Coloring 3-colorable Graphs

Pawan Kumar Aurora
Advised by Prof. Shashank K Mehta

Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

State of the Art Seminar

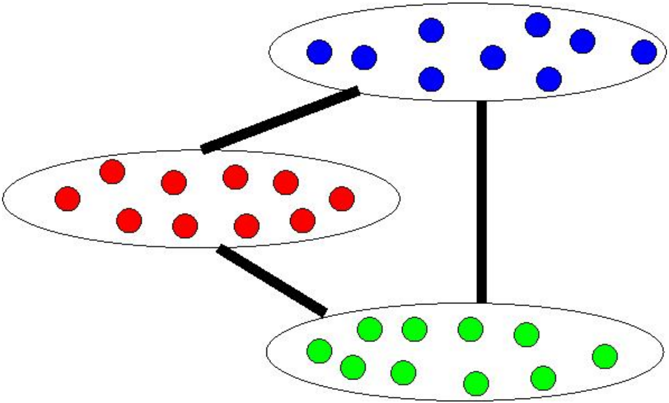
k-coloring

- A mapping $f : V(G) \to \{1, \ldots, k\}$ s.t. $f(u) \neq f(v)$ if $(u, v) \in E(G)$
- The chromatic number of a graph is the smallest k s.t. G can be k-colored
- It is NP-Hard to color a graph using optimal number of colors [1]
- The same is true also for graphs of constant chromatic number atleast 3

$k = 3$

- We will focus on 3-colorable graphs
- Objective: to color such a graph using as few colors as possible
- It is NP-Hard to color such a graph using 4 colors [2]
- Nothing better in terms of lower bounds is known
- Best upper bounds of the order of $|V|^{\epsilon}$ for $\epsilon > 0$
- Gap is HUGE

3-colorable Graphs



Applications

- Compiler Optimization: Assigning variables to registers
- Scheduling: Assigning jobs to time slots

Wigderson's Algorithm [3]

- Based on the following facts:
  1. The subgraph induced by the neighborhood of any vertex is 2-colorable
  2. 2-coloring is polynomial time solvable
  3. $\Delta + 1$ colors suffice to color any graph having maximum degree Δ
- Using facts 1 and 2, 2-color $N(v)$ for a vertex v having $deg(v) \geq \lceil \sqrt{n} \rceil$; remove colored vertices and iterate
- The remaining graph has $\Delta < \lceil \sqrt{n} \rceil$; color it using $\lceil \sqrt{n} \rceil$ colors using fact 3
- Total number of colors used: $O(\sqrt{n})$

Blum's Algorithm [4]

- Consider the following recurrence for some $\epsilon > 0$:
  $C(n) \leq 2 + C(n - \epsilon n/f(n))$
- Solving first for n' in the range $[n/2, n]$ we get:
  $C(n) \leq 2f(n)/\epsilon + C(n/2)$
- Solving the above recurrence gives $C(n) = O(f(n))$
- Repeatedly finding a 2-colorable set of size $\epsilon n/f(n)$ gives $O(f(n))$-coloring

Blum's Algorithm: How to get a set of desired size?

- Either directly find a set of desired size
  - For e.g. $N(v)$ for a vertex v having degree atleast $n/f(n)$
- Or combine several small sets to get a set of desired size
  - Find a 2-colorable set S having $|N(S)| \leq f(n)|S|$
    - For e.g. $N(v)$ for a vertex v having degree atmost $f(n)$
  - Remove both S and $N(S)$ from the graph while collecting S in a bucket
  - Repeat until the graph is reduced to less than half its original size
  - The bucket contains a 2-colorable set of size $\Omega(n/f(n))$

Blum's Algorithm: A simple Case

- For all pairs of vertices $u, v \in V(G)$, consider the 2-colorable set $S = N(u) \cap N(v)$
- For the case where $|S| \geq \frac{n}{f(n)^2}$, one of the following always holds:
  1. $|N(S)| \leq n/f(n)$ (same as $f(n)|S|$) $\Rightarrow$ Collect S
  2. $|N(S)| > n/f(n)$ and $N(S)$ is 2-colorable $\Rightarrow$ $N(S)$ is the desired set
  3. $|N(S)| > n/f(n)$ and $N(S)$ is not 2-colorable $\Rightarrow$ Next slide

Blum's Algorithm: $|N(S)| > \frac{n}{f(n)}$ and $N(S)$ is not 2-colorable

- The first condition is inconsequential, the second condition alone is enough
- $N(S)$ is not 2-colorable $\Rightarrow$ S is not monochromatic $\Rightarrow$ u and v belong to the same color class
- Merge u and v to w:
  $V = V - \{u, v\} \cup \{w\}$, $N(w) = N(u) \cup N(v)$
- Results in a graph having one less vertex without using any new color

Blum's Algorithm: Input Graph

- We can assume the following about our input graph:
  1. It has minimum degree atleast $f(n)$
  2. It has maximum degree atmost $n/f(n)$
  3. No two vertices share more than $n/f(n)^2$ neighbors
- The above can be enforced for any arbitrary $f(n)$
- However, the value of $f(n)$ is determined by how best we can handle the above graph

Blum's Algorithm: High Level Idea

- Remember that our aim is still to find a 2-colorable set of size $\Omega(n/f(n))$
- We will find a set that contains a large enough independent set
- Using an approximate vertex cover algorithm we will extract an independent set
- The size of the independent set obtained will determine the value of $f(n)$

Blum's Algorithm: Using Vertex Cover

- I is an independent set in a graph $G \Rightarrow V(G) - I$ is a vertex cover in G and vice versa
- An algorithm for vertex cover can be used to find an independent set
- Since both the problems are NP-Hard, we can only hope for an approximate result
- If VC is an optimal vertex cover in G, then we can find a vertex cover of size atmost $\left(2 - \frac{\log \log |V|}{2 \log |V|}\right) |VC|$ in G [5]

Blum's Algorithm: Using Vertex Cover

- $|I| \geq \frac{1}{2}\left(1 - \frac{1}{\log|T|}\right)|T| \Rightarrow |VC| \leq \frac{1}{2}\left(1 + \frac{1}{\log|T|}\right)|T|$
- We find a vertex cover of size atmost $\frac{1}{2}\left(1 + \frac{1}{\log|T|}\right)\left(2 - \frac{\log\log|T|}{2\log|T|}\right)|T| < \left[1 - \Omega\left(\frac{1}{\log|T|}\right)\right]|T|$
- That gives an independent set of size $\Omega\left(\frac{|T|}{\log|T|}\right)$
- **Note:** It would be useless to find a subset T that contains an independent set of size atleast $\frac{1}{2+\epsilon}\left(1 - \frac{1}{\log|T|}\right)|T|$

Blum's Algorithm: Motivation

- Consider three sets **red**, **blue** and **green** of roughly the same size
- For all pairs of vertices in different sets, add an edge with probability p
- The resulting graph is 3-colorable and has all the edges distributed uniformly at random
- For a vertex $v \in$ **red**, $N(v)$ is nearly half **blue** and half **green**
- So $N(N(v))$ is almost half **red**

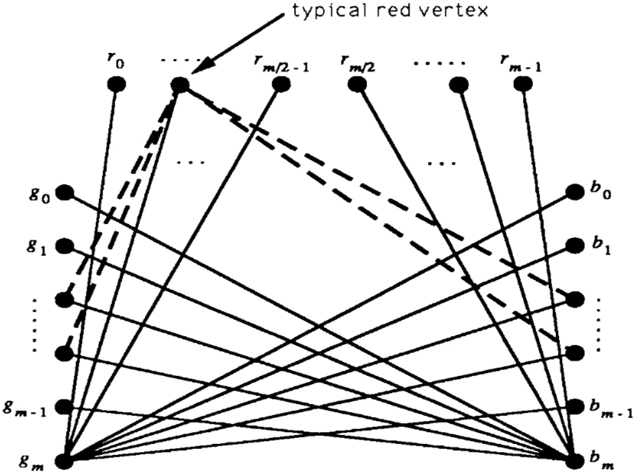
Blum's Algorithm: Reality

- But worst-case graphs are not random
- Can we atleast find a subset of $N(N(v))$ for some v that contains an independent set nearly half its size?
  - The answer is YES
- This exercise is useful only when the subset size is sufficiently larger than $f(n)$
  - Every vertex has a neighborhood of size atleast $f(n)$ which trivially contains an independent set atleast half its size

Blum's Algorithm: Finding desired subset-Step 1

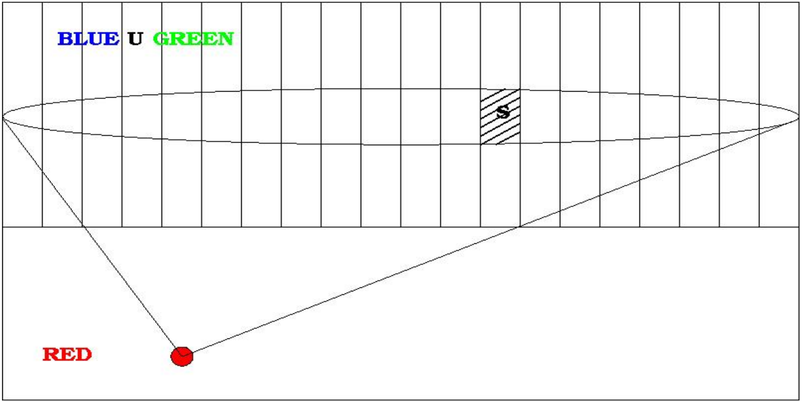
- Consider a vertex $v \in$ **red**
- Find a subset S of $N(v)$ s.t. nearly half of the edges incident on S enter into **red**
- Let **red** be the color with the most edges incident
  - Implies $D_{red}(\textbf{blue} \cup \textbf{green}) \geq \frac{1}{2} D(\textbf{blue} \cup \textbf{green})$
- However, it is not true that $D_{red}(N(v)) \geq \frac{1}{2} D(N(v))$ for any $v \in$ **red**
  - Vertices can have wildly varying degrees
- Solution lies in restricting the vertex degrees extremely tightly

Blum's Algorithm: Counter-example



- $D(\textbf{red}) = 5m$, $D(\textbf{green}) = D(\textbf{blue}) = (4 + \frac{1}{2})m$
- $\forall\ v \in \textbf{red}$: $D_{red}(N(v)) = 8 + \frac{m}{2}$, $D_{V-red}(N(v)) = 4 + m$
- $D(N(v)) = 12 + \frac{3}{2}m$. So, $D_{red}(N(v)) \approx \frac{1}{3}D(N(v))$

Blum's Algorithm: First Neighborhood

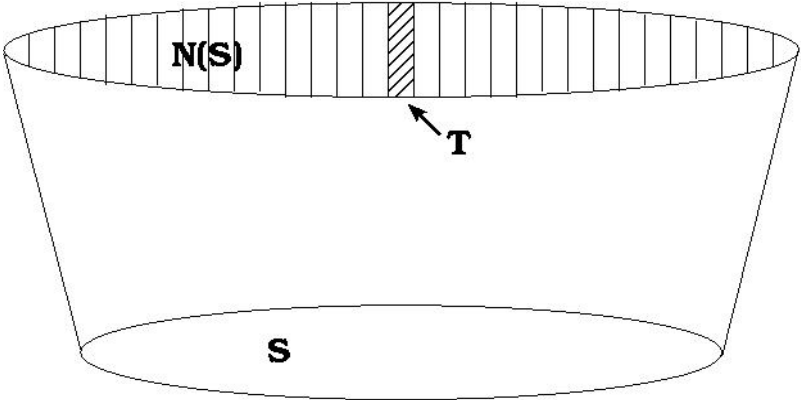


- $S = N(v) \cap \{v \in V \mid d(v) \in [(1+\delta)^j, (1+\delta)^{j+1})\}$, $\delta = \frac{1}{5\log n}$
- For some j, $D_{red}(S) \approx \frac{1}{2} D(S)$

Blum's Algorithm: Finding desired subset-Step 2

- Having obtained the set S, we now look at $N(S)$
- Even though $D_{red}(S) \approx \frac{1}{2}D(S)$, it is possible that many of the edges are incident on a few **red** vertices
- The same trick is used again and this time $N(S)$ is partitioned into bins
- Each bin has vertices lying in a close range in terms of their degree into S
- One of these bins is our desired subset

Blum's Algorithm: Second Neighborhood



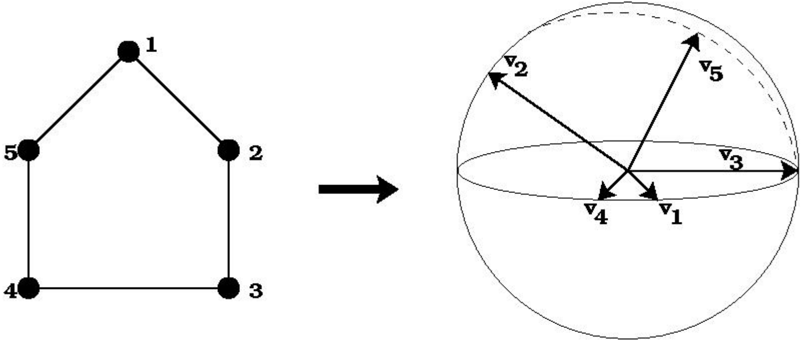
- $T = \{v \in N(S) \mid d_S(v) \in [(1+\delta)^i, (1+\delta)^{i+1})\}$
- For some i, $|T| = \tilde{\Omega}\left(\frac{f(n)^4}{n}\right)$; $\frac{|T \cap \textbf{red}|}{|T|} \geq \frac{1}{2}\left(1 - \frac{1}{\log n}\right)$

Blum's Algorithm: So what is our $f(n)$?

- Applying vertex cover to the set T, we get an independent set of size $\Omega\left(\frac{|T|}{\log|T|}\right) = \tilde{\Omega}\left(\frac{f(n)^4}{n}\right)$
- In order to be useful, we need $\tilde{\Omega}\left(\frac{f(n)^4}{n}\right) = \Omega\left(\frac{n}{f(n)}\right)$
- That gives an $\tilde{O}\left(n^{0.4}\right)$-coloring
- An $\tilde{O}\left(n^{0.375}\right)$-coloring can be obtained by handling certain dense regions differently

Karger-Motwani-Sudan's Algorithm [6]: High Level Idea

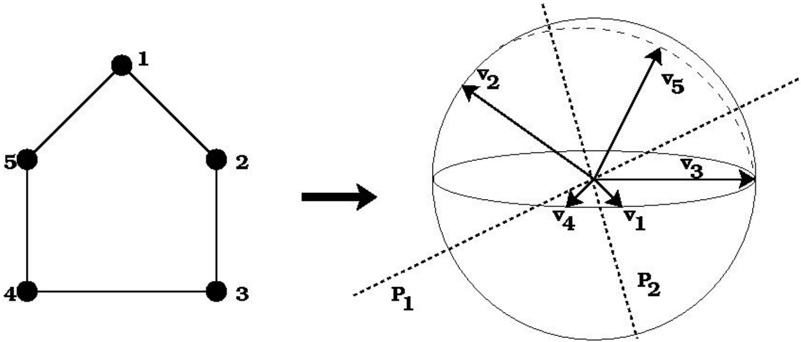
Consider the following embedding of a 5-cycle on the surface of a unit sphere:



Vertices are mapped to points on the unit sphere in such a manner that adjacent vertices get mapped to far away points

KMS Algorithm: High Level Idea

Consider cutting the unit sphere via the randomly chosen planes P_1 and P_2:



That divides the vertices into four groups. Giving each group a distinct color we get a legal approximate coloring.

KMS Algorithm: Finding the desired Embedding

- Let $v_1, v_2, \ldots, v_n$ be unit vectors in $\Re^n$
- Vector v_i corresponds to vertex i
- Minimizing $\langle v_i, v_j \rangle$ will keep v_i and v_j far apart
- Consider the following optimization problem:

$$
\begin{array}{rll}
\text{minimize} & \alpha & \\
\text{subject to} & \langle v_i, v_j \rangle \leq \alpha & \text{if } (i,j) \in E(G) \\
& \langle v_i, v_i \rangle = 1 & \\
& v_i \in \Re^n. &
\end{array}
$$

- An optimal solution to the above program will give us the desired embedding

KMS Algorithm: Finding the desired Embedding

- Unfortunately the program cannot be solved as is
  - Good news is there is a way around
- Consider a $n \times n$ symmetric positive semidefinite matrix M
- **Fact:** M can be decomposed into UU^T
- From the above, $M[i,j] = \langle u_i, u_j \rangle$ where $u_i = U[i,:]$ and $u_j = U[j,:]$

KMS Algorithm: Finding the desired Embedding

- Consider the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \alpha \\ \text{where} \quad & \{m_{ij}\} \text{ is positive semidefinite} \\ \text{subject to} \quad & m_{ij} \leq \alpha \quad \text{if } (i,j) \in E(G) \\ & m_{ij} = m_{ji} \\ & m_{ii} = 1. \end{aligned}$$

- An optimal solution to the above program can still give us the desired embedding
- And the above program can be solved efficiently

KMS Algorithm: Bounding the Optimal Value of α

- Consider the following k vectors in $\Re^n$:
  - Each vector has 0 in the last $n - k$ positions
  - Vector i has $-\sqrt{\frac{k-1}{k}}$ in the ith position and $\frac{1}{\sqrt{k(k-1)}}$ in the remaining $k - 1$ positions
- Clearly each vector has unit length and inner product of any two distinct vectors is $-\frac{1}{k-1}$
- For a k-colorable graph, the k colors coincide with the k vectors defined above
- So we have $\alpha \leq -\frac{1}{k-1}$ for a k-colorable graph

KMS Algorithm: Obtaining a coloring

- For a 3-colorable graph, the optimal value is atmost $-\frac{1}{2}$
- Vectors corresponding to adjacent vertices are atleast $2\pi/3$ radians (120 degrees) apart
- Using this fact, we can obtain a coloring via the following two methods:
  1. Hyperplane Partitions
  2. Vector Projections

KMS Algorithm: Hyperplane Partitions

- Random hyperplanes passing through the origin are used to cut the n-dimensional unit sphere
- Using h hyperplanes, we can obtain 2^h distinct regions
- Associate a distinct color with each region, giving each vertex the color of the region containing its vector
- It is possible that two adjacent vertices are given the same color (though with small probability)
- Legally colored vertices are removed and the algorithm is repeated on the graph remaining

KMS Algorithm: Hyperplane Partitions

- Given two vectors at an angle of θ, the probability that they are separated by a random hyperplane is θ/π
- We have $\theta \geq 2\pi/3$ as the angle between the vectors corresponding to the endpoints of an edge
- We say that an edge is cut by a hyperplane if these vectors are separated by the hyperplane
- So the probability of an edge being cut is atleast $2/3$
- A cut edge implies its endpoints belonging to different regions and hence getting different colors

KMS Algorithm: Hyperplane Partitions

- Pick $2 + \lceil \log_3 \Delta \rceil$ random hyperplanes independently
- Probability that an edge is not cut by any of these is atmost $(1 - 2/3)^{2+\lceil \log_3 \Delta \rceil} \leq 1/9\Delta$
- Let m' be the number of uncut edges
- $E[m'] \leq m/9\Delta \leq n/18 < n/8$, since $m \leq n\Delta/2$
- By Markov's Inequality, $Pr\{m' > n/4\} \leq 1/2$
- Thus, with probability atleast $1/2$ we have atmost $n/4$ uncut (monochromatic) edges

KMS Algorithm: Hyperplane Partitions

- Deleting one endpoint of each of the $n/4$ uncut edges leaves a set of atleast $3n/4$ legally colored vertices
- The number of colors used is $2^{2+\lceil \log_3 \Delta \rceil} = O(\Delta^{\log_3 2})$
- That translates to $O(n^{0.387})$ colors using Wigderson's technique
- Iterating on the deleted vertices we get an $O(n^{0.387})$-coloring
- No improvement over Blum's $O(n^{0.375})$-coloring

KMS Algorithm: Vector Projections

- Fix a parameter c and choose a random n-dimensional vector r
- Compute a subset S of vertices i with $\langle v_i, r \rangle \geq c$
- Let the subgraph induced on S have n' vertices and m' edges
- Delete one endpoint of each edge to leave an independent set on $n' - m'$ vertices
- For sufficiently large c, $n' \gg m'$ and we get an independent set of size roughly n'

KMS Algorithm: Vector Projections

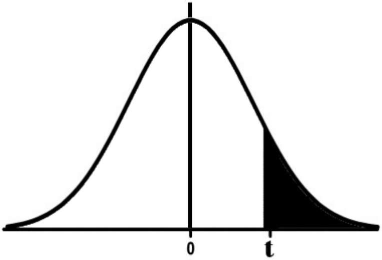
- $r = (r_1, \ldots, r_n)$, where r_i are independent random variables having the standard normal distribution
- The distribution function for r has density

$$f(y_1, \ldots, y_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-y_i^2/2} = \frac{1}{(2\pi)^{n/2}} e^{-\frac{1}{2}\sum_i y_i^2}$$

- Note that the density function depends only on the distance of the point from the origin
- Therefore the distribution of r is spherically symmetric

KMS Algorithm: Vector Projections

- For any unit vector $u \in \Re^n$, $\langle u, r \rangle$ is distributed according to the standard normal distribution



Shaded area corresponds to $N(t) = P[\langle u, r \rangle \geq t]$

KMS Algorithm: Vector Projections

- We have $P[\langle v_i, r\rangle \geq c] = N(c)$, so $E[n'] = nN(c)$
- $P[\langle v_1, r\rangle \geq c \text{ and } \langle v_2, r\rangle \geq c] \leq P[\langle (v_1 + v_2), r\rangle \geq 2c]$
$$= P\left[\langle \frac{v_1+v_2}{\|v_1+v_2\|}, r\rangle \geq \frac{2c}{\|v_1+v_2\|}\right]$$
$$= N\left(\frac{2c}{\|v_1+v_2\|}\right).$$
- $\|v_1 + v_2\| = \sqrt{v_1^2 + v_2^2 + 2\langle v_1, v_2\rangle} \leq \sqrt{2 - 2/2} = 1$
- $P[\langle v_1, r\rangle \geq c \text{ and } \langle v_2, r\rangle \geq c] \leq N(2c)$
- So, $E[m'] \leq mN(2c) \leq n\Delta N(2c)/2$ (Δ is max degree)
- Thus, $E[n' - m'] \geq nN(c) - n\Delta N(2c)/2$

KMS Algorithm: Vector Projections

- For every $x > 0$, $\phi(x)\left(\frac{1}{x} - \frac{1}{x^3}\right) < N(x) < \phi(x).\frac{1}{x}$
- From the above, we have $\frac{N(c)}{N(2c)} \geq 2\left(1 - \frac{1}{c^2}\right)e^{3c^2/2}$
- Solving for c so that $\Delta N(2c) < N(c)$, we get $c = \sqrt{\frac{2}{3}\ln\Delta}$
- For the above value of c, $E[n' - m'] \geq \tilde{\Omega}\left(\frac{n}{\Delta^{1/3}}\right)$
- Repeatedly coloring and removing independent sets of the above size gives an $\tilde{O}(\Delta^{1/3})$-coloring
- Using Wigderson's technique we get an $\tilde{O}(n^{0.25})$-coloring

Blum & Karger's Algorithm [7]

- The ideas of Blum and KMS are combined to get an $\tilde{O}(n^{3/14})$-coloring of a 3-colorable graph
- Similar in spirit to Wigderson's algorithm
- Blum's coloring tools are used to color a graph with large average degree
- When the remaining graph has a small average degree, KMS ideas are used to extract an independent set of reasonable size

Blum & Karger's Algorithm

- Consider a graph with average degree atmost $cn^{9/14}$
- So atleast half the vertices in the graph have degree less than $2cn^{9/14}$
- The subgraph induced by those vertices has maximum degree atmost $2cn^{9/14}$
- Using KMS algorithm, we can color the subgraph with $\tilde{O}(n^{3/14})$ colors
- From the coloring we can find an independent set of size $\tilde{\Omega}(n^{11/14})$
- Using the independent set we can make progress towards an $\tilde{O}(n^{3/14})$-coloring of the original graph

Directions for Future Work

- One idea constant in all the algorithms is finding a large set that can be colored using a constant number of colors
- Taking this idea forward, we would like to explore the possibility of finding large planar induced sub-graphs
- **Fact:** Planar graphs are 4-colorable [8]
- An interesting problem in its own right
- One Approach:
  - The subgraph induced by the vertices that lie along the diameter is clearly planar
  - Planarity testing is polynomial time solvable
  - Can we use the above facts to obtain a provably large induced planar subgraph?

Directions for Future Work

- There exist graphs having chromatic number atleast $n^{\Omega(1)}$ that can be embedded on the unit sphere s.t. $\langle v_i, v_j \rangle \leq -\frac{1}{2}$ $\forall\ (i,j) \in E(G)$
- So, having obtained an embedding as above, it is not possible to guarantee a coloring with $n^{o(1)}$ colors
- Can we add more constraints that are not satisfied by the class of graphs mentioned above?
- In that case, can we get an $n^{o(1)}$-coloring?

Directions for Future Work

- The graph obtained by removing the feedback vertex set is an induced forest which is 2-colorable
- Can we show that the size of a feedback vertex set in a 3-colorable graph is not too large?
- How about a partial feedback vertex set that removes only the odd cycles?
- Graphs that are both $C3$-free and $C5$-free have $N(v)$ and $N(N(v))$ as independent sets for any vertex v
- How well we can do for such graphs? Can we extend the same to general graphs?
- $C4$-free graphs: $|N(u) \cap N(v)| \leq 1 \; \forall \; u, v \in V(G)$; Using Blum's algorithm we get $\tilde{O}(n^{1/3})$; Can we do better?

References I

M. R. Garey and David S. Johnson.
The complexity of near-optimal graph coloring.
J. ACM, 23(1):43–49, 1976.

Sanjeev Khanna, Nathan Linial, and Shmuel Safra.
On the hardness of approximating the chromatic number.
Combinatorica, 20(3):393–415, 2000.

Avi Wigderson.
Improving the performance guarantee for approximate graph coloring.
J. ACM, 30(4):729–735, 1983.

Avrim Blum.
New approximation algorithms for graph coloring.
J. ACM, 41(3):470–516, 1994.

References II

Reuven Bar-Yehuda and Shimon Even.
A linear-time approximation algorithm for the weighted vertex cover problem.
J. Algorithms, 2(2):198–203, 1981.

David R. Karger, Rajeev Motwani, and Madhu Sudan.
Approximate graph coloring by semidefinite programming.
J. ACM, 45(2):246–265, 1998.

Avrim Blum and David R. Karger.
An $\tilde{O}(n^{3/14})$-coloring algorithm for 3-colorable graphs.
Inf. Process. Lett., 61(1):49–53, 1997.

K. Appel and W. Haken.
Every planar map is four colorable.
Illinois Journal of Mathematics, 21(3):429–490, 1977.