# Counting in Parameterized Comlexity

Dániel Marx

Institute for Computer Science and Control,
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

Joint work with Radu Curticapean and Holger Dell

Recent Advances in Parameterized Complexity
Tel Aviv, Israel, December 7, 2017

# Counting problems

Counting is harder than decision:

- Counting version of easy problems:
  not clear if they remain easy.
- Counting version of hard problems:
  not clear if we can keep the same running time.

# Counting problems

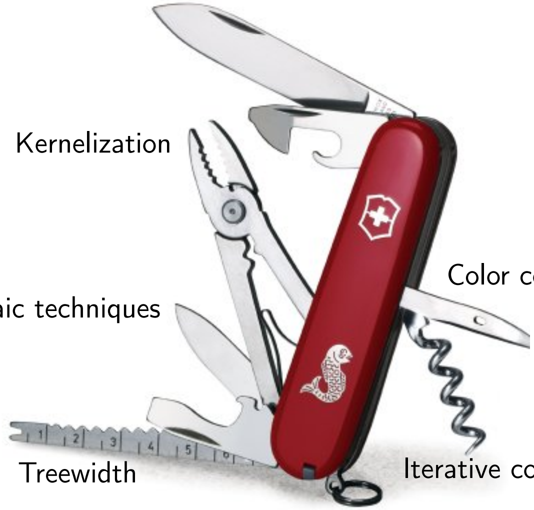
Counting is harder than decision:

- Counting version of easy problems:
  not clear if they remain easy.
- Counting version of hard problems:
  not clear if we can keep the same running time.

Working on counting problems is fun:

- You can revisit fundamental, "well-understood" problems.
- Requires a new set of lower bound techniques.
- Requires new algorithmic techniques.

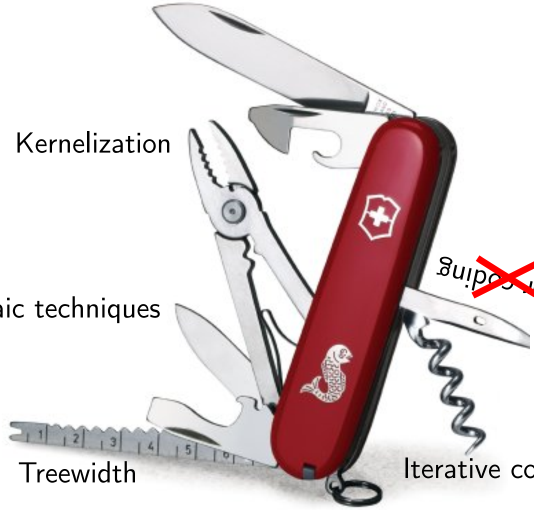
# FPT techniques



# FPT techniques . . . for counting?

Bounded-depth search trees

Kernelization



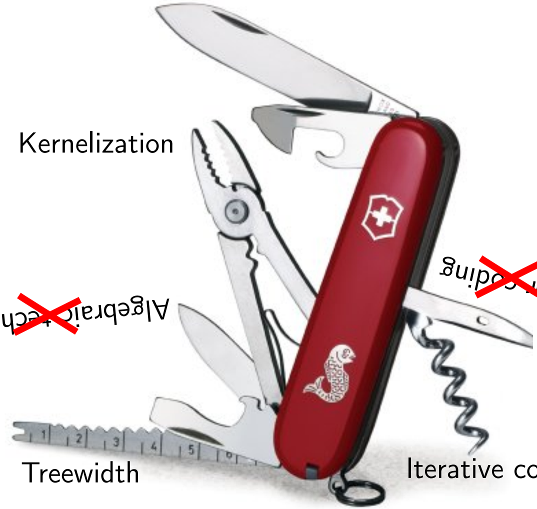
~~Color-coding~~

Algebraic techniques

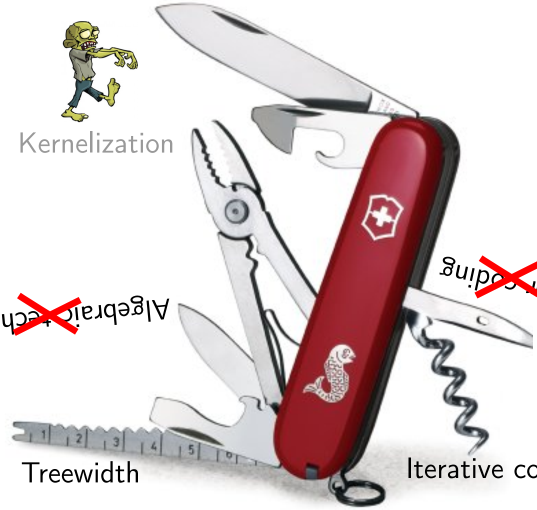
Treewidth

Iterative compression

# FPT techniques . . . for counting?



# FPT techniques . . . for counting?



# Counting complexity

- W[1]-hardness: "as hard as find a $k$-clique"
- #W[1]-hardness: "as hard as counting $k$-cliques"

Questions about counting versions of W[1]-hard problems:

- **Theoretical question:**
  Is the the counting version of a W[1]-hard problem #W[1]-hard?
- **More fine-grained question:**
  Can we get the same running time (e.g., $n^{O(\sqrt{k})}$) also for the counting version?

# Counting complexity

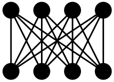
What can happen to the counting versions of an FPT or P problem?

1. The same algorithmic technique shows that the counting problem is FPT.
2. New algorithmic techniques are needed to show that the counting version is FPT.
3. New lower bound technique are needed to show that the counting version is #W[1]-hard.

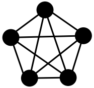
# Counting patterns

## Main question

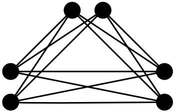
Which type of subgraph patterns are easy to count?



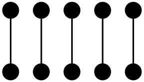
biclique



clique

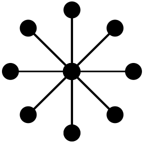


complete multipartite graph

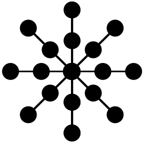


matching

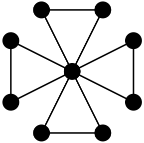
path



star



subdivided star

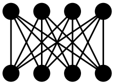


windmill

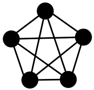
# Counting patterns

## Main question

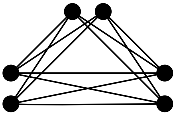
Which type of subgraph patterns are easy to count?



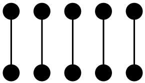
biclique



clique

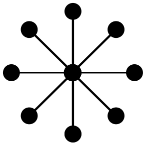


complete multipartite graph

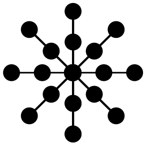


matching

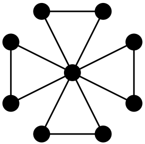
path



star



subdivided star

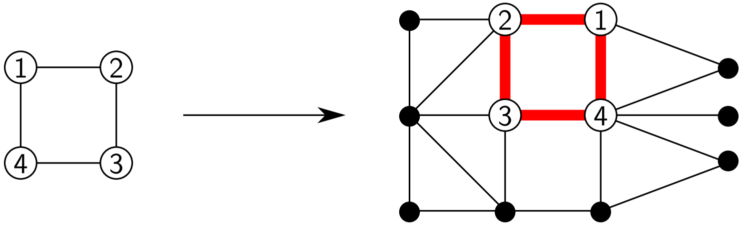


windmill

**Before that: counting homomorphisms!**

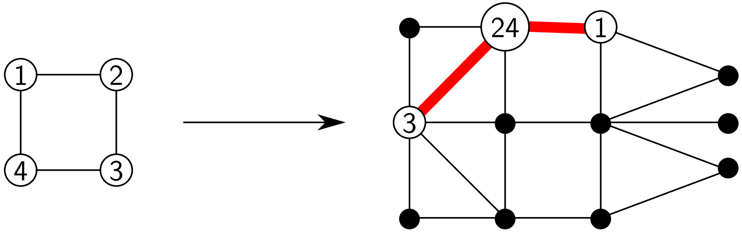
# Homomorphisms

A **homomorphism** from $H$ to $G$ is a mapping $f\colon V(H) \to V(G)$ such that if $ab$ is an edge of $H$, then $f(a)f(b)$ is an edge of $G$.



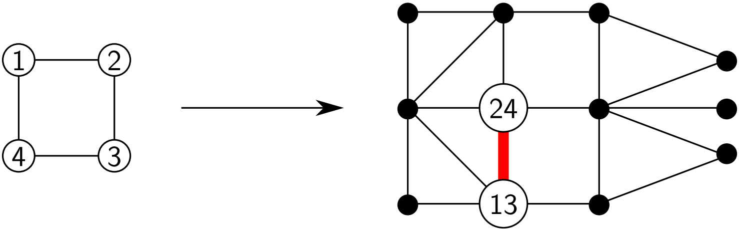
# Homomorphisms

A **homomorphism** from $H$ to $G$ is a mapping $f\colon V(H) \to V(G)$ such that if $ab$ is an edge of $H$, then $f(a)f(b)$ is an edge of $G$.



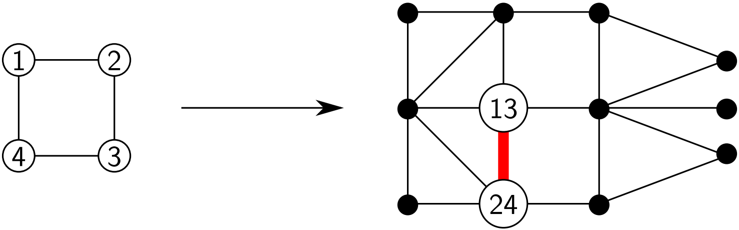
# Homomorphisms

A **homomorphism** from $H$ to $G$ is a mapping $f\colon V(H) \to V(G)$ such that if $ab$ is an edge of $H$, then $f(a)f(b)$ is an edge of $G$.



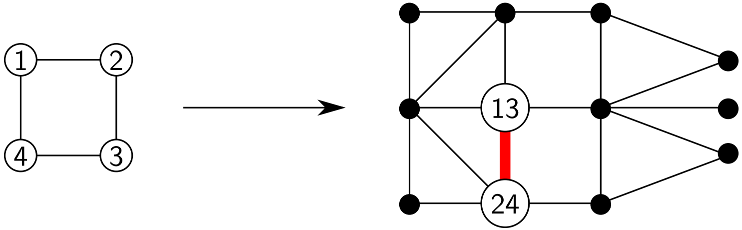
# Homomorphisms

A **homomorphism** from $H$ to $G$ is a mapping $f\colon V(H) \to V(G)$ such that if $ab$ is an edge of $H$, then $f(a)f(b)$ is an edge of $G$.



# Homomorphisms

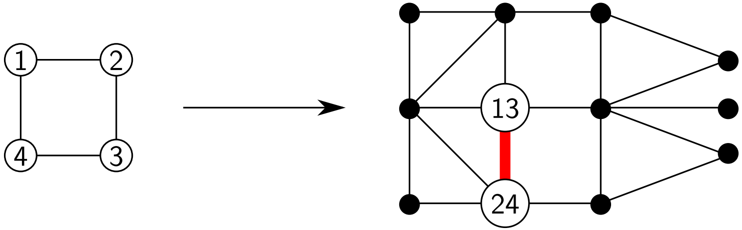
A **homomorphism** from $H$ to $G$ is a mapping $f\colon V(H) \to V(G)$ such that if $ab$ is an edge of $H$, then $f(a)f(b)$ is an edge of $G$.




Which pattern graphs $H$ are easy for counting homomorphisms?

# Homomorphisms

A **homomorphism** from $H$ to $G$ is a mapping $f\colon V(H) \to V(G)$ such that if $ab$ is an edge of $H$, then $f(a)f(b)$ is an edge of $G$.




Which pattern graphs $H$ are easy for counting homomorphisms?

**Theorem (trivial)**

For every fixed $H$, the problem $\#\textsc{Hom}(H)$ (count homomorphisms from $H$ to the given graph $G$) is polynomial-time solvable.

... because we can try all $|V(G)|^{|V(H)|}$ possible mappings $f\colon V(H) \to V(G)$.

# Counting homomorphisms

**Better question:**

$\#\textsc{Hom}(\mathcal{H})$
**Input:** graph $H \in \mathcal{H}$ and an arbitrary graph $G$.
**Task:** count the number of homomorphisms from $H$ to $G$.

**Goal:** characterize the classes $\mathcal{H}$ for which $\#\textsc{Hom}(\mathcal{H})$ is polynomial-time solvable.

# Counting homomorphisms

**Better question:**

| |
|---|
| $\#\textsc{Hom}(\mathcal{H})$<br>**Input:** graph $H \in \mathcal{H}$ and an arbitrary graph $G$.<br>**Task:** count the number of homomorphisms from $H$ to $G$. |

**Goal:** characterize the classes $\mathcal{H}$ for which $\#\textsc{Hom}(\mathcal{H})$ is polynomial-time solvable.

We have reasons to believe that there is no P vs. NP-complete dichotomy for $\#\textsc{Hom}(\mathcal{H})$. Instead of NP-completeness, we will use paramterized complexity for giving negative evidence.

We parameterize by $k = |V(H)|$, i.e., our goal is an $f(|V(H)|) \cdot n^{O(1)}$ time algorithm.

# Counting homomorphisms

**Theorem** [Dalmau and Jonsson 2004]

Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\textsc{Hom}(\mathcal{H})$ is polynomial-time solvable.
2. $\#\textsc{Hom}(\mathcal{H})$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded treewidth.

# Counting homomorphisms

**Theorem** [Dalmau and Jonsson 2004]

Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\textsc{Hom}(\mathcal{H})$ is polynomial-time solvable.
2. $\#\textsc{Hom}(\mathcal{H})$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded treewidth.

**Proof of the positive result:**

- Show that the problem can be solved in time $O(n^{c+1})$ if $H$ has treewidth $c$ (standard dynamic programing). [Díaz et al. 2002]

# Counting homomorphisms

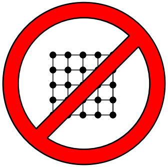
**Theorem** [Dalmau and Jonsson 2004]

Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\textsc{Hom}(\mathcal{H})$ is polynomial-time solvable.
2. $\#\textsc{Hom}(\mathcal{H})$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded treewidth.

**Excluded Grid Theorem** [Robertson and Seymour]

There is a function $f$ such that every graph with treewidth $f(k)$ contains a $k \times k$ grid minor.



# Counting homomorphisms

**Theorem** [Dalmau and Jonsson 2004]

Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\textsc{Hom}(\mathcal{H})$ is polynomial-time solvable.
2. $\#\textsc{Hom}(\mathcal{H})$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded treewidth.

**Proof of the negative result:**

1. Show that $\#\textsc{Hom}(\mathcal{H})$ is W[1]-hard if $\mathcal{H}$ is the class of grids.
2. Show that if $H$ contains $\boxplus_{k\times k}$ as minor, then $\#\textsc{Hom}(\boxplus_{k\times k})$ can be reduced to $\#\textsc{Hom}(H)$.
3. Use the Excluded Grid Theorem to show that this implies W[1]-hardness for every class $\mathcal{H}$ with unbounded treewidth.

# Counting subgraphs

Two highlights of classical complexity:

- Finding a perfect matching is polynomial-time solvable.
  [Edmonds 1965]
- Counting perfect matchings is #P-hard.
  [Valiant 1979]

# Counting subgraphs

Two highlights of classical complexity:

- Finding a perfect matching is polynomial-time solvable.
  [Edmonds 1965]
- Counting perfect matchings is #P-hard.
  [Valiant 1979]

[Flum and Grohe 2002] started the study of parameterized counting problems.

**Theorem**

Counting $k$-paths is #W[1]-hard.

**Question:** What about counting $k$-matchings?

# Counting $k$-matchings

Colorful history:

# Counting $k$-matchings

Colorful history:

- Weighted version is #W[1]-hard
  [Bläser and Curticapean 2012]

# Counting $k$-matchings

Colorful history:

- Weighted version is #W[1]-hard [Bläser and Curticapean 2012]
- Unweighted version is #W[1]-hard [Curticapean 2013] — complicated proof.

# Counting $k$-matchings

Colorful history:

- Weighted version is #W[1]-hard
  [Bläser and Curticapean 2012]
- Unweighted version is #W[1]-hard
  [Curticapean 2013] — complicated proof.
- Unweighted version is #W[1]-hard
  [Curticapean and M 2014] — simpler proof.

# Counting $k$-matchings

Colorful history:

- Weighted version is #W[1]-hard
  [Bläser and Curticapean 2012]
- Unweighted version is #W[1]-hard
  [Curticapean 2013] — complicated proof.
- Unweighted version is #W[1]-hard
  [Curticapean and M 2014] — simpler proof.
- Unweighted version is #W[1]-hard
  [Curticapean and M, unpublished] — even simpler proof.

# Counting $k$-matchings

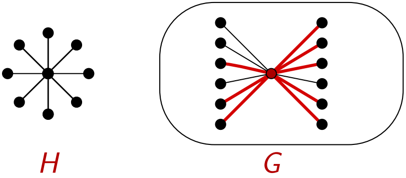
Colorful history:

- Weighted version is #W[1]-hard
  [Bläser and Curticapean 2012]
- Unweighted version is #W[1]-hard
  [Curticapean 2013] — complicated proof.
- Unweighted version is #W[1]-hard
  [Curticapean and M 2014] — simpler proof.
- Unweighted version is #W[1]-hard
  [Curticapean and M, unpublished] — even simpler proof.
- Unweighted version is #W[1]-hard
  [Curticapean, Dell, and M 2017] — tells the real story.

# Counting subgraphs

$\#\text{SUB}(\mathcal{H})$
**Input:** a graph $H \in \mathcal{H}$ and an arbitrary graph $G$.
**Task:** calculate the number of copies of $H$ in $G$.

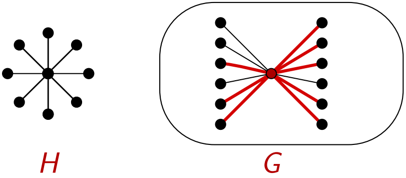
If $\mathcal{H}$ is the class of all stars, then $\#\text{SUB}(\mathcal{H})$ is easy: for each placement of the center of the star, calculate the number of possible different assignments of the leaves.



# Counting subgraphs

> $\#\text{SUB}(\mathcal{H})$
> **Input:** a graph $H \in \mathcal{H}$ and an arbitrary graph $G$.
> **Task:** calculate the number of copies of $H$ in $G$.

If $\mathcal{H}$ is the class of all stars, then $\#\text{SUB}(\mathcal{H})$ is easy: for each placement of the center of the star, calculate the number of possible different assignments of the leaves.




**Theorem** [Vassilevska Williams and Williams][Kowalik et al.]

If every graph in $\mathcal{H}$ has vertex cover number at most $c$, then $\#\text{SUB}(\mathcal{H})$ is polynomial-time solvable.

# Counting subgraphs

**Theorem** [Curticapean and M. 2014][Curticapean, Dell, and M. 2017]

Let $\mathcal{H}$ be a recursively enumerable class of graphs. If $\mathcal{H}$ has unbounded vertex cover number, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

($\nu(G) \le \tau(G) \le 2\nu(G)$, hence "unbounded vertex cover number" and "unbounded matching number" are the same.)

# Counting subgraphs

**Theorem** [Curticapean and M. 2014][Curticapean, Dell, and M. 2017]

Let $\mathcal{H}$ be a recursively enumerable class of graphs. If $\mathcal{H}$ has unbounded vertex cover number, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

($\nu(G) \le \tau(G) \le 2\nu(G)$, hence "unbounded vertex cover number" and "unbounded matching number" are the same.)

**Dichotomy theorem:**

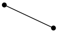
**Theorem**

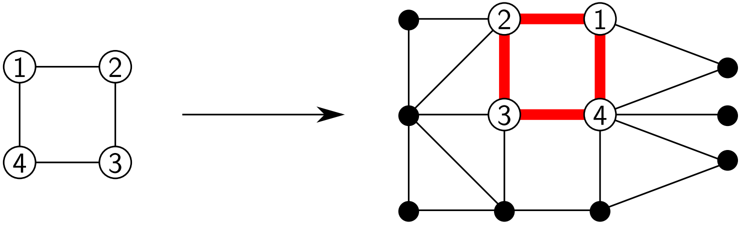
Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\textsc{Sub}(\mathcal{H})$ is polynomial-time solvable.
2. $\#\textsc{Sub}(\mathcal{H})$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded vertex cover number.

# Subgraphs $\Leftrightarrow$ homomorphisms

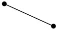
**Easy to check:**

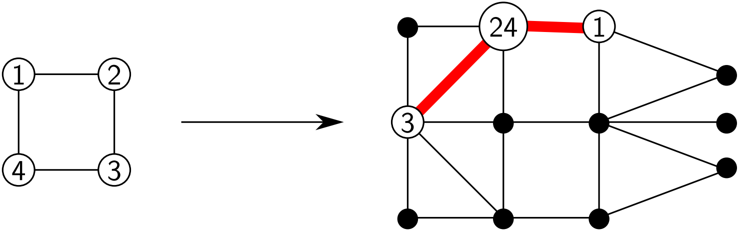
hom(  , $G$) $=$ 8sub(  , $G$) $+$ 4sub(  , $G$) $+$ 2sub(  , $G$)



# Subgraphs $\Leftrightarrow$ homomorphisms

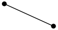
**Easy to check:**

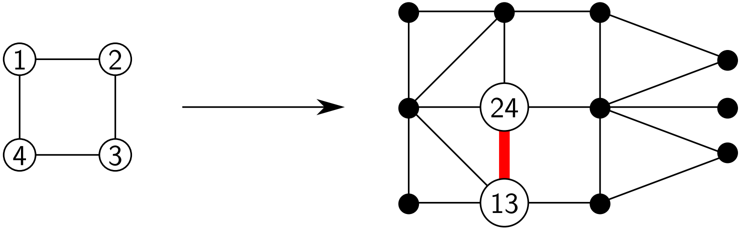
hom(  , $G$) = 8sub(  , $G$) + 4sub(  , $G$) + 2sub(  , $G$)



# Subgraphs ⇔ homomorphisms

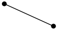
**Easy to check:**

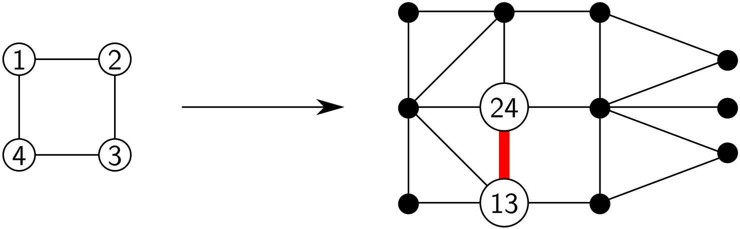
hom(, $G$) $=$ 8sub(, $G$) $+$ 4sub(, $G$) $+$ 2sub(, $G$)



# Subgraphs ⇔ homomorphisms

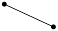
**Easy to check:**

hom( , $G$) $=$ 8sub( , $G$) $+$ 4sub( , $G$) $+$ 2sub( , $G$)




**Not completely obvious:**

The formula can be reversed by inclusion-exclusion.

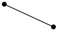
sub( , $G$) $= \frac{1}{8}$hom( , $G$) $- \frac{1}{4}$hom( , $G$) $+ \frac{1}{8}$hom( , $G$)

# General statements

**Definition**

surj$(H, G)$: number of surjective homomorphisms from $H$ to $G$ (every vertex and edge of $G$ appears in the image).

Homomorphisms can be counted by classifying according to the image $F$:

hom(  , $G$) $=$ 8sub(  , $G$) $+$ 4sub(  , $G$) $+$ 2sub(  , $G$)

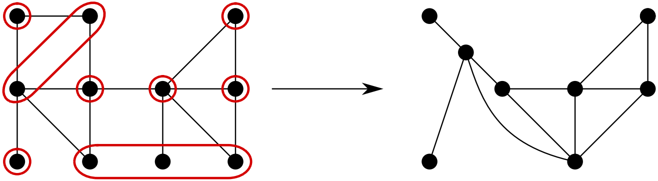
$$\Downarrow$$

$$\mathrm{hom}(H, G) = \sum_F \mathrm{surj}(H, F)\mathrm{sub}(F, G)$$

Which of the terms can be nonzero?

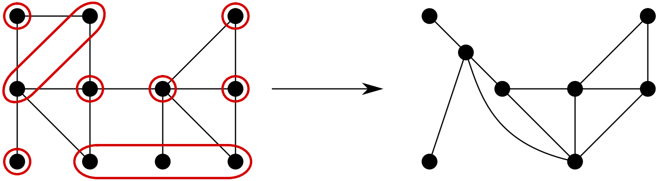
# Spasm

- $\mathrm{Part}_0(H)$: set of partitions of $V(H)$ where each class is an independent set.
- For $\Pi \in \mathrm{Part}_0(H)$, $H_{|\Pi}$ is obtained by contracting the classes of $\Pi$.

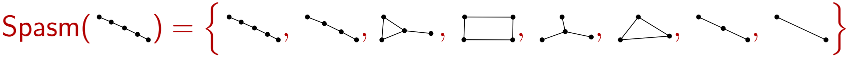


# Spasm

- $\mathrm{Part}_0(H)$: set of partitions of $V(H)$ where each class is an independent set.
- For $\Pi \in \mathrm{Part}_0(H)$, $H_{|\Pi}$ is obtained by contracting the classes of $\Pi$.



- $\mathrm{Spasm} = \{H_{|\Pi} \mid \Pi \in \mathrm{Part}_0(H)\}$



# Subgraphs $\Leftrightarrow$ homomorphisms

**From subgraphs to homomorphisms:**

$$\mathrm{hom}(H, G) = \sum_F \mathrm{surj}(H, F)\mathrm{sub}(F, G)$$

where $\mathrm{surj}(H, F) \neq 0$ if and only if $F \in \mathrm{Spasm}(H)$.

**From homomorphisms to subgraphs:** [Lovász 1967]

$$\mathrm{sub}(H, G) = \sum_F \beta_F \,\mathrm{hom}(F, G)$$

where $\beta_F \neq 0$ if and only if $F \in \mathrm{Spasm}(H)$.

## Subgraphs $\Leftrightarrow$ homomorphisms

**From subgraphs to homomorphisms:**

$$\mathrm{hom}(H, G) = \sum_F \mathrm{surj}(H, F)\mathrm{sub}(F, G)$$

where $\mathrm{surj}(H, F) \neq 0$ if and only if $F \in \mathrm{Spasm}(H)$.

... useless.

**From homomorphisms to subgraphs:** [Lovász 1967]

$$\mathrm{sub}(H, G) = \sum_F \beta_F \,\mathrm{hom}(F, G)$$

where $\beta_F \neq 0$ if and only if $F \in \mathrm{Spasm}(H)$.

# Subgraphs $\Leftrightarrow$ homomorphisms

**From subgraphs to homomorphisms:**

$$\mathrm{hom}(H, G) = \sum_{F} \mathrm{surj}(H, F)\mathrm{sub}(F, G)$$

where $\mathrm{surj}(H, F) \neq 0$ if and only if $F \in \mathrm{Spasm}(H)$.

. . . useless.

**From homomorphisms to subgraphs:** [Lovász 1967]

$$\mathrm{sub}(H, G) = \sum_{F} \beta_F \, \mathrm{hom}(F, G)$$

where $\beta_F \neq 0$ if and only if $F \in \mathrm{Spasm}(H)$.

Extremely useful for applications in algorithms and complexity!

# Algorithmic applications

$$\mathrm{sub}(H, G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_F \, \mathrm{hom}(F, G)$$

The maximum treewidth in $\mathrm{Spasm}(H)$ gives an upper bound on complexity:

### Corollary

If every graph in $\mathrm{Spasm}(H)$ has treewidth at most $c$, then $\mathrm{sub}(H, G)$ can be computed in time $O(n^{c+1})$.

# Algorithmic applications

### Corollary

If every graph in $\mathrm{Spasm}(H)$ has treewidth at most $c$, then $\mathrm{sub}(H, G)$ can be computed in time $O(n^{c+1})$.

**Observe:** If $H$ has $k$ edges, then every graph in $\mathrm{Spasm}(H)$ has at most $k$ edges.

# Algorithmic applications

**Corollary**

If every graph in $\mathrm{Spasm}(H)$ has treewidth at most $c$, then $\mathrm{sub}(H, G)$ can be computed in time $O(n^{c+1})$.

**Observe:** If $H$ has $k$ edges, then every graph in $\mathrm{Spasm}(H)$ has at most $k$ edges.

**Theorem** [Scott and Sorkin 2007]

Every graph with $\leq k$ edges has treewidth at most $0.174k + O(1)$.

**Corollary**

If $H$ has $k$ edges, then $\mathrm{sub}(H, G)$ can be computed in time $n^{0.174k+O(1)}$.

# Counting $k$-paths

**Corollary**

If $H$ has $k$ edges, then $\mathrm{sub}(H, G)$ can be computed in time $n^{0.174k+O(1)}$.

**Example:** Counting $k$-paths

- Brute force: $O(n^k)$.
- Meet in the middle [Björklund et al. 2009],[Koutis and Williams 2016]: $O(n^{0.5k})$.
- [Björklund et al. 2014]: $n^{0.455k+O(1)}$.
- New! counting homomorphisms in the spasm: $n^{0.174k+O(1)}$.

# Count small cycles

**Theorem** [Alon, Yuster, and Zwick 1997]

For $k \leq 7$, we can compute $\mathrm{sub}(C_k, G)$ in time $n^\omega$ (where $\omega < 2.373$ is the matrix-multiplication exponent).

# Count small cycles

**Theorem** [Alon, Yuster, and Zwick 1997]

For $k \leq 7$, we can compute $\text{sub}(C_k, G)$ in time $n^\omega$ (where $\omega < 2.373$ is the matrix-multiplication exponent).

We can recover this result:

- Check: if $k \leq 7$, then every graph in $\text{Spasm}(C_k, G)$ has treewidth at most $2$.
- For treewidth 2, the $O(n^{2+1})$ homomorphism algorithm can be improved to $O(n^\omega)$ with fast matrix multiplication.
- $\Rightarrow O(n^\omega)$ algorithm for $\text{sub}(C_k, G)$ if $k \leq 7$.

# Vertex cover

**Theorem**

If $H$ has vertex cover number $c$, then $\hom(H, G)$ can be computed in time $O(n^{c+1})$.

**Proof:** For $F \in \mathsf{Spasm}(H)$, we have $\mathsf{tw}(F) \leq \mathsf{vc}(F) \leq \mathsf{vc}(H) \leq c$.

**Corollary**

If $\mathcal{H}$ is a class of graphs with bounded vertex cover number, then $\#\textsc{Sub}(\mathcal{H})$ is FPT parameterized by $|V(H)|$.

(Can be improved to polynomial time.)

# Complexity applications

$$\mathrm{sub}(H, G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_F \hom(F, G)$$

**Note:** Every $\beta_F$ is nonzero.

## Complexity applications

$$\mathrm{sub}(H, G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_F \hom(F, G)$$

**Note:** Every $\beta_F$ is nonzero.

Reductions:

- **Obvious:**
  if we can compute $\hom(F, G)$ for any $F \in \mathrm{Spasm}(H)$
  $\Rightarrow$ we can compute $\mathrm{sub}(H, G)$.

# Complexity applications

$$\mathrm{sub}(H, G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_F \,\mathrm{hom}(F, G)$$

**Note:** Every $\beta_F$ is nonzero.

Reductions:

- **Obvious:**
  if we can compute $\mathrm{hom}(F, G)$ for any $F \in \mathrm{Spasm}(H)$
  $\Rightarrow$ we can compute $\mathrm{sub}(H, G)$.
- **Highly nontrivial:**
  if we can compute $\mathrm{sub}(H, G)$
  $\Rightarrow$ we can compute $\mathrm{hom}(F, G)$ for any $F \in \mathrm{Spasm}(H)$.

# Complexity applications

$$\mathrm{sub}(H, G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_F \, \mathrm{hom}(F, G)$$

**Note:** Every $\beta_F$ is nonzero.

Reductions:

- **Obvious:**
  if we can compute $\mathrm{hom}(F, G)$ for any $F \in \mathrm{Spasm}(H)$
  $\Rightarrow$ we can compute $\mathrm{sub}(H, G)$.
- **Highly nontrivial:**
  if we can compute $\mathrm{sub}(H, G)$
  $\Rightarrow$ we can compute $\mathrm{hom}(F, G)$ for any $F \in \mathrm{Spasm}(H)$.

Complexity of $\mathrm{hom}(F, G)$ for any $F \in \mathrm{Spasm}(H)$ is a **lower bound** on the complexity of $\mathrm{sub}(H, G)$.

# Matrices

Fix an enumeration of graphs with $\leq k$ edges with nondecreasing number of edges.

- Hom matrix: row $i$, column $j$ is $\hom(H_i, H_j)$.
- Sub matrix: row $i$, column $j$ is $\mathrm{sub}(H_i, H_j)$.
- Surj matrix: row $i$, column $j$ is $\mathrm{surj}(H_i, H_j)$.

# Matrices

Fix an enumeration of graphs with $\leq k$ edges with nondecreasing number of edges.

- Hom matrix: row $i$, column $j$ is $\mathrm{hom}(H_i, H_j)$.
- Sub matrix: row $i$, column $j$ is $\mathrm{sub}(H_i, H_j)$.
- Surj matrix: row $i$, column $j$ is $\mathrm{surj}(H_i, H_j)$.

$$\mathrm{hom}(H, G) = \sum_F \mathrm{surj}(H, F)\mathrm{sub}(F, G)$$
$$\Downarrow$$
$$\mathrm{Hom} = \mathrm{Surj} \cdot \mathrm{Sub}$$

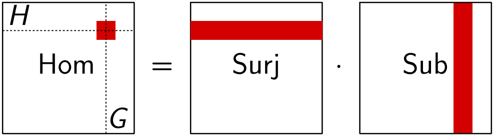
$$\boxed{\mathrm{Hom}} = \boxed{\mathrm{Surj}} \cdot \boxed{\mathrm{Sub}}$$

# Matrices

Fix an enumeration of graphs with $\leq k$ edges with nondecreasing number of edges.

- Hom matrix: row $i$, column $j$ is $\hom(H_i, H_j)$.
- Sub matrix: row $i$, column $j$ is $\mathrm{sub}(H_i, H_j)$.
- Surj matrix: row $i$, column $j$ is $\mathrm{surj}(H_i, H_j)$.

$$\hom(H, G) = \sum_F \mathrm{surj}(H, F)\mathrm{sub}(F, G)$$
$$\Downarrow$$
$$\mathrm{Hom} = \mathrm{Surj} \cdot \mathrm{Sub}$$

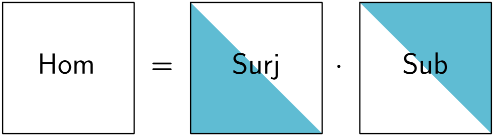


# Matrices

Fix an enumeration of graphs with $\leq k$ edges with nondecreasing number of edges.

- Hom matrix: row $i$, column $j$ is $\hom(H_i, H_j)$.
- Sub matrix: row $i$, column $j$ is $\mathrm{sub}(H_i, H_j)$.
- Surj matrix: row $i$, column $j$ is $\mathrm{surj}(H_i, H_j)$.

$$\hom(H, G) = \sum_F \mathrm{surj}(H, F)\mathrm{sub}(F, G)$$
$$\Downarrow$$
$$\mathrm{Hom} = \mathrm{Surj} \cdot \mathrm{Sub}$$




The Hom matrix is invertible!

# Categorial product

One of the standard graph products:

**Definition**

$G_1 \times G_2$ has vertex set $V(G_1) \times V(G_2)$ and $(v_1, v_2)$ and $(v_1', v_2')$ adjacent in $G_1 \times G_2 \iff v_1 v_1' \in E(G_1)$ and $v_2 v_2' \in E(G_2)$.

[missing figure]

Exercise:

$$\hom(H, G_1 \times G_2) = \hom(H, G_1) \cdot \hom(H, G_2)$$

# Extracting a term

**Lemma**

Given an algorithm for $\mathrm{sub}(H, G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_F \hom(F, G)$ (with $\beta_F \neq 0$), we can compute $\hom(F, G)$ for any $F \in \mathrm{Spasm}(H)$.

Use the algorithm on $Z \times G$ for every $Z$ with $\leq k = |E(H)|$ edges.

$$\sum_{F \in \mathrm{Spasm}(H)} \beta_F \cdot \hom(F, Z \times G) = b_Z$$

# Extracting a term

**Lemma**

Given an algorithm for $\mathsf{sub}(H, G) = \sum_{F \in \mathsf{Spasm}(H)} \beta_F \, \mathsf{hom}(F, G)$ (with $\beta_F \neq 0$), we can compute $\mathsf{hom}(F, G)$ for any $F \in \mathsf{Spasm}(H)$.

Use the algorithm on $Z \times G$ for every $Z$ with $\leq k = |E(H)|$ edges.

$$\sum_{F \in \mathsf{Spasm}(H)} \beta_F \cdot \mathsf{hom}(F, Z) \cdot \mathsf{hom}(F, G) = b_Z$$

# Extracting a term

**Lemma**

Given an algorithm for $\mathrm{sub}(H, G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_F \hom(F, G)$ (with $\beta_F \neq 0$), we can compute $\hom(F, G)$ for any $F \in \mathrm{Spasm}(H)$.

Use the algorithm on $Z \times G$ for every $Z$ with $\leq k = |E(H)|$ edges.

$$\sum_{F \in \mathrm{Spasm}(H)} \hom(F, Z) \cdot \beta_F \cdot \hom(F, G) = b_Z$$

# Extracting a term

**Lemma**

Given an algorithm for $\mathrm{sub}(H,G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_F \hom(F,G)$ (with $\beta_F \neq 0$), we can compute $\hom(F,G)$ for any $F \in \mathrm{Spasm}(H)$.

Use the algorithm on $Z \times G$ for every $Z$ with $\leq k = |E(H)|$ edges.

$$\sum_{F \in \mathrm{Spasm}(H)} \hom(F,Z) \cdot \beta_F \cdot \hom(F,G) = b_Z$$

$$\mathrm{Hom}^T \cdot \begin{matrix} \beta_{F_1} \cdot \hom(F_1, G) \\ \vdots \\ \beta_{F_t} \cdot \hom(F_t, G) \end{matrix} = \begin{matrix} b_{Z_1} \\ \vdots \\ b_{Z_t} \end{matrix}$$

# Extracting a term

**Lemma**

Given an algorithm for $\mathrm{sub}(H, G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_F \,\mathrm{hom}(F, G)$ (with $\beta_F \neq 0$), we can compute $\mathrm{hom}(F, G)$ for any $F \in \mathrm{Spasm}(H)$.

Use the algorithm on $Z \times G$ for every $Z$ with $\leq k = |E(H)|$ edges.

$$\sum_{F \in \mathrm{Spasm}(H)} \mathrm{hom}(F, Z) \cdot \beta_F \cdot \mathrm{hom}(F, G) = b_Z$$

$$\mathrm{Hom}^T \cdot \begin{pmatrix} \beta_{F_1} \cdot \mathrm{hom}(F_1, G) \\ \vdots \\ \beta_{F_t} \cdot \mathrm{hom}(F_t, G) \end{pmatrix} = \begin{pmatrix} b_{Z_1} \\ \vdots \\ b_{Z_t} \end{pmatrix}$$

The Hom matrix is invertible, so we can solve this system of equations!

# Hardness results

**Theorem**

Counting $k$-matchings is W[1]-hard.

# Hardness results

**Theorem**

Counting $k$-matchings is W[1]-hard.

**Proof:** As $K_k \in \mathrm{Spasm}(M_{\binom{k}{2}})$, counting $k$-cliques can be reduced to counting $\binom{k}{2}$-matchings. $\square$

# Hardness results

**Theorem**

Counting $k$-matchings is W[1]-hard.

**Proof:** As $K_k \in \mathrm{Spasm}(M_{\binom{k}{2}})$, counting $k$-cliques can be reduced to counting $\binom{k}{2}$-matchings. $\square$

- With standard techniques, we can show that there is no $f(k)n^{o(k/\log k)}$ time algorithm, assuming ETH.
- For other counting problems, hardness boils down to finding a graph of large treewidth in the spasm.

# Hardness results

**Theorem**

If $\mathcal{H}$ is a class of graphs with unbounded vertex cover number, then $\#\text{Sub}(\mathcal{H})$ is W[1]-hard.

**Proof:**

- Let $\mathcal{H}' = \bigcup_{H \in \mathcal{H}} \text{Spasm}(H)$.
- **Lemma:** If $H$ has vertex cover numer $k$, then $\text{Spasm}(H)$ contains a graph with treewidth $\Omega(k)$.
- As $\mathcal{H}$ has unbounded vertex cover number, $\mathcal{H}'$ has unbounded treewidth.
- Thus $\#\text{Hom}(\mathcal{H}')$ is W[1]-hard [Dalmau and Jonsson 2004].
- We can reduce $\#\text{Hom}(\mathcal{H}')$ to $\#\text{Sub}(\mathcal{H})$. $\square$

# Dichotomy result

### Theorem

Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\text{Sub}(\mathcal{H})$ is polynomial-time solvable.
2. $\#\text{Sub}(\mathcal{H})$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded vertex cover number.

# Dichotomy result

**Theorem**

Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\textsc{Sub}(\mathcal{H})$ is polynomial-time solvable.
2. $\#\textsc{Sub}(\mathcal{H})$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded vertex cover number.

**Ingredients:**

- Formula $\mathrm{sub}(H,G) = \sum_{F \in \mathrm{Spasm}(H)} \beta_{F,H}\, \mathrm{hom}(F,G)$.
- Complexity of $\mathrm{hom}(F,G)$ is well understood from earlier work: treewidth determines it.
- **Algorithmic result:** bounded vc-number implies that treewidth is bounded in the spasm.
- **Hardness result:**
  1. For $F \in \mathrm{Spasm}(H)$, $\mathrm{hom}(F,G)$ can be reduced to $\mathrm{hom}(H,G)$.
  2. If vc-number is unbounded, then the spasm contains graphs of large treewidth.

# Outlook

- Similar approach for counting **induced** subgraphs.
- Graph motif parameters: those that can be computed from counting induced subgraphs of bounded size.
- Linear combination of homomorphisms seems to be the most fundamental form of description.

# Advertisement

**Postdoc positions available in parameterized algorithms and complexity!**

Institute for Computer Science and Control
Hungarian Academy of Sciences
Budapest, Hungary