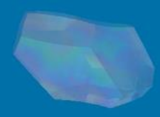
Have 2 cupcakes.

15-251: Great Theoretical Ideas in Computer Science
Lecture 29

Why Max-Cut is my favorite problem



(a TCS tale)

Faculty Course Evaluations

`https://cmu.smartevals.com`

Please fill one in!!

Unlike in other lectures,
I don't necessarily expect you
to understand everything today.

I'll be glossing over details
for the sake of the story.

Problems not known to be solvable in polynomial time, not known to be NP-hard:

1. Factoring

2. Graph-Isomorphism

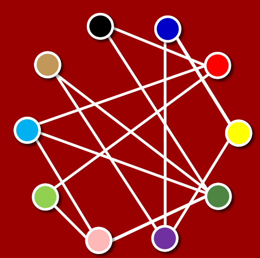
Handbook on Algorit[illegible] and Theory of Computation [illegible]

"The [illegible] of **natural** problems in NP ha[illegible]lved themselves [illegible] being **either in P or NP-compl[illegible]** [illegible]ss you uncover a specific con[illegible] of [the above] inte[illegible] it is more likel[illegible] problem simply need[illegible] work."

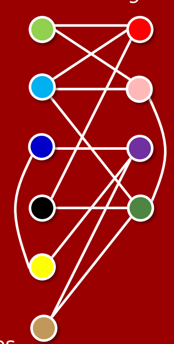
NOT TRUE

My fave problem: **Max-Cut**

Input: a graph



Output: a "bipartition"



Goal: max # of crossing ("cut") edges

My favorite problem: Max-Cut

Problem:
Find a bipartition achieving at least **90%** of the maximum possible # of cut edges.

We don't know if it's doable in polynomial time.
We don't know if it's NP-hard.

Similar situation for many approx. alg. problems.

But I'll tell you about Max-Cut, because it's my favorite problem.

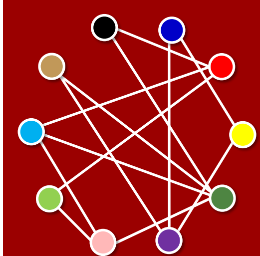
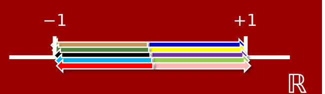
Achieving 50% of max in poly time.

We saw two algorithms for this:

Lecture 10: "Local search".
Lecture 18: Choose a random bipartition.

It took about 20 years to find a better algorithm.

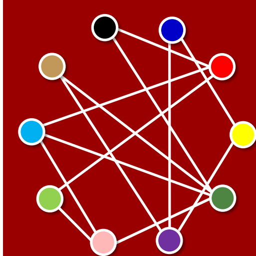
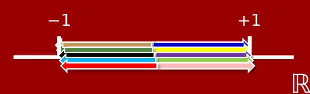
A better algorithm for Max-Cut

Want a bipartition into left (−1) and right (+1).

Start with an arbitrary one.

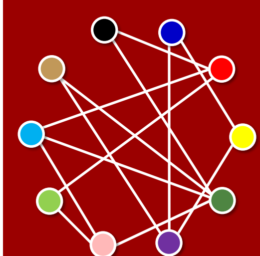
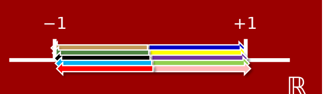
A better algorithm for Max-Cut

"Local Search" looks at vertices, flips them if this improves.

This is too drastic.

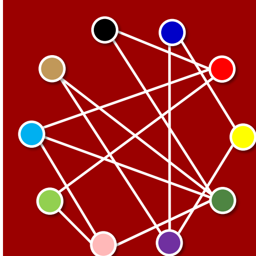
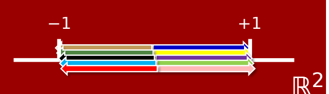
A better algorithm for Max-Cut

If only we could "partly flip".

This is too drastic.

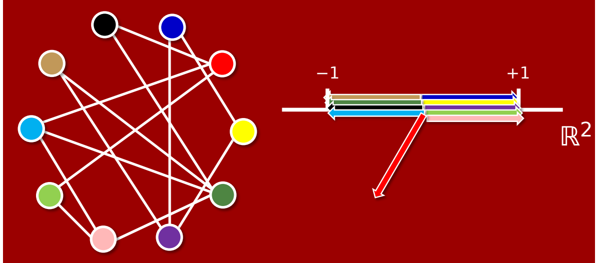
A better algorithm for Max-Cut

If only we could "partly flip".

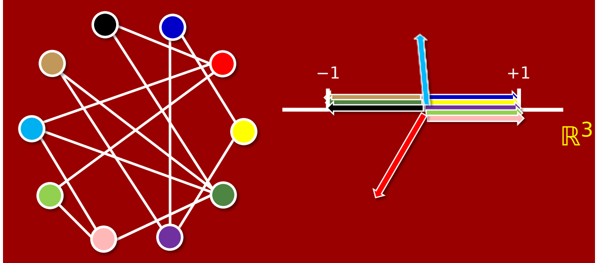
OK, just do it, using 2nd dimension.

A better algorithm for Max-Cut



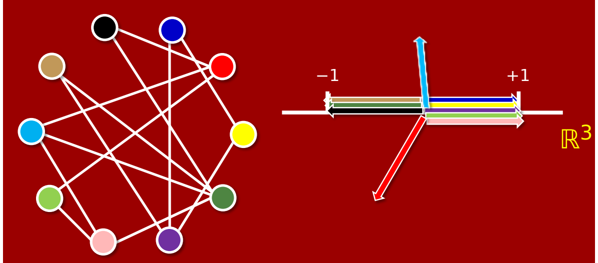
Imagine each arrow is repelled by the other arrows it has an edge to.

A better algorithm for Max-Cut



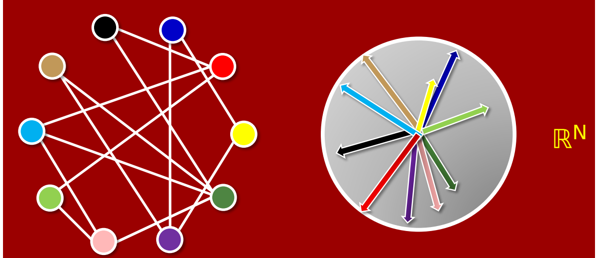
Imagine each arrow is repelled by the other arrows it has an edge to.

A better algorithm for Max-Cut



Keep going, letting vectors repel, into higher dimensions if necessary.

A better algorithm for Max-Cut

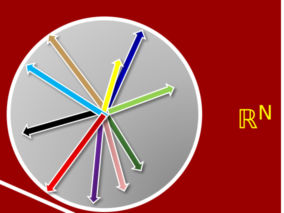


End goal: a unit vector σ_v for each $v \in V$, maximizing $\sum_{(u,v)\in E} \left(\frac{1}{2} - \frac{1}{2}\sigma_u \cdot \sigma_v\right)$

A better algorithm for Max-Cut

dot-product of vector for vertex u & vector for vertex v

(−1 if they're opposite, +1 if they're identical)



End goal: a unit vector σ_v for each $v \in V$, maximizing $\sum_{(u,v)\in E} \left(\frac{1}{2} - \frac{1}{2}\sigma_u \cdot \sigma_v\right)$

A better algorithm for Max-Cut

Amazing: Can find the optimal vectors in poly(n) time!

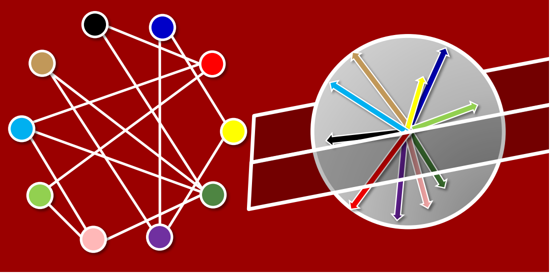
How? *It's a long and very interesting story…*

Starts with a Nov. 7, 1979 New York Times headline: *"A Soviet Discovery Rocks World of Mathematics"*

Ends with some awesome linear algebra.

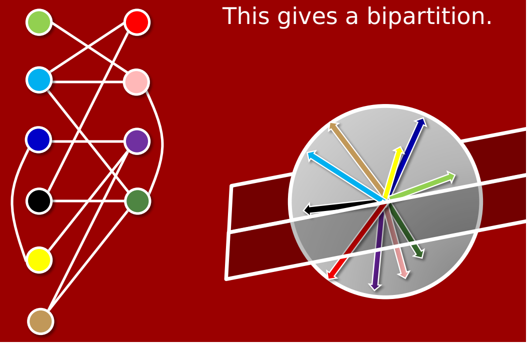
A better algorithm for Max-Cut

Last step: Pick a random hyperplane thru 0.
This gives a bipartition.



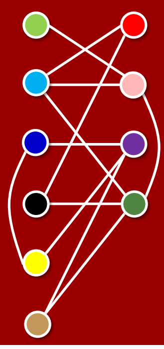
A better algorithm for Max-Cut

Last step: Pick a random hyperplane thru 0.
This gives a bipartition.



A better algorithm for Max-Cut

Last step: Pick a random hyperplane thru 0.
This gives a bipartition.



Not too hard analysis:

Expected # of edges cut is
≥ 87.8% of max-cut.


Michel Goemans + David Williamson 1994

A better algorithm for Max-Cut

Last step: Pick a random hyperplane thru 0.
This gives a bipartition.

Not too hard analysis:

Expected # of edges cut is
≥ 87.8% of max-cut.

actually 87.856720578485160421730103367…%

$$= \frac{2}{\pi \sin\theta^*}$$ where θ^* is soln. of $\tan(\theta/2) = \theta$.

(But who's counting?)

My favorite problem: Max-Cut

Problem:
Find a bipartition achieving at least **90%**
of the maximum possible # of cut edges.

We don't know if it's doable in polynomial time.
We don't know if it's NP-hard.

As of 1994, we know 87.8% is doable.
What about NP-hardness?

NP-hardness for Max-Cut

1972: NP-hard to achieve 100% of the maximum.

These days, it's a homework-level problem.

NP-hardness for Max-Cut

1992: Proof of the famous "PCP Theorem" (mentioned in Lecture 27).

PCP = Probabilistically Checkable Proofs.

Implies that 99.99999999%-approximation for Max-Cut is NP-hard.

What do PCPs have to do with approximation algorithms?

It's a long and very interesting story…

NP-hardness for Max-Cut

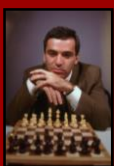
A "PCP" can somehow be thought of as a game.

Its 2 players are somehow cooperating "provers", playing a Max-Cut-like game.

PCP Theorem somehow gives a "game" they can win at most 99.99999999% of the time.

NP-hardness for Max-Cut

A "PCP" can somehow be thought of as a game.



We want to see how hard we can make it.

Idea: Something called **Parallel Repetition**.

NP-hardness for Max-Cut

A "PCP" can somehow be thought of as a game.

We want to see how hard we can make it.

Idea: Something called **Parallel Repetition**.

NP-hardness for Max-Cut

A "PCP" can somehow be thought of as a game.

Ran Raz '94

Proved this makes game much harder.

Unfortunately, it's now a weird game.

NP-hardness for Max-Cut

Ran Raz '94

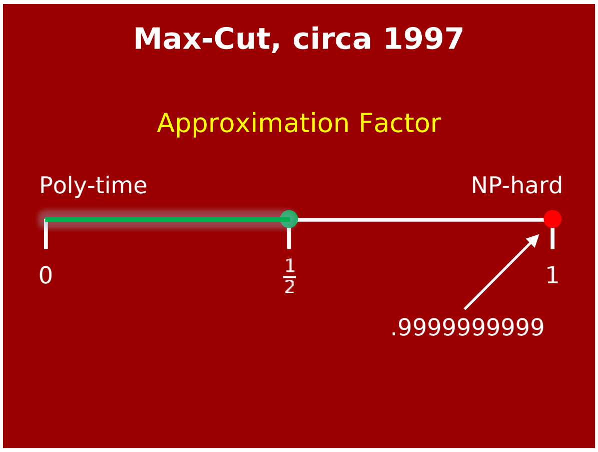
"Finding a 0.01% optimal solution to WeirdParallelChess is NP-hard."

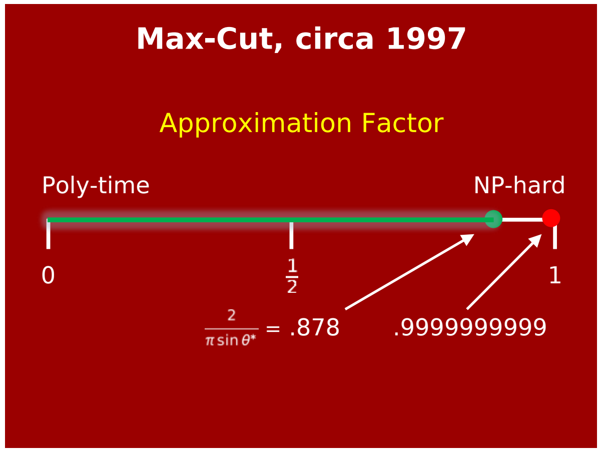
NP-hardness reduction ⇓ (involving lots of Fourier Analysis of Boolean Functions)

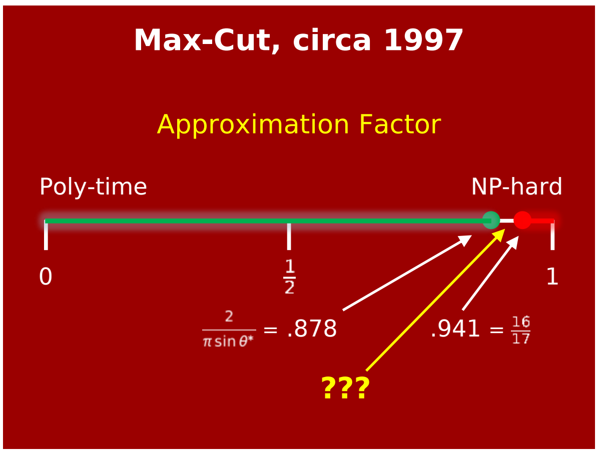
Johan Håstad '97

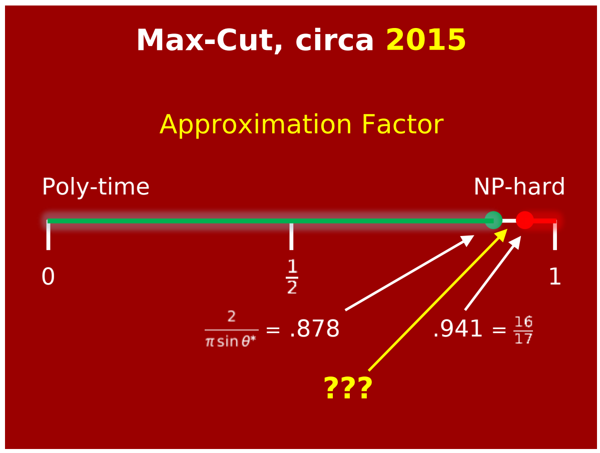
"Finding a 94.1% optimal solution to Max-Cut is NP-hard."

16/17, but who's counting?









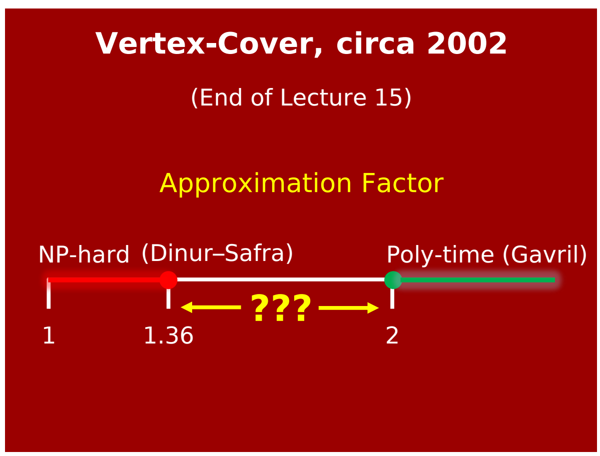
You might yawn, but to me it's awesome & terrible.

Max-Cut is maybe the simplest algorithms problem.

Not knowing if 90%-approximating can be done in O(n) time, or that it requires $2^{\Omega(n)}$ time, is terrible.

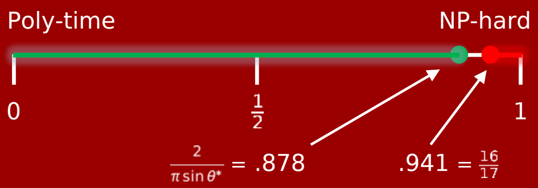
That number between .878 and .941 is, to me, like, the "fine structure constant" in theoretical physics.

Only more fundamental.



Max-Cut, circa 1997

Approximation Factor



Some interesting things did happen in the last 18 years…

WeirdParallelChess is so complicated!

Wouldn't it be cool if a *simpler game were equally hard*?

A game where, for every "move" of one player, there is a **unique** (forced) best move for the other player?

Subhash Khot '02

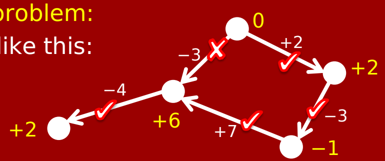
I call it… the **Unique Games Conjecture**.

Unique Games Conjecture

Turns out, Khot's simpler game is equivalent to…

Topography problem:

Input looks like this:

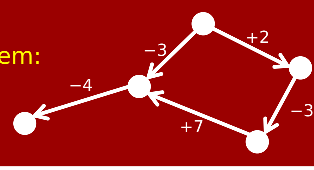


Think: nodes = cities, edges = elevation differences.

Goal: label cities by elevations, so that as many elevation differences as possible are right.

Unique Games Conjecture

Topography problem:



Unique Games Conjecture ≡

Given a Topography input where it's possible to get ≥ 99.99% of the differences right,

it is NP-hard to find a solution getting ≥ 0.01% of the differences right.

NP-hardness for Max-Cut

Ran Raz '94

"Finding a 0.01% optimal solution to WeirdParallelChess is NP-hard."

NP-hardness reduction ⇓ (involving lots of Fourier Analysis of Boolean Functions)

Johan Håstad '97

"Finding a 94.1% optimal solution to Max-Cut is NP-hard."

16/17, but who's counting?

NP-hardness for Max-Cut

Subhash Khot, 2002

Conjecture: "Finding a 0.01% optimal solution to Unique-Games is NP-hard."

NP-hardness reduction ⇓ (involving lots of Fourier Analysis of Boolean Functions)

Johan Håstad '97

"Finding a 94.1% optimal solution to Max-Cut is NP-hard."

16/17, but who's counting?

NP-hardness for Max-Cut (?)

Subhash Khot, 2002

Conjecture: “Finding a 0.01% optimal solution to Unique-Games is NP-hard.”

NP-hardness reduction ⇓ (involving lots of Fourier Analysis of Boolean Functions)

“Finding a **???**% optimal solution to Max-Cut is NP-hard.”

NP-hardness for Max-Cut (!)

Subhash Khot, 2002

Conjecture: “Finding a 0.01% optimal solution to Unique-Games is NP-hard.”

NP-hardness reduction ⇓ (involving lots of Fourier Analysis of Boolean Functions)

“Finding a solution better than **87.8567205784**% of optimal for Max-Cut is NP-hard.”

Proving the reduction worked required proving a new theorem called “Majority Is Stablest Theorem”.

NP-hardness reduction ⇓ (involving lots of Fourier Analysis of Boolean Functions)

Proving the reduction worked required proving a new theorem called “Majority Is Stablest Theorem”.

Imagine a 2-party election where each vote has a small probability ε of being miscounted.

Proving the reduction worked required proving a new theorem called “Majority Is Stablest Theorem”.

Imagine a 2- … re each vote has a small pr … g miscounted.

What' … ounts affect th … election?

It depends on the “voting scheme”. (Simple majority, “electoral college”, etc…)

Proving the reduction worked required proving a new theorem called “Majority Is Stablest Theorem”.

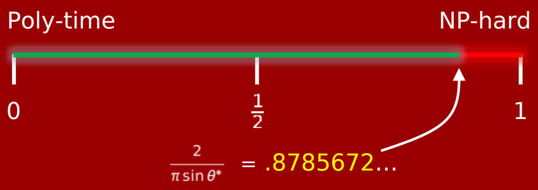
Theorem: Among “fair” voting schemes, the one least susceptible to miscounts is Majority.

What does **voting theory** have to do with NP-hardness reductions?

It’s a long and very interesting story…

The Max-Cut picture

Approximation Factor



IF…

you believe the “Unique Games Conjecture”

Unique Games Conjecture

If you believe it, we get perfect understanding of the approximability of Max-Cut, Vertex-Cover.

And in fact ALL Constraint Satisfaction Problems.

Prasad Raghavendra, 2009

However, many people **disbelieve it**!

Unique Games Conjecture

50% of researchers believe it, try to prove it;
50% of research *disbelieve* it, try to *disprove* it.

Unique Games Conjecture

50% of researchers believe it, try to prove it;
50% of research *disbelieve* it, try to *disprove* it.

Unique Games Conjecture

50% of researchers believe it, try to prove it;
50% of research *disbelieve* it, try to *disprove* it.

Because of this, I actually think it’s **more** interesting than **P** vs. **NP**.

Because, except for weirdos like Anıl, pretty much everyone agrees **P** ≠ **NP**.

jk, Anıl

Final story:

The time Guy Kindler, Uri Feige, and I tried to prove the Unique Games Conjecture.

It was 2005–2006, we were all working at Microsoft Research.

We had some plan to prove it via a twist on Parallel Repetition.

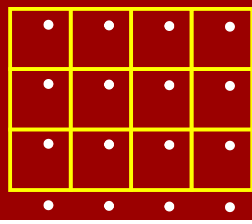
Everything boiled down to a problem about **foam**.

Why? *It's a long and very interesting story.*

The cubical **foam** problem:

Say that a shape 'tiles d-dim. space cubically' if, when you shift it by all integer amounts in all d directions in $\mathbb{R}^d$, it exactly covers space.

How small can its surface area / perimeter be?



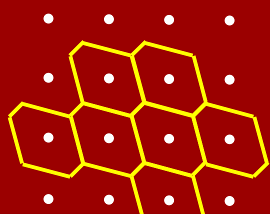
the integer points in $\mathbb{R}^2$

perimeter: 4

The cubical **foam** problem:

Say that a shape 'tiles d-dim. space cubically' if, when you shift it by all integer amounts in all d directions in $\mathbb{R}^d$, it exactly covers space.

How small can its surface area / perimeter be?

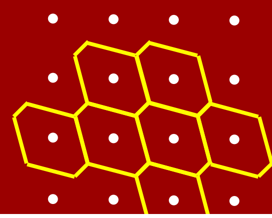


the integer points in $\mathbb{R}^2$

perimeter: $\sqrt{6}+\sqrt{2}\approx 3.84$

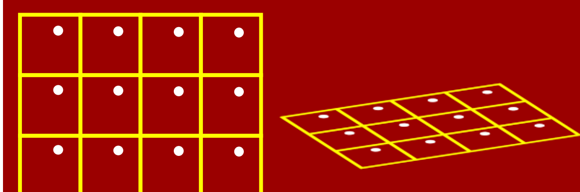
This is the optimal solution in 2 dimensions.

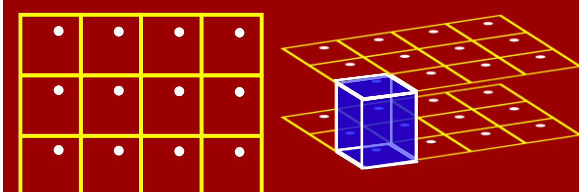
What about 3 dimensions?

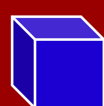


the integer points in $\mathbf{R}^2$

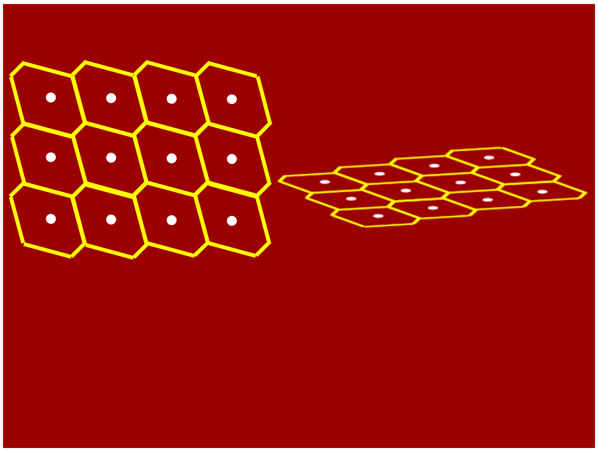
perimeter: $\sqrt{6}+\sqrt{2}\approx 3.84$

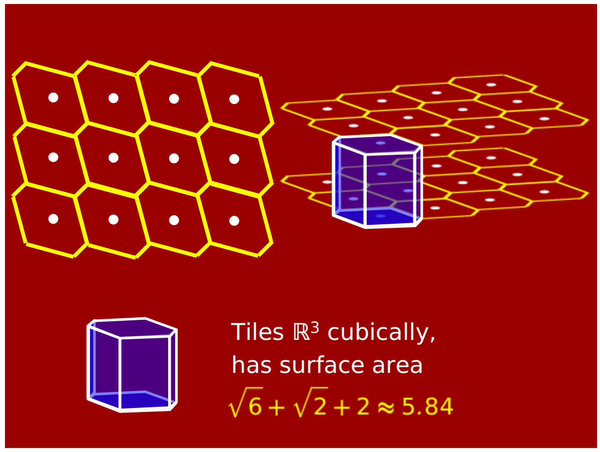


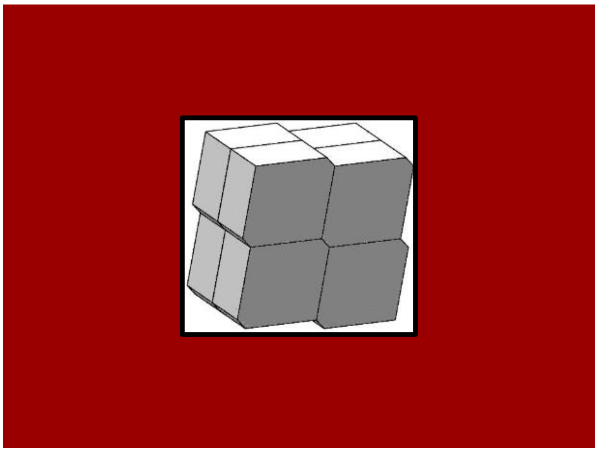


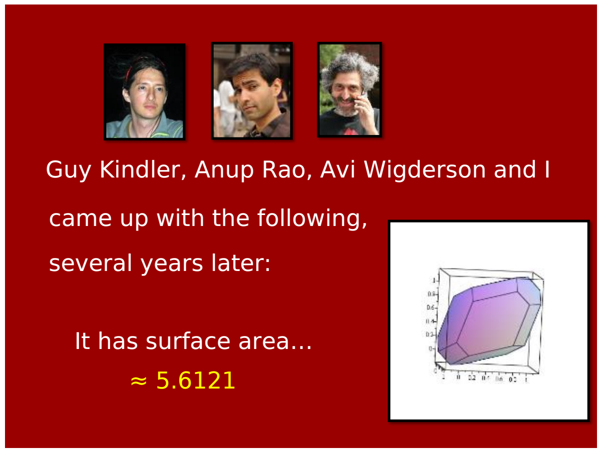


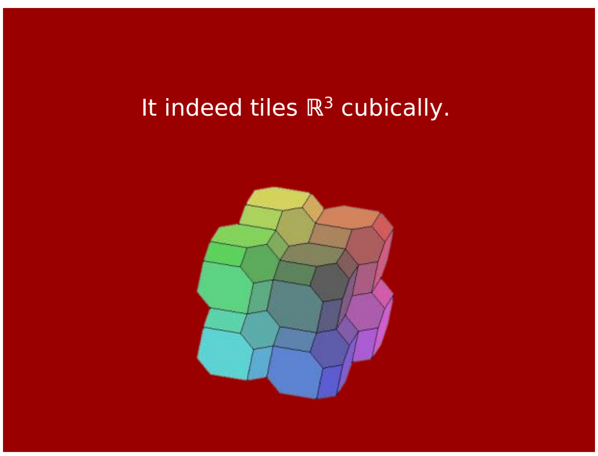
Tiles $\mathbb{R}^3$ cubically, has surface area 6.

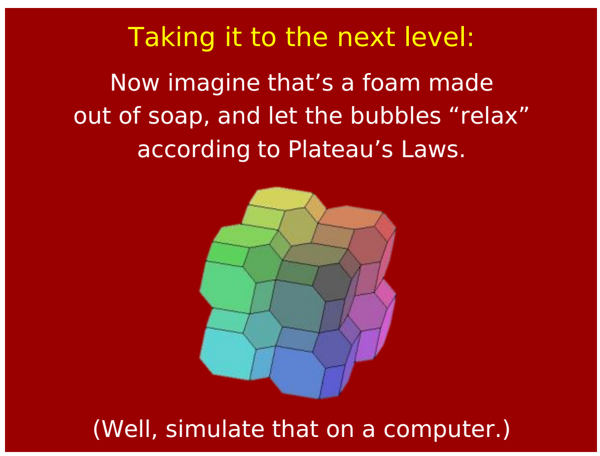


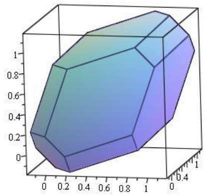

Tiles $\mathbb{R}^3$ cubically,
has surface area
$\sqrt{6}+\sqrt{2}+2 \approx 5.84$




Guy Kindler, Anup Rao, Avi Wigderson and I
came up with the following,
several years later:
It has surface area…
≈ 5.6121

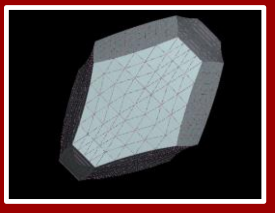

It indeed tiles $\mathbb{R}^3$ cubically.

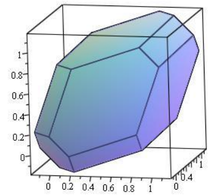

Taking it to the next level:
Now imagine that's a foam made
out of soap, and let the bubbles "relax"
according to Plateau's Laws.
(Well, simulate that on a computer.)

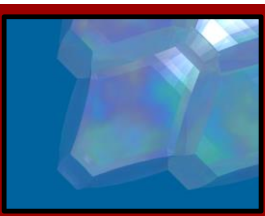


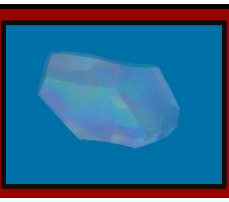
New surface area:
≈ 5.602

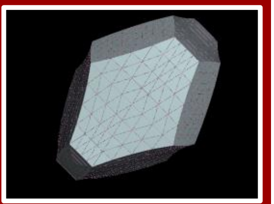
(Best solution I know.)









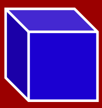


Back to the Unique Games Conjecture story.

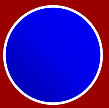
For that, we cared about
the **high**-dimensional version.

What can you say about the surface area
of shapes that tile $\mathbb{R}^d$ cubically?

What can you say about surface area
of shapes which tile $\mathbb{R}^d$ cubically?



You can always use the cube:
surface area $2d$. OTOH…



Any tiling shape will have volume 1.

Any volume-1 shape has at least as much surface area as vol.-1 sphere.

Which in d dimensions is $\approx \sqrt{d}$

Let $A(d)$ be the least surface area of
a shape which tiles $\mathbb{R}^d$ cubically.

We know $\sqrt{d} \lesssim A(d) \leq 2d$.

For our Unique Games Conjecture plan to work,
all we needed was that the correct answer
was NOT $A(d) = \Theta(\sqrt{d})$.

We really believed that $A(d) = \Theta(d)$.

I mean, come on:
How can a shape tile space in a cubical pattern
without kind of looking like a cube??

Well, uh, apparently it can.
The gang & I proved that $A(d) = \Theta(\sqrt{d})$.

Only consolation: we got to write a paper
called *"Spherical Cubes"*.

Theoretical Computer Science

Max-Cut: it’s the most basic algorithms problem.

But understanding its computational complexity

took us from geometry,

to probabilistic proofs,

to voting theory,

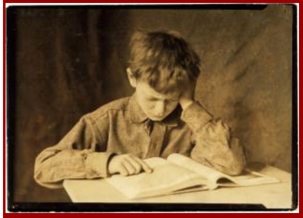
to foams…

That’s what’s cool about Theoretical Comp. Science:
beautiful intersections with all of math and science.

Reminder: Faculty Course Evaluations

`https://cmu.smartevals.com`

Study Guide



Lectures 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24.