Coloring sparse random k-colorable graphs in polynomial expected time

Julia Böttcher, TU München, Germany

Definitions

$V(G) = \{1, \ldots, n\} =$ vertext set of G.

$\chi(G) =$ chromatic number of G.

Definitions

$V(G) = \{1, \ldots, n\}$ = vertext set of G.

$\chi(G)$ = chromatic number of G.

Algorithm $\mathcal{A}$ runs in polynomial expected running time:

$$\sum_{|G|=n} t_{\mathcal{A}}(G) \cdot \mathbf{P}[G] \quad \text{is polynomial.}$$

($t_{\mathcal{A}}(G)$: running time of $\mathcal{A}$ on input G)

Approximating the chromatic number

Coloring graphs:

THEOREM — HALLDÓRSON 1993

$\chi(G)$ can be approximated within $\frac{n(\log\log n)^2}{(\log n)^3}$.

Approximating the chromatic number

Coloring graphs:

THEOREM HALLDÓRSON 1993

$\chi(G)$ can be approximated within $\frac{n(\log\log n)^2}{(\log n)^3}$.

THEOREM ENGEBRETSEN,HOLMERIN 2003

Approximating $\chi(G)$ within $n^{1-\mathcal{O}\left(1/\sqrt{\log\log n}\right)}$ is hard.

Approximating the chromatic number

Coloring graphs:

THEOREM HALLDÓRSON 1993

$\chi(G)$ can be approximated within $\frac{n(\log\log n)^2}{(\log n)^3}$.

THEOREM ENGEBRETSEN, HOLMERIN 2003

Approximating $\chi(G)$ within $n^{1-\mathcal{O}(1/\sqrt{\log\log n})}$ is hard.

Coloring 3-chromatic graphs:

THEOREM BLUM, KARGER 1997

G can efficiently be colored with $n^{3/14+o(1)}$ colors.

THEOREM KHANNA, LINIAL, SAFRA 1993

4-coloring G is hard.

k-coloring random k-colorable graphs

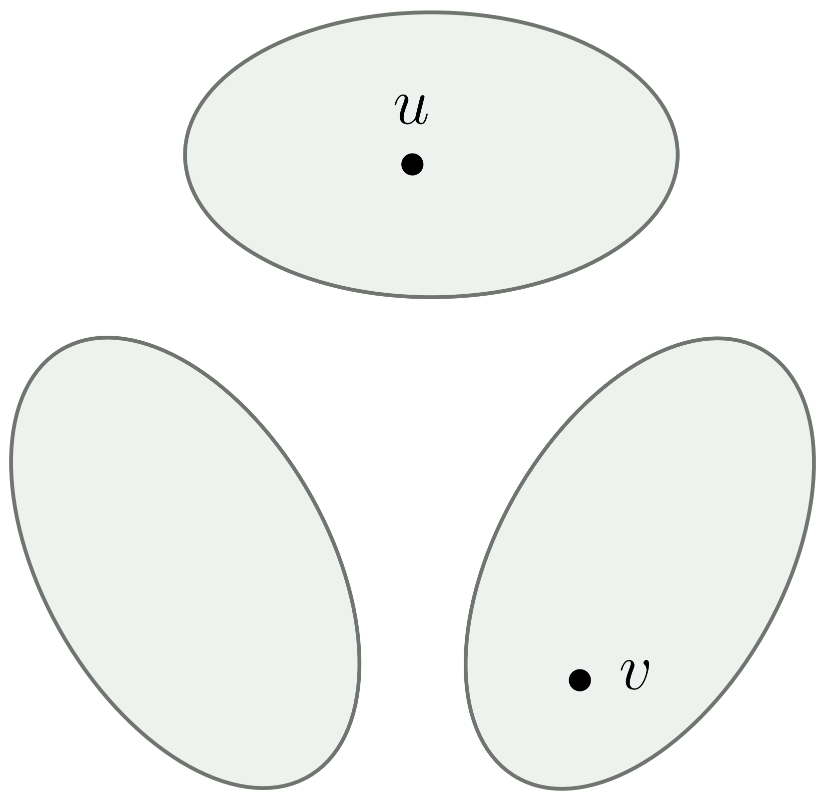
Coloring uniformly distributed k-colorable graphs:

- Algorithms that work w.h.p. KUCERA 1977; TURNER 1988
- In polynomial expected time. DYER AND FRIEZE 1989

k-coloring random k-colorable graphs

Coloring uniformly distributed k-colorable graphs:

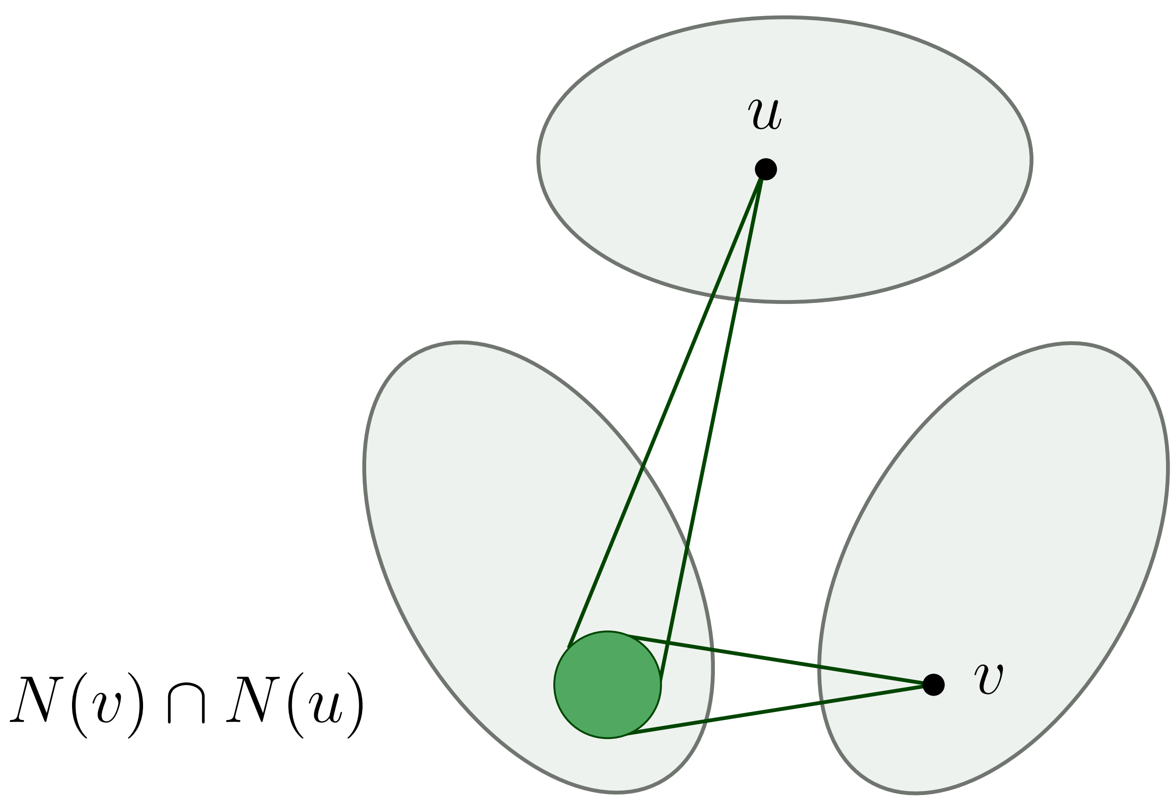
- Algorithms that work w.h.p. — KUCERA 1977; TURNER 1988
- In polynomial expected time. — DYER AND FRIEZE 1989



k-coloring random k-colorable graphs

Coloring uniformly distributed k-colorable graphs:

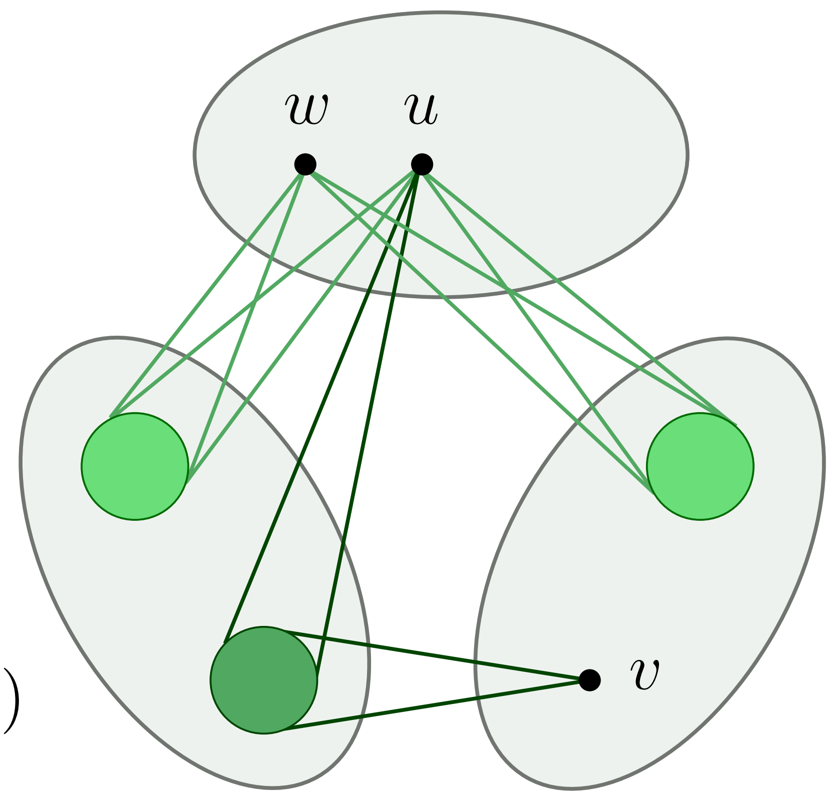
- Algorithms that work w.h.p. — KUCERA 1977; TURNER 1988
- In polynomial expected time. — DYER AND FRIEZE 1989



k-coloring random k-colorable graphs

Coloring uniformly distributed k-colorable graphs:

- Algorithms that work w.h.p. KUCERA 1977; TURNER 1988
- In polynomial expected time. DYER AND FRIEZE 1989

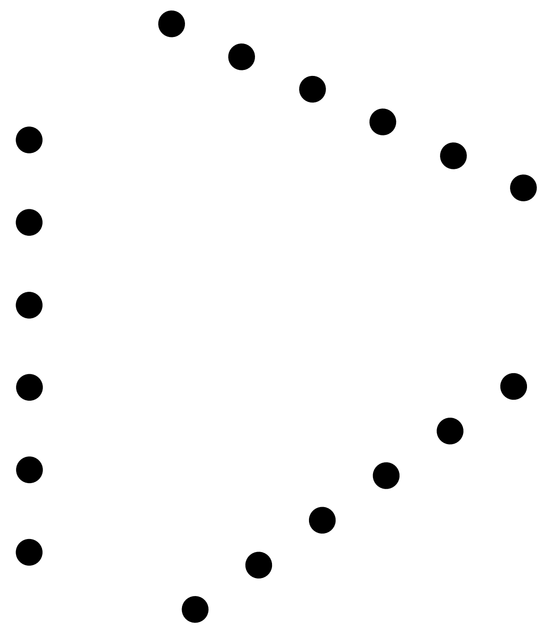


$N(v) \cap N(u)$

The random k-colorable graph $\mathcal{G}_{n,p,k}$

Construct $\mathcal{G}_{n,p,k}$ as follows:

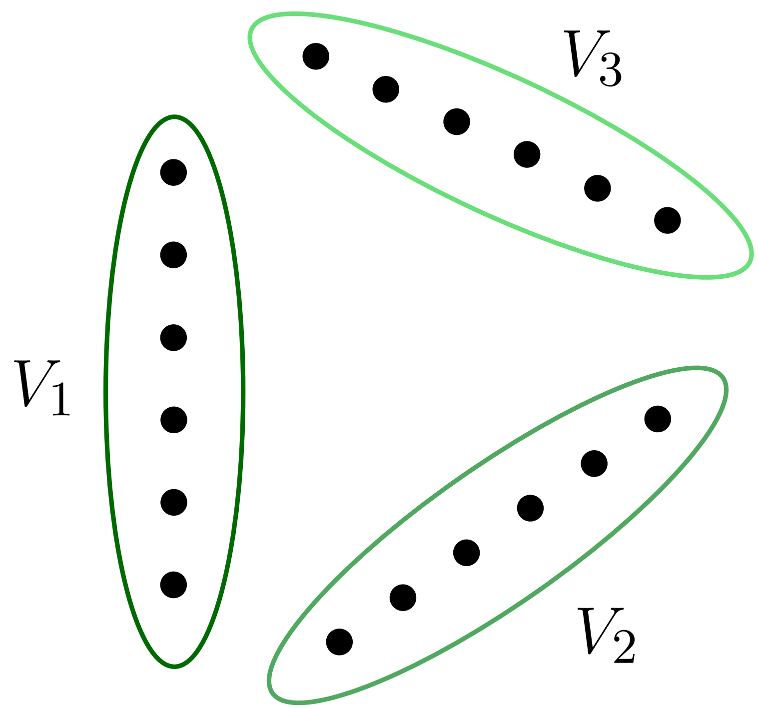
- Partition V into k color classes $V_1, \ldots, V_k$ of equal size.
- For $i \neq j$ insert edges between V_i and V_j with probability $p = d/n$.



The random k-colorable graph $\mathcal{G}_{n,p,k}$

Construct $\mathcal{G}_{n,p,k}$ as follows:

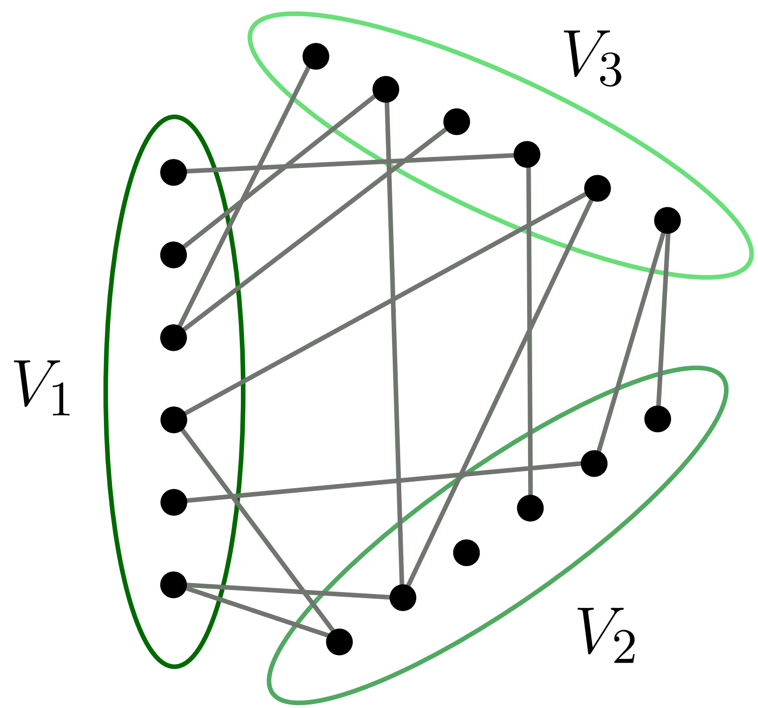
- Partition V into k color classes $V_1, \ldots, V_k$ of equal size.
- For $i \neq j$ insert edges between V_i and V_j with probability $p = d/n$.



The random k-colorable graph $\mathcal{G}_{n,p,k}$

Construct $\mathcal{G}_{n,p,k}$ as follows:

- Partition V into k color classes $V_1, \ldots, V_k$ of equal size.
- For $i \neq j$ insert edges between V_i and V_j with probability $p = d/n$.



k-coloring random k-colorable graphs

Coloring uniformly distributed k-colorable graphs:

- Algorithms that work w.h.p. KUCERA 1977; TURNER 1988
- In polynomial expected time. DYER AND FRIEZE 1989

k-coloring random k-colorable graphs

Coloring uniformly distributed k-colorable graphs:

- Algorithms that work w.h.p. KUCERA 1977; TURNER 1988
- In polynomial expected time. DYER AND FRIEZE 1989

Coloring $\mathcal{G}_{n,p,k}$:

- W.h.p.
  - for $p \geq n^{\epsilon}/n$, BLUM AND SPENCER 1995
  - for $p \geq c/n$. ALON AND KAHALE 1997
- In polynomial expected time
  - for $p \geq n^{\epsilon}/n$, SUBRAMANIAN 2000
  - for $p \geq c \cdot \ln n/n$, COJA-OGHLAN 2004
  - for $p \geq c/n$.

An Algorithm for 3-Coloring $\mathcal{G}_{n,p,3}$

Algorithm 1: COLOR$\mathcal{G}_{n,p,3}$(G)

Input: a graph $\mathcal{G}_{n,p,3} = G = (V, E)$

Output: a valid coloring for $\mathcal{G}_{n,p,3}$

begin

/** The initial phase **/

Construct a coloring that fails on $< \epsilon n$ vertices ;

Refine the initial coloring ;

Repeatedly uncolor vertices having few neighbors of some other color ;

Color the components of uncolored vertices ;

end

An Algorithm for 3-Coloring $\mathcal{G}_{n,p,3}$

Algorithm 1: COLOR$\mathcal{G}_{n,p,3}$(G)

Input: a graph $\mathcal{G}_{n,p,3} = G = (V, E)$

Output: a valid coloring for $\mathcal{G}_{n,p,3}$

begin

/$**$ `The initial phase` $**$/

Construct a coloring that fails on $< \epsilon n$ vertices ;

/$**$ `The iterative recoloring procedure` $**$/

Refine the initial coloring ;

Repeatedly uncolor vertices having few neighbors of some other color ;

Color the components of the uncolored vertices ;

end

An Algorithm for 3-Coloring $\mathcal{G}_{n,p,3}$

Algorithm 1: COLOR$\mathcal{G}_{n,p,3}$(G)

Input: a graph $\mathcal{G}_{n,p,3} = G = (V, E)$

Output: a valid coloring for $\mathcal{G}_{n,p,3}$

begin

```
    /** The initial phase **/
    Construct a coloring that fails on < εn vertices ;
    /** The iterative recoloring procedure **/
    Refine the initial coloring ;
    /** The uncoloring procedure **/
    Repeatedly uncolor vertices having few neighbors of
    some other color ;

    Color the components of the uncolored vertices ;
```

end

An Algorithm for 3-Coloring $\mathcal{G}_{n,p,3}$

Algorithm 1: COLOR$\mathcal{G}_{n,p,3}$(G)

Input: a graph $\mathcal{G}_{n,p,3} = G = (V, E)$

Output: a valid coloring for $\mathcal{G}_{n,p,3}$

begin

/$**$ The initial phase $**$/

Construct a coloring that fails on $< \epsilon n$ vertices ;

/$**$ The iterative recoloring procedure $**$/

Refine the initial coloring ;

/$**$ The uncoloring procedure $**$/

Repeatedly uncolor vertices having few neighbors of some other color ;

/$**$ The extension step $**$/

Color the components of the uncolored vertices ;

end

An Algorithm for 3-Coloring $\mathcal{G}_{n,p,3}$

Algorithm 1: COLOR$\mathcal{G}_{n,p,3}$(G)

Input: a graph $\mathcal{G}_{n,p,3} = G = (V, E)$
Output: a valid coloring for $\mathcal{G}_{n,p,3}$

begin
for $0 \leq y \leq n$ **do**
foreach 3-coloring of each $Y \subseteq V$ of size y **do**
/** The initial phase **/
Construct a coloring that fails on $< \epsilon n$ vertices ;
/** The iterative recoloring procedure **/
Refine the initial coloring ;
/** The uncoloring procedure **/
Repeatedly uncolor vertices having few neighbors of some other color ;
/** The extension step **/
Color the components of the uncolored vertices ;
if we have a valid coloring of G **then** **return** ;
end

An SDP for MAX-3-CUT

Problem: Find a 3-cut $\bigcup V_i = [n]$ s.t. $\sum_{i \neq j} \mathrm{e}(V_i, V_j)$ is maximal.

Idea: Use variables taking one of 3 different values
(3 unit vectors $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ with $\langle \mathbf{s}_i \,|\, \mathbf{s}_j \rangle = -1/2$).

An SDP for MAX-3-CUT

Problem: Find a 3-cut $\bigcup V_i = [n]$ s.t. $\sum_{i \neq j} \mathrm{e}(V_i, V_j)$ is maximal.

Idea: Use variables taking one of 3 different values
(3 unit vectors $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ with $\langle \mathbf{s}_i \,|\, \mathbf{s}_j \rangle = -1/2$).

Max-3-cut:

$$\boxed{\begin{array}{l} \max \quad \displaystyle\sum_{ij \in E(G)} \frac{2}{3} \left(1 - \langle \mathbf{v}_i \,|\, \mathbf{v}_j \rangle\right), \\ \text{s.t. } \mathbf{v}_i \in \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3\}, \forall i \in V \end{array}}$$

An SDP for MAX-3-CUT

Problem: Find a 3-cut $\bigcup V_i = [n]$ s.t. $\sum_{i \neq j} \mathrm{e}(V_i, V_j)$ is maximal.

Idea: Use variables taking one of 3 different values
(3 unit vectors $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ with $\langle \mathbf{s}_i \,|\, \mathbf{s}_j \rangle = -1/2$).

Max-3-cut:

$$\boxed{\begin{aligned} &\max \quad \sum_{ij \in E(G)} \frac{2}{3} \left(1 - \langle \mathbf{v}_i \,|\, \mathbf{v}_j \rangle\right), \\ &\text{s.t. } \mathbf{v}_i \in \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3\}, \forall i \in V \end{aligned}}$$

Relaxation:

FRIEZE,JERRUM 1997

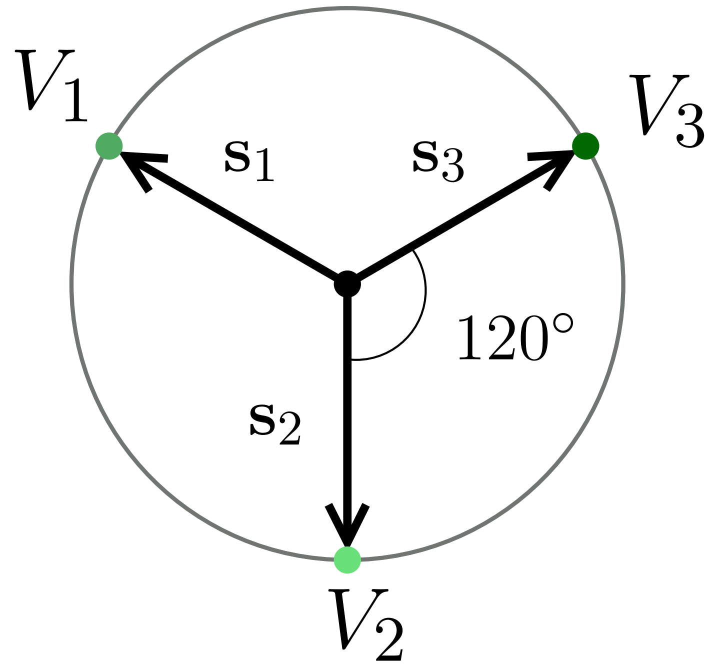
$$\boxed{\begin{aligned} &\max \quad \sum_{ij \in E(G)} \frac{2}{3} \left(1 - \langle \mathbf{x}_i \,|\, \mathbf{x}_j \rangle\right), \\ &\text{s.t.} \quad \mathbf{x}_i \in \mathbb{R}^n, \|\mathbf{x}_i\| = 1 \quad \forall i \in V \\ &\qquad \langle \mathbf{x}_i \,|\, \mathbf{x}_j \rangle \geq -\frac{1}{2} \quad \forall i, j \in V \end{aligned}} \qquad (\mathcal{SDP}_3)$$

$\mathcal{SDP}_3$ and 3-colorable graphs

- For a 3-colorable graph Max-3-cut $= |E|$

$\mathcal{SDP}_3$ and 3-colorable graphs

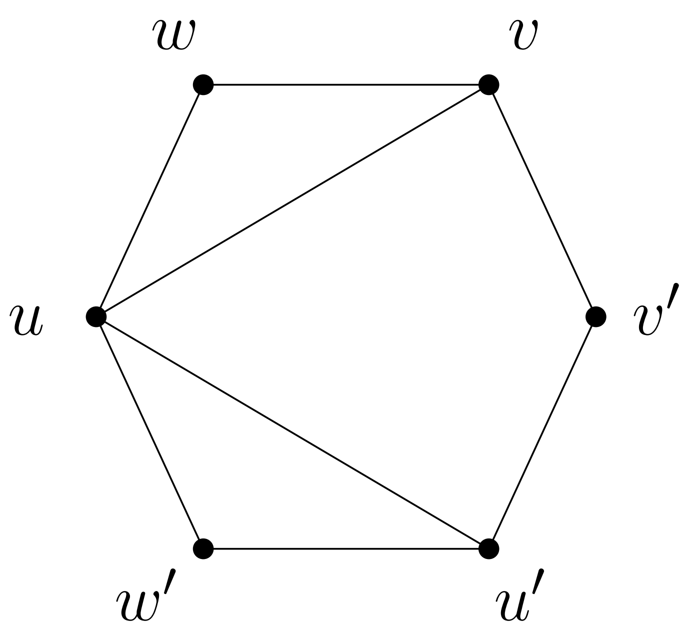
- For a 3-colorable graph Max-3-cut $= |E|$
- Realization of a maximum 3-cut in $\mathcal{SDP}_3$: map the color classes to the vectors $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$.

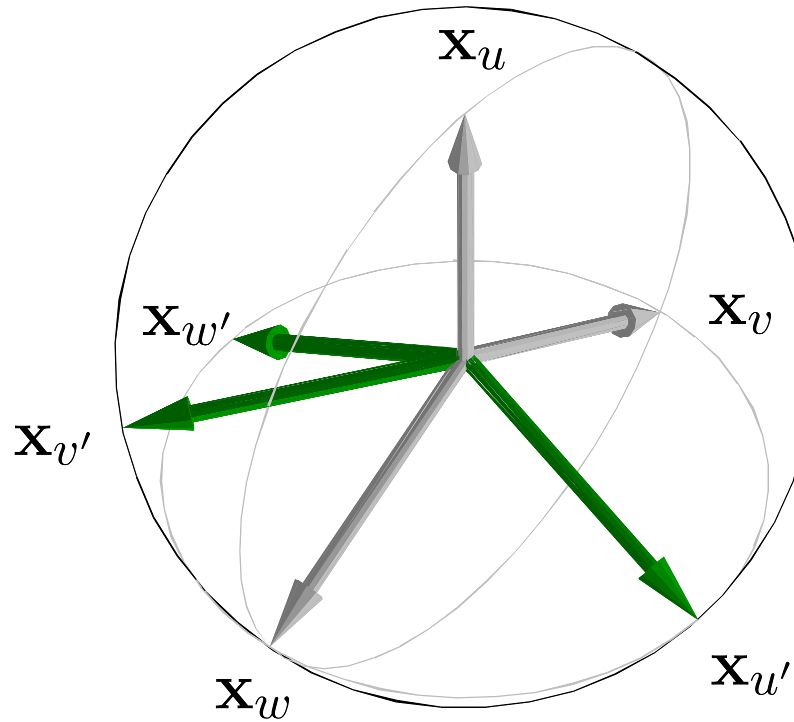


An optimal solution for $\mathcal{SDP}_3$ on 3-colorable graphs with color classes V_1,V_2,V_3

$\mathcal{SDP}_3$ and 3-colorable graphs (ctd.)

Positions of the vectors of an optimal solution to $\mathcal{SDP}_3$ are "far away" from this ideal picture in general:





An optimal solution for $\mathcal{SDP}_3$ on a 3-colorable graphs in 3 dimensions

An Algorithm for 3-Coloring $\mathcal{G}_{n,p,3}$

Algorithm 1: COLOR$\mathcal{G}_{n,p,3}$(G)

Input: a graph $\mathcal{G}_{n,p,3} = G = (V, E)$

Output: a valid coloring for $\mathcal{G}_{n,p,3}$

begin
- **for** $0 \leq y \leq n$ **do**
  - **foreach** 3-coloring of each $Y \subseteq V$ of size y **do**
    - /$**$ The initial phase $**$/
    - Construct a coloring that fails on $< \epsilon n$ vertices ;
    - /$**$ The iterative recoloring procedure $**$/
    - Refine the initial coloring ;
    - /$**$ The uncoloring procedure $**$/
    - Repeatedly uncolor vertices having few neighbors of some other color ;
    - /$**$ The extension step $**$/
    - Color the components of uncolored vertices ;
    - **if** we have a valid coloring of G **then** **return** ;

end

Finding an initial coloring of $\mathcal{G}_{n,p,3}$

- $\mathcal{X} = (\mathbf{x}_v)_{v \in V}$: optimal solution to $\mathcal{SDP}_3(G)$
- μ-neighborhood of v : $\mathbf{N}^\mu(v) := \{v' \in V \mid \langle \mathbf{x}_v \,|\, \mathbf{x}_{v'} \rangle > 1 - \mu\}$

Finding an initial coloring of $\mathcal{G}_{n,p,3}$

- $\mathcal{X} = (\mathbf{x}_v)_{v \in V}$: optimal solution to $\mathcal{SDP}_3(G)$
- μ-neighborhood of v : $\mathbf{N}^\mu(v) := \{v' \in V \mid \langle \mathbf{x}_v \,|\, \mathbf{x}_{v'} \rangle > 1 - \mu\}$

LEMMA

For all $\epsilon < 1/2$ there is a $\mu < 1/2$ s.t. t.f. holds with probability greater than $1 - \exp(-4n/3)$: For each $i \in \{1, 2, 3\}$ there is a vertex $v_i \in V_i$ with

- $|\mathbf{N}^\mu(v_i) \cap V_i| \geq (1 - \epsilon)n/3$ and
- $|\mathbf{N}^\mu(v_i) \cap V_j| < \epsilon n/3$ for all $j \neq i$

Finding an initial coloring of $\mathcal{G}_{n,p,3}$

Goal: A coloring of $\mathcal{SDP}_3$ that fails on $< \epsilon n$ vertices.

THEOREM Coja-Oghlan, Moore, Sanwalani

With probability at least $1 - \exp(-2n)$ t.f. holds

$$\mathcal{SDP}_3(\mathcal{G}_{n,p}) \leq \frac{2}{3}\binom{n}{2}p + \mathcal{O}\left(\sqrt{n^3 p(1-p)}\right).$$

Finding an initial coloring of $\mathcal{G}_{n,p,3}$

Goal: A coloring of $\mathcal{SDP}_3$ that fails on $< \epsilon n$ vertices.

THEOREM Coja-Oghlan, Moore, Sanwalani

With probability at least $1 - \exp(-2n)$ t.f. holds

$$\mathcal{SDP}_3(\mathcal{G}_{n,p}) \leq \frac{2}{3}\binom{n}{2}p + \mathcal{O}\left(\sqrt{n^3 p(1-p)}\right).$$

Idea: Construct $G^* \in \mathcal{G}_{n,p}$ from $G \in \mathcal{G}_{n,p,3}$ by inserting additional edges with probability p within each color class.

LEMMA

With probability at least $1 - \exp(-3n/2)$ t.f. holds

$$\mathcal{SDP}_3(G^*) - \mathcal{SDP}_3(G) \leq \mathcal{O}(n\sqrt{pn}).$$

Concluding Remarks

- k-coloring is hard / difficult to approximate.
- k-colorable graphs can be k-colored in polynomial expected time.
- On sparse graphs semidefinite programming helps.

Concluding Remarks

- k-coloring is hard / difficult to approximate.
- k-colorable graphs can be k-colored in polynomial expected time.
- On sparse graphs semidefinite programming helps.

There are only 3 colors, 10 digits,
and 7 notes; it's what we do with
them that's important.

RUTH ROSS

Concluding Remarks

- k-coloring is hard / difficult to approximate.
- k-colorable graphs can be k-colored in polynomial expected time.
- On sparse graphs semidefinite programming helps.

There are only 3 colors, 10 digits,
and 7 notes; it's what we do with
them that's important.

RUTH ROSS