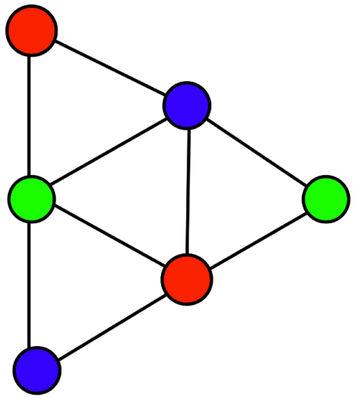
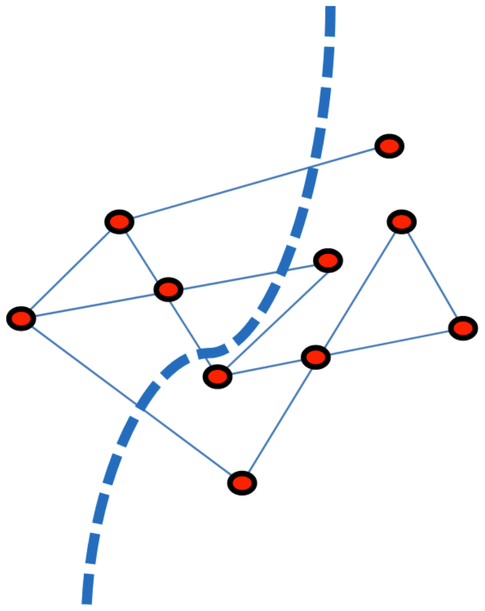
Approximately Coloring a 3-colorable Graph Using Semidefinite Programming





"coloring a 3-colorable graph in poly-time using as few colors as possible"

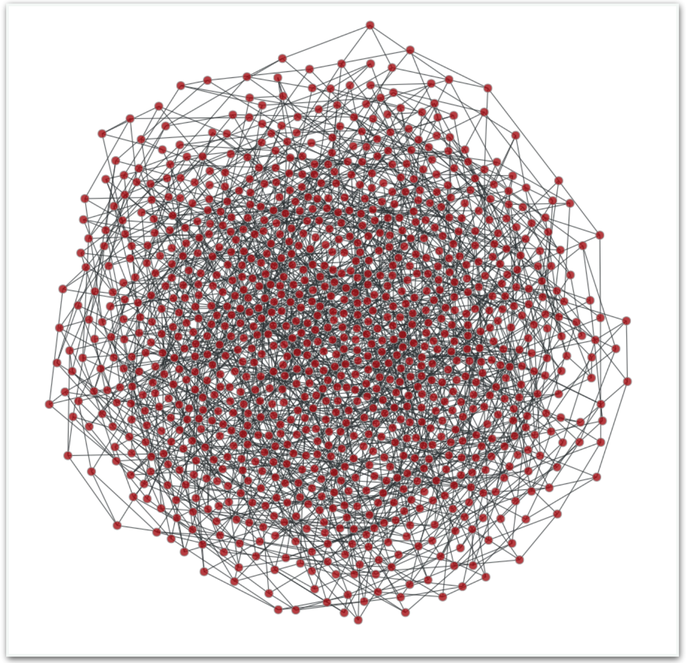
Presenter: Tzu-Chi Yen
April 30th, 2019

What does it mean to approximately color a graph in terms of n?

Let's say — we have a 3-colorable graph with 1,000 vertices…

For $O\left(n^{0.5}\right)$ We need ~32 colors; actually we'll need ~128 colors, as we will see.

For $O\left(n^{0.19996}\right)$ We need ~4 colors.



How can we color it using as fewer colors as we can??

😭

No implementation (yet), only mathematical sketches

Other facts

Coloring a 3-colorable graph using 4 colors is NP-hard. Same is true for 5 colors.

Actually, it is NP-hard to color with any constant number of colors.

A brief history of approximate coloring a 3-colorable graph

$O\left(n^{0.5}\right)$	Wigderson; STOC'82	purely *combinatorial* approach - 1
$\tilde{O}\left(n^{0.4}\right)$	Blum; STOC'89	purely *combinatorial* approach - 2
$\tilde{O}\left(n^{0.375}\right)$	Blum; FOCS'90	purely *combinatorial* approach - 3
$\tilde{O}\left(n^{0.25}\right)$	Karger, Motwani, & Sudan; FOCS'94	purely *combinatorial* approach - 1 & first *SDP* approach
$\tilde{O}\left(n^{0.2143}\right)$	Blum & Karger; IPL'97	purely *combinatorial* approach - 3 & first *SDP* approach
$\tilde{O}\left(n^{0.2111}\right)$	Arora, Chlamtac, & Charikar; STOC'06	purely *combinatorial* approach - 3 & b*etter SDP* approach (sparsest cut)
$O\left(n^{0.2072}\right)$	Chlamtac; FOCS'07	purely *combinatorial* approach - 3 & *even better SDP* approach (sparsest cut)
$O\left(n^{0.2049}\right)$	Kawarabayashi & Thorup; FOCS'12	purely *combinatorial* approach - 4 & *even better SDP* approach
$O\left(n^{0.19996}\right)$	Kawarabayashi & Thorup; STACS'14	

Applications of graph k-colorability

- Scheduling
- Register allocation
- Image segmentation
- Optimization
- Parallel numerical computation

Strategies for combating hard problems

— In class, we've learned that the integer linear program (ILP) formulation can be used to model several hard problems exactly.

— But since ILPs are super hard as well, we relax them into linear programs (LPs). And solve those instead.

— After carefully rounding the solution to a LP relaxation, we attain an **approximation** of an optimal solution for the original hard problem.

— Generally, there are 2 ways to further improve the approximation.

One: adding new "valid" constraints to the ILP, and relax to a new LP. Solve it.

Two: Look for higher-order relaxations that can be solved in poly-time, such as SDP.

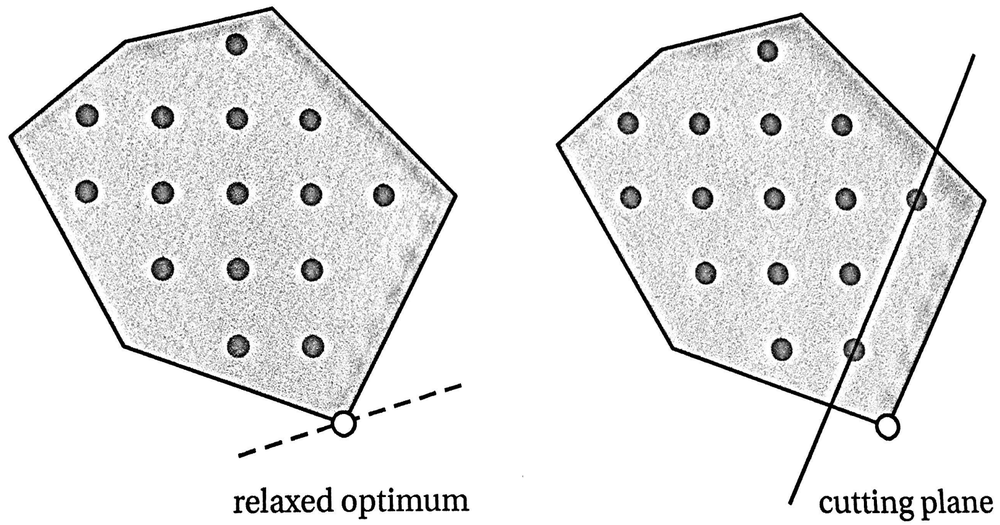


FIGURE 9.43: A cutting plane brings us closer to an integer solution.

Early Results: Wigderson's trick

Basic observations:

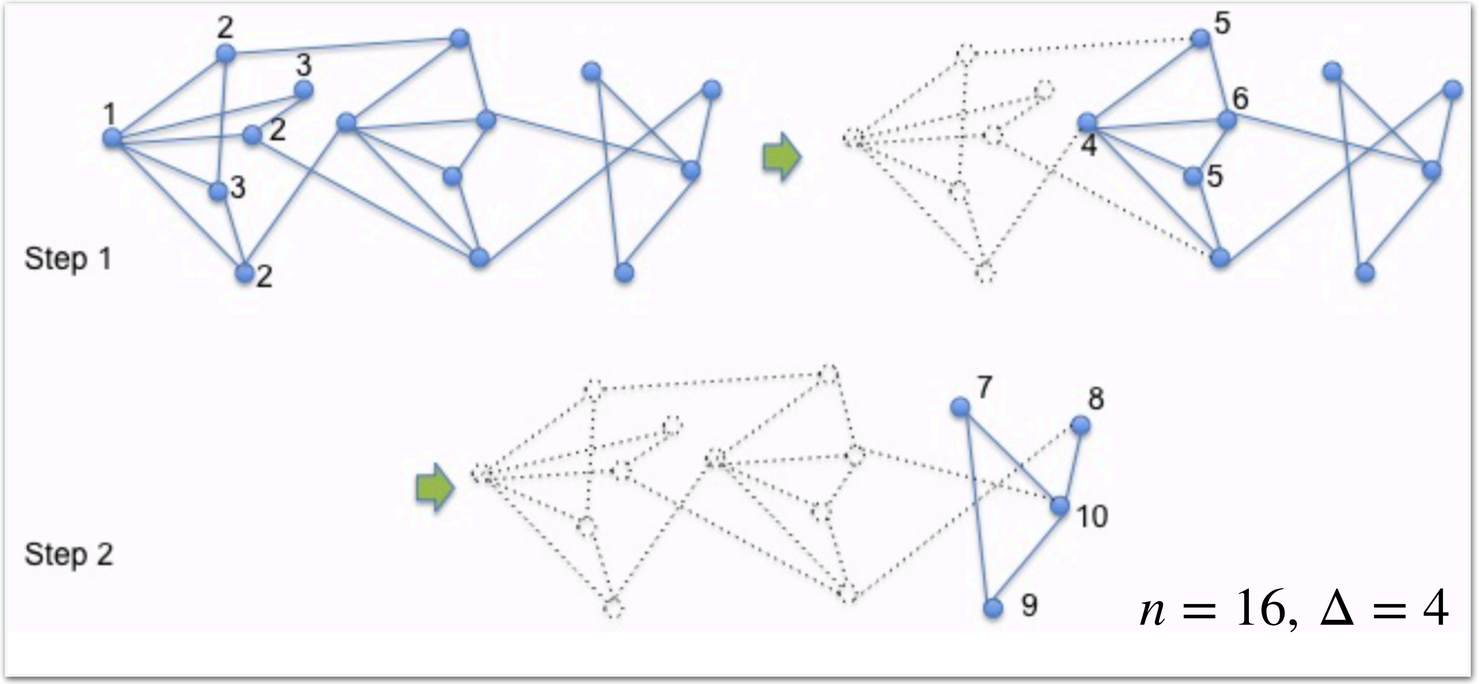
i) **2Col** is easy.
ii) If G has max-degree Δ, it is easy to color G with Δ+1 colors.
iii) 3-vs.-n coloring is easy.

Intuition:
graph with more higher degree vertices is easier!

1. When a node's degree $d \geq \Delta = n^{0.5}$, color it and its neighborhood well*, delete these SAT vertices.

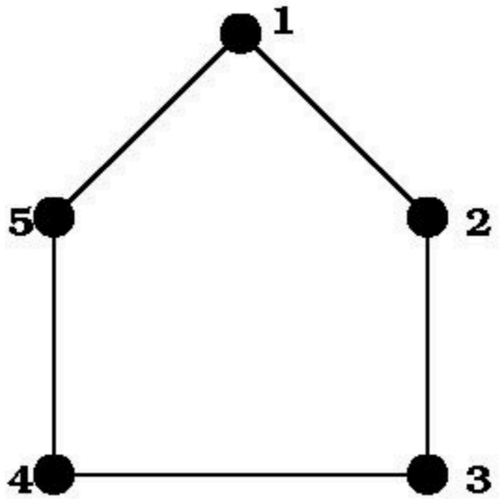
2. Repeat 1 until only lower-degree nodes left.

3. Use the greedy algorithm to color the rest.



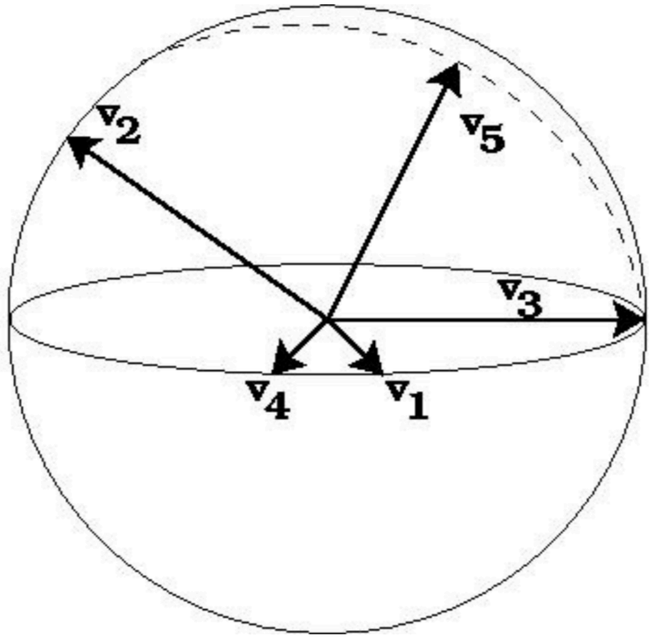
* YES, it's easy — that's a 3-colorable graph!

Vector Programming (VP) Formulation for Graph Coloring



just a 5-cycle

An embedding



Vector Program

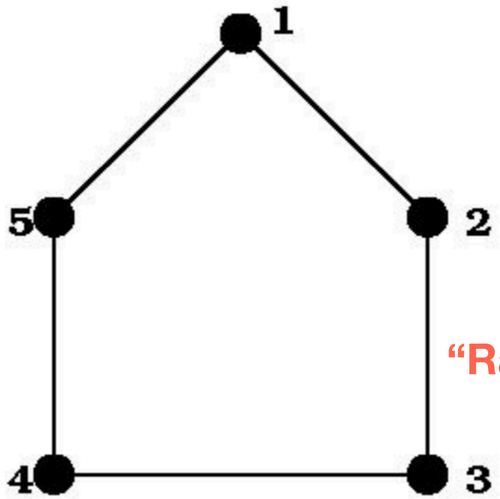
minimize λ

subject to $\langle v_i, v_j \rangle \leq \lambda, \forall (i,j) \in E,$

$\langle v_i, v_i \rangle = 1, \forall i \in V,$

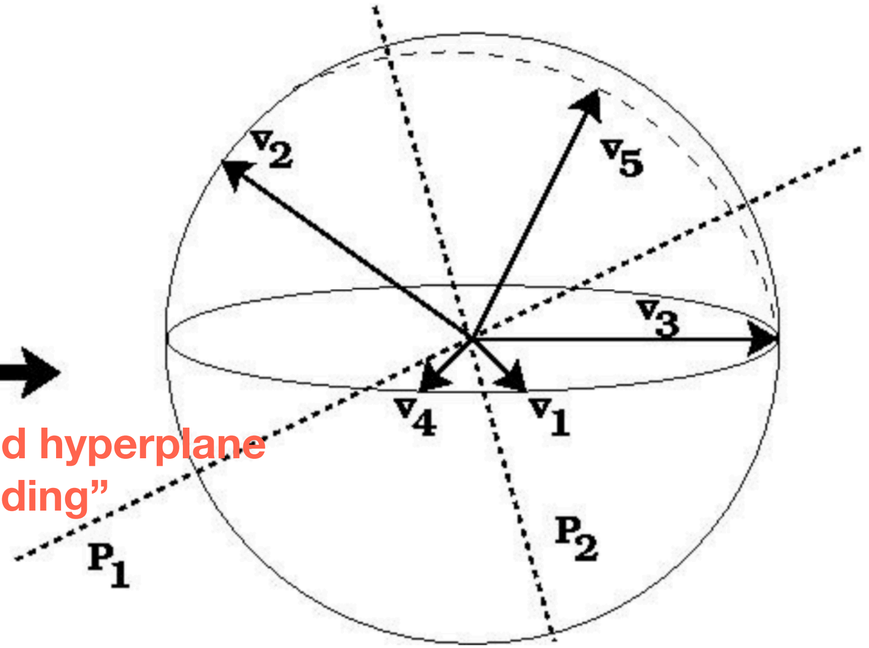
$v_i \in R^n, \forall i \in V.$

Vector Programming (VP) Formulation for Graph Coloring



just a 5-cycle

"Randomized hyperplane rounding"



Vector Program

minimize λ

subject to $\langle v_i, v_j \rangle \leq \lambda, \forall (i,j) \in E,$

$\langle v_i, v_i \rangle = 1, \forall i \in V,$

$v_i \in R^n, \forall i \in V.$

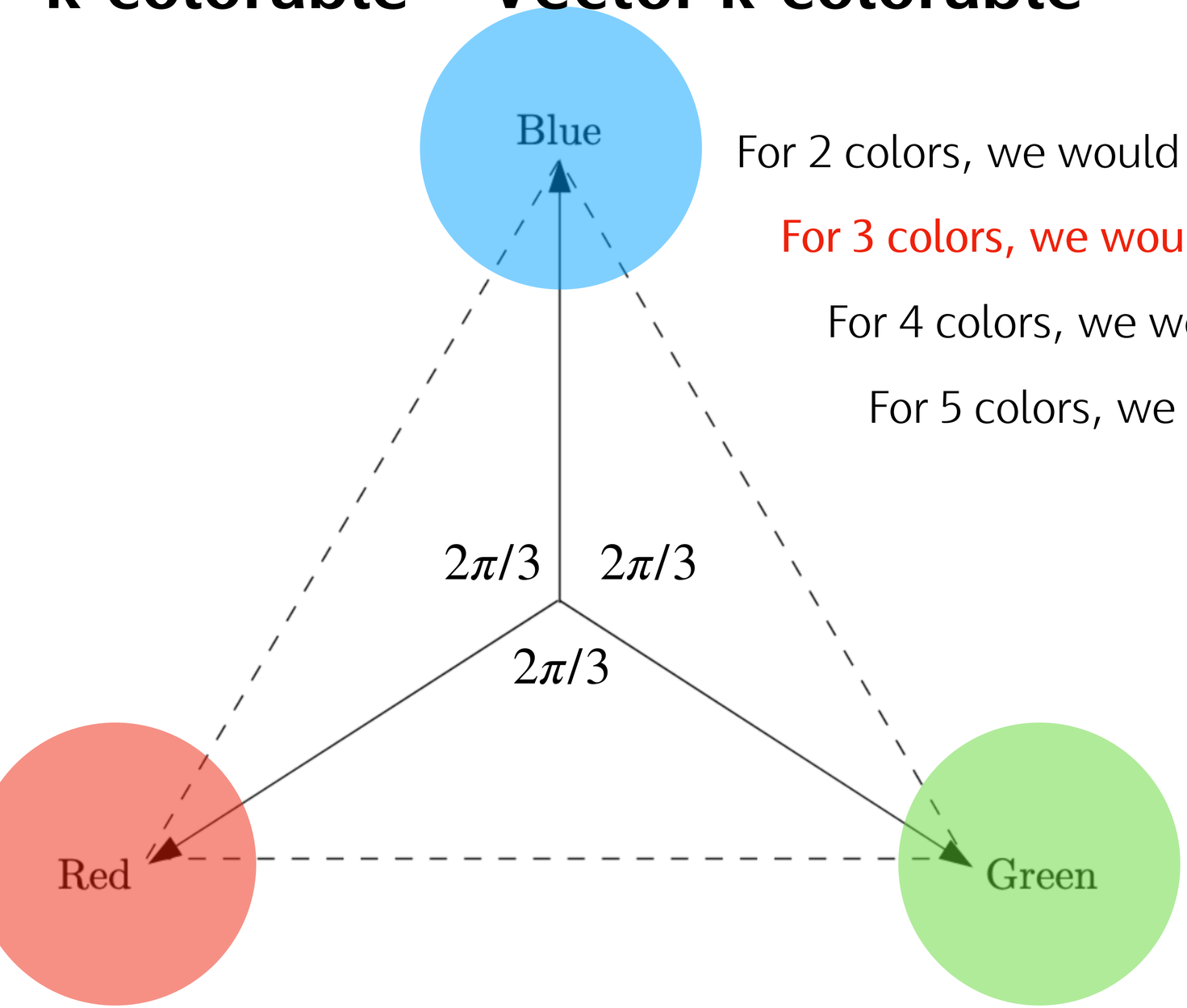
Question 1: Feasible solution to the VP == legal coloring for the graph ??

Not really. But we can recursively remove the satisfied subset, and the remaining graph is still 3-colorable.

Question 2: With a promised 3-coloring, what is an upper bound for λ^* ?

We'll show $\lambda^* \leq -\frac{1}{2}$.

k-colorable = Vector k-colorable



For 2 colors, we would have 2 vectors in 1D, -1 dot product.

For 3 colors, we would have 3 vectors in 2D, -1/2 dot product.

For 4 colors, we would have 4 vectors in 3D, -1/3 dot product.

For 5 colors, we would have 5 vectors in 4D, -1/4 dot product.

Vector Programming (VP) is equivalent to semidefinite program (SDP)

Vector Program (also SDP)

minimize $-1/2$

subject to $\langle v_i, v_j \rangle \leq -1/2, \forall (i,j) \in E,$

$\langle v_i, v_i \rangle = 1, \forall i \in V,$

$v_i \in R^n, \forall i \in V.$

"Objective function is a constant."

This is a feasibility problem.

The corresponding SDP

minimize $0 * \mathbf{Y} - 1/2$

subject to $\mathbf{Y} = [y_{ij}] = \langle v_i, v_j \rangle \succeq 0,$ **Positive semidefinite**

$y_{ii} = 1, \forall i \in V.$

For any err > 0, SDP can be solved within an addictive error of err, in poly-time in n and log(1/err), using the ellipsoid algorithm.

* SDP can also be solved efficiently by "interior point methods"
* Currently, SDPs with 1000's of variables are solvable.

Randomized Rounding: converting from SDP solution to coloring

Goal: avoid using too many colors + avoid introducing monochromatic edges

The independent set

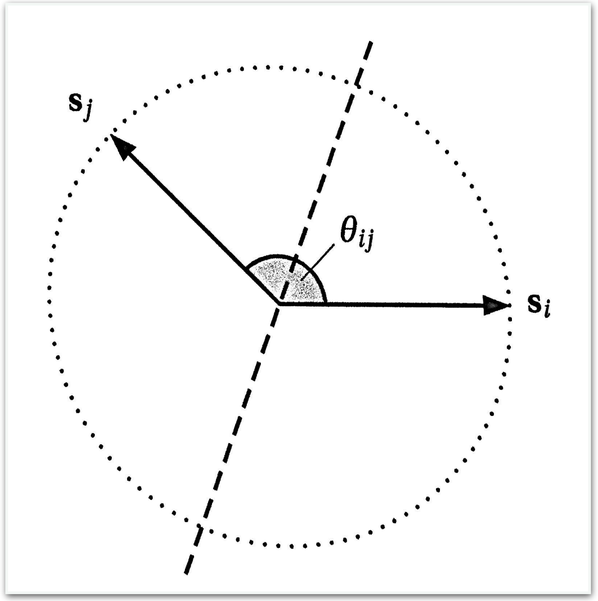
The random Gaussian (normal) vector

A to be chosen threshold

$$S(\epsilon) = \{v \mid \overrightarrow{g} \cdot \overrightarrow{v} \geq \epsilon\}$$

The vertex to include in the set

The vertex's embedding vector



Randomized Rounding: converting from SDP solution to coloring

Goal: avoid using too many colors + avoid introducing monochromatic edges

- Idea: Color the independent set with one color, never use that color again, delete that subset, and the remaining graph is still 3-colorable.
- Repeat until less than **** vertices, then just use **** new colors.
- The **** is dependent on Δ (max degree) & the details of the rounding.

- Theorem: There is a poly-time algorithm that produces $\tilde{O}\left(\Delta^{1/3}\sqrt{\ln \Delta}\right)$ coloring for a 3-colorable graph. "With high probability"

Summary of the Independent-Set-Coloring subroutine

1. Solve SDP.
2. Pick a random vector r.
3. Pick a set $S(\epsilon)$ which is ϵ-close to r.
4. Let $S'(\epsilon) \subseteq S(\epsilon)$ be vertices with degree zero in $S(\epsilon)$.
5. Remove the vertices in the independent set $S'(\epsilon)$ from the graph.
6. Repeat from Step 2 while the graph is non-empty.

Generally, while $\Delta \geq n^{3/4}$, do `Wigderson`.

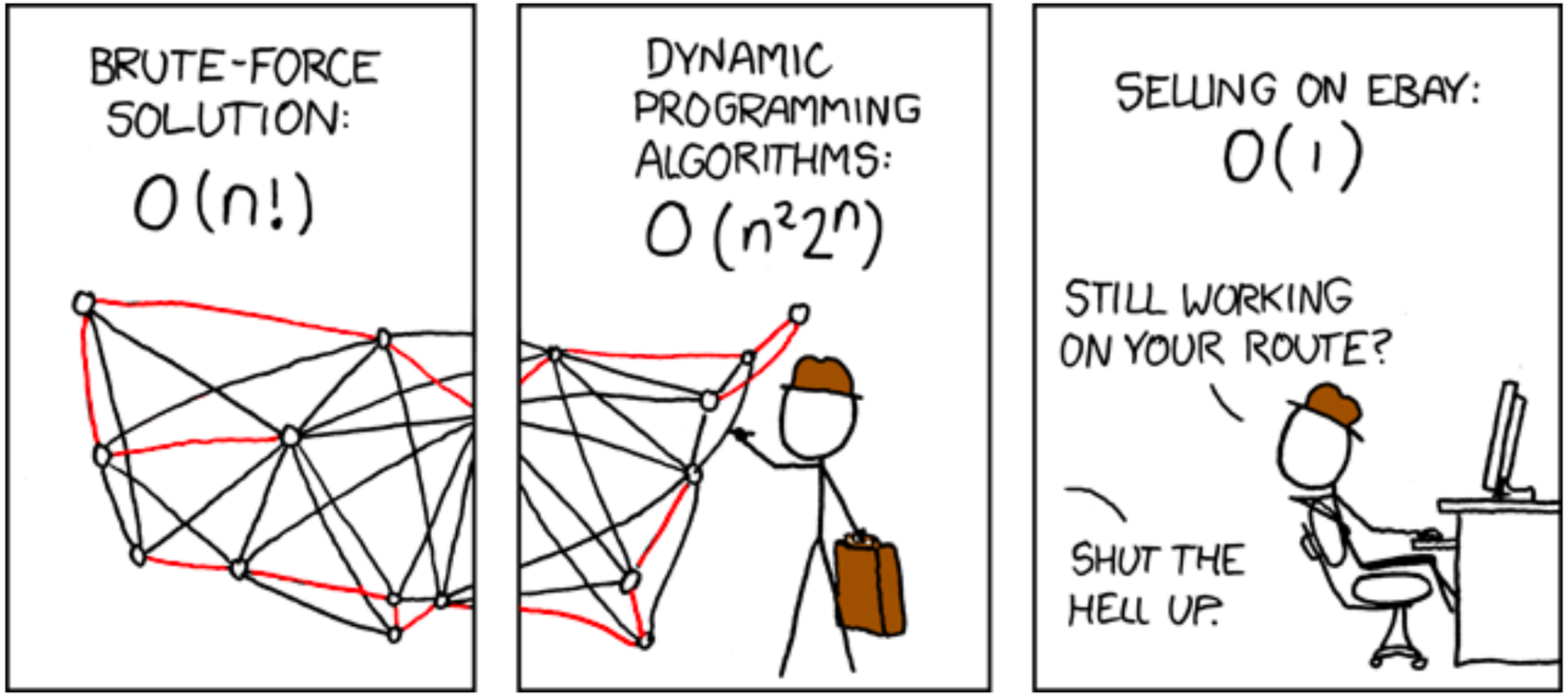
When $\Delta < n^{3/4}$, color the rest via `KMS` (or called `Independent-Set-Coloring`).

We can do the loop no more than $n^{1/4}$ times, $O(n^{1/4})$ colors.

Total $\tilde{O}\left(n^{1/4}\right)$ colors.

Intuitions for a Faster Algorithm

- Improved rounding scheme
- More clever SDP relaxation



TSP in the modern age. https://xkcd.com/399/

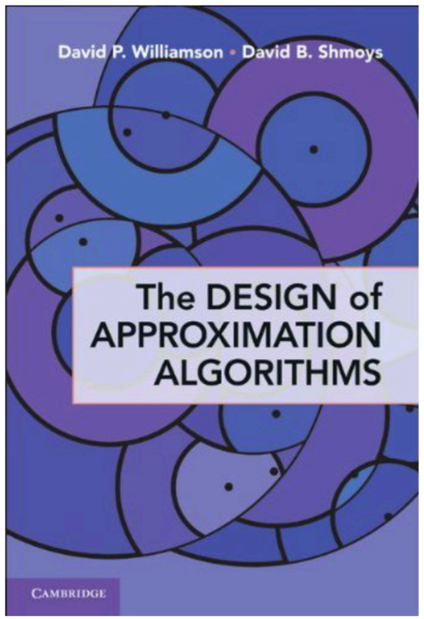
Conclusion

- Today we introduced intuitions for VP, SDP, and random hyperplane rounding.
- "Coloring 3-colorable graphs" is one of the 10 open problems in David P. Williamson's book.
- Alternative combinatorial & SDP approach improves its upper bound.

Interesting future work:

— What graph structure characterizes a "hard case" for these algorithms?

— How can we generate "random colorable graphs" that enable such numerical experiments?



Electronic version at www.designofapproxalgs.com

Avi Wigderson (b. 1956)

Interestingly, the first known approximation algorithm to coloring a 3-colorable graph is Avi Wigderson's Master's Thesis work.

He is awarded the 2019 Knuth Prize for his work that revolutionized our understanding of randomness in computation.

Thanks!