Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Algorithmen & Komplexität

Institut für Informatik

Exact Algorithms

Sommer Term 2020

Lecture 6. Graph Coloring

Based on: [Exact Exponential Algorithms: §3.1.2, §4.3]

Further reading: [Parameterized Algorithms: §10.1.3]

(slides by J. Spoerhase, Th. van Dijk, S. Chaplick, and A. Wolff)

Alexander Wolff

Lehrstuhl für Informatik I

Graph Coloring

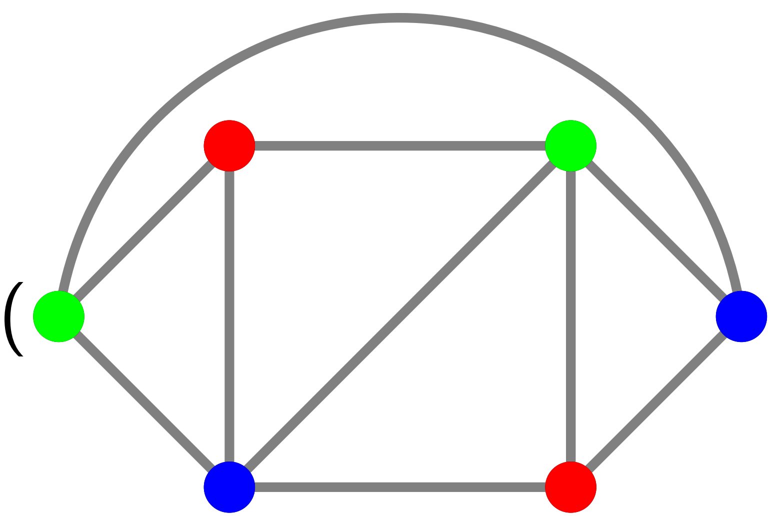
Given: Graph $G = (V, E)$

Find: *feasible coloring*, i.e., assign a color to each vertex so that adjacent vertices get different colors.

Objective: minimize the number of colors used.

Chromatic number:
$\chi(G) = \min_k G$ is k-colorable

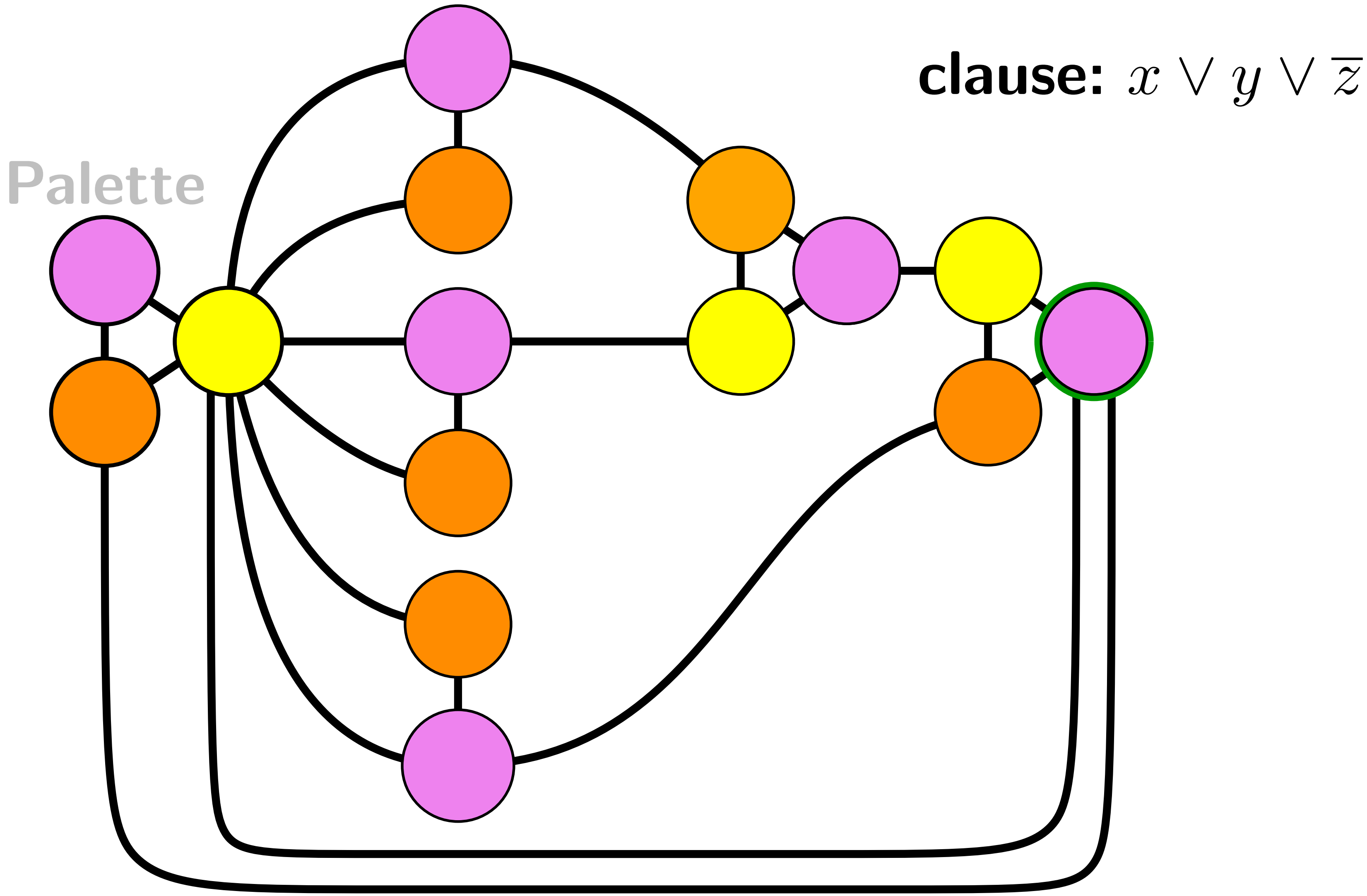
Color class:
Set of vertices with the same color.

$\chi($  $) = 3$

Complexity

Thm. k-Coloring is NP-complete **[Karp 1972]**

Thm. 3-Coloring is NP-complete



k-Coloring by Lawler [1976]

Binomial Thm. $\sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} = (x+y)^n$

Let $C_k(S) := G[S]$ is k-colorable.

$$C_k(S) \;=\; \big(\; \exists S' \subseteq S:\;\; C_1(S') \wedge C_{k-1}(S \setminus S') \;\big)$$

maximal

Determine: $C_k(V)$ **Algorithm:** Dynamic program

Runtime (fixed k): $\displaystyle\sum_{S \subseteq V} \sum_{S' \subseteq S} 1 \;=\; 3^n$

Better runtime (k fixed):

$$\sum_{S \subseteq V} \sum_{\substack{S' \subseteq S \\ S' \text{ maxIS}}} 1 \;=\; \sum_{S \subseteq V} \sqrt[3]{3}^{\,|S|} = \sum_{c=0}^{n} \binom{n}{c} \sqrt[3]{3}^{\,c} = (\sqrt[3]{3}+1)^n$$

$$\in O(2.4423^n)$$

3-Coloring (Exercise from 2016)

Trivial: $O^*(3^n)$

maximal [Lawler 1976]

G 3-colorable $\Leftrightarrow$ $\exists S$: S is independent, $G[V \setminus S]$ 2-colorable

Algorithm: enumerate all $S \subseteq V$ and check properties

Runtime: $O^*(2^n)$ $O^*(\sqrt[3]{3}^n) \subset O(1.4423^n)$

Schiermeyer 1994: $O(1.398^n)$

Beigel, Eppstein 1995: $O(1.3446^n)$

Beigel, Eppstein 2005: $O(1.3289^n)$

(3, 2)-CSP
+ reduction rules
+ case distinction

4-Coloring (Exercise from 2016)

G 4-colorable $\Leftrightarrow$ $\exists X \,\dot\cup\, Y = V$: $G[X]$ and $G[Y]$ 2-colorable.

Algorithm: Enumerate all $X \subseteq V$ and check the properties.

Runtime: $O^*(2^n)$

G 4-colorable $\Leftrightarrow$ $\exists S$: S maximal IS and $G[V \setminus S]$ 3-colorable

Algorithm: Enumerate sets S and check properties.

Runtime: $O^*(\sqrt[3]{3}^n \cdot \sqrt[3]{3}^n) = O^*(3^{2n/3}) \subset O^*(2.0801^n)$

but our 3-coloring instance is smaller than n...
W.l.o.g., $|S| \geq n/4$.

$O^*(\sqrt[3]{3}^n \cdot \sqrt[3]{3}^{\frac{3}{4}n}) = O^*(3^{\frac{1}{3}n + \frac{1}{3} \cdot \frac{3}{4}n}) \subset O(1.8982^n)$

Independent Sets by Byskov [2004]

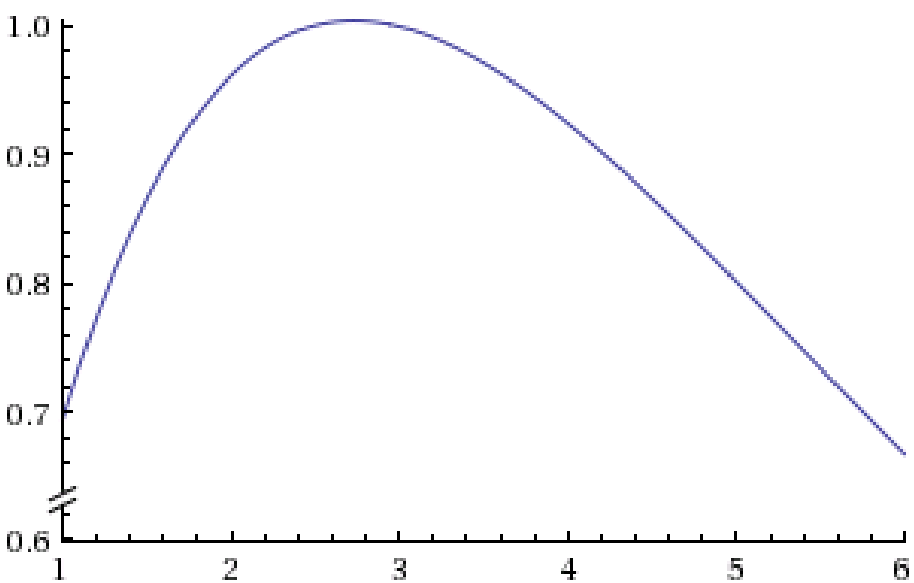
Def. $I^{=k}(G) :=$ maximal independent sets of size k

Thm. $\forall d \in \mathbb{N} : |I^{=k}| \leq d^{(d+1)k-n}(d+1)^{n-dk}$

Proof. As in Lecture 1.

$B(n) \leq s \cdot B(n-s) \leq s \cdot 3^{(n-s)/3} = \frac{s}{3^{s/3}} \cdot 3^{n/3} \leq 3^{n/3}$

\# leaves in the search tree



$s \longrightarrow$

This time: $B(n,k) = \ldots$

k-Coloring by Byskov [2004]

Def. $I^{=k}(G) :=$ maximal independent sets of size k

Procedure 1:

For each maximal independent set $I \subseteq V$ with $|I| \geq n/k$:
Check if $G[V \setminus I]$ is $(k-1)$-colorable.

$T_2(\cdot)$ **polynomial**

Runtime for k-coloring: $$\sum_{j=\lceil n/k \rceil}^{n} |I^{=j}(G)| \cdot T_{k-1}(n-j)$$

Procedure 2:

For each partition $X \cup\!\!\!\cdot\, Y = V$:
Check if $G[X]$ is $\lfloor k/2 \rfloor$-colorable and $G[Y]$ is $\lceil k/2 \rceil$-colorable.

Runtime for k-coloring: $$\sum_{j=0}^{n} \binom{n}{j} \cdot T_{k/2}(j)$$ **as by Lawler**

k-Coloring by Byskov [2004]

Thm. $\forall d \in \mathbb{N} : |I^{=k}| \leq d^{(d+1)k-n}(d+1)^{n-dk}$

3-Coloring: $O(1.3289^n)$ (Beigel & Eppstein 2005)

4-Coloring: $O(1.7504^n)$ $O(1.7272^n)$ Fomin, Gaspers, Saurabh (2007)

5-Coloring: $O(2.1592^n)$ $O(2.1364^n)$

6-Coloring: $O(2.3289^n)$

k-Coloring $O^*(2.4423^n)$ (Lawler 1976)

Coloring by Björklund & Husfeldt [2006] and Koivisto [2006]

Theorem. For n-vertex graphs, the graph coloring problem can be solved in $O^*(2^n)$ time. [FOCS'06]

Andreas Björklund

Thore Husfeldt

Mikko Koivisto

Color Classes and Set Partitioning

Color class: Set of vertices of the same color.
Each color class is an independent set.

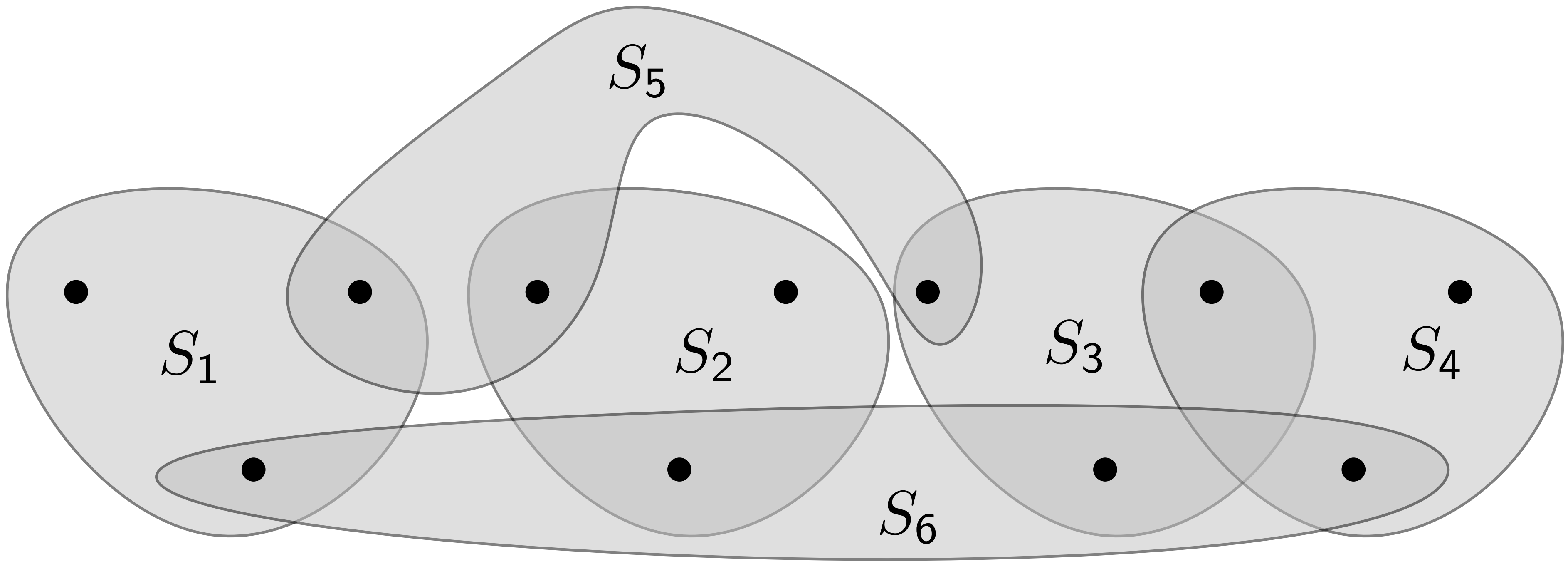
Alternatively:
Find the smallest number of independent sets
so that each node is in exactly one of these independent sets.

Cardinality Set Cover

Given: Set U and family $\mathcal{S} \subseteq 2^U$ with $\bigcup \mathcal{S} = U$,

Find: *Cover* $\mathcal{S}' \subseteq \mathcal{S}$ with $\bigcup \mathcal{S}' = U$.

Objective: Minimize the cardinality $|\mathcal{S}'|$ of the cover!



Graph Coloring via Cardinality Set Cover

Let $U = V(G)$ and $\mathcal{S} = \mathcal{I}$,
where $\mathcal{I}$ is the family of maximal independent sets of G.

Problem: Color classes must be disjoint!

What if we have a non-disjoint cover?
$V = I_1 \cup I_2 \cup \cdots \cup I_k$

Make it disjoint: for each $j = 1, \ldots, k$ (in order), set

$$I'_j := I_j - \bigcup_{j' < j} I_{j'}$$

$\Rightarrow V = I'_1 \,\dot{\cup}\, I'_2 \,\dot{\cup}\, \cdots \,\dot{\cup}\, I'_k$ is a k-coloring.

The family $\mathcal{I}$ can be enumerated in $O^*(2^n)$ time :-)

Variations & Definitions

Consider SC-instances $(U, \mathcal{S})$
where U is explicit but $\mathcal{S}$ is only *implicitly* given.

That is, we assume that $\mathcal{S}$ can be enumerated in $O^*(2^n)$ time,
where $n = |U|$ (w.l.o.g. $S \not\subseteq S'$ for any $S \neq S' \in \mathcal{S}$).

A k-*cover* is a set family $\mathcal{S}' \subseteq \mathcal{S}$ with $|\mathcal{S}'| = k$ and $\bigcup \mathcal{S}' = U$.

An *ordered* k-*cover* is a k-tuple $(S_1, \ldots, S_k)$ with $S_i \in \mathcal{S}$ and
$\bigcup_{i=1}^k S_i = U$.

Objective:
An algorithm to determine the *number* c_k of ordered k-covers.

The Number of Ordered k-Covers

For each $W \subseteq U$, let

$$\mathcal{S}[W] = \{\, S \in \mathcal{S} \mid S \cap W = \varnothing \,\}$$
$$s[W] = |\mathcal{S}[W]|$$

Lemma. The number of ordered k-covers of $(U, \mathcal{S})$ is

$$c_k = \sum_{W \subseteq U} (-1)^{|W|} s[W]^k.$$

Proof of the Lemma

$\mathcal{S}[W] = \{\, S \in \mathcal{S} \mid S \cap W = \varnothing \,\}$

$s[W] = |\mathcal{S}[W]|$

Lemma. The number of ordered k-covers of $(U, \mathcal{S})$ is

$$c_k = \sum_{W \subseteq U} (-1)^{|W|} s[W]^k.$$

Proof. Apply IE-Theorem:

- Objects: (ordered) k-tuples $(S_1, \ldots, S_k) \in \mathcal{S}^k$
- Properties: for each $u \in U$: $u \in \bigcup_i S_i$

$\bar{N}[W] =$ number of k-tuples avoiding $W = s[W]^k$

(We allow duplicates.)

$$c_k = N(U) \overset{\text{IE}}{=} \sum_{W \subseteq U} (-1)^{|W|} \bar{N}[W]$$

$$= \sum_{W \subseteq U} (-1)^{|W|} s[W]^k \qquad \square$$

Computing the Number of k-Covers

Theorem. The number of ordered k-covers of $(U, \mathcal{S})$ can be determined in $O^*(2^n)$ time.

Proof. Compute $s[W]$ for *every* $W \subseteq U$ via a DP in $O^*(2^n)$ total time.

Order the elements $U = \{u_1, \ldots, u_n\}$.

Want to compute $c_k = \sum_{W \subseteq U} (-1)^{|W|} s[W]^k$.

$\ldots$ but how to find $s[W]$?

A “Backward” DP

Let $g_i(W) = |\{S \in \mathcal{S}[W] \colon \{u_1, \ldots, u_i\} \setminus W \subseteq S\}|$
$= \#$ subsets of $\mathcal{S}$ avoiding W and containing $\{u_1, \ldots, u_i\} \setminus W$.

Note: $s[W] = g_0(W)$ for every $W \subseteq U$

Recurrence:

$$g_n(W) = \begin{cases} 1 & \text{if } U \setminus W \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

and, for $0 < i \leq n$:

$$g_{i-1}(W) = \begin{cases} g_i(W) & \text{if } u_i \in W \\ \underbrace{g_i(W)}_{\#\text{ subsets with } u_i} + \underbrace{g_i(W \cup \{u_i\})}_{\#\text{ subsets without } u_i} & \text{if } u_i \notin W \end{cases}$$

A "Backward" DP (cont'd)

Algorithm Num-k-Covers($U, \mathcal{S}$)

foreach $W \subseteq U$ **do**
- $g_n(W) \leftarrow 0$

foreach $S \in \mathcal{S}$ **do**
- $g_n(U \setminus S) \leftarrow 1$

for $i \leftarrow n$ **downto** 1 **do**
- **foreach** $W \subseteq U$ **do**
  - **if** $u_i \in W$ **then**
    - $g_{i-1}(W) \leftarrow g_i(W)$
  - **else**
    - $g_{i-1}(W) \leftarrow g_i(W) + g_i(W \cup \{u_i\})$

foreach $W \subseteq U$ **do**
- $s[W] \leftarrow g_0(W)$

Finally, compute $c_k = \sum_{W \subseteq U} (-1)^{|W|} s[W]^k$. $\square$

Runtime and Space Consumption

Corollary. The graph coloring problem can be solved in $O^*(2^n)$ time and space.

Proof. Determine the smallest k so that there is a k-cover for $(V, \mathcal{I})$. $\square$

This yields $\chi(G)$. **But how do we get a coloring?**

How much slower do we get if we insist on **polynomial space?**