, Julis Romo Rabinowitz 399

  - Bachir (pset 3&4a) - Tuesday Jan 9, 1-3 PM, Julis Romo Rabinowitz 399

  - Cemil (pset 4b&5) - Wednesday Jan 10, 1-3 PM , Julis Romo Rabinowitz 399

  - Sinem (pset 6&7) - Thursday Jan 11, 1-3 PM , Julis Romo Rabinowitz 399

  - Jeff (pset 8) - Friday Jan 12, 1-3 PM , Julis Romo Rabinowitz 399

  - Georgina (practice midterms and finals) – Monday Jan 15, 1-3 PM, Julis Romo Rabinowitz 399

- There will be a 3 hour comprehensive review session by me.

  - Tuesday, January 16, 6-9 PM, in Friend 006.

- There will be pizza!
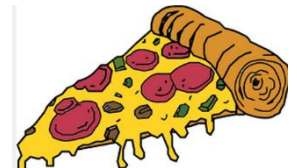
Last but not least…

- It's been a great pleasure having you all in my class this fall!

- Go make optimal decisions in your lives!

- And keep in touch!

<div align="right">

AAA.
December 14, 2017

</div>

Notes & References

■Notes:

-    Chapter 8 of [DPV08] mentions undecidability and the halting problem. Chapter 9 of [DPV08] is optional but a fun read.

■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.