# Vector Representations of Graphs and the Maximum Cut Problem

David P. Williamson
Berkeley GraphXD
February 26, 2018

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

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

  - Two examples of approximation algorithms for the maximum cut problem

  - When is it hard to compute efficiently approximately?

- Some concluding thoughts

- "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

# Discrete Optimization Problems

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

- Lots of these formulated as graph problems.

- 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*?"
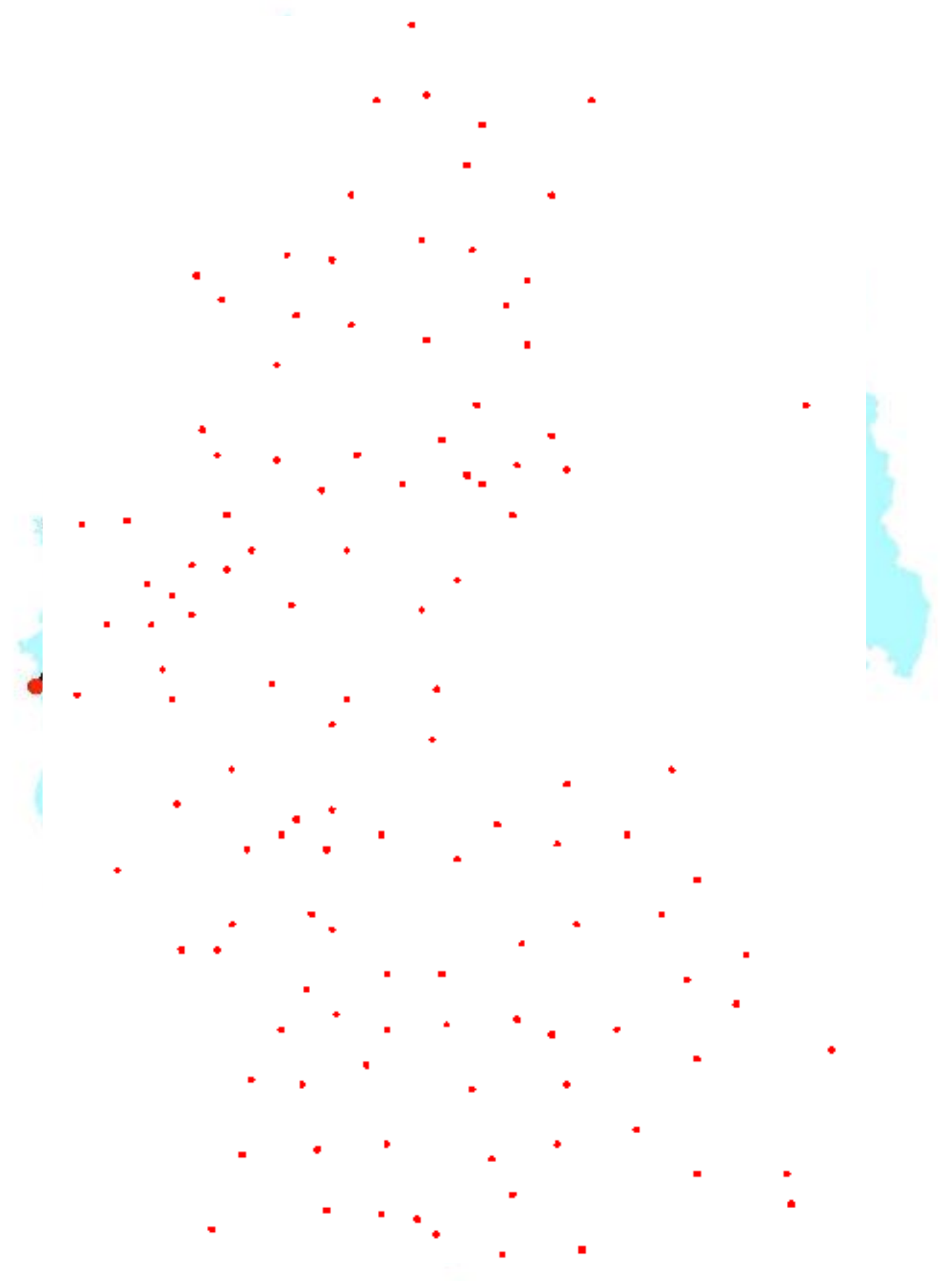
# The Obvious Finite Algorithm

- 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 $6 \times 10^{198}$.  1 tour/ns still is about $10^{182}$ years.

# "Good" algorithms

a given class of problems?

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 ...

Edmonds (1967):

sponding to each edge, an optimum branching

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 is easy to formalize relative say to a Turing machine,

on each edge. This is a version of the well-known 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.
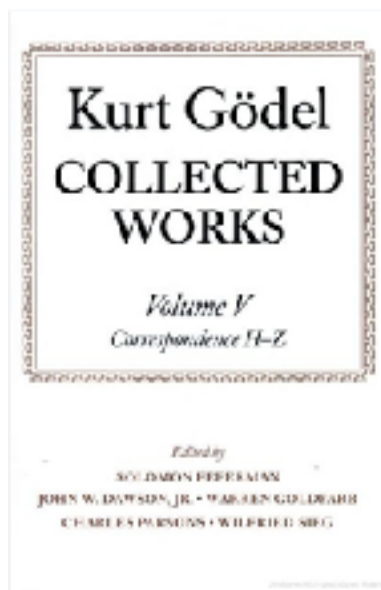
A good algorithm is known for finding in any graph
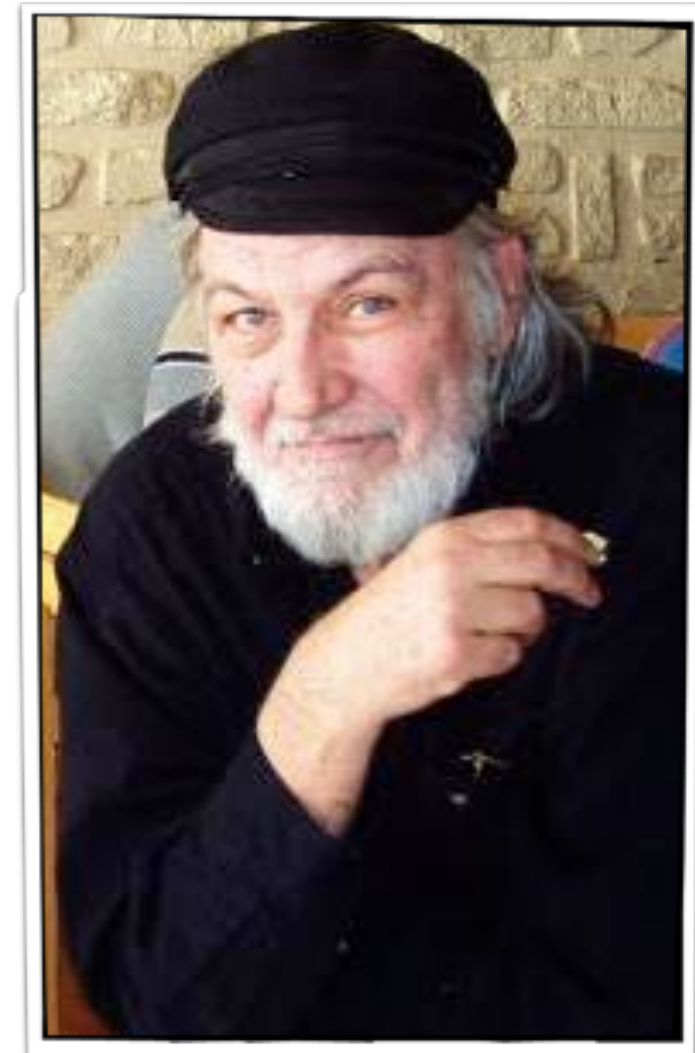
- Today: *polynomial-time algorithms* are considered the theoretical measure of a good, efficient algorithm.

- *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 Errun-
genschaften der Medizin, wenn möglich, zu einer vollständigen Heilung
Since ~ I hear, you are feeling stronger now, I would like take the
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 Tur-
ing 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 re-
ductions are definitely involved in the case of other finitist problems, e.g.,
when computing the quadratic remainder symbol by repeated applica-
tion 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
$\varphi$) 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



**Clay Mathematics Institute**
*Dedicated to increasing and disseminating mathematical knowledge*

HOME | ABOUT CMI | PROGRAMS | NEWS & EVENTS | AWARDS | SCHOLARS | PUBLICATIONS

## First Clay Mathematics Institute Millennium Prize Announced

### Prize for Resolution of the Poincaré Conjecture Awarded to Dr. Grigoriy Perelman

**March 18, 2010.** The Clay Mathematics Institute (CMI) announces today that Dr. Grigoriy Perelman of St. Petersburg, Russia, is the recipient of the Millennium Prize for resolution of the Poincaré conjecture. The citation for the award reads:

*The Clay Mathematics Institute hereby awards the Millennium Prize for resolution of the Poincaré conjecture to Grigoriy Perelman.*

More ...

## The Millennium Prize Problems

In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) established seven *Prize Problems*. The Prizes were conceived to record some of the most difficult problems with which mathematicians were grappling at the turn of the second millennium; to elevate in the consciousness of the general public the fact that

▸ Birch and Swinnerton-Dyer Conjecture
▸ Hodge Conjecture
▸ Navier-Stokes Equations
▸ P vs NP
▸ Poincaré Conjecture
▸ Riemann Hypothesis
▸ Yang-Mills Theory

▸ Rules
▸ Millennium Meeting Videos

**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.

Nevertheless, the computer science landscape

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 problem begins to loom large just as an interesting theoretical question in

*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 $α < 1$; 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.

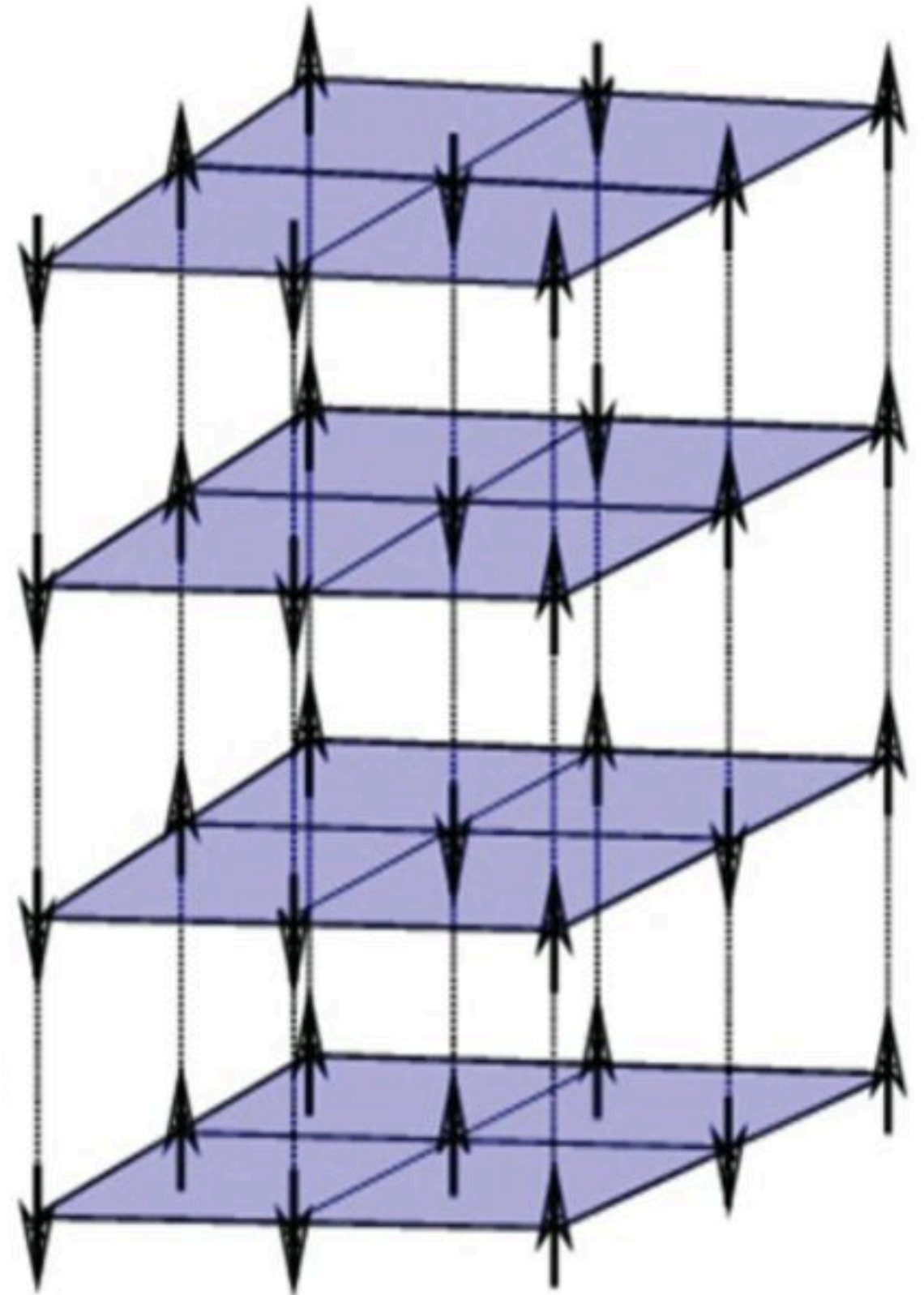- There is a polynomial-time algorithm if the graph is *planar*.
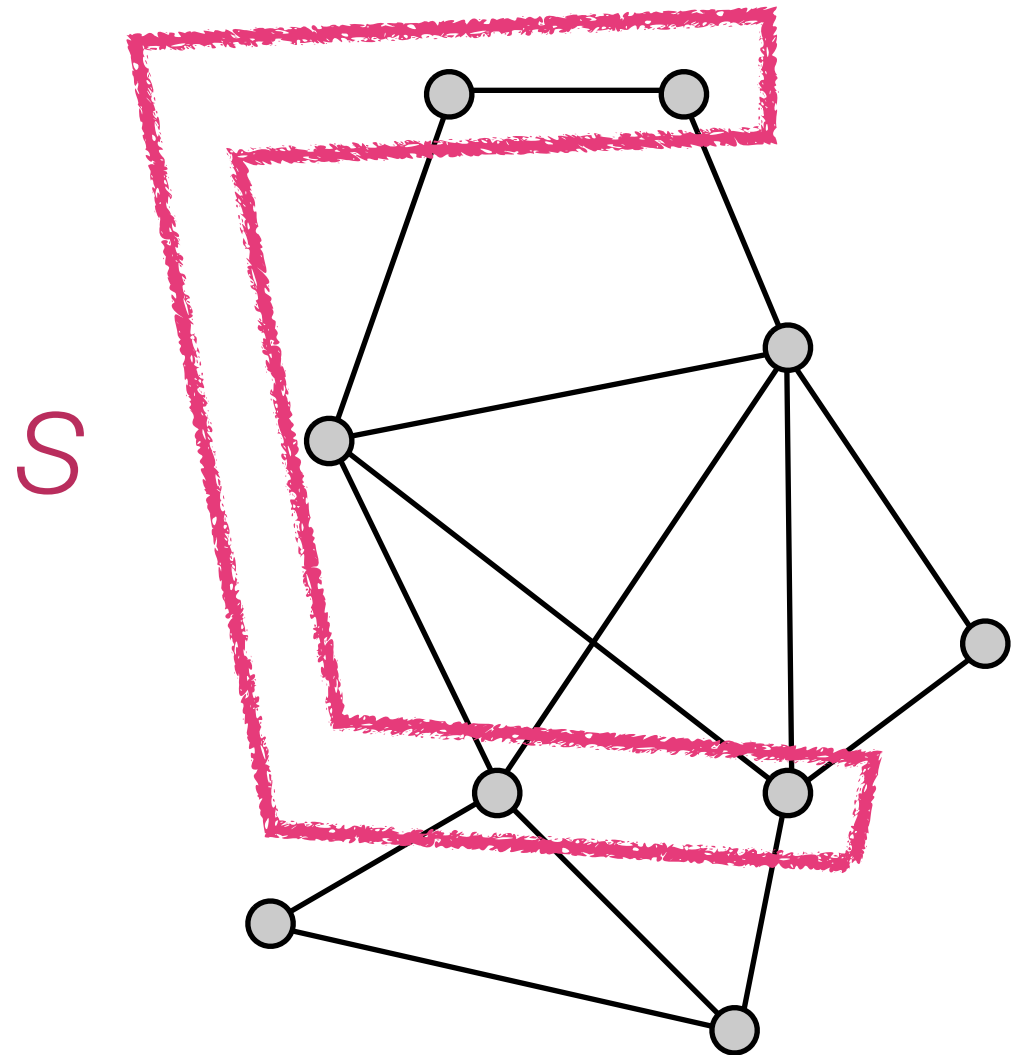
# Applications

- A few, when each edge has a weight, and we want to find a cut of maximum total weight

  - Finding minimal energy state of Ising spin glass model

  - Minimizing number of 'vias' drilled in two-sided printed circuit board.

- Barahona, Grötschel, Jünger, Reinert (1988)

- But some weights will be negative.



Heim et al, *Science*, 2015

# 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.
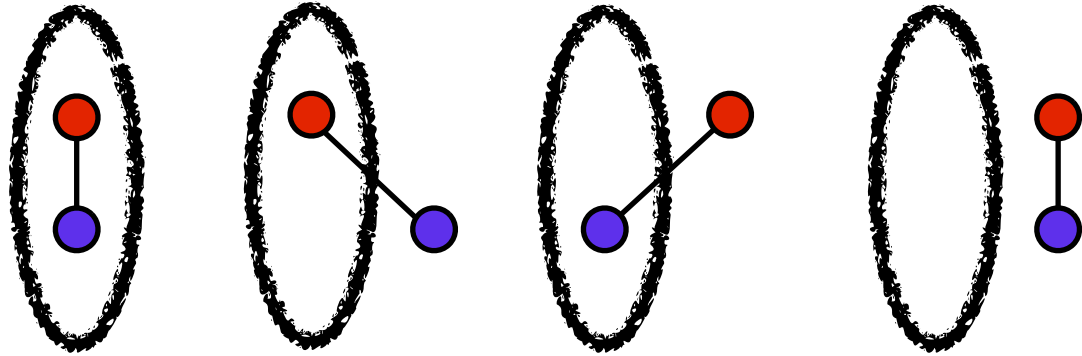


- Thus expected number of edges in the cut is 1/2 $|E|$, which is at least half the optimal value.

# An alternate approach (Goemans, W 1995)

- Suppose we introduce an *n*-dimensional unit vector $v_i$ (where *n=|V|*) for each vertex *i∈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∈V.*

- Then if we set *S={ i∈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

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

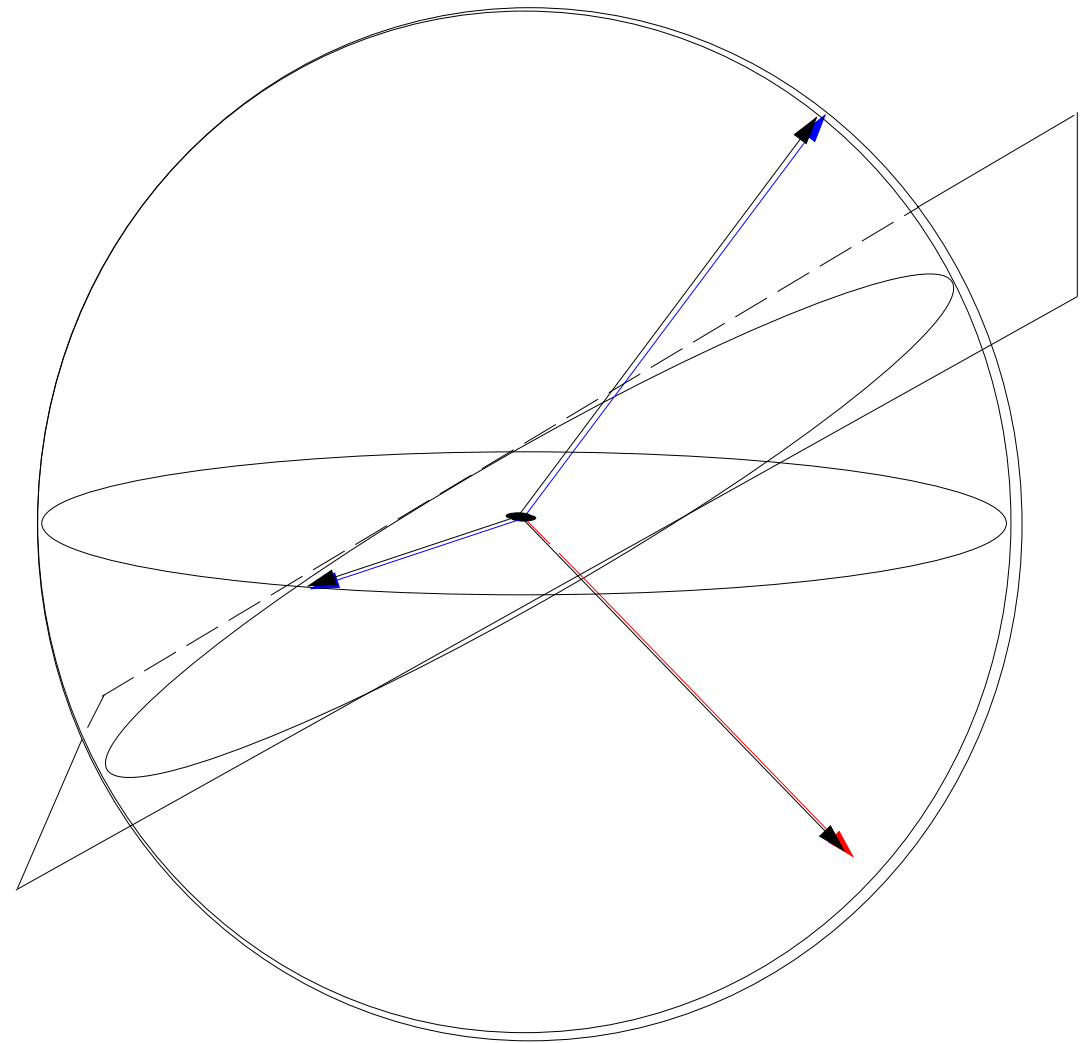  if the vectors are *arbitrary n-*dimensional unit-length vectors (via *semidefinite programming*).

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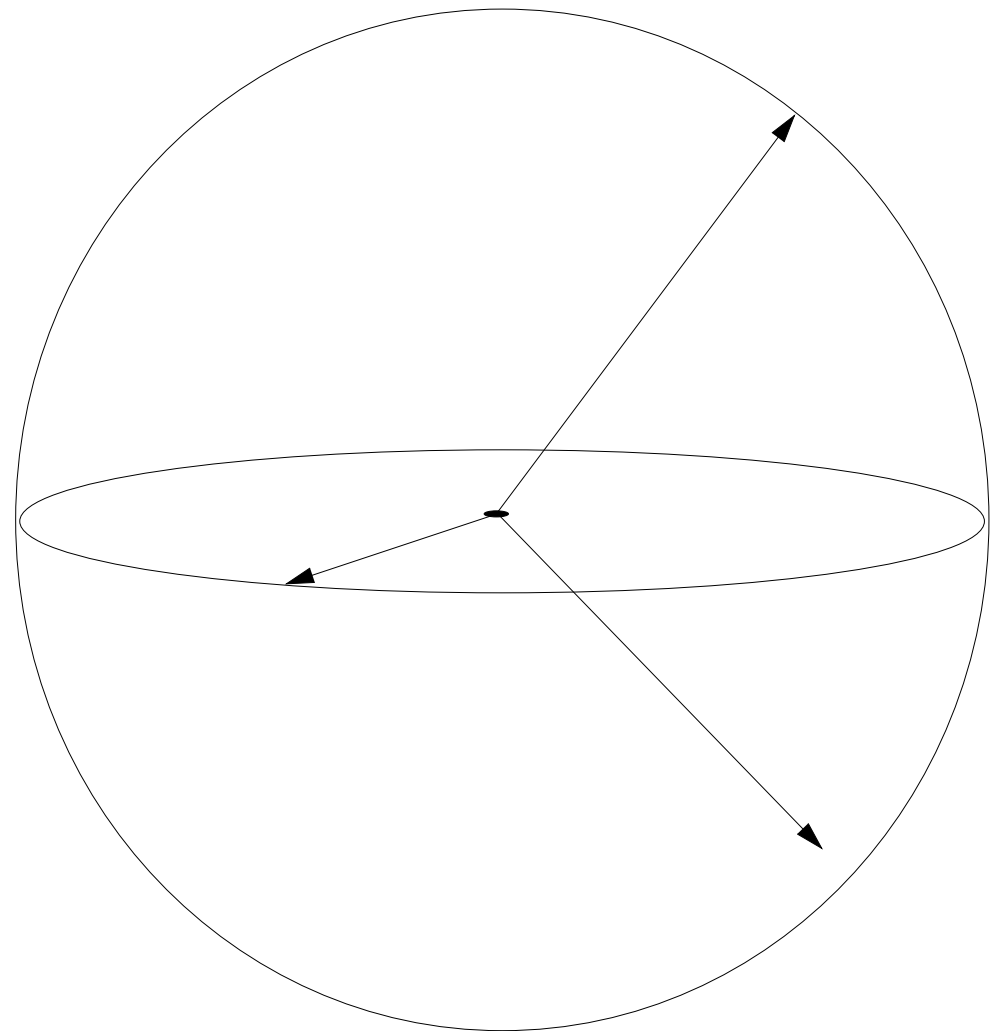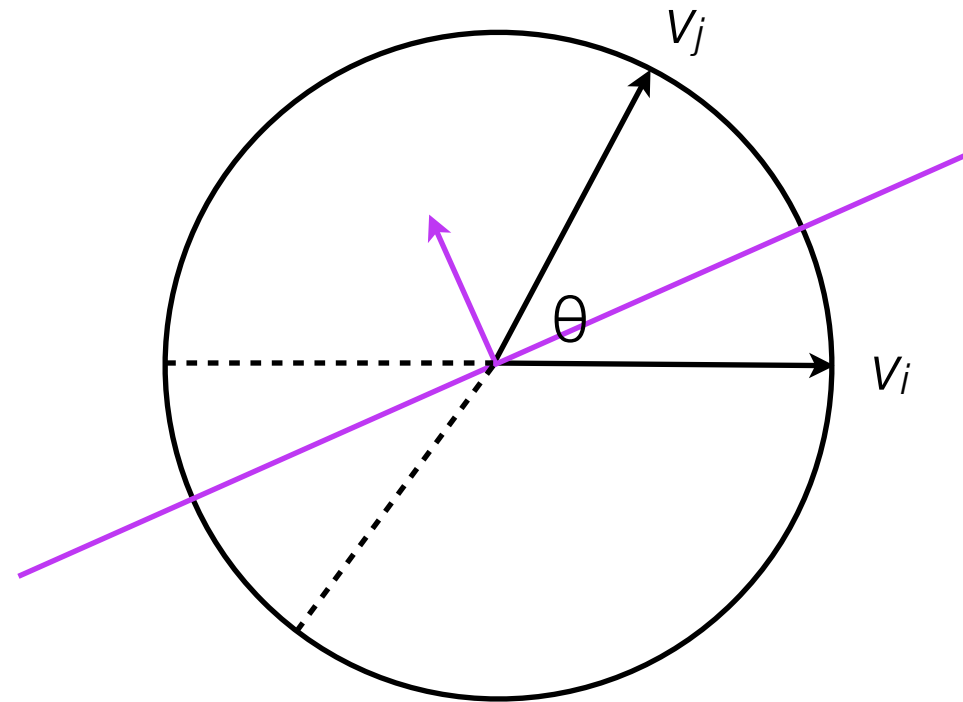
# Getting a solution

- 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*∈*V: $v_i$ · r ≥ 0* }.

- What is expected number of edges in this cut?
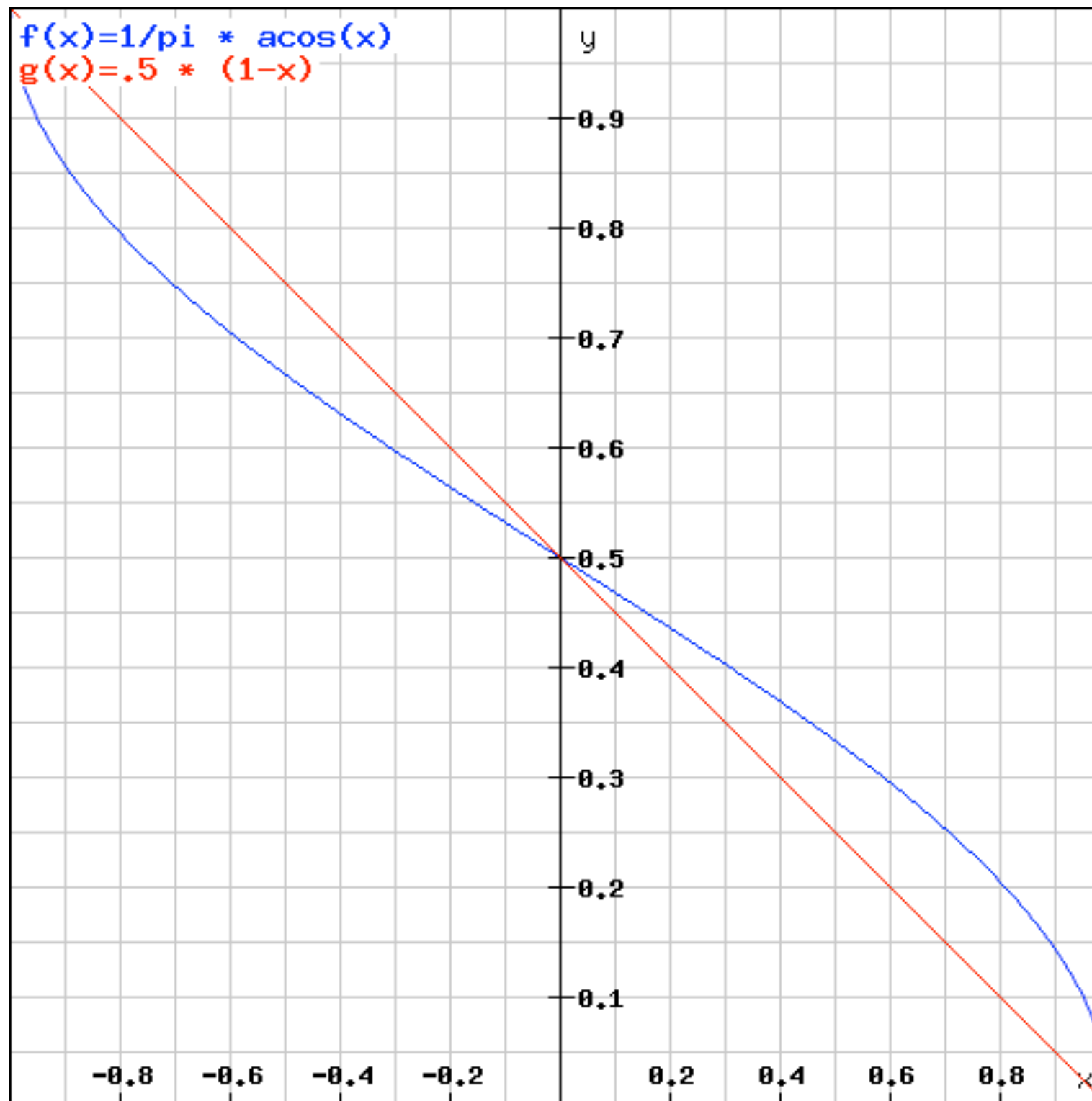
- Consider the plane containing vectors $v_i$ and $v_j$, and the projection of random vector $r$ to this plane.

- Of the $2\pi$ possible orientations of the projected random vector, $2\theta$ 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 $v_i \cdot v_j = \|v_i\|\|v_j\|\cos\theta = \cos\theta.$

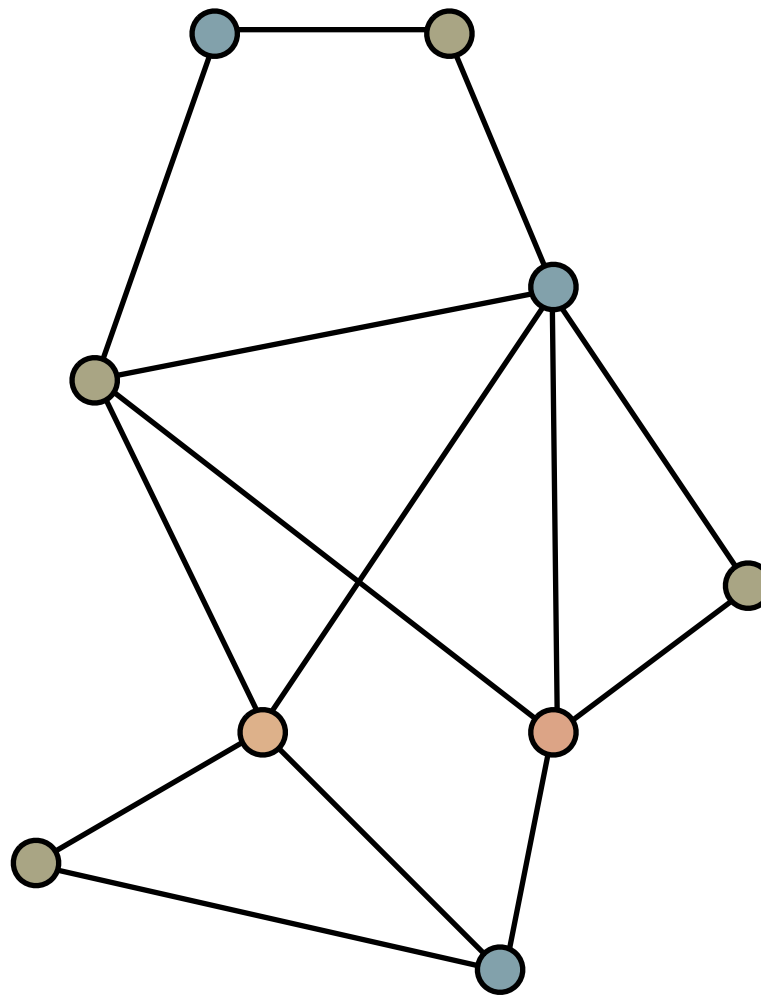$$\alpha = \min_{-1 \le x \le 1} \frac{\frac{1}{\pi}\arccos(x)}{\frac{1}{2}(1-x)} \ge .87856$$

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.

- Graph coloring: Find the minimum number of colors to assign to nodes such that each edge has differently colored endpoints.



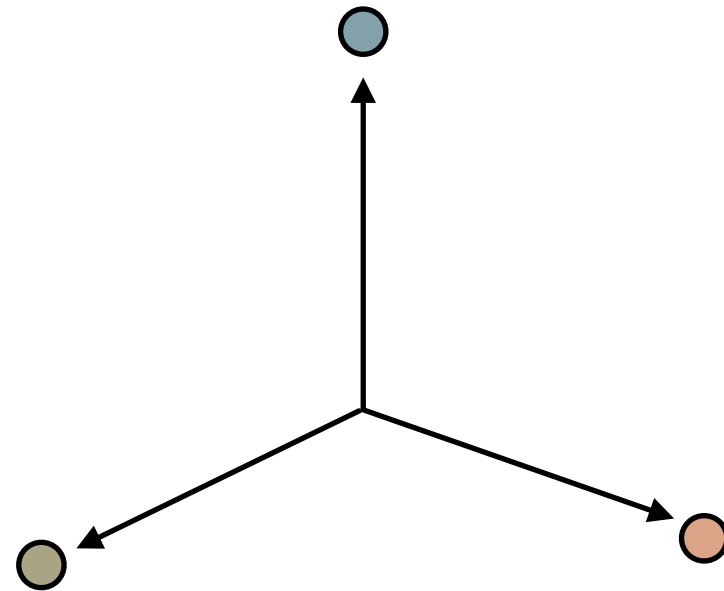- This graph is *3-colorable*: Can assign 3 colors to the nodes.

# More vectors

- Can solve the following in polynomial time for arbitrary *n*-dimensional unit vectors:

$$\min \max_{(i,j) \in E} v_i \cdot v_j$$

- If the graph is 3-colorable, then solution is at most -1/2; in general, if the graph is *k*-colorable, then there is a solution of at most -1/(*k*-1).

- Draw $t$ random vectors, and give node one of $2^t$ colors, one for each possible way of being above/below the $t$ random hyperplanes.

- The probability of two nodes joined by edge being on the same side of one random hyperplane is at most

$$1 - \frac{\theta}{\pi} = \frac{1}{3}$$

- So the probability of two nodes joined by an edge being on the same side of $t$ random hyperplanes is

$$\left(\frac{1}{3}\right)^t$$

- For good choice of $t,$ can color 3-colorable graph with O($n^{.387}$) colors (Karger, Motwani, Sudan 1998)

- Kawarabayashi and Thorup (2017) can color a 3-colorable graph with $O(n^{1.9996})$ colors.

# 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.

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

$$x_1 + x_3 + x_9 \equiv 0 (\mathrm{mod}\ 2)$$
$$x_2 + x_3 + x_{15} \equiv 1 (\mathrm{mod}\ 2)$$
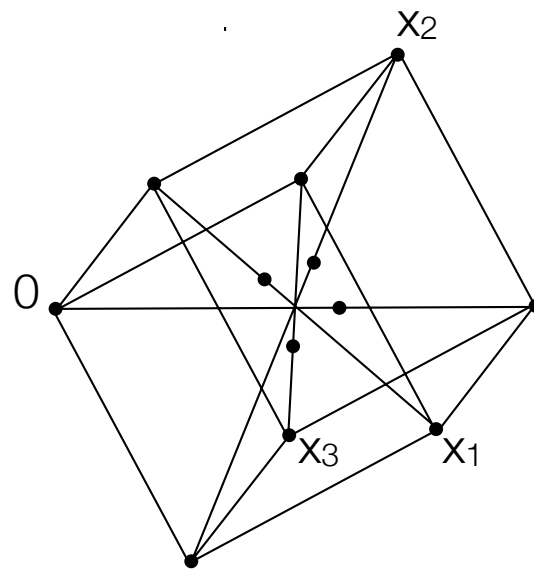$$x_1 + x_7 + x_{12} \equiv 0 (\mathrm{mod}\ 2)$$

$$\vdots$$

- 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 $(16/17+\varepsilon)$-approximation algorithm for $\varepsilon > 0$ unless P = NP. $(16/17 \approx 0.941)$.

# 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.

$$x_5 - x_3 \equiv 3 (\text{mod } 21)$$
$$x_3 - x_2 \equiv 2 (\text{mod } 21)$$
$$x_{19} - x_5 \equiv 15 (\text{mod } 21)$$

$$\vdots$$

# 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.
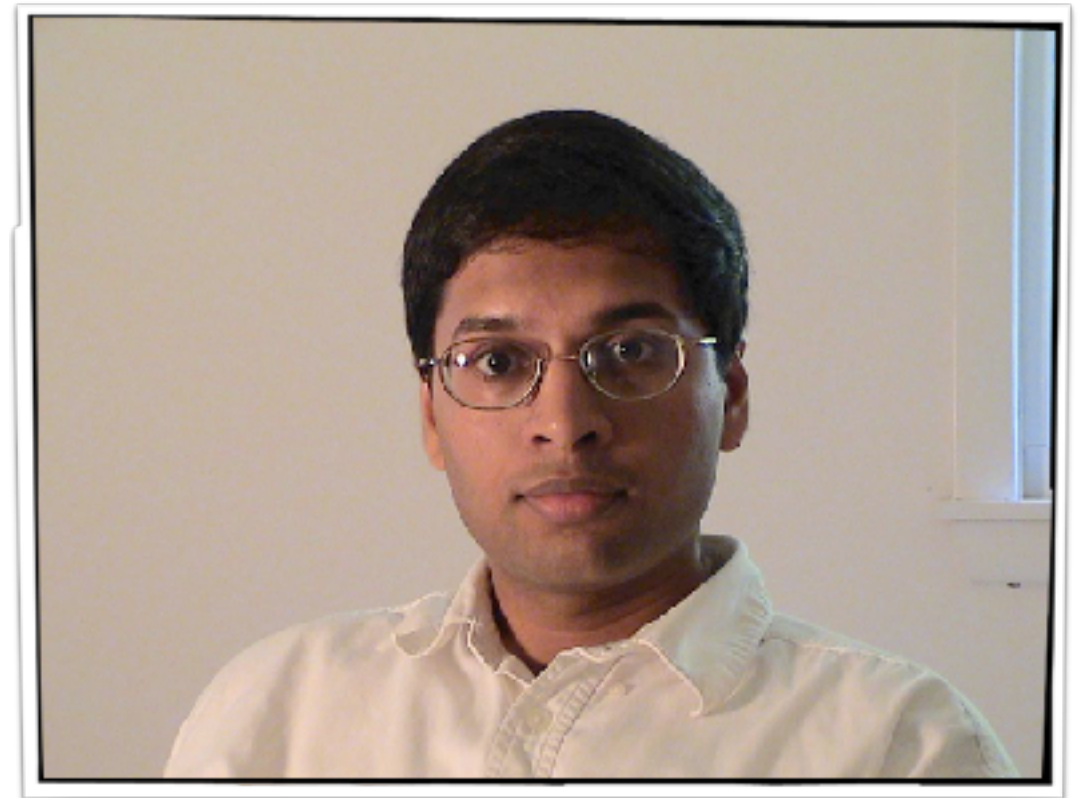
# Some consequences

- If the conjecture is true, then there is no (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).

- If a variant of the conjecture is true, then it is as hard as any problem in NP to color a 3-colorable graph with *k* colors for any constant *k* (Dinur, Mossel, Regev 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)

  - Not so recently: In Euclidean plane, given any $\varepsilon > 0$, there is a $(1 + \varepsilon)$-approximation algorithm (Arora 1998, Mitchell 1999).

  - Recent: A 1.4-approximation algorithm for a special case of metric spaces (Mömke, Svensson 2016; Sebo, Vygen 2014).

  - No $(\alpha-\varepsilon)$-approximation algorithm for $\alpha=123/122$ and $\varepsilon > 0$ unless P = NP.

- What else can be done with these vector representations of graphs?

  - Finding a sparse cut in a graph (Arora, Rao, Vazirani 2009)

# A critique?

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

- Edmonds (1965)

> between finite and non-finite
>
> 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.
>
> However, if only to delimit the search for good, practical algorithms, it
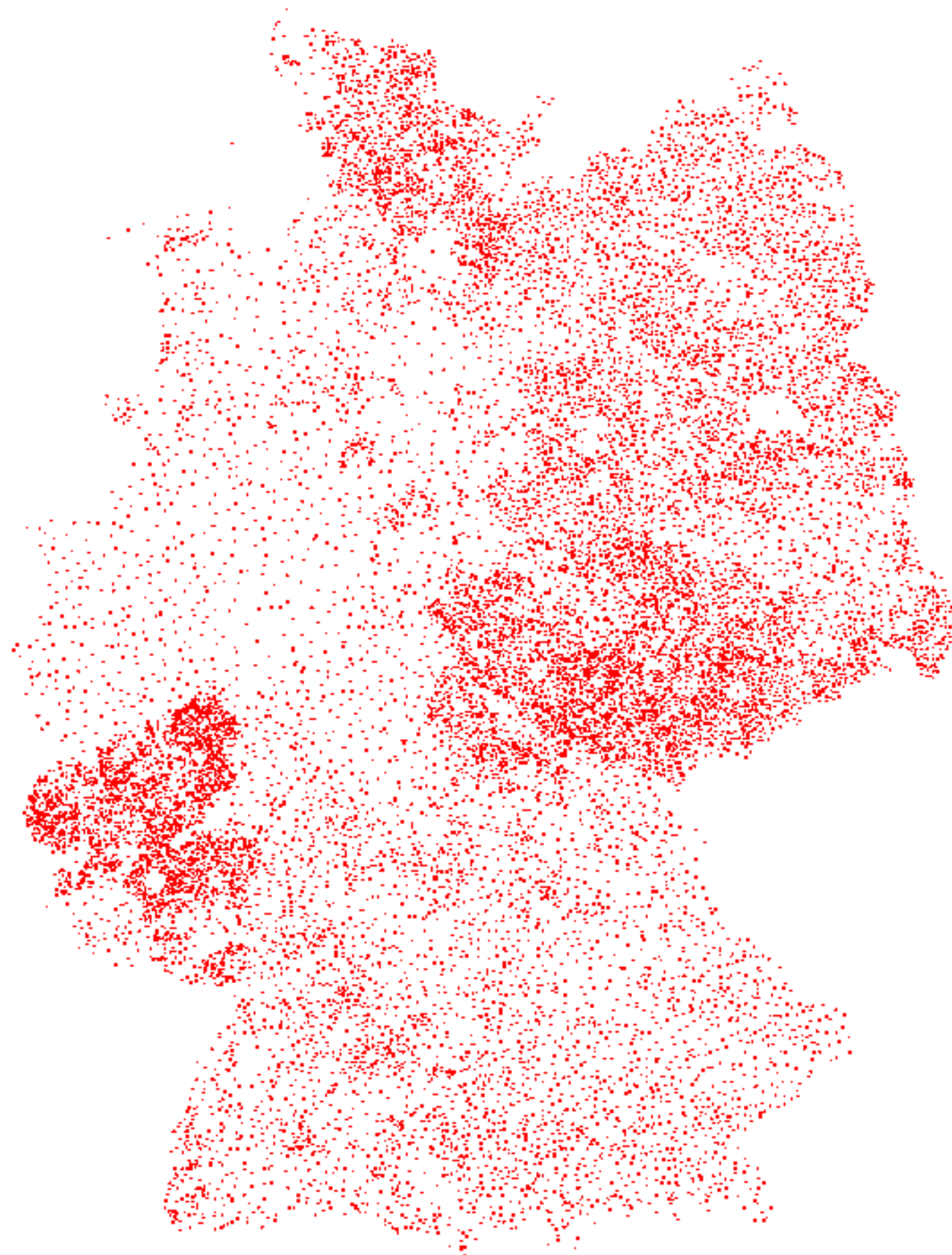
- Computational work in solving particular instances of hard discrete optimization problems has been remarkable. Perhaps we need a better theory to capture this reality?

  - More nuanced notion of efficient computation than polynomial time?

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

*"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.