What Computers Can Compute (Approximately)

David P. Williamson
TU Chemnitz
9 June 2011

Outline

- The 1930s-40s: What can computers compute?

- The 1960s-70s: What can computers compute efficiently?

- The 1990s-: What can computers compute efficiently approximately?

  - Two examples of approximation algorithms

  - When is it hard to compute efficiently approximately?

- Some concluding thoughts

Some terminology (imprecise)

- "Problem"

    - Traditional mathematics usage: e.g. Fermat's Last Problem

    - Computational usage: Find an algorithm (computer program) such that given *any* valid input, the desired output is produced.

    - A *decision problem*: The output is one of "Yes" or "No".

Input $\longrightarrow$ Algorithm $\longrightarrow$ Output

The 1930s-40s: Can computers solve any problem?

- No.

- Alan Turing (1936): Cannot solve the Halting Problem (a decision problem):

  - Given a computer program P, and an input x for that program, output "Yes" if the program ever terminates on the given input, "No" otherwise.
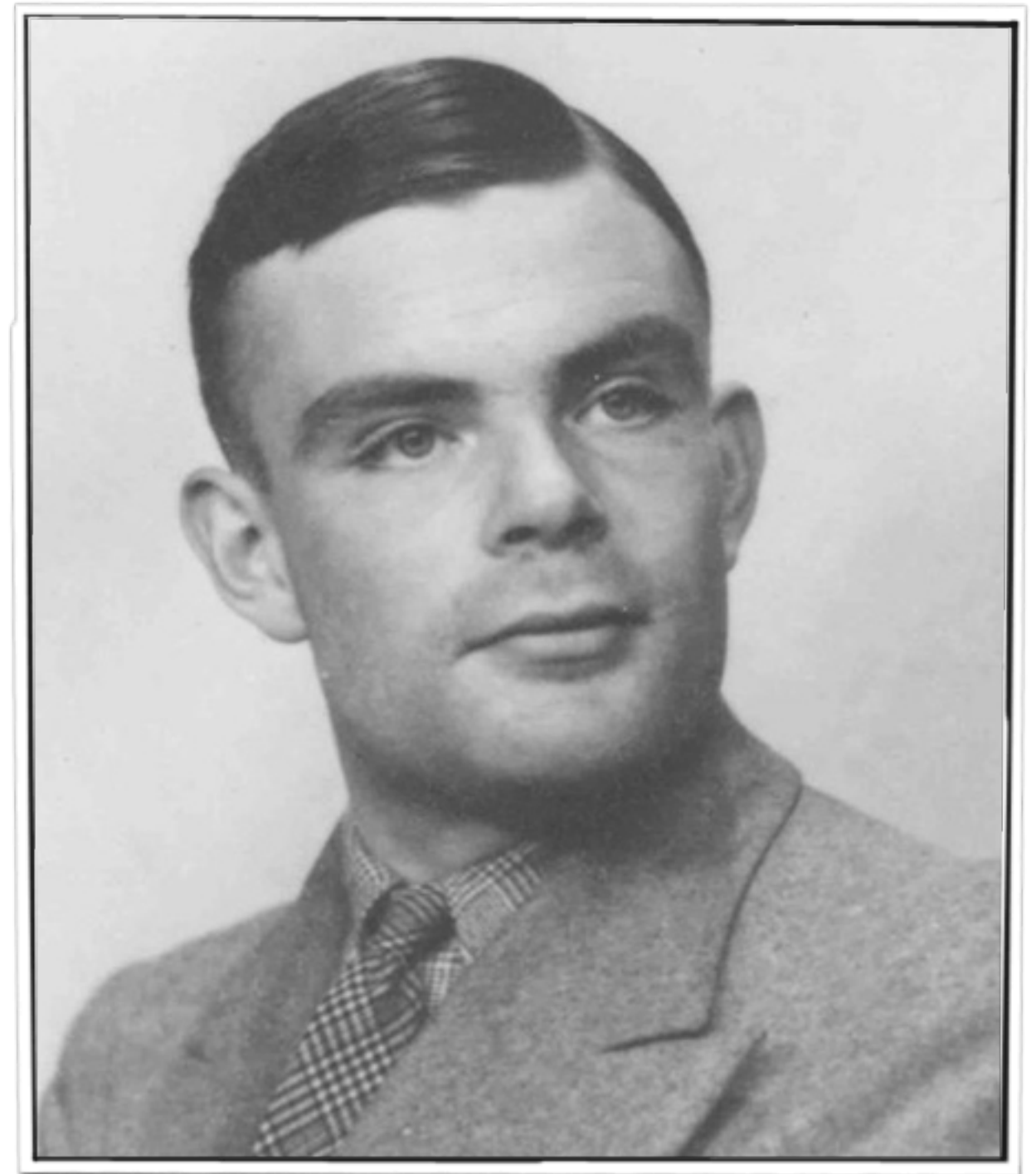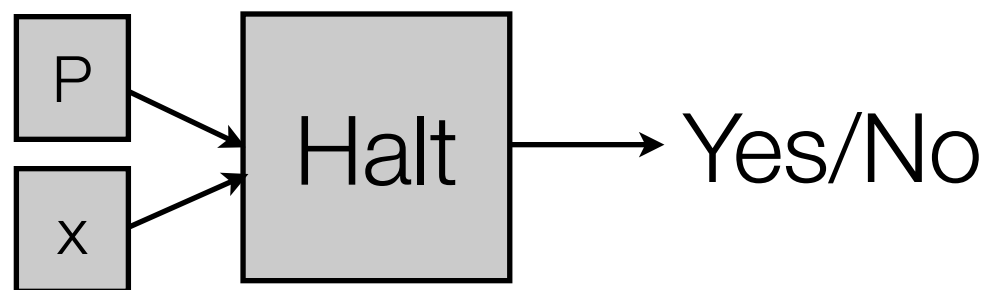
The proof

By contradiction. Suppose such a program, Halt(P,x), exists, that determines if program P halts on input x. Then consider the following program MyHalt(P)

MyHalt(P)
If Halt(P,P)
    then loop forever
Else
    Stop

Let P_1, P_2, ... be a list of all programs that take a single input, and consider whether program P_i halts when given P_j as input (H = halts, N = doesn't halt).

Which program is MyHalt?

Inputs

Programs

	P_1	P_2	P_3	P_4	P_5
P_1	H	N	H	H	N
P_2	N	H	N	N	H
P_3	N	N	N	N	N
P_4	H	H	N	N	H
P_5	H	H	H	H	H

MyHalt

N	N	H	H	N

Discrete Optimization Problems

- Appears in many places: scheduling jobs on computers, locating facilities, building networks, stocking inventory,...

- Famous example: *the traveling salesman problem (TSP).*

    - Given n cities and the distances between each pair of cities, find the shortest *tour* that visits each city once and returns to the starting point.

- Decision version of TSP: Additional input of a number C, "Is the length of the shortest tour at most C?"
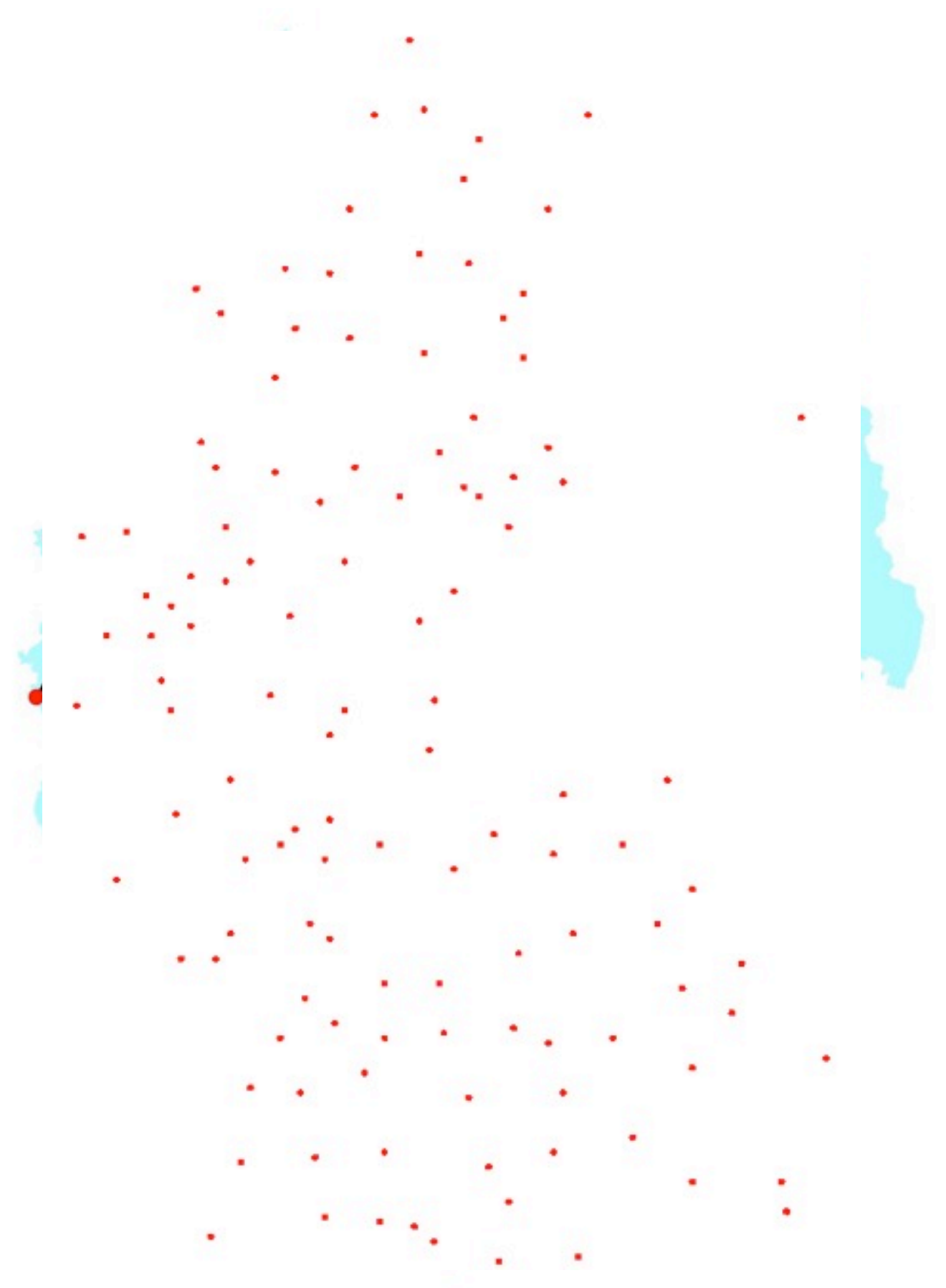
The Obvious Finite Algorithm

- Consider all $n!$ possible orderings of the cities and compute the length of the tour for that ordering. Keep track of the shortest one found.

- Problem: $n!$ grows pretty quickly with n. 120! is about 6×10^{198}. 1 tour/ns still is about 10^{182} years.

"Good" algorithms

Edmonds (1965):

One can find many classes of problems, besides maximum matching and its generalizations, which have algorithms of exponential order but seemingly none better. An example known to organic chemists is that of deciding whether two given graphs are isomorphic. For practical purposes the difference between algebraic and exponential order is often more crucial than the difference between finite and non-finite.

It would be unfortunate for any rigid criterion to ...

Edmonds (1967):

We say an algorithm is *good* if there is a polynomial function $f(n)$ which, for every positive-integer valued n, is an upper bound on the "amount of work" the algorithm does for any input of "size" n. The concept ... formalize ... relative say to a T... ...chine

traveling saleman problem [cf. 4]. I conjecture that there is no good algorithm for the traveling saleman problem. My reasons are the same as for any mathematical conjecture: (1) It is a legitimate mathematical possibility, and (2) I do not know.

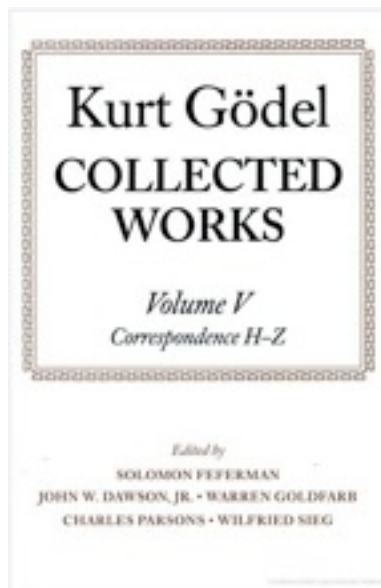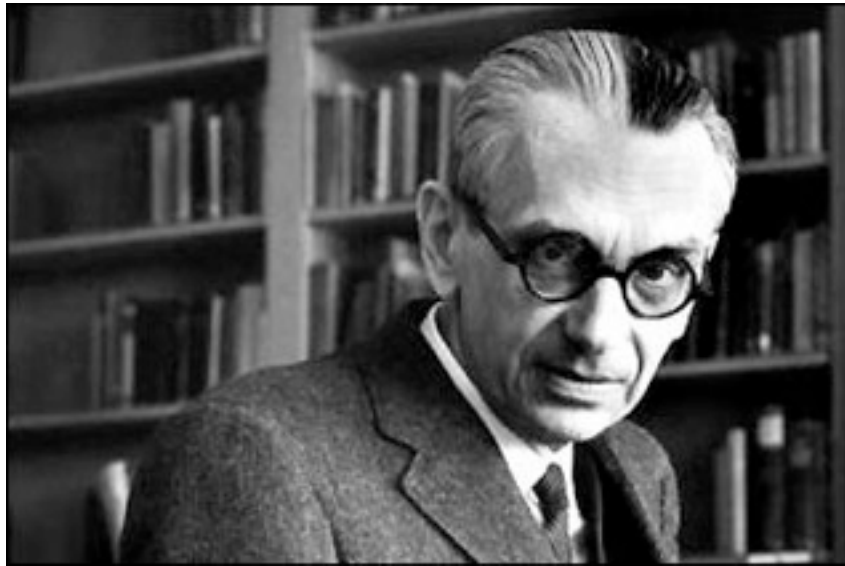A good algorithm is known for finding ... any graph
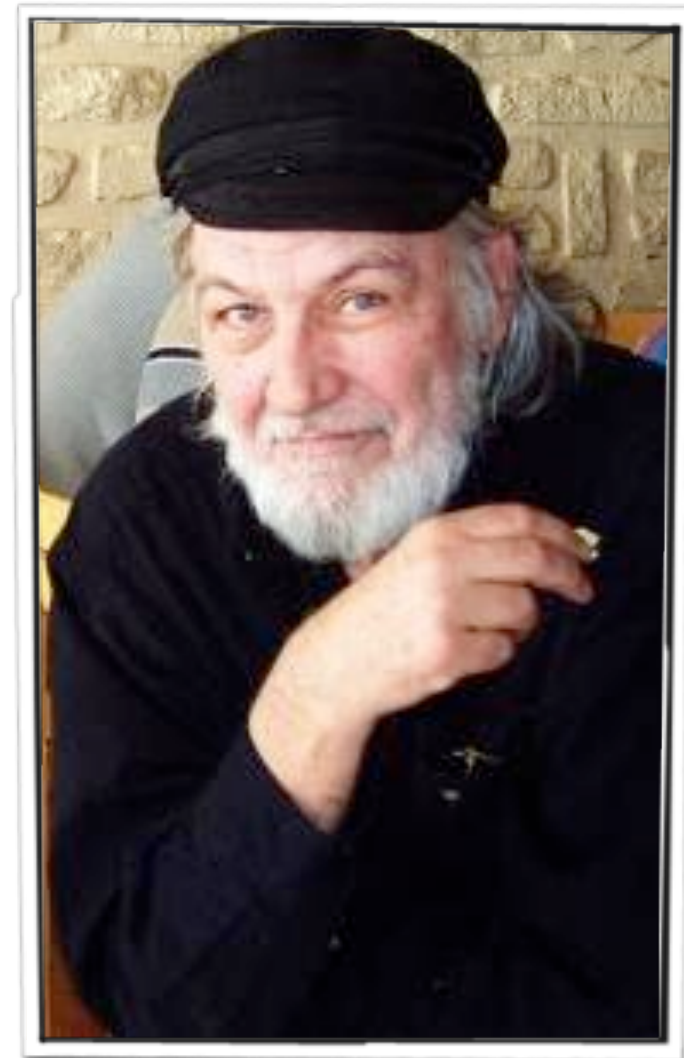
P vs. NP

- Today: *polynomial-time algorithms* are considered the theoretical measure of a good, efficient algorithm.

- *P* is the class of all decision problems solvable by a polynomial-time algorithm.

- *NP* is (roughly) the set of all decision problems for which we can "check" in polynomial time whether the answer is "Yes" (or "No") if someone gives us a "proof".

- (Cook, Levin 1971, Karp 1972) Given a polynomial-time algorithm for the decision version of the TSP, we can get a polynomial-time algorithm for any problem in *NP*.

$$P = NP?$$

A 1956 letter from Kurt Gödel to John von Neumann

Kurt Gödel
COLLECTED
WORKS

Volume V
Correspondence H–Z

Edited by
SOLOMON FEFERMAN
JOHN W. DAWSON, JR. · WARREN GOLDFARB
CHARLES PARSONS · WILFRIED SIEG

Ihr Zustand sich bald noch weiter bessert u. dass die neuesten Errungenschaften der Medizin, wenn möglich, zu einer vollständigen Heilung

liberty to write to you about a mathematical problem; your view on it would be of great interest to me: Obviously, it is easy to construct a Turing machine that allows us to decide, for each formula F of the restricted functional calculus[a] and every natural number n, whether F has a proof of length n [length=number of symbols]. Let $\psi(F, n)$ be the number of steps required for the machine to do that, and let $\varphi(n) = \max_F \ \psi(F, n)$.

The question is, how rapidly does $\varphi(n)$ grow for an optimal machine? It is possible to show that $\varphi(n) \geq Kn$. If there really were a machine with $\varphi(n) \sim Kn$ (or even just $\sim Kn^2$) then that would have consequences of the greatest significance. Namely, this would clearly mean that the thinking of a mathematician in the case of yes-or-no questions could be completely[1] replaced by machines, in spite of the unsolvability of the Entscheidungsproblem. n would merely have to be chosen so large that, when the machine does not provide a result, it also does not make any sense to think about the problem. Now it seems to me to be quite within the realm of possibility that $\varphi(n)$ grows that slowly. For 1.) $\varphi(n) \geq Kn$ seems to be the only estimate obtainable by generalizing the proof of the unsolvability of the Entscheidungsproblem; 2.) $\varphi(n) \sim Kn$ (or $\sim Kn^2$) just means that the number of steps when compared to pure trial and error can be reduced from N to $\log N$ (or $\log N^2$). Such significant reductions are definitely involved in the case of other finitist problems, e.g., when computing the quadratic remainder symbol by repeated application of the law of reciprocity. It would be interesting to know what the case would be, e.g., in determining whether a number is prime, and how significantly *in general* for finitist combinatorial problems the number of steps can be reduced when compared to pure trial and error. I do not know whether you have heard that "Post's problem" (whether there are degrees of unsolvability among the problems $(\exists y)\varphi(y, x)$ with recursive φ) was solved positively by a quite young man by the name of Richard Friedberg.[b] The solution is very elegant. Unfortunately, Friedberg is not going to study mathematics, but rather medicine (seemingly under the influence of his father).

unter dem Einfluss seines Vaters).

P vs. NP one of the seven Clay Millenium Problems

It's one of the fundamental mathematical problems of our time, and its importance grows with the rise of powerful computers.

BY LANCE FORTNOW

The Status of the P versus NP Problem

The software written for this illustration makes a stylized version of a network graph that draws connections between elements based on proximity. The graph constantly changes as the elements sort themselves.

WHEN EDITOR-IN-CHIEF MOSHE Vardi asked me to write this piece for *Communications*, my first reaction was the article could be written in two words:

Still open.

When I started graduate school in the mid-1980s, many believed that the quickly developing area of circuit complexity would soon settle the **P** versus **NP** problem, whether every algorithmic problem with efficiently verifiable solutions have efficiently computable solutions. But circuit complexity and other approaches to the problem have stalled and we have little reason to believe we will see a proof separating **P** from **NP** in the near future.

increased, the cost of computing has dramatically decreased, not to mention the power of the Internet. Computation has become a standard tool in just about every academic field. Whole subfields of biology, chemistry, physics, economics and others are devoted to large-scale computational modeling, simulations, and problem solving.

As we solve larger and more complex problems with greater computational power and cleverer algorithms, the problems we cannot tackle begin to stand out. The theory of **NP**-completeness helps us understand these limitations and the **P** versus **NP** prob-

*Communications of the ACM,* Sept. 2009

So what now?

- One possible response for discrete optimization problems: polynomial-time algorithms to find *near* optimal solutions.

- An α-*approximation algorithm* is a polynomial-time approximation algorithm that produces solutions within a factor of α of the optimal.

  - E.g. A 3/2-approximation algorithm for the TSP would find a tour of length at most 3/2 times the length of the shortest tour.

  - For maximization problems, assume $\alpha < 1$; a 1/2-approximation algorithm finds a solution whose value is at least 1/2 that of an optimal solution.

  - Can consider *randomized* algorithms, in which case we want the expected value of the solution to be within α of optimal.

- Significant amount of work on a wide range of problems; will choose just one as an example.

The maximum cut problem

- Given an undirected graph *G= (V,E)*, find a set *S* of vertices that maximizes the number of edges with exactly one endpoint in *S* (edges *in the cut*).

- As with the traveling salesman problem, a polynomial-time algorithm for this problem would imply P=NP.

S
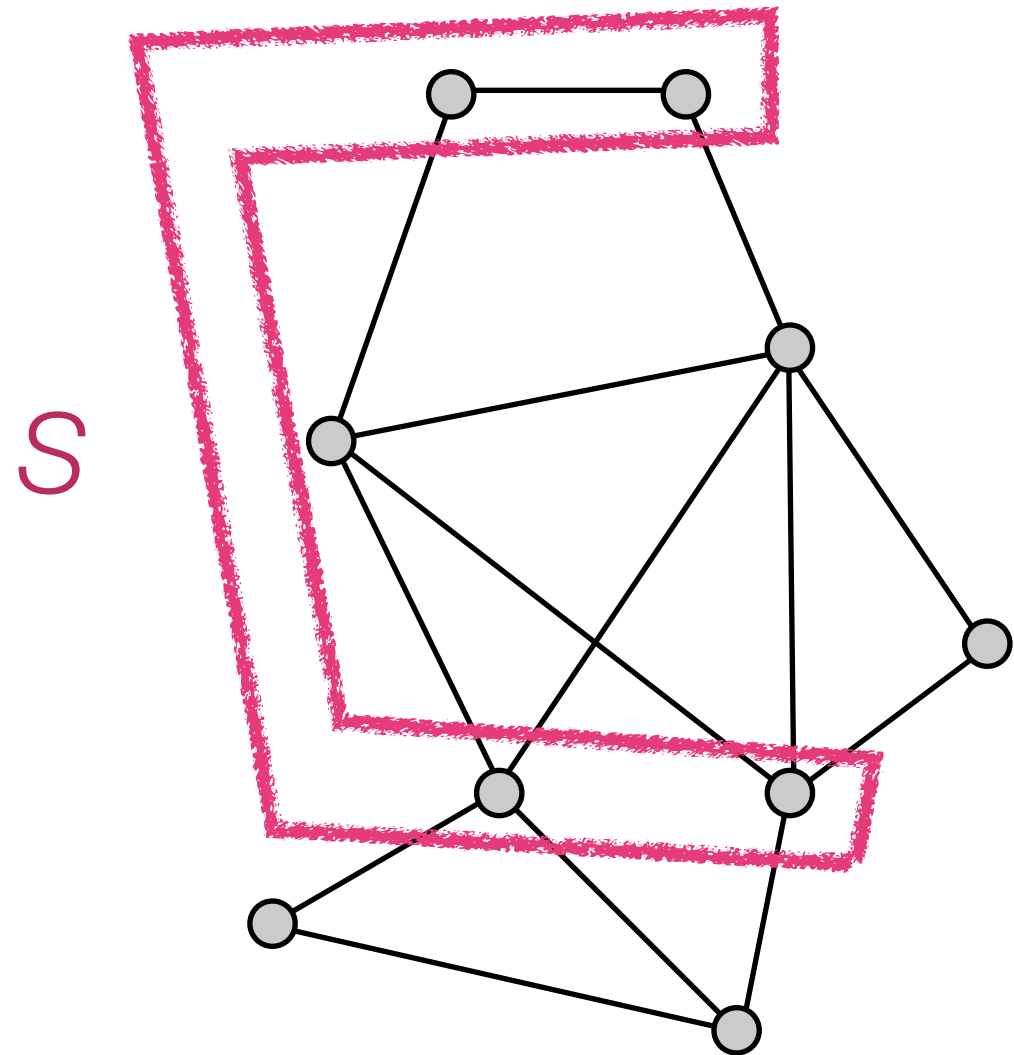
A randomized approximation algorithm (Erdős 1967)

- Choose set of vertices S uniformly at random.

- Then probability that any given edge is in the cut is 1/2.
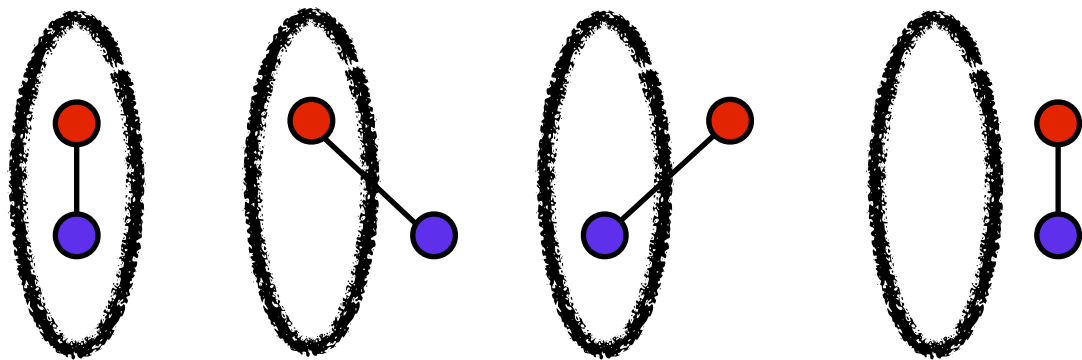
- Thus expected number of edges in the cut is 1/2 $|E|$, which is at least half the optimal value.

An alternate approach (Goemans, W 1995)

- Suppose we introduce an *n*-dimensional unit vector v_i (where $n=|V|$) for each vertex $i \in V$ and we ask for

$$\max \frac{1}{2} \sum_{(i,j) \in E} (1 - v_i \cdot v_j)$$

  with either $v_i=(-1,0,0,...)$ or $v_i=(1,0,0,...)$ for each $i \in V$.

- Then if we set $S=\{ i \in V: v_i=(-1,0,0,...) \}$, the number of edges in the cut is

$$|\{(i,j) \in E : v_i \neq v_j\}| = \frac{1}{2} \sum_{(i,j) \in E : v_i \neq v_j} (1 - v_i \cdot v_j)$$

$$= \frac{1}{2} \sum_{(i,j) \in E} (1 - v_i \cdot v_j)$$

A relaxation
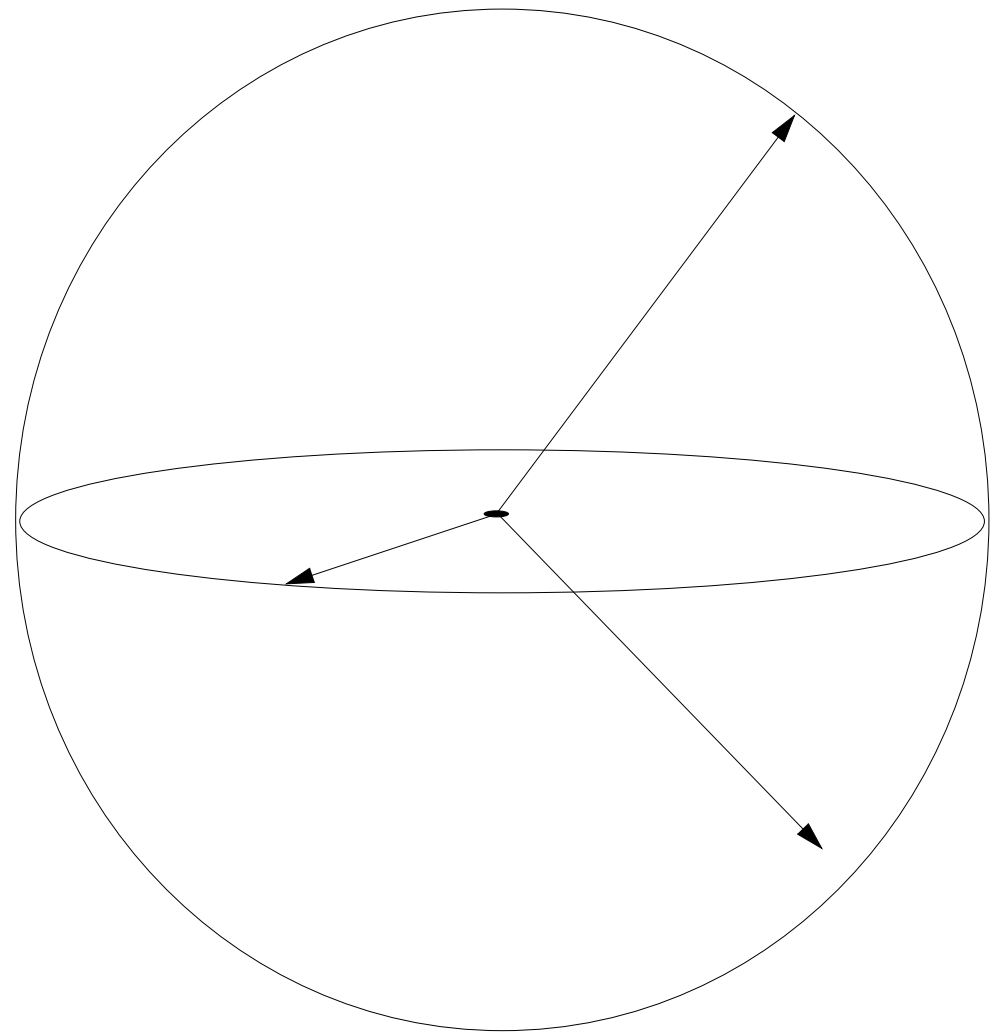
- We *can* solve the following in polynomial time

$$\max \frac{1}{2} \sum_{(i,j)\in E} (1 - v_i \cdot v_j)$$

 if the vectors are *arbitrary n-*dimensional unit-length vectors (via *semidefinite programming*).

- Note that if *OPT* is the number of edges in the cut in an optimal solution, and *Z* is the quantity above, Z ≥ *OPT.*
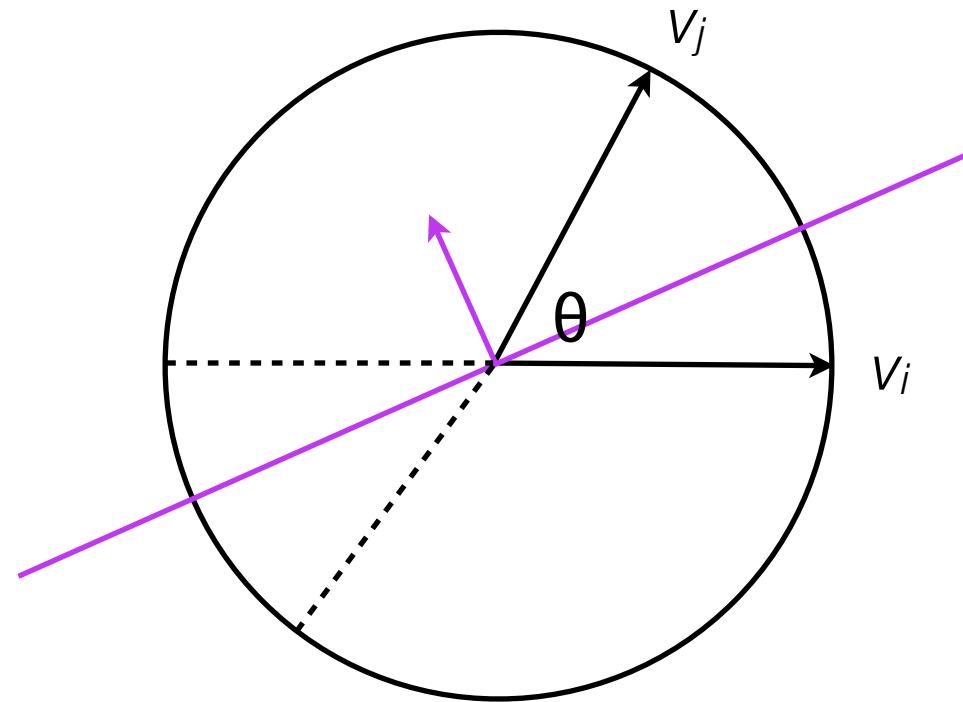
Getting a solution

- We draw a random *n*-dimensional vector *r* from the multivariate normal distribution (i.e. each component r_i from *N(0,1)*). Let this be the normal to a hyperplane through the origin of the unit sphere.

- Let *S*={ $i \in V: v_i \cdot r \geq 0$ }.

- What is expected number of edges in this cut?
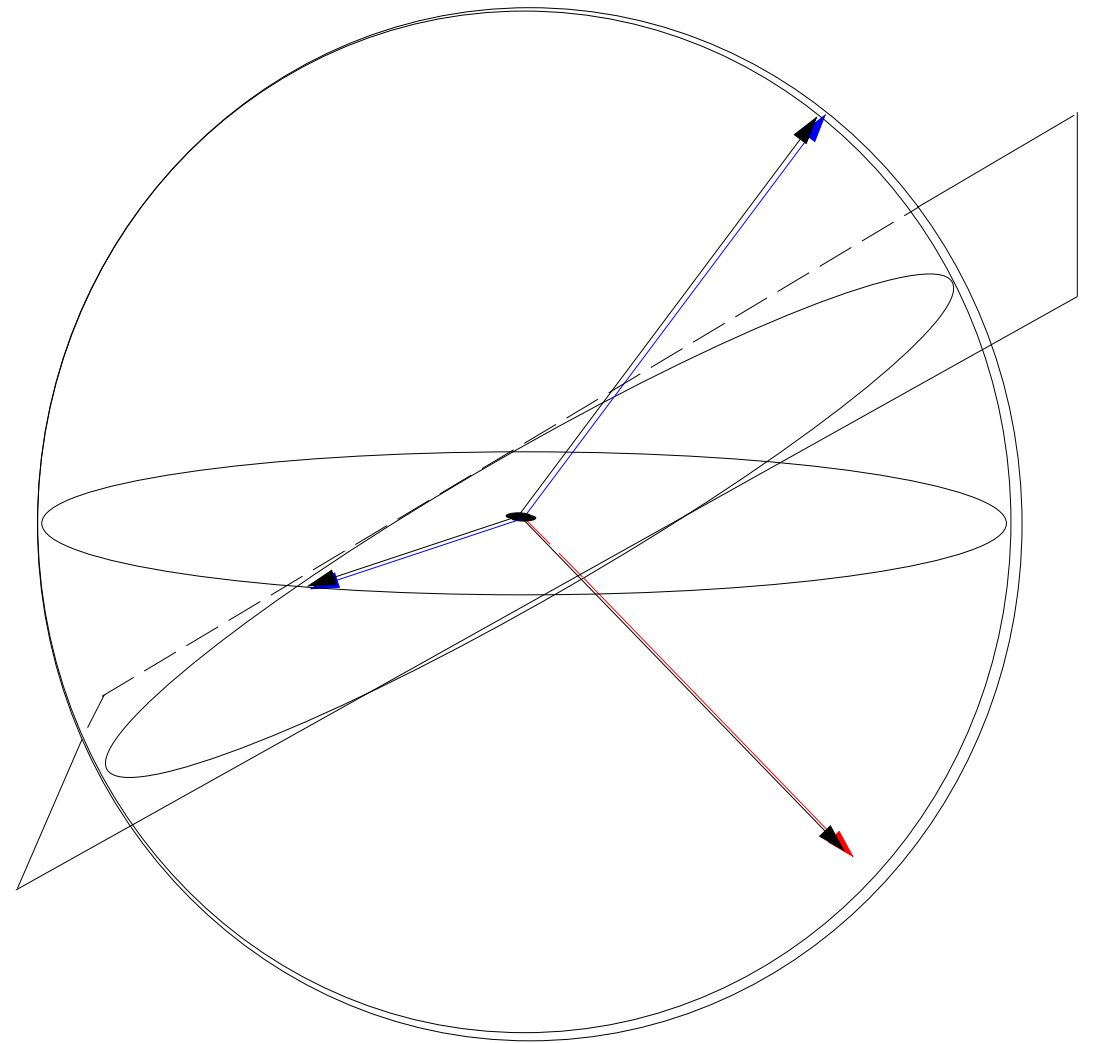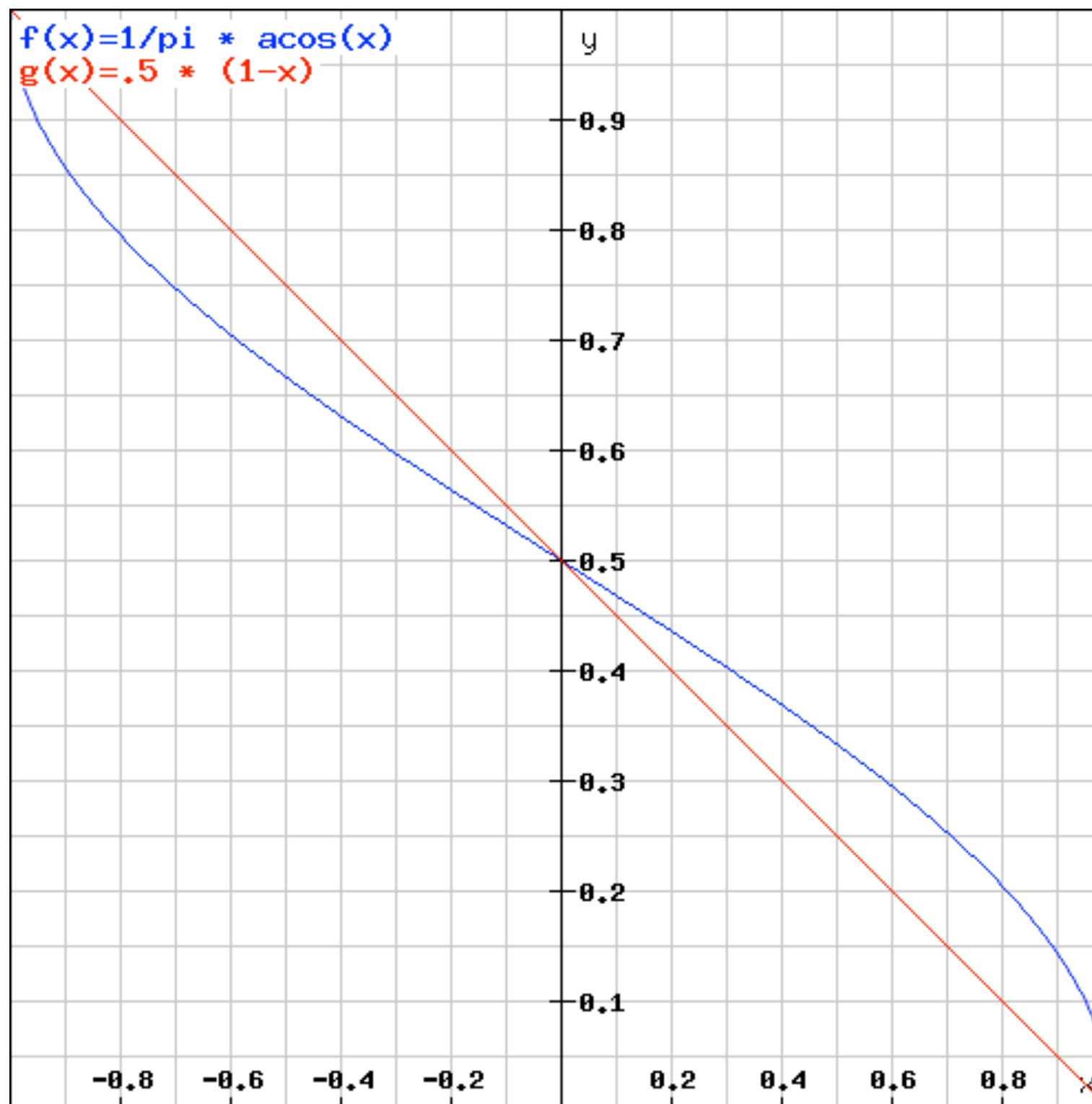
Probability that edge (i,j) is in the cut

- Consider the plane containing vectors v_i and v_j, and the projection of random vector r to this plane.

- Of the 2π possible orientations of the projected random vector, 2θ of them correspond to v_i and v_j on opposite sides of the hyperplane (and hence edge (i,j) in the cut). So the probability is

$$\frac{2\theta}{2\pi} = \frac{\theta}{\pi} = \frac{1}{\pi}\arccos(v_i \cdot v_j)$$

since $v_i \cdot v_j = \|v_i\|\|v_j\|\cos\theta = \cos\theta$.

$$\alpha = \min_{-1 \le x \le 1} \frac{\frac{1}{\pi}\arccos(x)}{\frac{1}{2}(1-x)} \ge .87856$$

The analysis

Then the expected number of edges in the cut is

$$\sum_{(i,j)\in E} \frac{1}{\pi} \arccos(v_i \cdot v_j) \geq .87856 \cdot \frac{1}{2} \sum_{(i,j)\in E} (1 - v_i \cdot v_j)$$

$$= .87856 \cdot Z$$

$$\geq .87856 \cdot OPT.$$

This gives us a .87856-approximation algorithm for the maximum cut problem.

What can computers compute approximately efficiently?

- A little hard to say, when we don't even know what is computable in polynomial time.

- However, there is a significant line of work showing that for a particular problem, if there is an α-approximation algorithm for a particular α, then P=NP.

- Huge breakthrough in the early 1990s showing this for a wide range of problems; many improvements since then.

Example

- Håstad (1996) considers the problem of maximizing the number of satisified equations of three variables over GF[2]; e.g.

$$x_1 + x_3 + x_9 \equiv 0 \pmod{2}$$
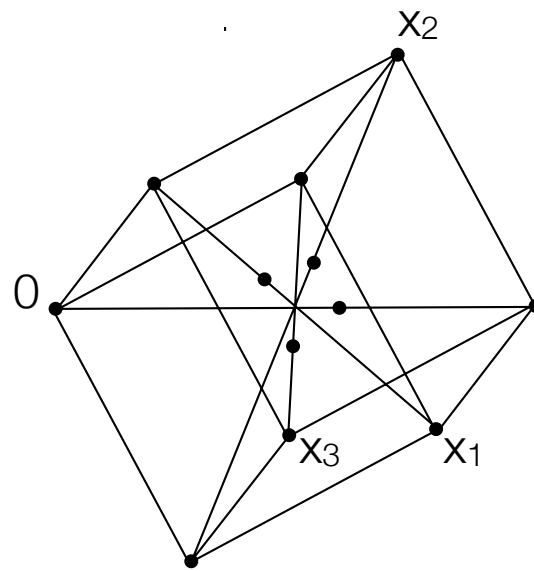$$x_2 + x_3 + x_{15} \equiv 1 \pmod{2}$$
$$x_1 + x_7 + x_{12} \equiv 0 \pmod{2}$$

$$\vdots$$

- Håstad shows that if there is any (1/2 + ε)-approximation algorithm for constant ε > 0, then P = NP.

- But there is a very simple 1/2-approximation algorithm!

What about the maximum cut problem?

- Bellare, Goldreich, Sudan 1998 and Trevisan, Sorkin, Sudan, W 2000 show how to translate the previous result into one for the maximum cut problem.

- Show that there is no (16/17+ε)-approximation algorithm for ε > 0 unless P = NP. (16/17 ≈ 0.941).

The 2000s: The Unique Games Problem and the Unique Games Conjecture

- The *unique games problem*: For a parameter k, find values of $x_i \in \{0, ..., k\text{-}1\}$ to maximize the number of satisfied difference equations mod k. E.g.

$$x_5 - x_3 \equiv 3 \pmod{21}$$

$$x_3 - x_2 \equiv 2 \pmod{21}$$

$$x_{19} - x_5 \equiv 15 \pmod{21}$$

$$\vdots$$

The Unique Games Conjecture

- The conjecture was formulated by Subhash Khot in 2002.

- Conjecture: If P≠NP, then for all δ > 0, there exists a *k* such that in polynomial time it is not possible to distinguish between sets of difference equations mod *k* in which at least a 1-δ fraction of the equations are satisfiable, and those for which at most a δ fraction are satisfiable.

Some consequences

- If the conjecture is true, then there is no (2-ε)-approximation algorithm for the vertex cover problem for any constant ε > 0 unless P = NP (Khot, Regev 2008).

- If the conjecture is true, then there is no (α+ε)-approximation algorithm for the maximum cut problem for any constant ε > 0 unless P = NP (Khot, Kindler, Mossel, O'Donnell 2007; Mossel, O'Donnell, Oleszkiewicz 2008), where

$$\alpha = \min_{-1 \leq x \leq 1} \frac{\frac{1}{\pi} \arccos(x)}{\frac{1}{2}(1 - x)} \geq .87856$$

- If the conjecture is true, then for every maximum constraint satisfaction problem there is a (ρ-ε)-approximation algorithm, and there can be no (ρ+ε)-approximation algorithm unless P=NP (Raghavendra 2008; Raghavendra, Steurer 2009).

Some open questions

- Resolve the Unique Games Conjecture.

- How well can the Traveling Salesman Problem be approximated for cities in a general metric space?

  - Long known: A 1.5-approximation algorithm (Christofides 1976)

  - Last decade: In Euclidean plane, given any $\varepsilon > 0$, there is a $(1 + \varepsilon)$-approximation algorithm (Arora 1998, Mitchell 1999).

  - Very recent: A 1.461-approximation algorithm for a special case of metric spaces (Mömke, Svensson April 2011).

 - No $(\alpha\text{-}\varepsilon)$-approximation algorithm for $\alpha=221/220$ and $\varepsilon > 0$ unless P = NP.

A critique?

- Perhaps too much of a gap between polynomial time as a theoretical measure of efficiency and computational realities?

- Edmonds (1965)

> It would be unfortunate for any rigid criterion to inhibit the practical development of algorithms which are either not known or known not to conform nicely to the criterion. Many of the best algorithmic ideas known today would suffer by such theoretical pedantry. In fact, an outstanding open question is, essentially: "how good" is a particular algorithm for linear programming, the simplex method? And, on the other hand, many important algorithmic ideas in electrical switching theory are obviously not "good" in our sense.
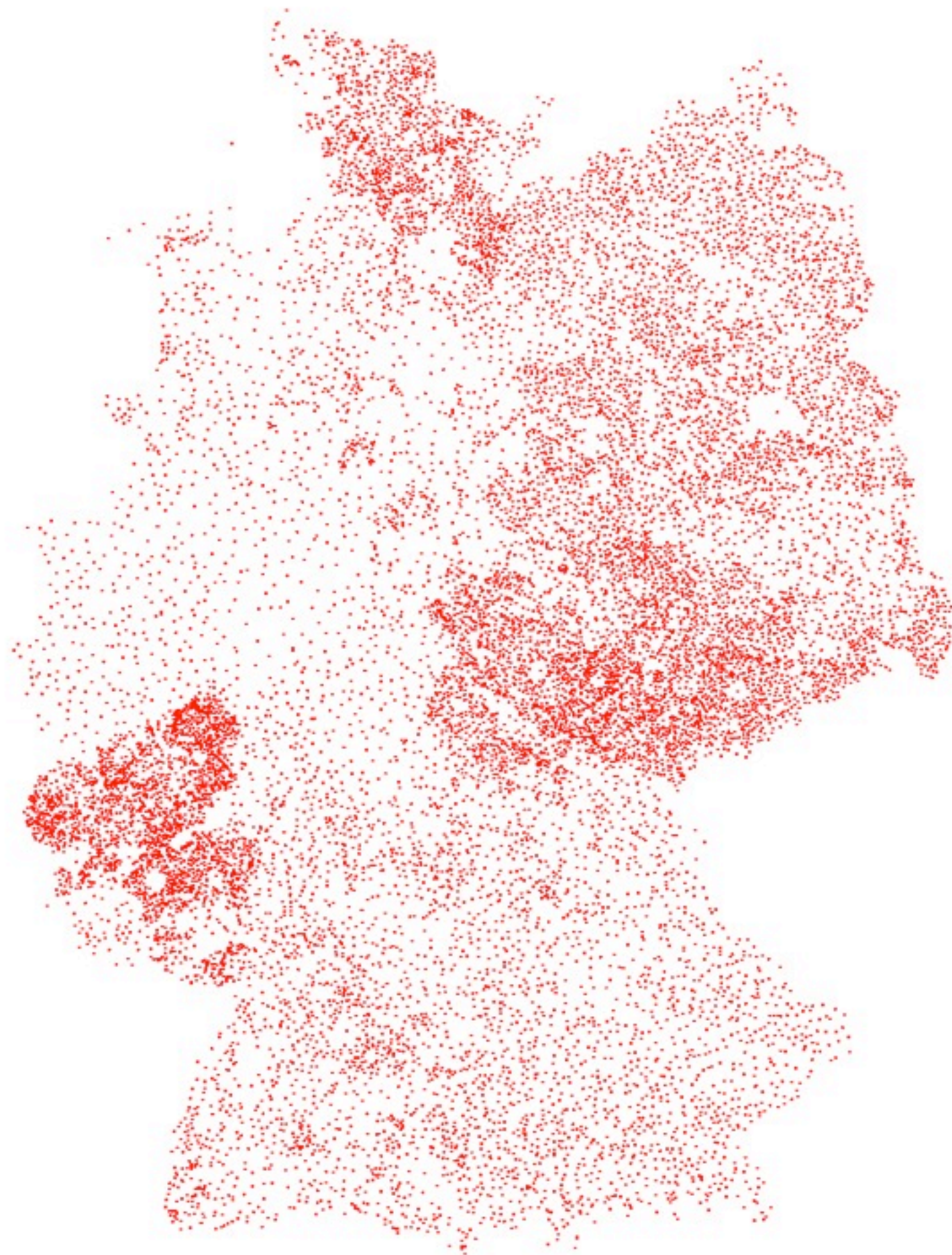
Conclusions

- Computational work in solving particular instances of hard discrete optimization problems has been remarkable. Perhaps we need a better theory to capture this reality?

  - More nuanced notion of efficient computation than polynomial time?

  - Some notion of 'real-life' instances of problems (such as TSP)?

Conclusions

"Or who shut in the sea with doors,
    when it burst out from the womb,
when I made clouds its garment
    and thick darkness its swaddling band,
and prescribed limits for it
    and set bars and doors,
and said, 'Thus far shall you come, and no farther,
    and here shall your proud waves be stayed'?"
        -- Job 38:8-11

- We've come a long way in understanding the power of efficient approximate computation for discrete optimization.

- But it is all relative to our understanding of efficient computation.