Fixed Parameter Algorithms and Kernelization

Saket Saurabh

The Institute of Mathematica Sciences, India

Pre-WorKer 2011Talk, Vienna, 1st September

Classical complexity

A brief review:

- We usually aim for **polynomial-time** algorithms: the running time is $O(n^c)$, where n is the input size.
- Classical polynomial-time algorithms: shortest path, mathching, minimum spanning tree, 2SAT, convext hull, planar drawing, linear programming, etc.
- It is unlikely that polynomial-time algorithms exist for **NP-hard** problems.
- Unfortunately, many problems of interest are NP-hard: Hamiltonian cycle, 3-coloring, 3SAT, etc.
- We expect that these problems can be solved only in exponential time (i.e., c^n).

Can we say anything nontrivial about NP-hard problems?

Parameterized complexity

Main idea: Instead of expressing the running time as a function $T(n)$ of n, we express it as a function $T(n, k)$ of the input size n and some parameter k of the input.

In other words: we do not want to be efficient on all inputs of size n, only for those where k is small.

Parameterized complexity

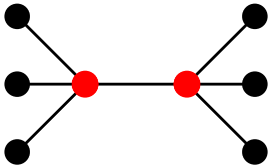
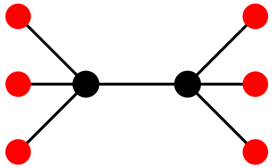
Main idea: Instead of expressing the running time as a function $T(n)$ of n, we express it as a function $T(n, k)$ of the input size n and some parameter k of the input.

In other words: we do not want to be efficient on all inputs of size n, only for those where k is small.

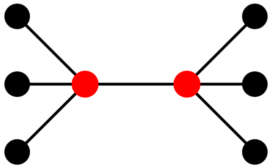
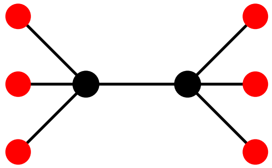
What can be the parameter k?

- The size k of the solution we are looking for.
- The maximum degree of the input graph.
- The diameter of the input graph.
- The length of clauses in the input Boolean formula.
- ...

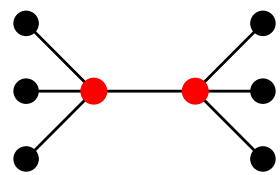
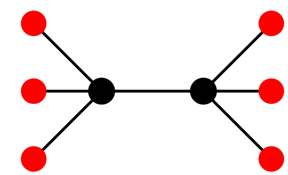
Parameterized complexity

Problem:	MINIMUM VERTEX COVER	MAXIMUM INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?
		
Complexity:	NP-complete	NP-complete

Parameterized complexity

Problem:	MINIMUM VERTEX COVER	MAXIMUM INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices? 	Is it possible to find k independent vertices? 
Complexity:	NP-complete	NP-complete
Complete enumeration:	$O(n^k)$ possibilities	$O(n^k)$ possibilities

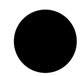
Parameterized complexity

Problem:	MINIMUM VERTEX COVER	MAXIMUM INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?
		
Complexity:	NP-complete	NP-complete
Complete enumeration:	$O(n^k)$ possibilities	$O(n^k)$ possibilities
	$O(2^k n^2)$ algorithm exists 	No $n^{o(k)}$ algorithm known 

Bounded search tree method

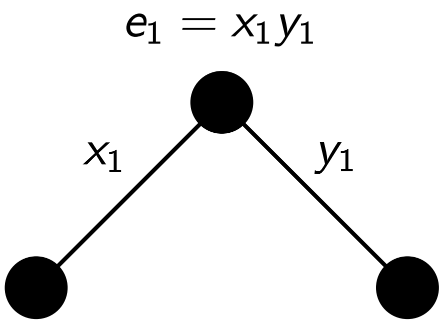
Algorithm for MINIMUM VERTEX COVER:

$e_1 = x_1 y_1$



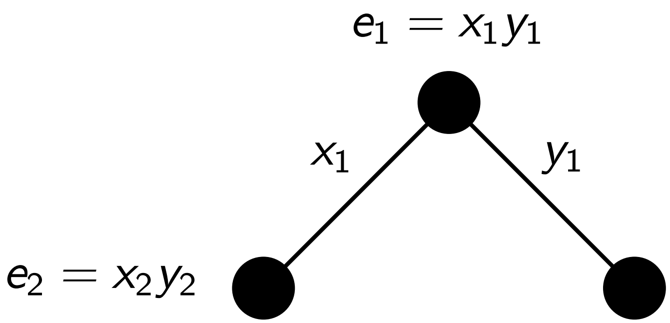
Bounded search tree method

Algorithm for MINIMUM VERTEX COVER:



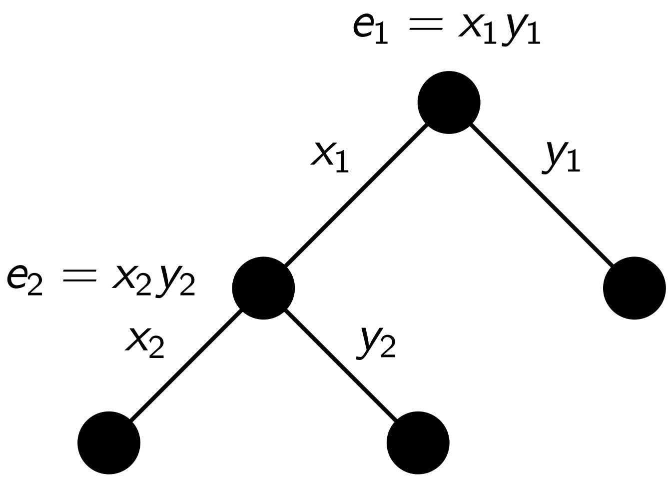
Bounded search tree method

Algorithm for MINIMUM VERTEX COVER:



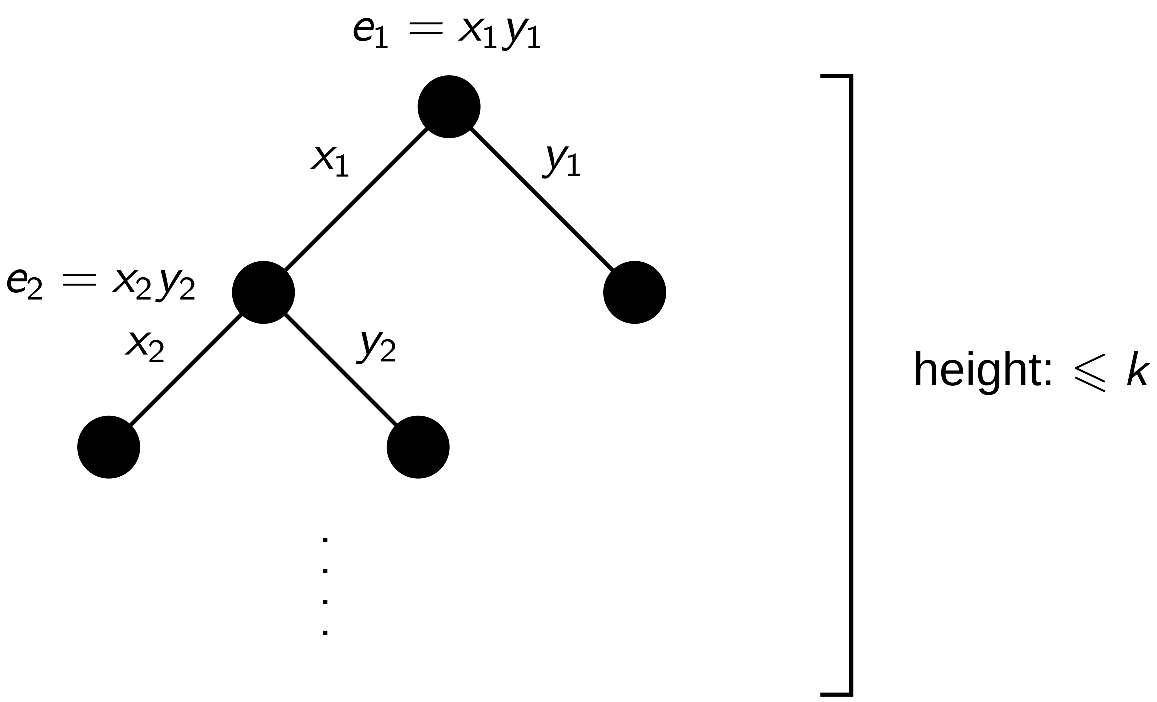
Bounded search tree method

Algorithm for MINIMUM VERTEX COVER:



Bounded search tree method

Algorithm for MINIMUM VERTEX COVER:



Height of the search tree is $\leqslant k \Rightarrow$ number of leaves is $\leqslant 2^k \Rightarrow$ complete search requires $2^k \cdot$ poly steps.

Fixed-parameter tractability

Definition: A **parameterization** of a decision problem is a function that assigns an integer parameter k to each input instance x.

The parameter can be

- explicit in the input (for example, if the parameter is the integer k appearing in the input (G, k) of VERTEX COVER), or
- implicit in the input (for example, if the parameter is the diameter d of the input graph G).

Main definition:

> A parameterized problem is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

Fixed-parameter tractability

Definition: A **parameterization** of a decision problem is a function that assigns an integer parameter k to each input instance x.

Main definition:

> A parameterized problem is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

Example: MINIMUM VERTEX COVER parameterized by the required size k is FPT: we have seen that it can be solved in time $O(2^k + n^2)$.

Better algorithms are known: e.g, $O(1.2832^k k + k|V|)$.

Main goal of parameterized complexity: to find FPT problems.

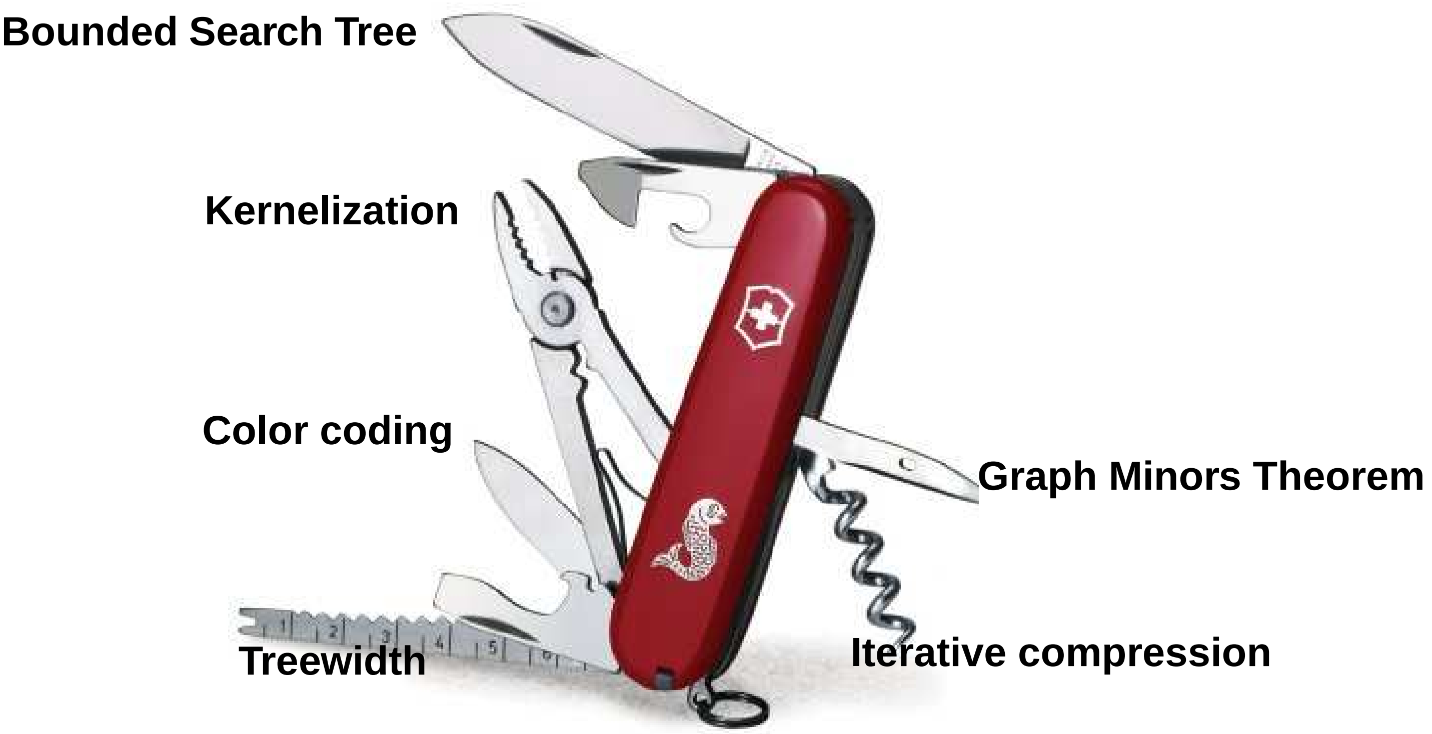
FPT problems

Examples of NP-hard problems that are FPT:

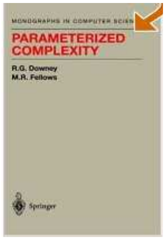
- Finding a vertex cover of size k.
- Finding a path of length k.
- Finding k disjoint triangles.
- Drawing the graph in the plane with k edge crossings.
- Finding disjoint paths that connect k pairs of points.
- ...

FPT algorithmic techniques

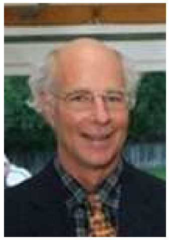
- Significant advances in the past 20 years or so (especially in recent years).
- Powerful toolbox for designing FPT algorithms:

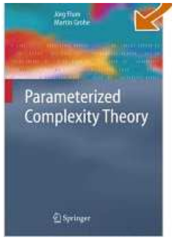


Books

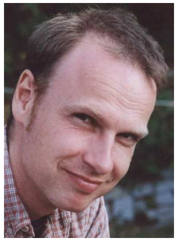


Downey-Fellows: Parameterized Complexity, Springer, 1999



Flum-Grohe: Parameterized Complexity Theory, Springer, 2006

Niedermeier: Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.

Kernelization

Kernelization

Definition: **Kernelization** is a polynomial-time transformation that maps an instance (I, k) to an instance (I', k') such that

- (I, k) is a yes-instance if and only if (I', k') is a yes-instance,
- $k' \leqslant k$, and
- $|I'| \leqslant f(k)$ for some function $f(k)$.

Kernelization

Definition: **Kernelization** is a polynomial-time transformation that maps an instance (I, k) to an instance (I', k') such that

- (I, k) is a yes-instance if and only if (I', k') is a yes-instance,
- $k' \leqslant k$, and
- $|I'| \leqslant f(k)$ for some function $f(k)$.

Simple fact: If a problem has a kernelization algorithm, then it is FPT.

Proof: Solve the instance (I', k') by brute force.

Kernelization

Definition: **Kernelization** is a polynomial-time transformation that maps an instance (I, k) to an instance (I', k') such that

- (I, k) is a yes-instance if and only if (I', k') is a yes-instance,
- $k' \leqslant k$, and
- $|I'| \leqslant f(k)$ for some function $f(k)$.

Simple fact: If a problem has a kernelization algorithm, then it is FPT.
Proof: Solve the instance (I', k') by brute force.

Converse: Every FPT problem has a kernelization algorithm.
Proof: Suppose there is an $f(k)n^c$ algorithm for the problem.

- If $f(k) \leqslant n$, then solve the instance in time $f(k)n^c \leqslant n^{c+1}$, and output a trivial yes- or no-instance.
- If $n < f(k)$, then we are done: a kernel of size $f(k)$ is obtained.

Kernelization for **VERTEX COVER**

General strategy: We devise a list of reduction rules, and show that if none of the rules can be applied and the size of the instance is still larger than $f(k)$, then the answer is trivial.

Reduction rules for VERTEX COVER instance (G, k):

Rule 1: If v is an isolated vertex $\Rightarrow (G \setminus v, k)$
Rule 2: If $d(v) > k \Rightarrow (G \setminus v, k - 1)$

Kernelization for VERTEX COVER

General strategy: We devise a list of reduction rules, and show that if none of the rules can be applied and the size of the instance is still larger than $f(k)$, then the answer is trivial.

Reduction rules for VERTEX COVER instance (G, k):

Rule 1: If v is an isolated vertex $\Rightarrow (G \setminus v, k)$
Rule 2: If $d(v) > k \Rightarrow (G \setminus v, k-1)$

If neither Rule 1 nor Rule 2 can be applied:

- If $|V(G)| > k(k+1) \Rightarrow$ There is no solution (every vertex should be the neighbor of at least one vertex of the cover).
- Otherwise, $|V(G)| \leqslant k(k+1)$ and we have a $k(k+1)$ vertex kernel.

Kernelization for VERTEX COVER

Let us add a third rule:

Rule 1: If v is an isolated vertex $\Rightarrow (G \setminus v, k)$

Rule 2: If $d(v) > k \Rightarrow (G \setminus v, k-1)$

Rule 3: If $d(v) = 1$, then we can assume that its neighbor u is in the solution $\Rightarrow (G \setminus (u \cup v), k-1)$.

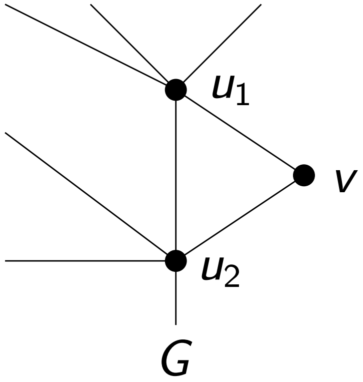
If none of the rules can be applied, then every vertex has degree at least 2.
$\Rightarrow |V(G)| \leqslant |E(G)|$

- If $|E(G)| > k^2 \Rightarrow$ There is no solution (each vertex of the solution can cover at most k edges).
- Otherwise, $|V(G)| \leqslant |E(G)| \leqslant k^2$ and we have a k^2 vertex kernel.

Kernelization for VERTEX COVER

Let us add a fourth rule:

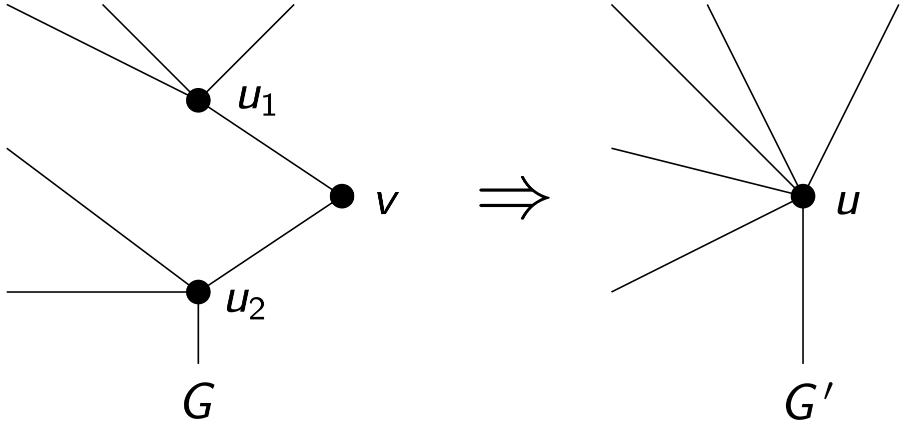
Rule 4a: If v has degree 2, and its neighbors u_1 and u_2 are adjacent, then we can assume that u_1, u_2 are in the solution $\Rightarrow (G \setminus \{u_1, u_2, v\}, k-2)$.



Kernelization for VERTEX COVER

Let us add a fourth rule:

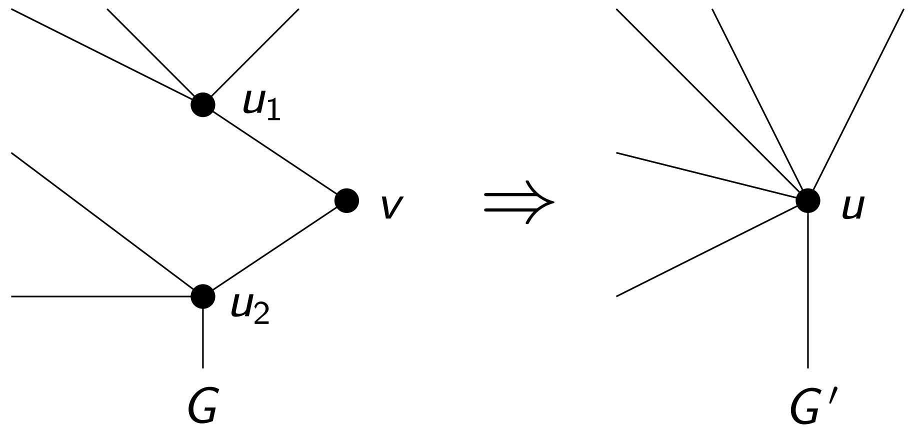
Rule 4b: If v has degree 2, then G' is obtained by identifying the two neighbors of v and deleting $v \Rightarrow (G', k-1)$.



Kernelization for VERTEX COVER

Let us add a fourth rule:

Rule 4b: If v has degree 2, then G' is obtained by identifying the two neighbors of v and deleting v $\Rightarrow$ $(G', k-1)$.



Correctness:

Let S' be a vertex cover of size $k-1$ for G'.

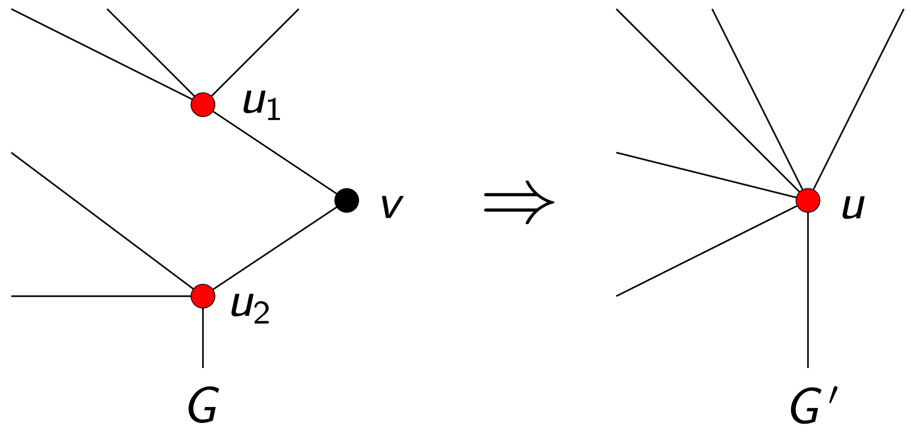
If $u \in S \Rightarrow (S' \setminus u) \cup \{u_1, u_2\}$ is a vertex cover of size k for G.

If $u \notin S \Rightarrow S' \cup v$ is a vertex cover of size k for G.

Kernelization for Vertex Cover

Let us add a fourth rule:

Rule 4b: If v has degree 2, then G' is obtained by identifying the two neighbors of v and deleting $v \Rightarrow (G', k-1)$.



Correctness:

Let S' be a vertex cover of size $k-1$ for G'.

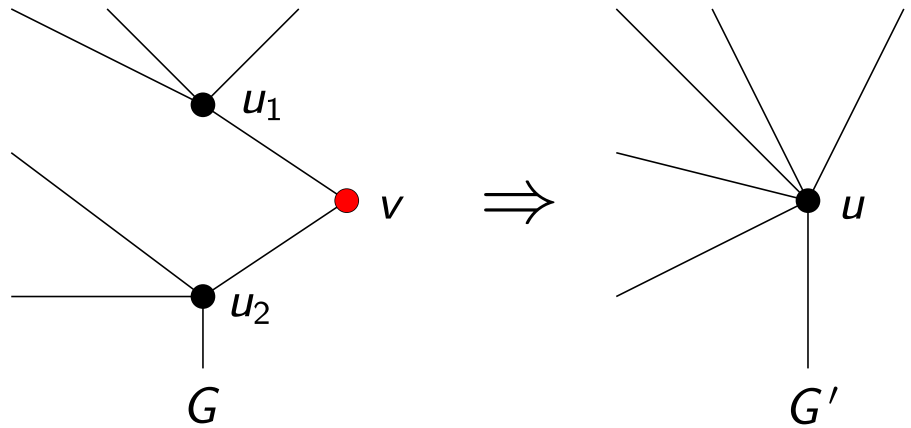
If $u \in S \Rightarrow (S' \setminus u) \cup \{u_1, u_2\}$ is a vertex cover of size k for G.

If $u \notin S \Rightarrow S' \cup v$ is a vertex cover of size k for G.

Kernelization for VERTEX COVER

Let us add a fourth rule:

Rule 4b: If v has degree 2, then G' is obtained by identifying the two neighbors of v and deleting $v \Rightarrow (G', k-1)$.



Correctness:

Let S' be a vertex cover of size $k-1$ for G'.

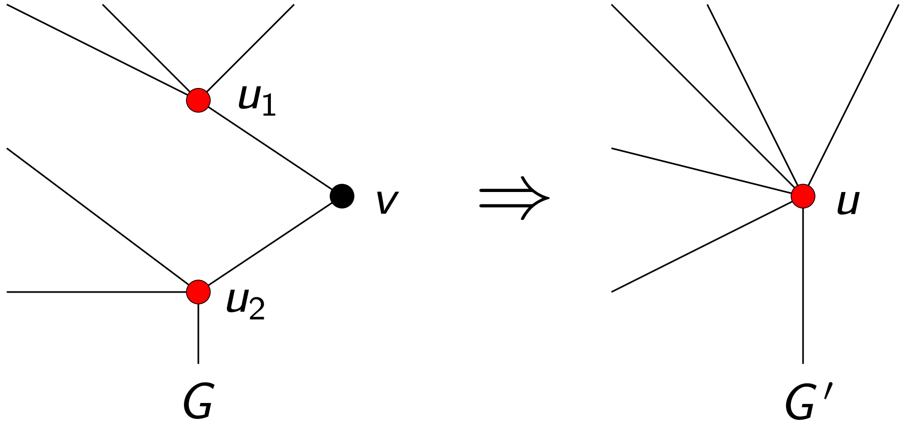
If $u \in S \Rightarrow (S' \setminus u) \cup \{u_1, u_2\}$ is a vertex cover of size k for G.

If $u \notin S \Rightarrow S' \cup v$ is a vertex cover of size k for G.

Kernelization for VERTEX COVER

Let us add a fourth rule:

Rule 4b: If v has degree 2, then G' is obtained by identifying the two neighbors of v and deleting $v \Rightarrow (G', k-1)$.



Correctness:

Let S be a vertex cover of size k for G.

If $u_1, u_2 \in S \Rightarrow (S \setminus \{u_1, u_2, v\}) \cup u$ is a vertex cover of size $k-1$ for G'.

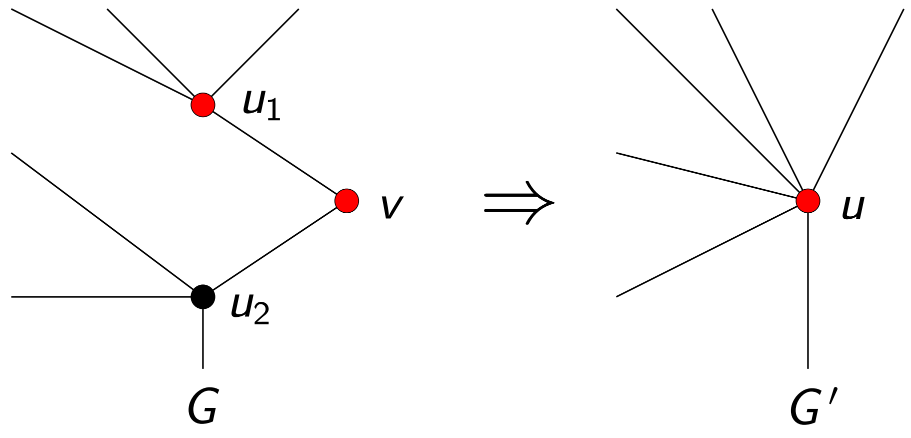
If exactly one of u_1 and u_2 is in S, then $v \in S \Rightarrow (S \setminus \{u_1, u_2, v\}) \cup u$ is a vertex cover of size $k-1$ for G'.

If $u_1, u_2 \notin S$, then $v \in S \Rightarrow (S \setminus v)$ is a vertex cover of size $k-1$ for G'.

Kernelization for VERTEX COVER

Let us add a fourth rule:

Rule 4b: If v has degree 2, then G' is obtained by identifying the two neighbors of v and deleting $v \Rightarrow (G', k-1)$.



Correctness:

Let S be a vertex cover of size k for G.

If $u_1, u_2 \in S \Rightarrow (S \setminus \{u_1, u_2, v\}) \cup u$ is a vertex cover of size $k-1$ for G'.

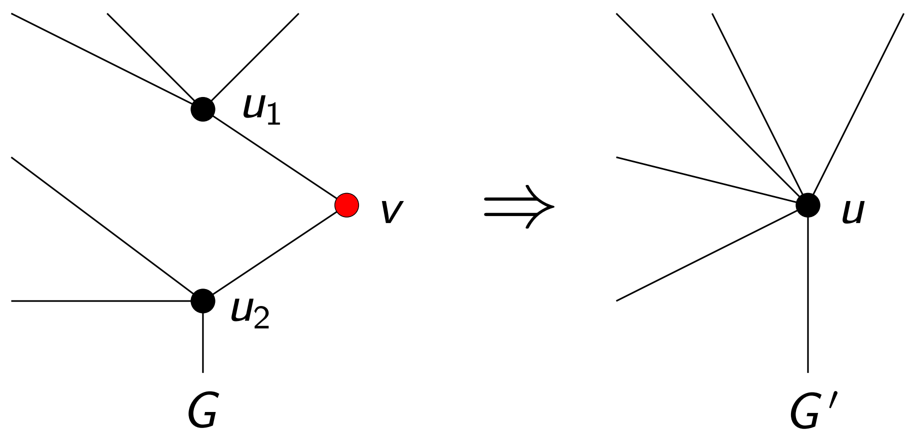
If exactly one of u_1 and u_2 is in S, then $v \in S \Rightarrow (S \setminus \{u_1, u_2, v\}) \cup u$ is a vertex cover of size $k-1$ for G'.

If $u_1, u_2 \notin S$, then $v \in S \Rightarrow (S \setminus v)$ is a vertex cover of size $k-1$ for G'.

Kernelization for VERTEX COVER

Let us add a fourth rule:

Rule 4b: If v has degree 2, then G' is obtained by identifying the two neighbors of v and deleting $v \Rightarrow (G', k-1)$.



Correctness:

Let S be a vertex cover of size k for G.

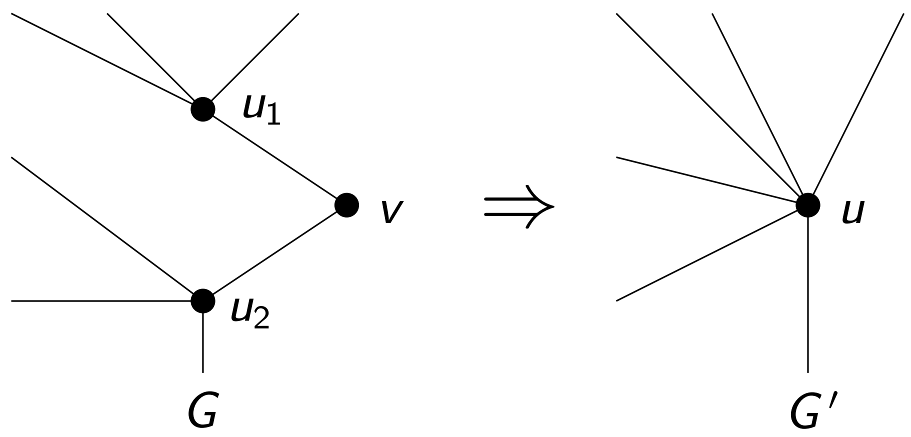
If $u_1, u_2 \in S \Rightarrow (S \setminus \{u_1, u_2, v\}) \cup u$ is a vertex cover of size $k-1$ for G'.
If exactly one of u_1 and u_2 is in S, then $v \in S \Rightarrow (S \setminus \{u_1, u_2, v\}) \cup u$ is a vertex cover of size $k-1$ for G'.

If $u_1, u_2 \notin S$, then $v \in S \Rightarrow (S \setminus v)$ is a vertex cover of size $k-1$ for G'.

Kernelization for Vertex Cover

Let us add a fourth rule:

Rule 4b: If v has degree 2, then G' is obtained by identifying the two neighbors of v and deleting $v \Rightarrow (G', k-1)$.

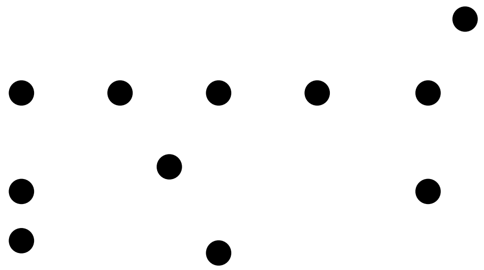


Kernel size:

- If $|E(G)| > k^2 \Rightarrow$ There is no solution (each vertex of the solution can cover at most k edges).
- Otherwise, $|V(G)| \leqslant 2|E(G)|/3 \leqslant \frac{2}{3}k^2$ and we have a $\frac{2}{3}k^2$ vertex kernel.

COVERING POINTS WITH LINES

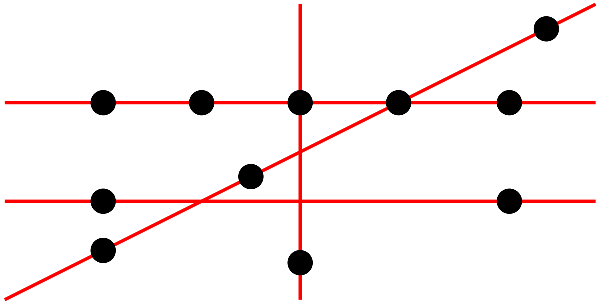
Task: Given a set P of n points in the plane and an integer k, find k lines that cover all the points.



Note: We can assume that every line of the solution covers at least 2 points, thus there are at most n^2 candidate lines.

Covering Points with Lines

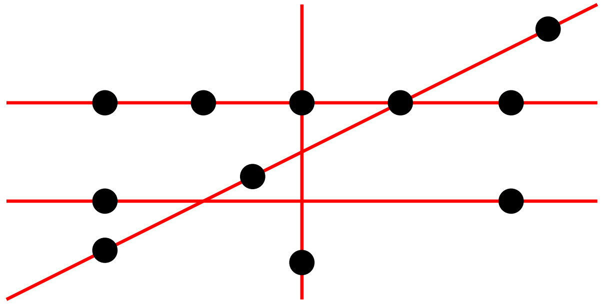
Task: Given a set P of n points in the plane and an integer k, find k lines that cover all the points.



Note: We can assume that every line of the solution covers at least 2 points, thus there are at most n^2 candidate lines.

COVERING POINTS WITH LINES

Task: Given a set P of n points in the plane and an integer k, find k lines that cover all the points.



Note: We can assume that every line of the solution covers at least 2 points, thus there are at most n^2 candidate lines.

Reduction Rule:

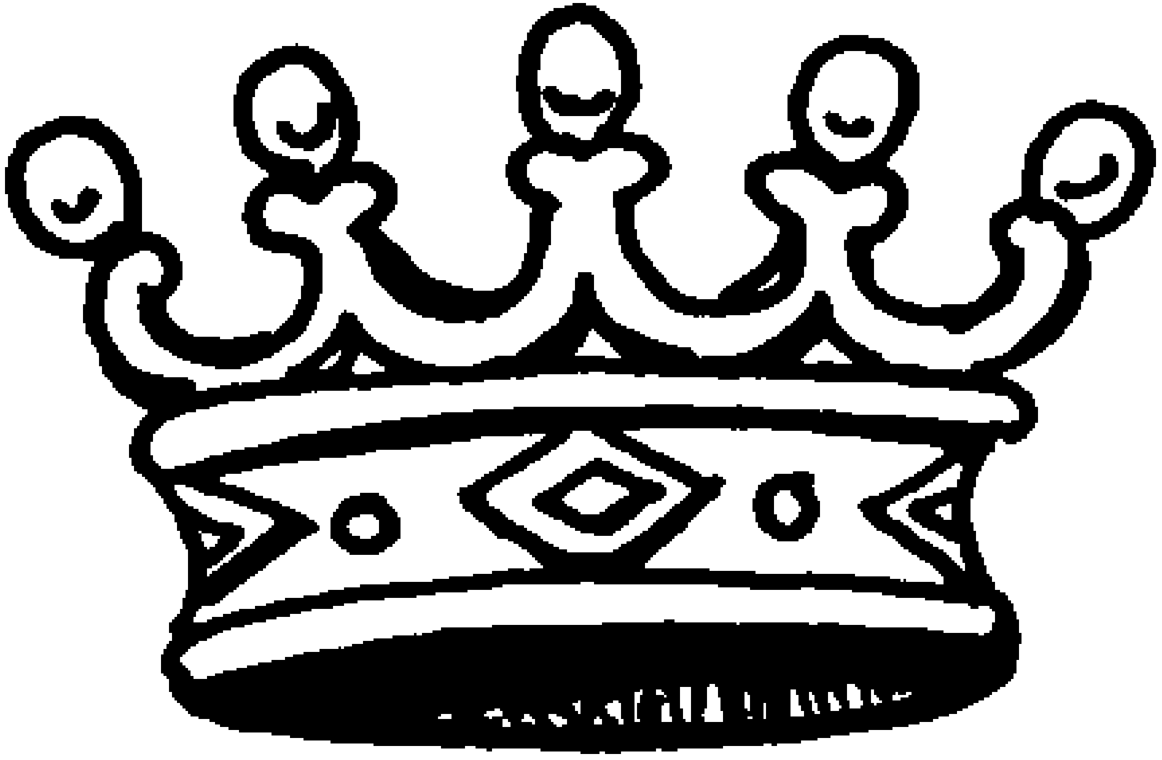
If a candidate line covers a set S of more than k points $\Rightarrow$ $(P \setminus S, k-1)$.

If this rule cannot be applied and there are still more than k^2 points, then there is no solution $\Rightarrow$ Kernel with at most k^2 points.

Kernelization

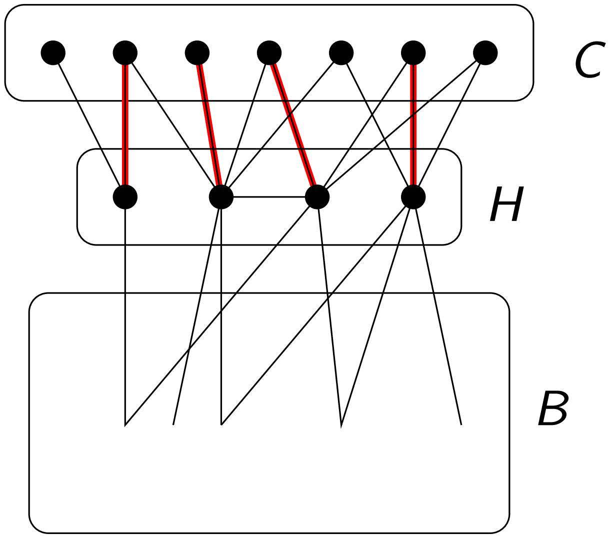
- Kernelization can be thought of as a polynomial-time preprocessing before attacking the problem with whatever method we have. “It does no harm” to try kernelization.
- Some kernelizations use lots of simple reduction rules and require a complicated analysis to bound the kernel size…
- … while other kernelizations are based on surprising nice tricks (Next: Crown Reduction and the Sunflower Lemma).
- Possibility to prove lower bounds.

Crown Reduction



Crown Reduction

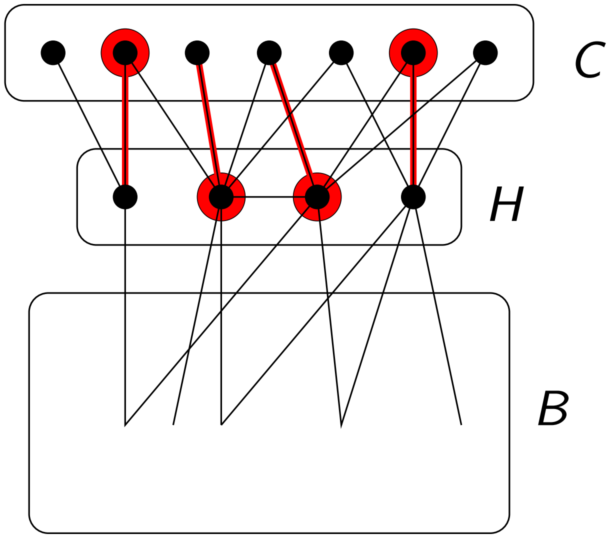
Definition: A **crown decomposition** is a partition $C \cup H \cup B$ of the vertices such that



- C is an independent set,
- there is no edge between C and B,
- there is a matching between C and H that covers H.

Crown Reduction

Definition: A **crown decomposition** is a partition $C \cup H \cup B$ of the vertices such that



- C is an independent set,
- there is no edge between C and B,
- there is a matching between C and H that covers H.

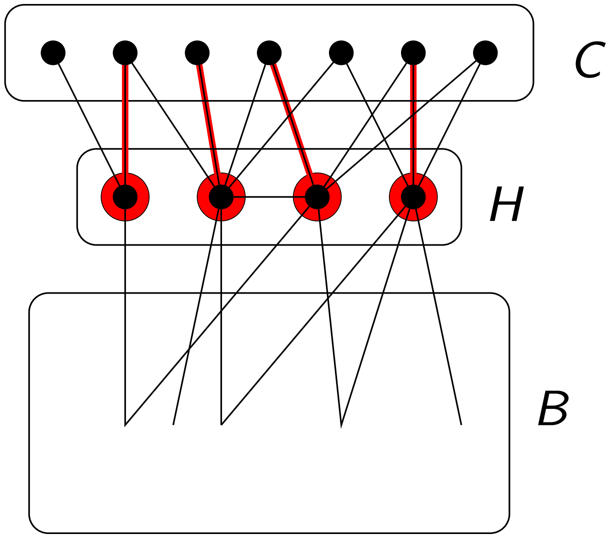
Crown rule for VERTEX COVER:

The matching needs to be covered and we can assume that it is covered by H (makes no sense to use vertices of C)

$\Rightarrow (G \setminus (H \cup C), k - |H|)$.

Crown Reduction

Definition: A **crown decomposition** is a partition $C \cup H \cup B$ of the vertices such that



- C is an independent set,
- there is no edge between C and B,
- there is a matching between C and H that covers H.

Crown rule for VERTEX COVER:

The matching needs to be covered and we can assume that it is covered by H (makes no sense to use vertices of C)

$\Rightarrow (G \setminus (H \cup C), k - |H|)$.

Crown Reduction

Key lemma:

Lemma: Given a graph G without isolated vertices and an integer k, in polynomial time we can either

- find a matching of size $k+1$,
- find a crown decomposition,
- or conclude that the graph has at most $3k$ vertices.

Crown Reduction

Key lemma:

Lemma: Given a graph G without isolated vertices and an integer k, in polynomial time we can either

- find a matching of size $k+1$, $\Rightarrow$ No solution!
- find a crown decomposition, $\Rightarrow$ Reduce!
- or conclude that the graph has at most $3k$ vertices.

  $\Rightarrow$ $3k$ vertex kernel!

This gives a 3k vertex kernel for VERTEX COVER.

Proof

Lemma: Given a graph G without isolated vertices and an integer k, in polynomial time we can either

- find a matching of size $k+1$,
- find a crown decomposition,
- or conclude that the graph has at most $3k$ vertices.

For the proof, we need the classical Kőnig's Theorem.

$\tau(G)$: size of the minimum vertex cover
$\nu(G)$: size of the maximum matching (independent set of edges)

Theorem: [Kőnig, 1931] If G is **bipartite,** then

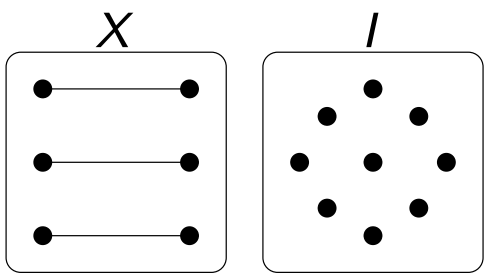
$$\tau(G) = \nu(G)$$

Proof

Lemma: Given a graph G without isolated vertices and an integer k, in polynomial time we can either

- find a matching of size $k+1$,
- find a crown decomposition,
- or conclude that the graph has at most $3k$ vertices.

Proof: Find (greedily) a maximal matching; if its size is at least $k+1$, then we are done. The rest of the graph is an independent set I.



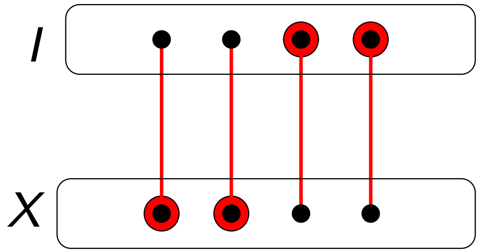
Proof

Lemma: Given a graph G without isolated vertices and an integer k, in polynomial time we can either

- find a matching of size $k + 1$,
- find a crown decomposition,
- or conclude that the graph has at most $3k$ vertices.

Proof: Find (greedily) a maximal matching; if its size is at least $k + 1$, then we are done. The rest of the graph is an independent set I.

Find a maximum matching/minimum vertex cover in the bipartite graph between X and I.



Proof

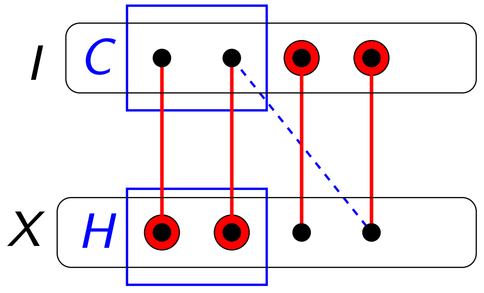
Lemma: Given a graph G without isolated vertices and an integer k, in polynomial time we can either

- find a matching of size $k+1$,
- find a crown decomposition,
- or conclude that the graph has at most $3k$ vertices.

Proof:

Case 1: The minimum vertex cover contains at least one vertex of X

$\Rightarrow$ There is a crown decomposition.



Proof

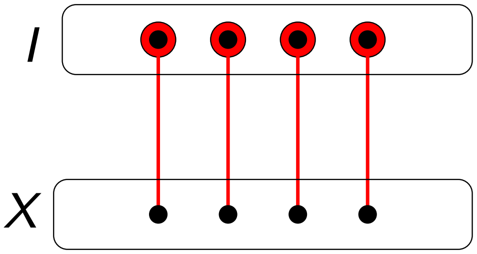
Lemma: Given a graph G without isolated vertices and an integer k, in polynomial time we can either

- find a matching of size $k+1$,
- find a crown decomposition,
- or conclude that the graph has at most $3k$ vertices.

Proof:

Case 1: The minimum vertex cover contains at least one vertex of X

$\Rightarrow$ There is a crown decomposition.



Case 2: The minimum vertex cover contains only vertices of I $\Rightarrow$ It contains every vertex of I

$\Rightarrow$ There are at most $2k + k$ vertices.

DUAL OF VERTEX COLORING

Parameteric dual of k-COLORING. Also known as SAVING k COLORS.

Task: Given a graph G and an integer k, find a vertex coloring with $|V(G)| - k$ colors.

Crown rule for DUAL OF VERTEX COLORING:

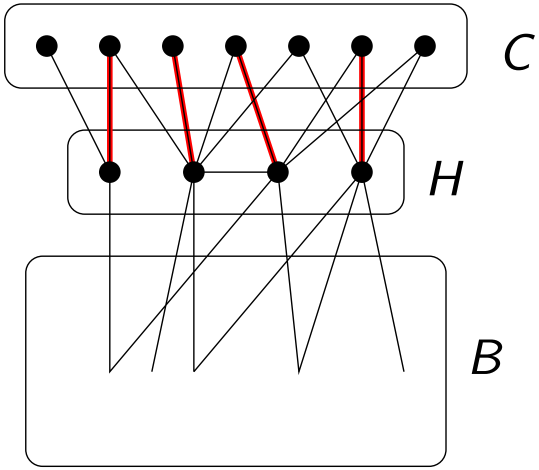
DUAL OF VERTEX COLORING

Parameteric dual of k-COLORING. Also known as SAVING k COLORS.

Task: Given a graph G and an integer k, find a vertex coloring with $|V(G)| - k$ colors.

Crown rule for DUAL OF VERTEX COLORING:

Suppose there is a crown decomposition for the **complement graph** $\overline{G}$.



- C is a clique in G: each vertex needs a distinct color.
- Because of the matching, it is possible to color H using only these $|C|$ colors.
- These colors cannot be used for B.
- $(G \setminus (H \cup C), k - |H|)$

DUAL OF VERTEX COLORING

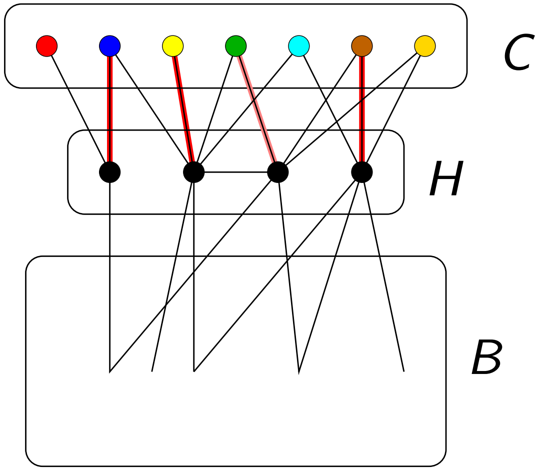
Parameteric dual of k-COLORING. Also known as SAVING k COLORS.

Task: Given a graph G and an integer k, find a vertex coloring with $|V(G)| - k$ colors.

Crown rule for DUAL OF VERTEX COLORING:

Suppose there is a crown decomposition for the **complement graph** $\overline{G}$.

- C is a clique in G: each vertex needs a distinct color.
- Because of the matching, it is possible to color H using only these $|C|$ colors.
- These colors cannot be used for B.
- $(G \setminus (H \cup C), k - |H|)$



Dual of Vertex Coloring

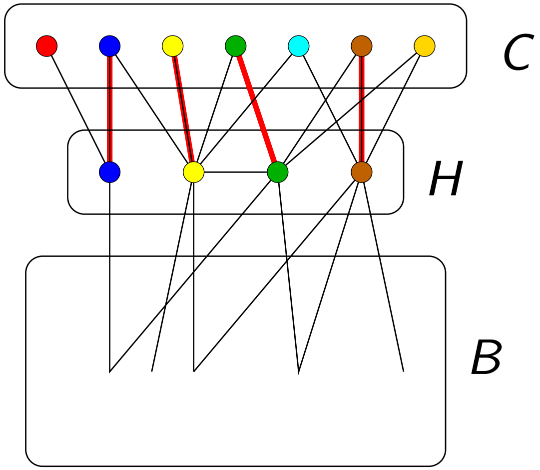
Parameteric dual of k-COLORING. Also known as SAVING k COLORS.

Task: Given a graph G and an integer k, find a vertex coloring with $|V(G)| - k$ colors.

Crown rule for DUAL OF VERTEX COLORING:

Suppose there is a crown decomposition for the **complement graph** $\overline{G}$.

- C is a clique in G: each vertex needs a distinct color.
- Because of the matching, it is possible to color H using only these $|C|$ colors.
- These colors cannot be used for B.
- $(G \setminus (H \cup C), k - |H|)$



Crown Reduction for DUAL OF VERTEX COLORING

Use the key lemma for the complement $\overline{G}$ of G:

Lemma: Given a graph G without isolated vertices and an integer k, in polynomial time we can either

- find a matching of size $k + 1$, $\Rightarrow$ YES: we can save k colors!
- find a crown decomposition, $\Rightarrow$ Reduce!
- or conclude that the graph has at most $3k$ vertices.

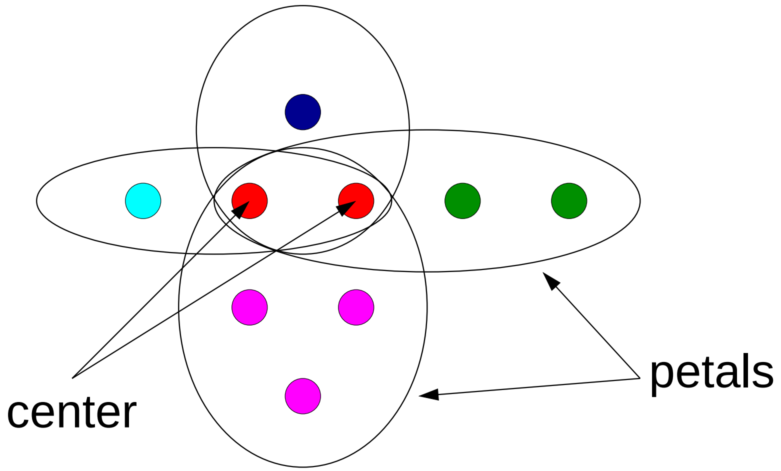
  $\Rightarrow$ $3k$ vertex kernel!

This gives a 3k vertex kernel for DUAL OF VERTEX COLORING.

Sunflower Lemma

Sunflower lemma

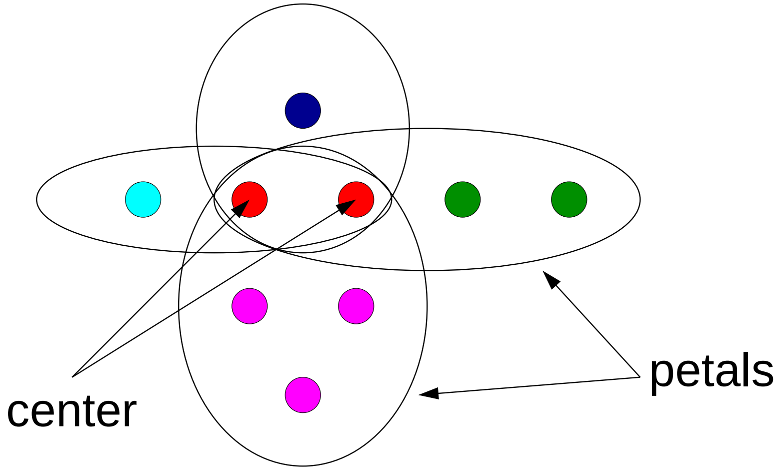
Definition: Sets S_1, S_2, ... , S_k form a **sunflower** if the sets $S_i \setminus (S_1 \cap S_2 \cap \cdots \cap S_k)$ are disjoint.



Lemma: [Erdős and Rado, 1960] If the size of a set system is greater than $(p-1)^d \cdot d!$ and it contains only sets of size at most d, then the system contains a sunflower with p petals. Furthermore, in this case such a sunflower can be found in polynomial time.

Sunflowers and d-HITTING SET

d-HITTING SET: Given a collection $\mathcal{S}$ of sets of size at most d and an integer k, find a set S of k elements that intersects every set of $\mathcal{S}$.



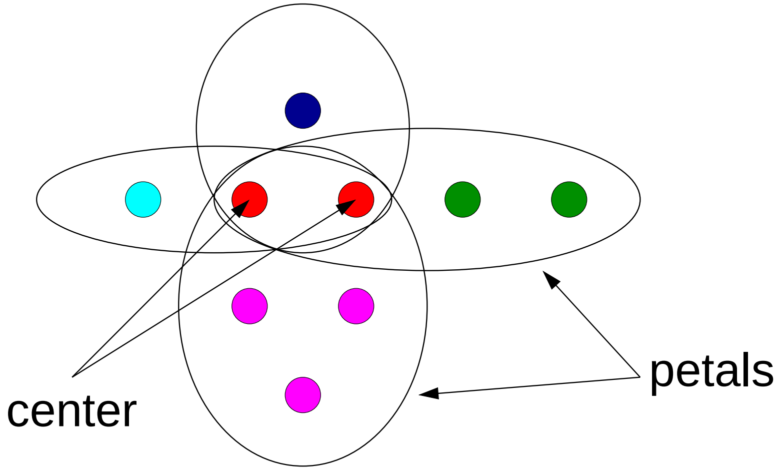
Reduction Rule: If $k+1$ sets form a sunflower, then remove these sets from $\mathcal{S}$ and add the center C to $\mathcal{S}$ (S does not hit one of the petals, thus it has to hit the center).

Note: if the center is empty (the sets are disjoint), then there is no solution.

If the rule cannot be applied, then there are at most $O(k^d)$ sets.

Sunflowers and d-HITTING SET

d-HITTING SET: Given a collection $\mathcal{S}$ of sets of size at most d and an integer k, find a set S of k elements that intersects every set of $\mathcal{S}$.



Reduction Rule (variant): Suppose more than $k+1$ sets form a sunflower.

- If the sets are disjoint $\Rightarrow$ No solution.
- Otherwise, keep only $k+1$ of the sets.

If the rule cannot be applied, then there are at most $O(k^d)$ sets.

Conclusions

- Many nice techniques invented so far — and probably many more to come.
- A single technique might provide the key for several problems.
- How to find new techniques? By attacking the open problems!
- Theory is incomplete if there is no way to say sorry we cant! — recently theory has evolved to say problems do not have polynomial kernels!!!