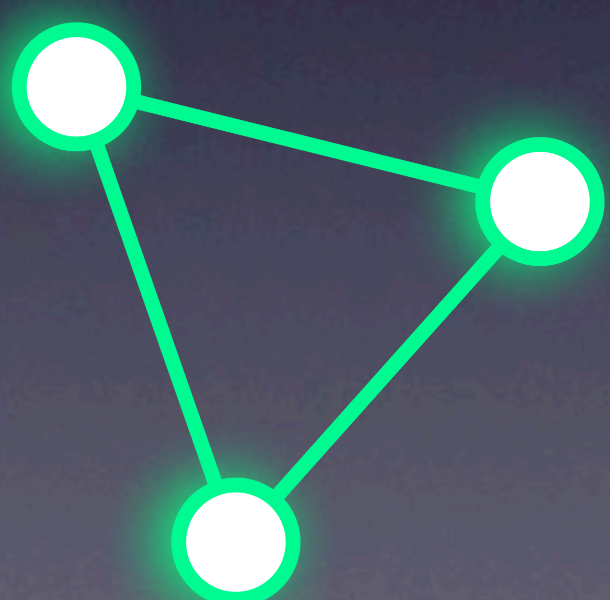
Selected surprises in subgraph counting

Petteri Kaski

Helsinki Institute for Information Technology HIIT &
Department of Information and Computer Science
Aalto University, Helsinki

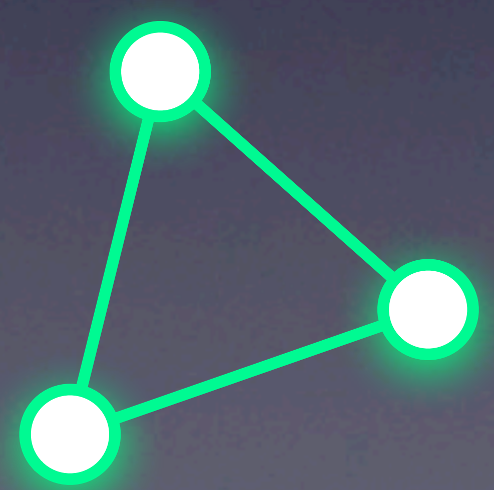


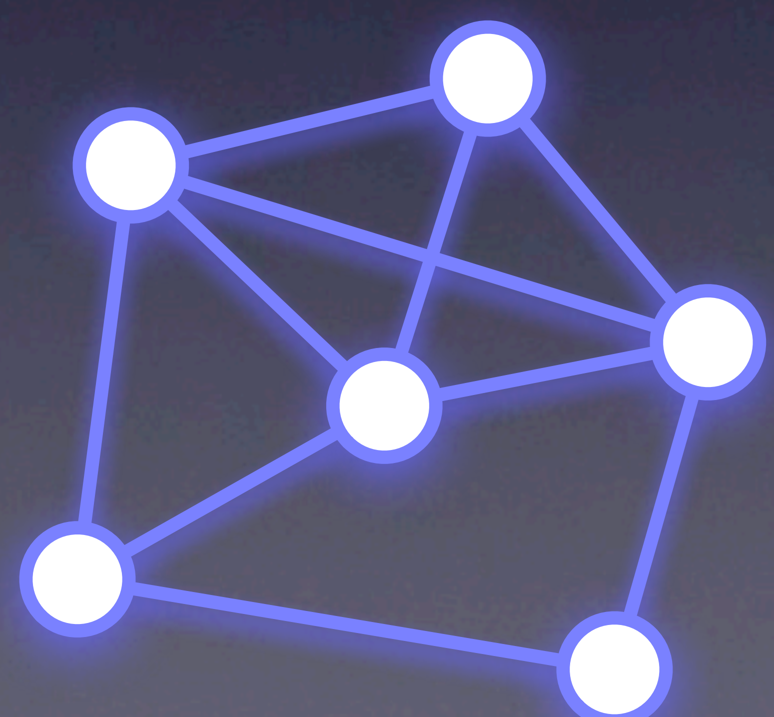
24th Estonian Theory Days
Saka Mõis
26 October 2013

Aalto University

Combinatorics

- Existence
  - Is there a solution ?
- Enumeration
  - How many distinct solutions are there ?

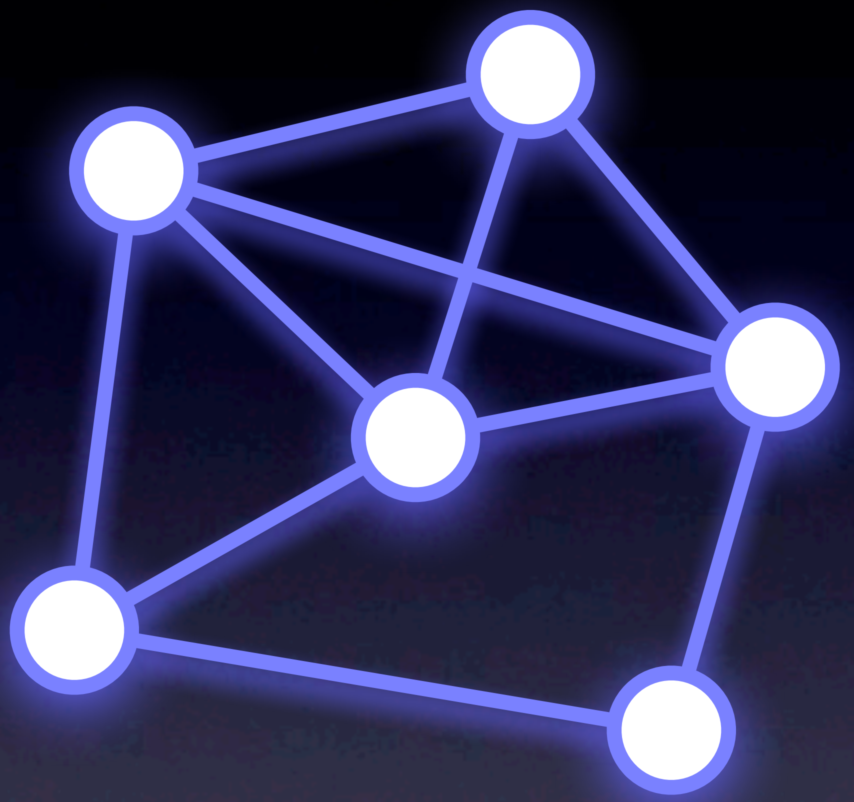




Algorithms / Complexity

- What resources are sufficient / necessary to ...
    - ... decide whether a solution exists
    - ... enumerate the solutions
- Resources (worst-case asymptotic)
    - running time
    - space usage

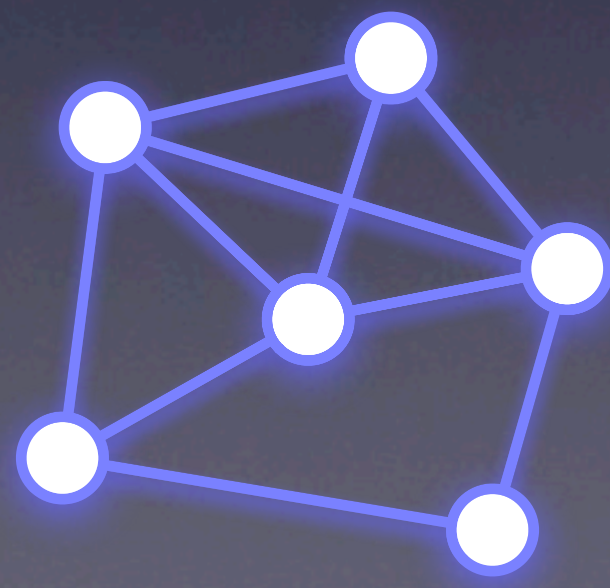
Subgraph problems



- Input:
  - **Host** graph H
- Task (existence):
  - Is there a subgraph of H with property P ?
- Task (enumeration):
  - How many subgraphs of H have property P ?

Graph inputs

- Resource usage is measured as a function of the host graph
  - number of vertices (=n)
  - number of edges (=m)
  - a problem-specific parameter (=k)



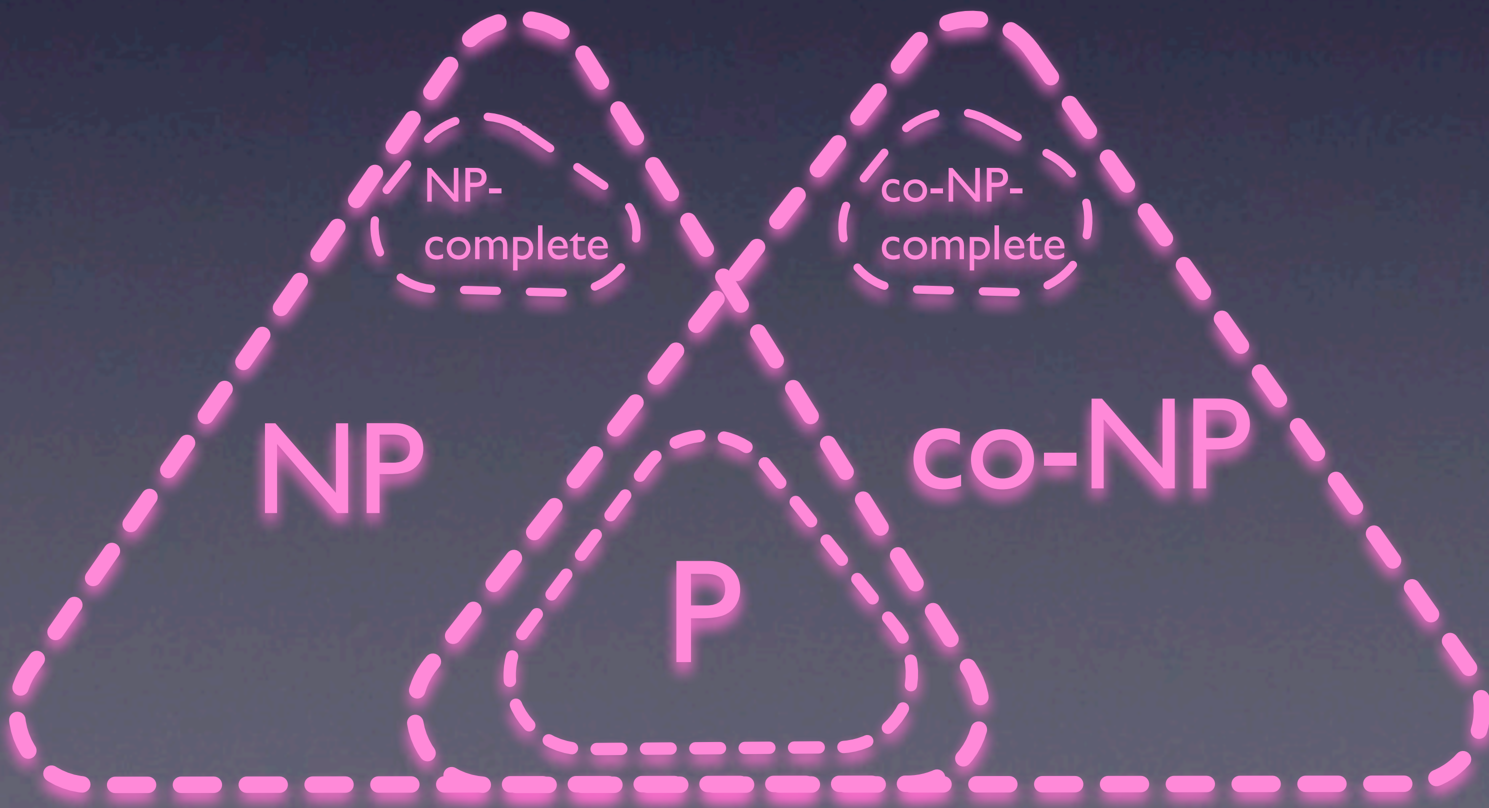
Existence: Triangle?

Enumeration: #Triangles

Classical dichotomy

- Tractable problem
  ~ polynomial resources suffice
- Intractable problem
  ~ super-polynomial resources necessary
  (*conjectured* necessary)

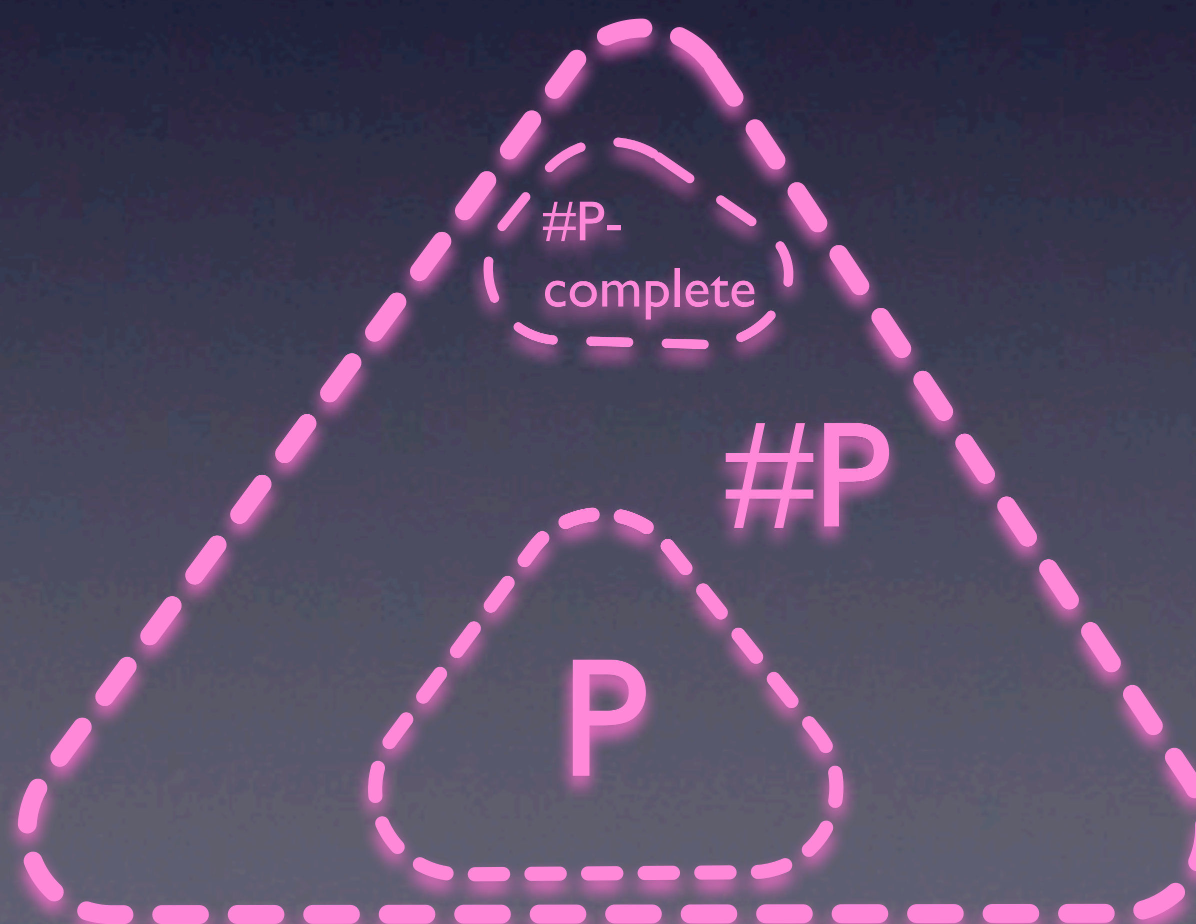
Existence
(Cook, Levin)



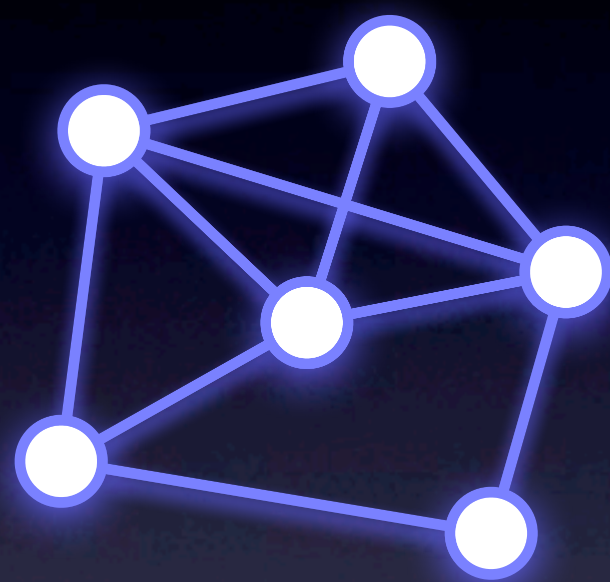
Classical dichotomy

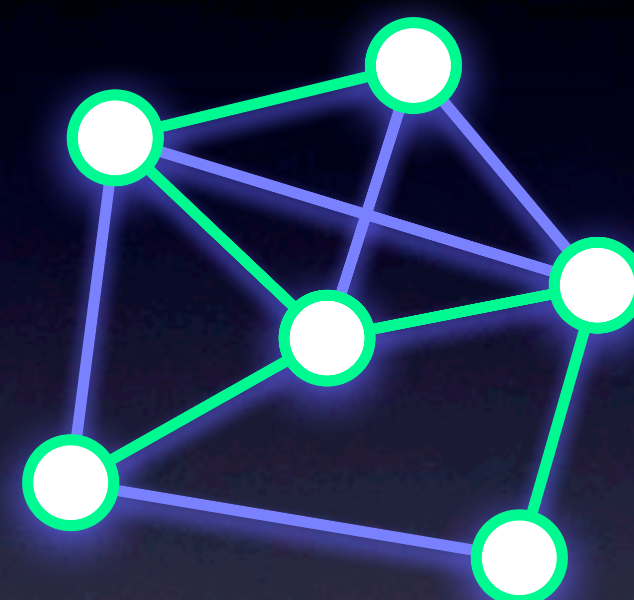
- Tractable problem
  ~ polynomial resources suffice
- Intractable problem
  ~ super-polynomial resources necessary
  (*conjectured* necessary)

Enumeration
(Valiant)



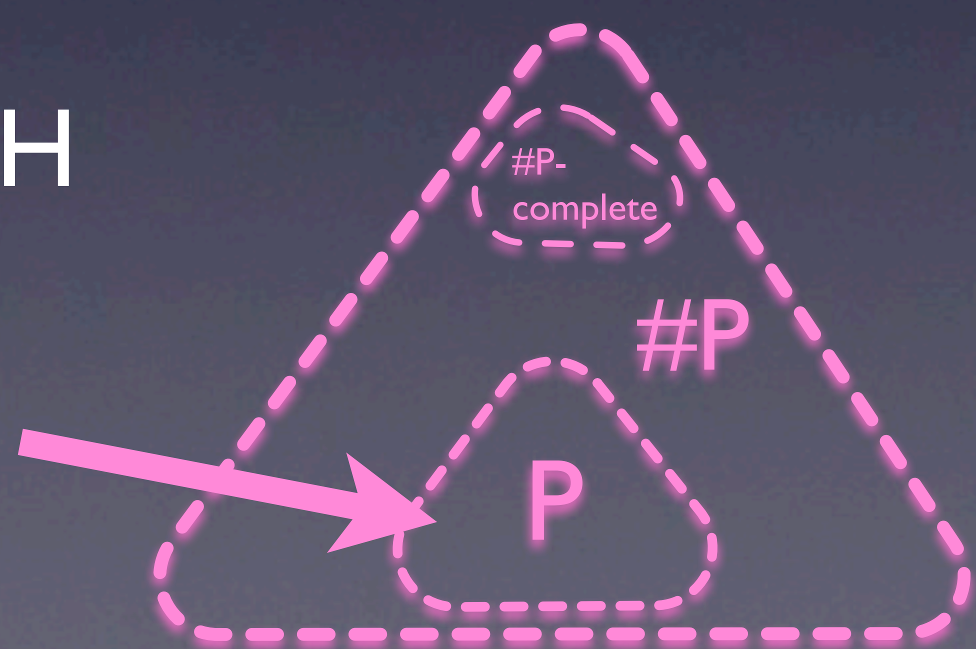
Example: #Spanning Trees



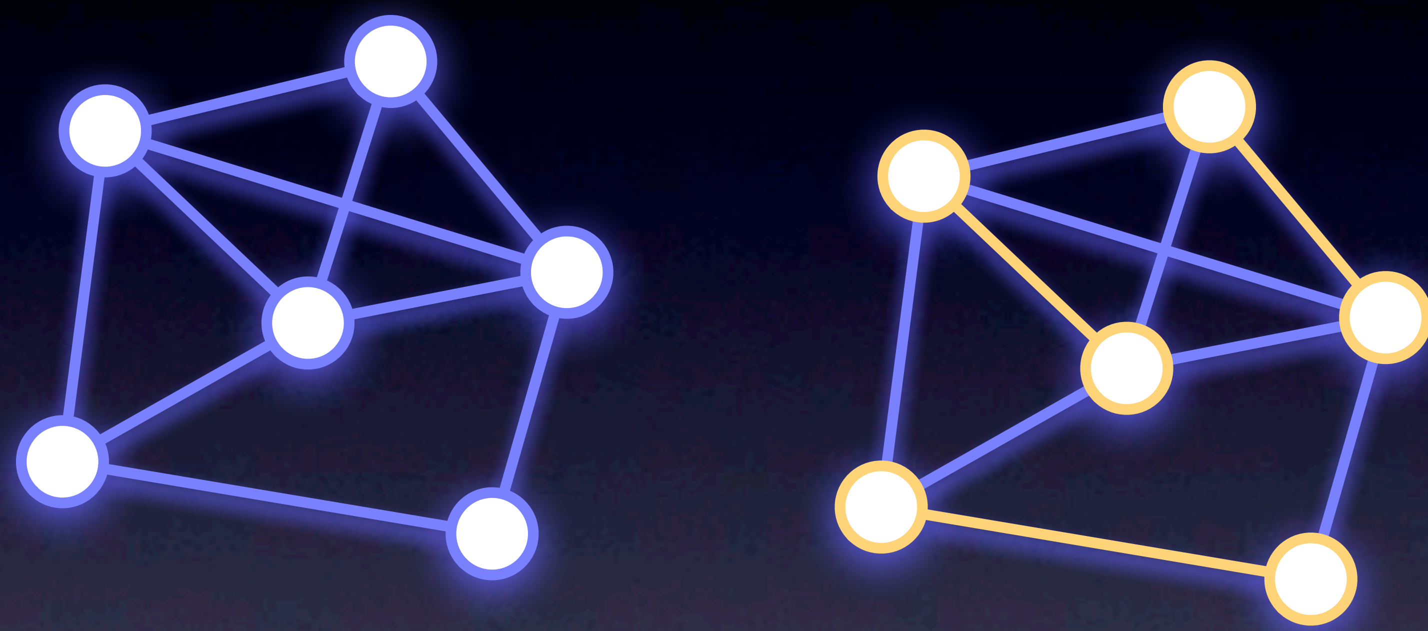


- Input: A graph H
- Enumerate: the spanning trees in H

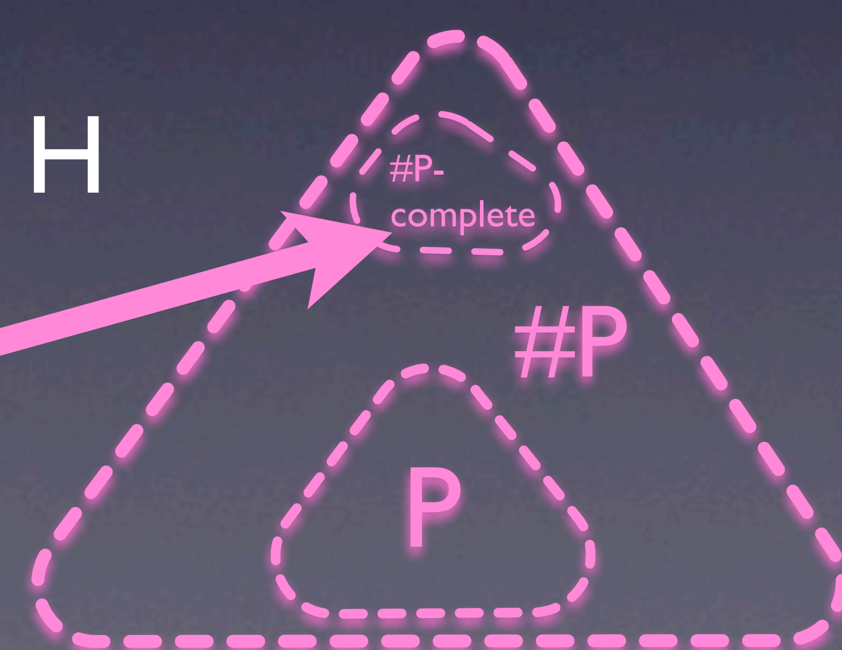
polynomial-time solvable (Kirchoff)



Example:
#Perfect Matchings



- Input: A graph H
- Enumerate: the perfect matchings in H



#P-complete (Valiant)

Coping with intractability

- Super-polynomial resources (*conjectured*) necessary

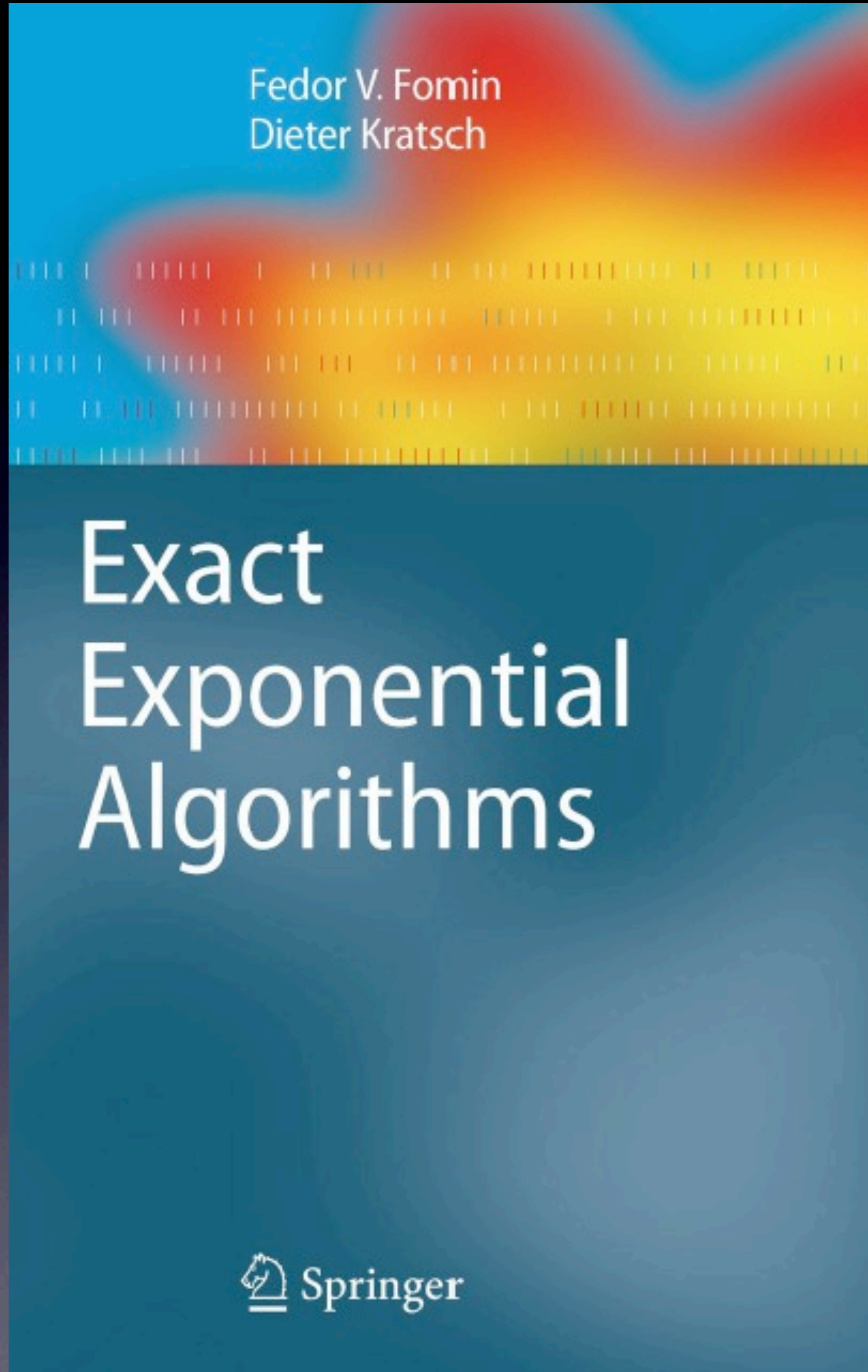
  ~ all right,
  but what is the best we can do ?

  ~ super-polynomial (≈ exponential) is
  a lot of resources, so surely we can do better
  than brute-force ?

  → Exact exponential algorithms

  ~ what is it that makes the problem hard ?

  → Parameterized algorithms



(Fomin & Kratsch 2010)

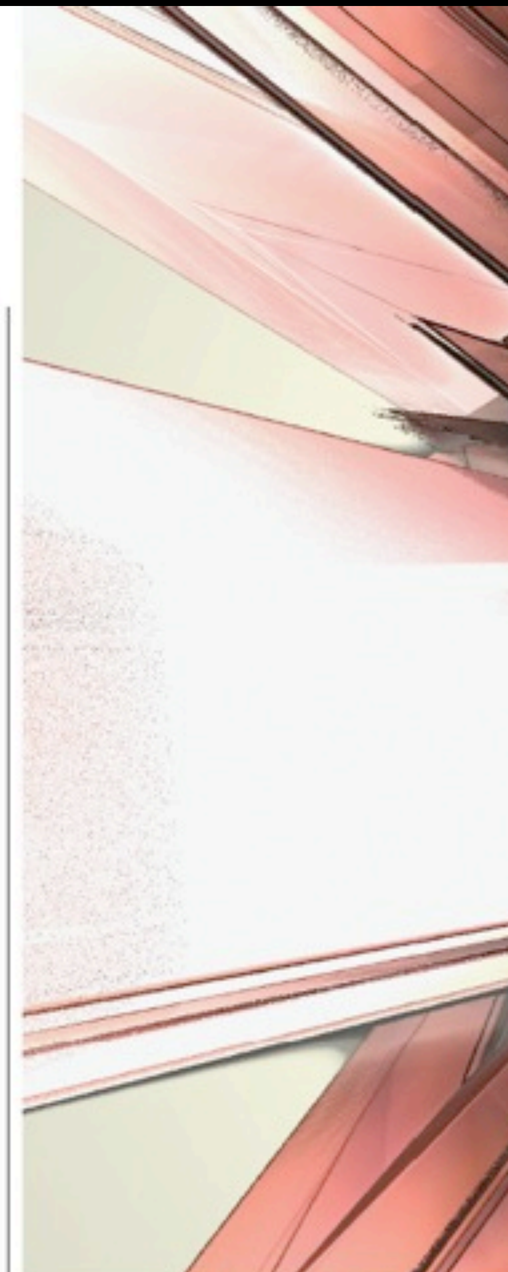
review articles

DOI:10.1145/2428556.2428575

Discovering surprises in the face of intractability.

BY FEDOR V. FOMIN AND PETTERI KASKI

Exact Exponential Algorithms



MANY COMPUTATIONAL PROBLEMS have been shown to be intractable, either in the strong sense that no algorithm exists at all—the canonical example being the undecidability of the Halting Problem—or that no *efficient* algorithm exists. From a theoretical perspective perhaps the most intriguing case occurs with the family of *NP*-complete problems, for which *it is not known* whether the problems are intractable. That is, despite extensive research, neither is an efficient algorithm known, nor has the existence of one been rigorously ruled out.[16]

To cope with intractability, advanced techniques such as *parameterized algorithms*[10,13,31] (that isolate the exponential complexity to a specific structural parameter of a problem instance) and *approximation algorithms*[34] (that produce a solution whose value is guaranteed to be within a known factor of the value of an optimum solution) have been developed. But what can we say about finding exact solutions of non-parameterized instances of intractable problems? At first glance, the general case of an *NP*-complete problem is a formidable opponent: when faced with a problem whose instances

» key insights

- While it remains open whether or not *P* equals *NP*, significant progress in the area of exhaustive search has been made in the last few years. In particular, many *NP*-complete problems can now be solved significantly faster by exhaustive search. The area of exact exponential algorithms studies the design of such techniques.
- While many exact exponential algorithms date back to the early days of computing, a number of beautiful surprises have emerged recently.

(Fomin & K., CACM, March 2013)

Motivation for this talk

- “Most” subgraph counting problems are hard
  - #P-complete when unparameterized
  - #W[1]-hard when parameterized with the “natural” parameter k
- Yet, there exist “positive surprises” in the form of algorithms that are substantially more efficient than brute force ...
- ... we review selected examples

Outline

- Selected techniques in subgraph counting
  - Inclusion–exclusion & linear equations
  - Split-and-list & fast matrix multiplication
  - Zeta and Möbius transforms
- Applications (“selected surprises”):
  - #Perfect Matchings
  - Deletion–contraction & Tutte polynomial
  - #k-Matchings

Example:
#Perfect Matchings

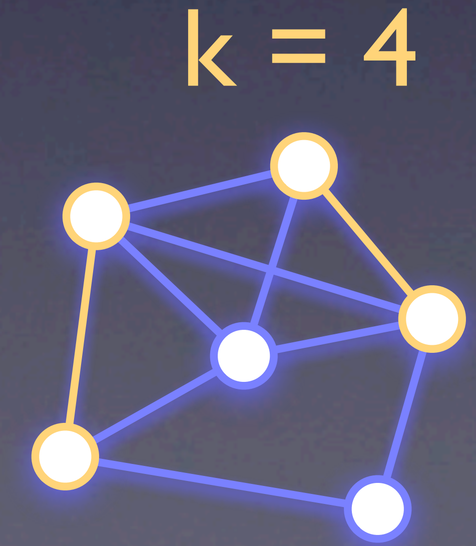
Time	Space	
$\mathcal{O}^*(n!)$	$O^*(1)$	(Brute force)
$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(2^n)$	(Dynamic programming)
$\mathcal{O}^*(2^n)$	$O^*(1)$	Björklund & Husfeldt 2008
$O^*(1.733^n)$	$O^*(1.733^n)$	
$\mathcal{O}^*(1.619^n)$	$\mathcal{O}^*(1.619^n)$	Koivisto 2009
$O^*(1.942^n)$	$O^*(1)$	Nederlof 2010
$\mathcal{O}^*(1.415^n)$	$O^*(1)$	Björklund 2012
		Cygan & Pilipczuk 2013

n = number of vertices

$\mathcal{O}^*(\ )$ suppresses a factor polynomial in *n*

Example: #k-Matchings

- k-matching
  = matching that touches k vertices
- A perfect matching has k = n
- What happens if we keep
  k fixed and let n → ∞?
- Complexity:
  #W[1]-complete (Curticapean 2013)



Example: #k-Matchings

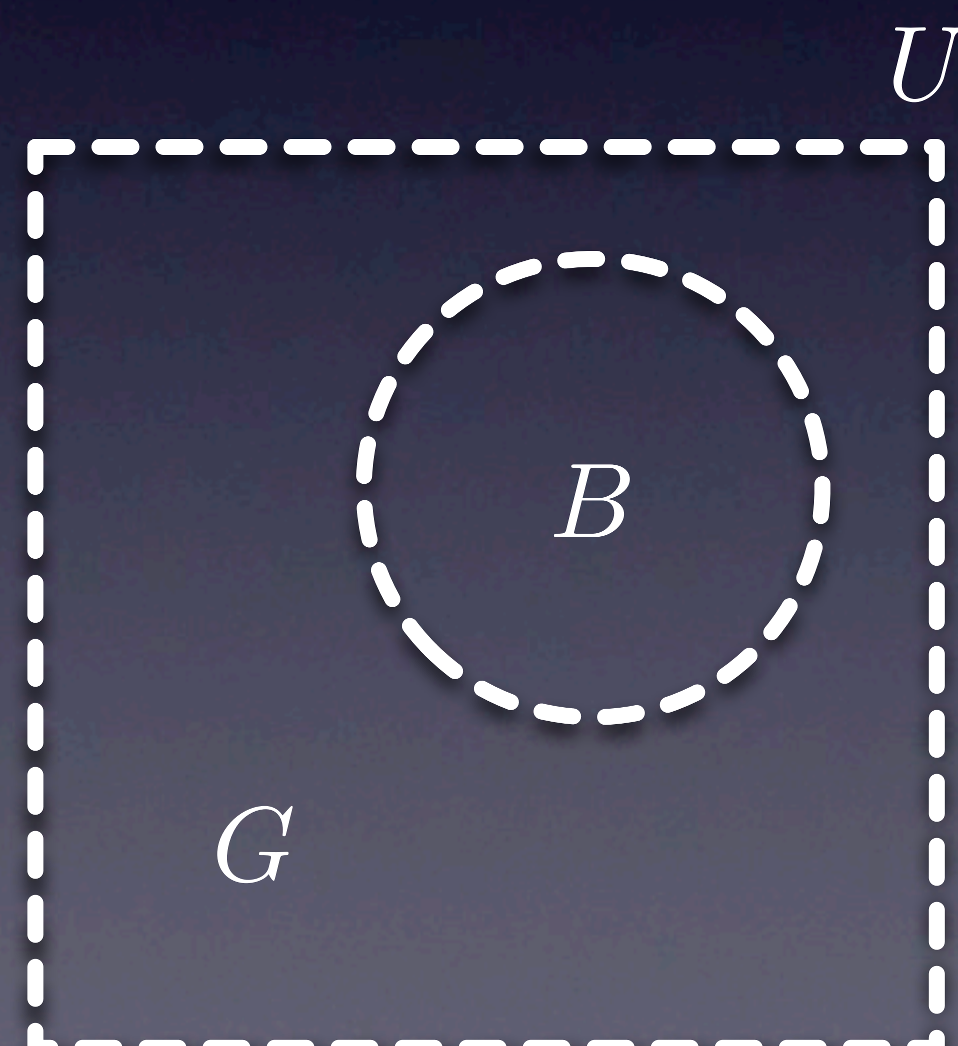
Time	
$n^{k+O(1)}$	(Brute force)
$n^{\omega k/3+O(1)}$	Nešetřil & Poljak 1985
$n^{k/2+O(1)}$	Vassilevska & Williams 2009
	Koutis & Williams 2009
	Björklund, Husfeldt, K. & Koivisto 2009
$n^{0.4547k+O(1)}$	Björklund, K. & Kowalik 2014

$2 \leq \omega < 2.3727$ (Vassilevska Williams 2012)

The good, the bad, and the universe

- Good objects G ⊆ U
- Bad objects B = U \ G
- All objects (the universe U)
- |G| = |U| – |B|

- |U| and |B| are easy to count?
- ... if so, then |G| is easy too!

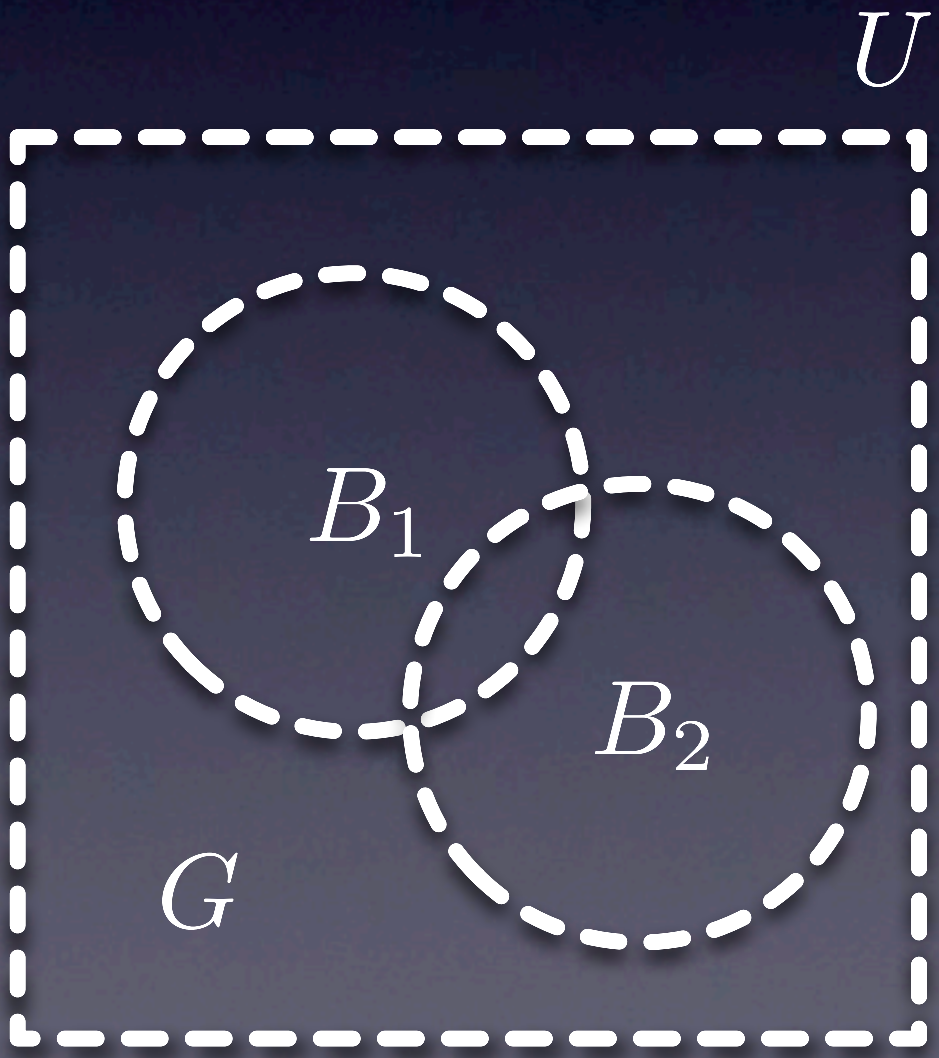


Two ways to be bad?

$$|G| = |U| - |B_1| - |B_2| + |B_1 \cap B_2|$$

Easy to compute?

... then so is |G|

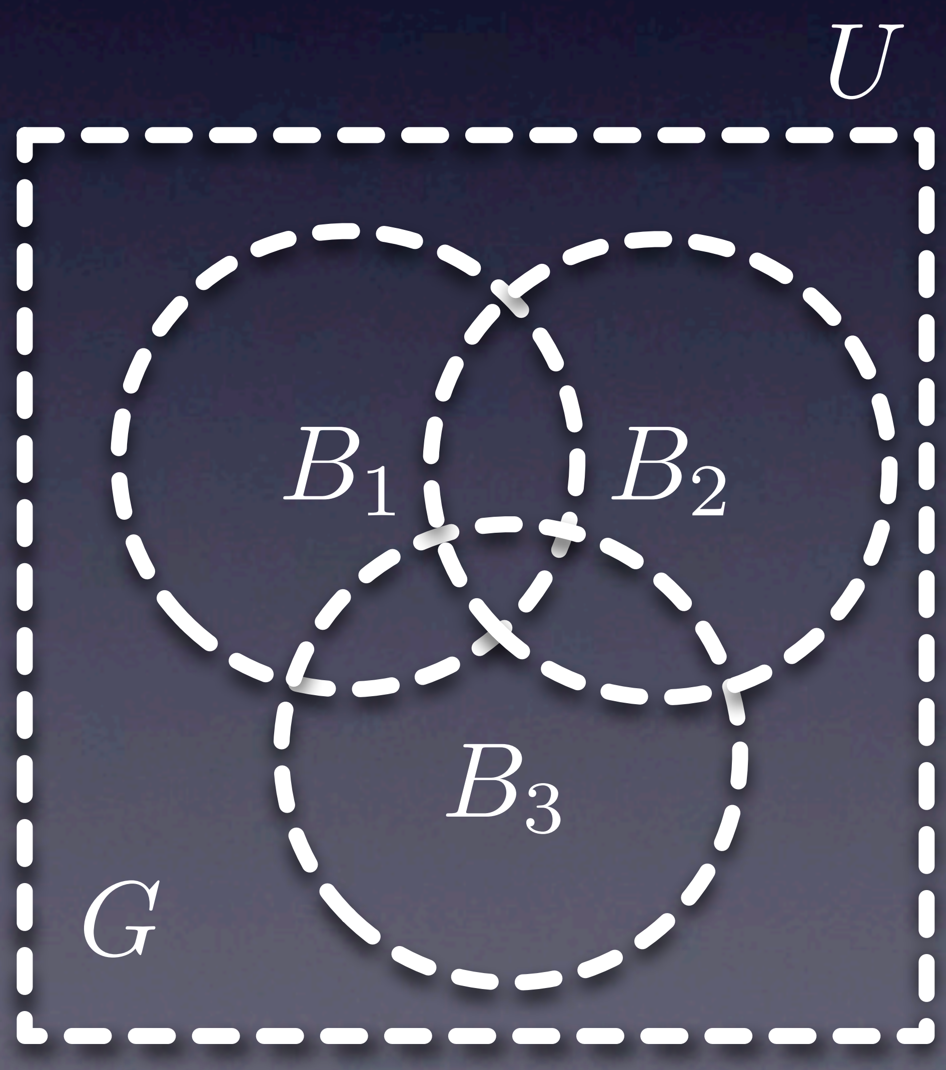


Three ways to be bad?

$$\begin{aligned}|G| = |U| \\ &- |B_1| - |B_2| - |B_3| \\ &+ |B_1 \cap B_2| + |B_1 \cap B_3| + |B_2 \cap B_3| \\ &- |B_1 \cap B_2 \cap B_3|\end{aligned}$$

Easy to compute?

... then so is |G|



The principle of inclusion and exclusion

- Let U be a finite **universe**
- Let $B_1, B_2, \ldots, B_n \subseteq U$ be **bad** properties
- An $x \in U$ is **good** if it has no bad property
- Let $G \subseteq U$ be the set of all good objects
- Then,

$$|G| = \sum_{I \subseteq \{1,2,\ldots,n\}} (-1)^{|I|} \left| \bigcap_{j \in I} B_j \right|$$

Sum with 2^n terms

Easy to compute?

Surprise 1:
#Perfect Matchings

Warmup: #Perfect Matchings in $O^*(2^n)$ time and $O^*(1)$ space

- #P-complete (Valiant 1979)
- $O^*(2^n)$ time and $O^*(1)$ space (Björklund & Husfeldt 2008)

The bad, the good, ...

- Input: Graph H with vertex set {1, 2, ..., n}

- Object = Ordered n/2-tuple of edges of H
- U = all objects
- B_j = objects that do not touch vertex j = 1, 2, ..., n
- G = objects that touch every vertex j = 1, 2, ..., n

- #Perfect Matchings in H = |G| / (n/2)!

Algorithm

#Perfect Matchings in H =

$$= \frac{1}{(n/2)!} \sum_{I \subseteq \{1,2,\ldots,n\}} (-1)^{|I|} \left| \bigcap_{j \in I} B_j \right|$$

$$= \frac{1}{(n/2)!} \sum_{I \subseteq \{1,2,\ldots,n\}} (-1)^{|I|} \underline{\underline{m(H[\{1,2,\ldots,n\} \setminus I])}}^{n/2}$$

Number of edges in the subgraph of H with vertices in I deleted

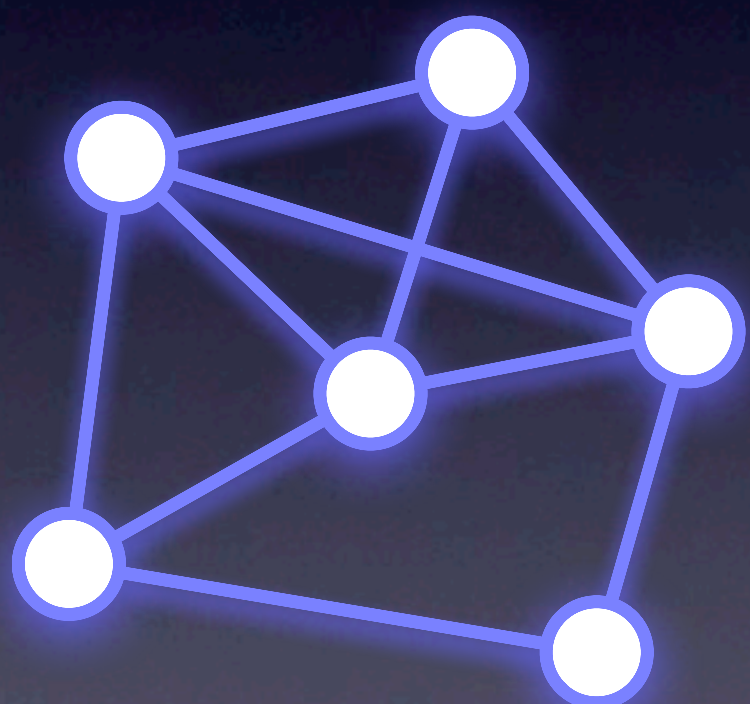
Time $O^*(2^n)$, space $O^*(1)$ to evaluate the sum

Current best:
#Perfect Matchings
in $O^*(2^{n/2})$ time
and $O^*(1)$ space

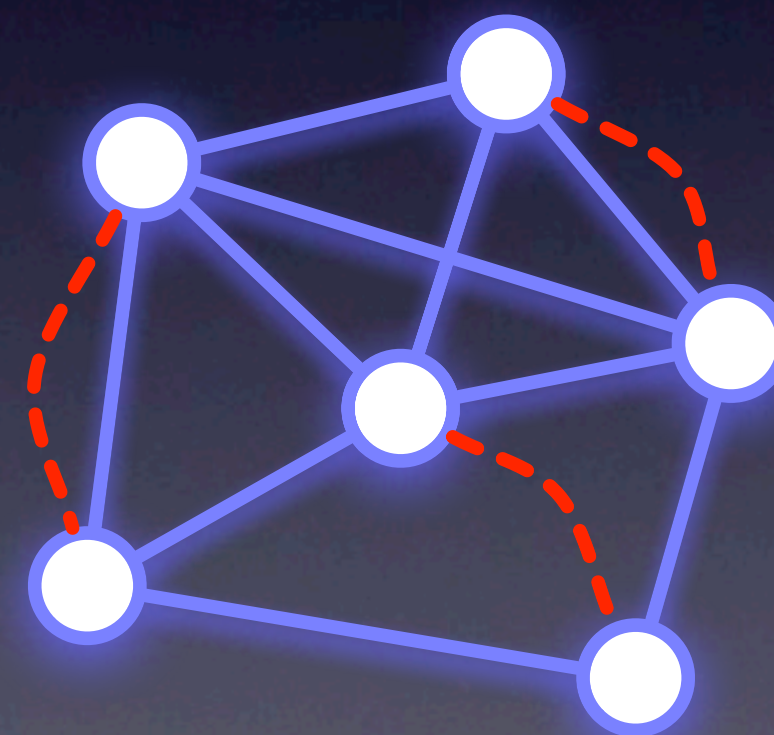
(Björklund 2012)
(Cygan & Pilipczuk 2013)

Key trick

Insert n/2 fixed
“virtual edges”
that form
a perfect matching

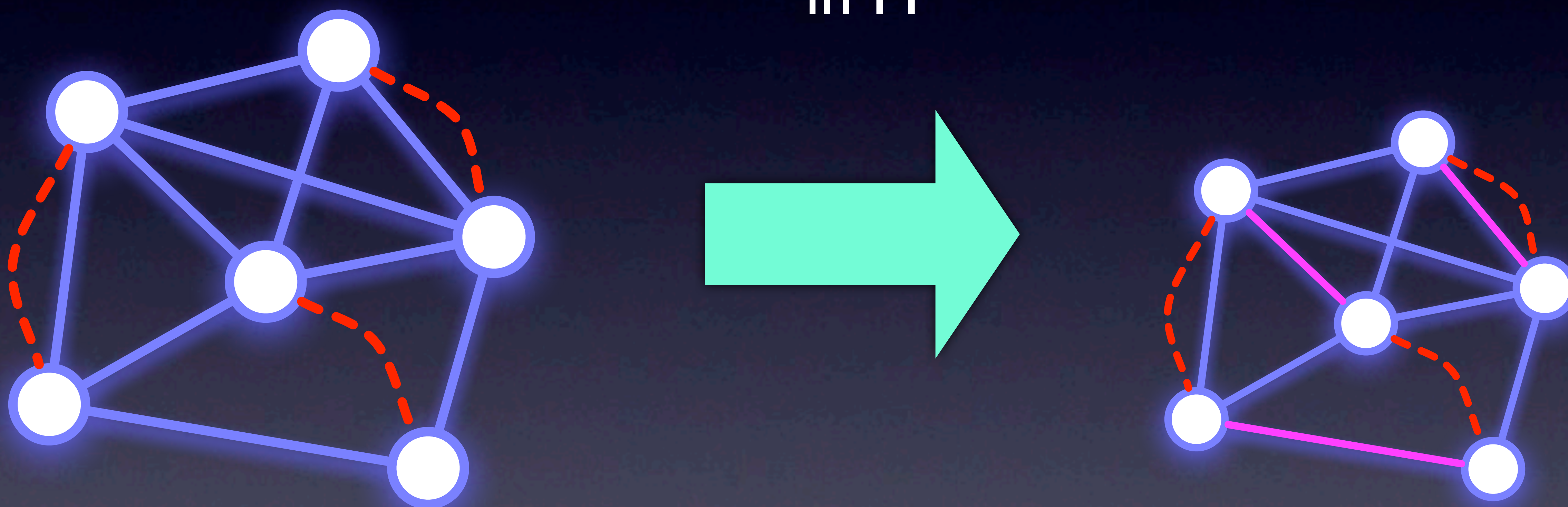


H
(input, n vertices)



Key observation

Now take any
perfect matching
in H



We get a
set of alternating symmetric
closed walks that traverses
every virtual edge

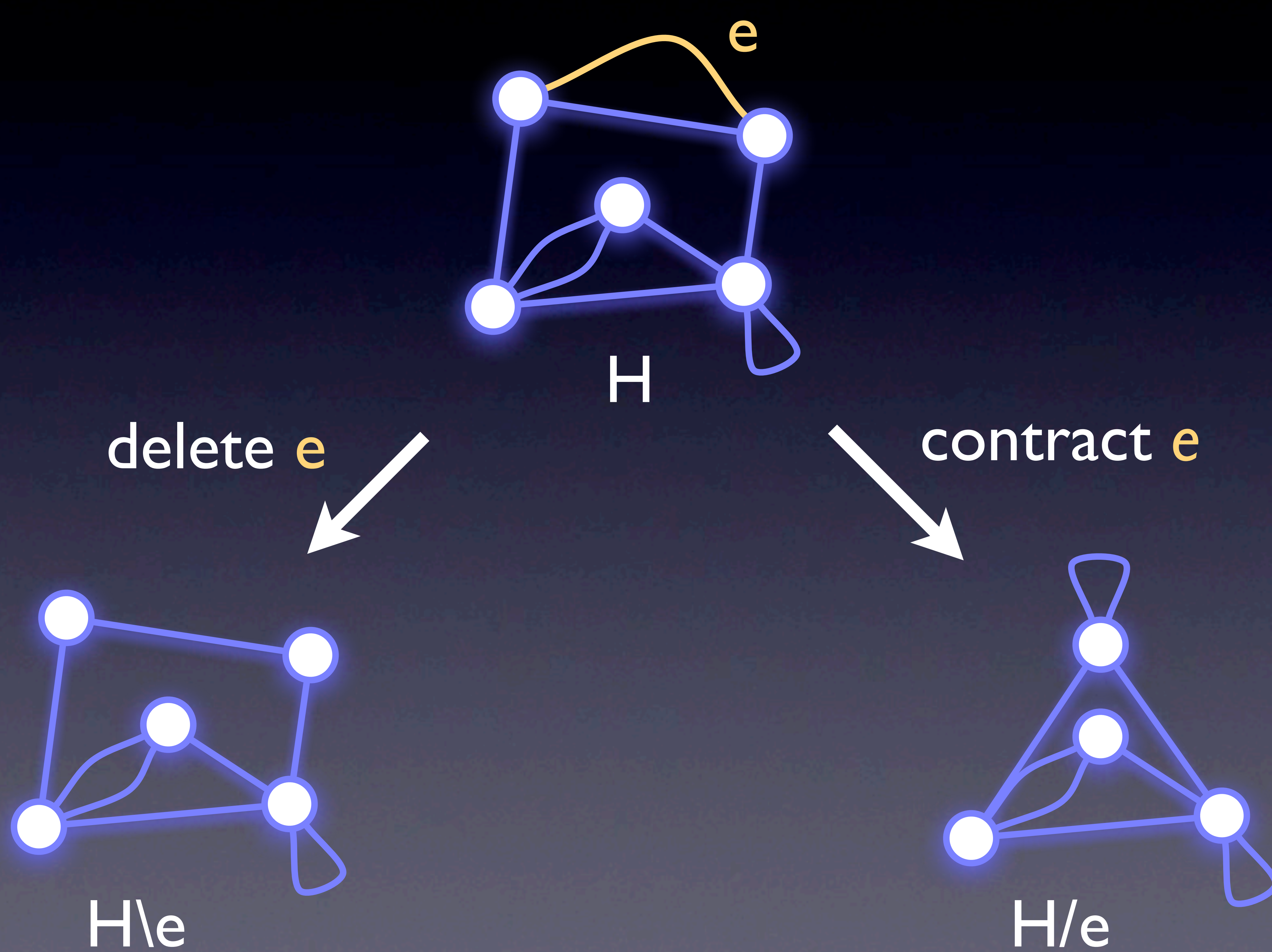
The bad, the good, ...

- Input: Graph H with vertex set {1, 2, ..., n}
- Object = Multiset of alternating symmetric*
  closed walks with n edges
- B_e = objects that do not traverse virtual edge e
- G = objects that traverse every virtual edge
  (= perfect matchings)
- #Perfect Matchings in H = |G|
- Time $O^*(2^{n/2})$, space $O^*(1)$

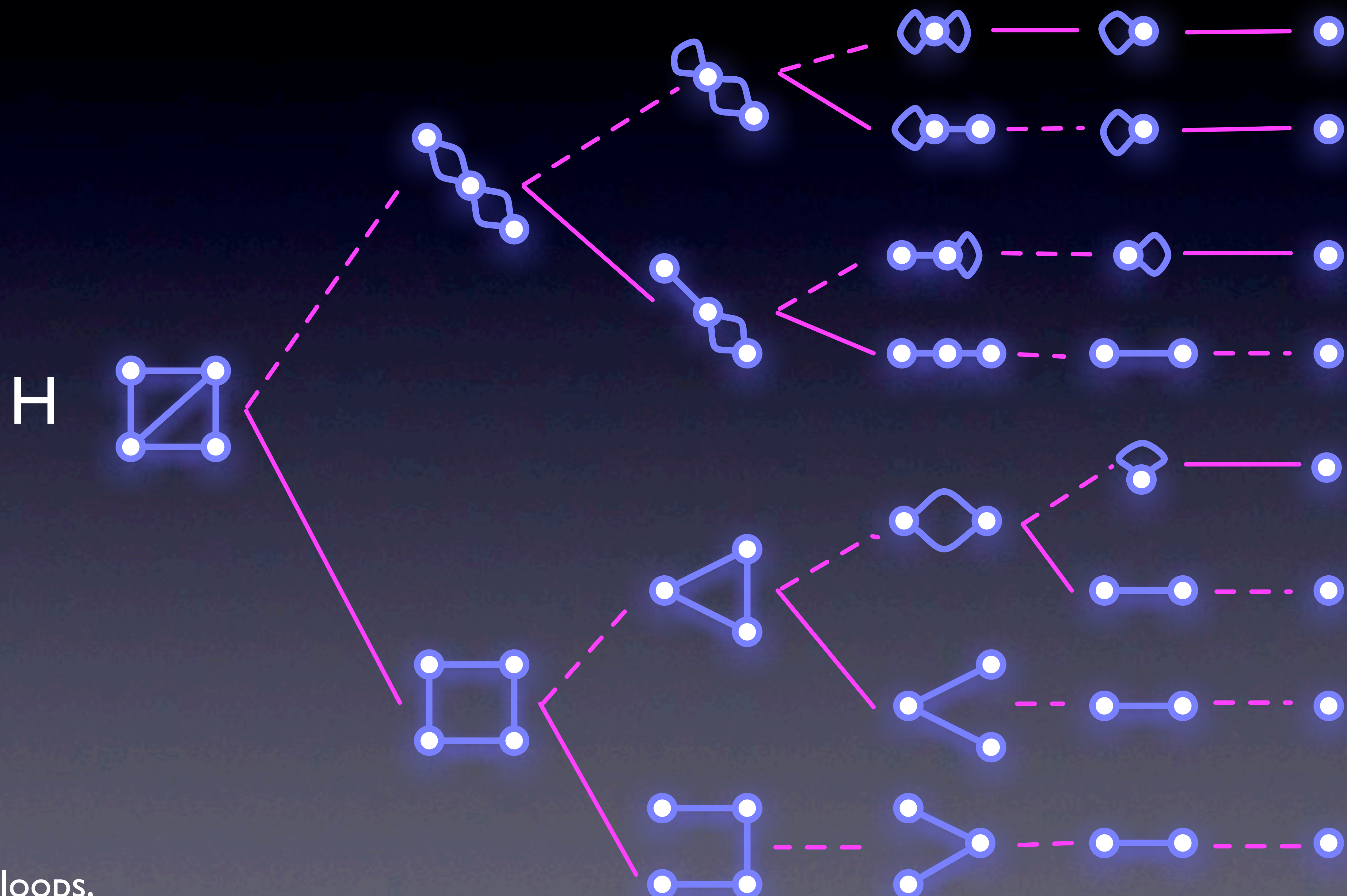
*
1) undirected/unoriented &
2) no individualized
start/end vertex

Surprise 2:
Deletion–contraction

Deletion and contraction



Deletion–contraction tree



(delete loops,
contract cut-edges)

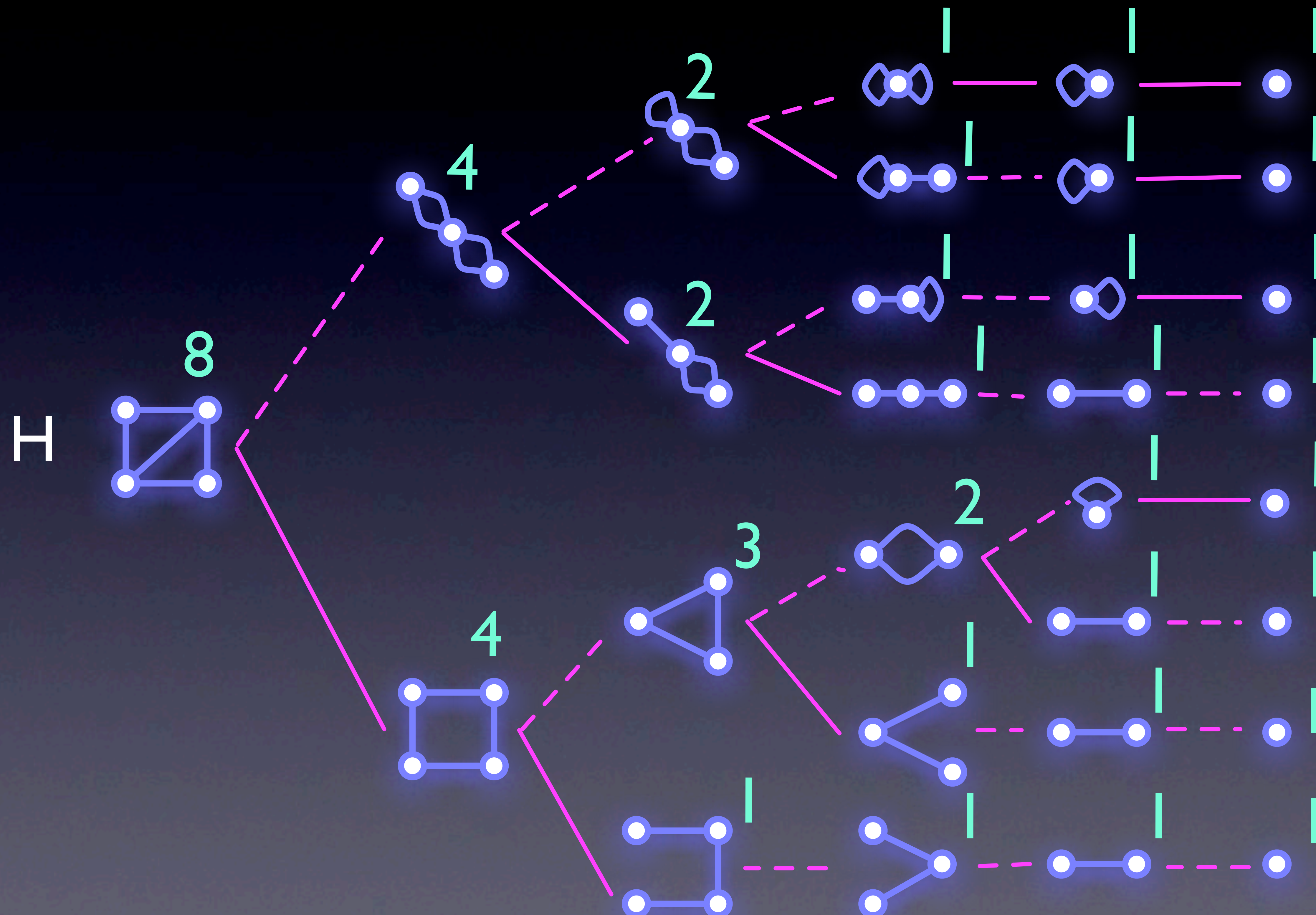
Deletion–contraction recurrences

- Many basic graph invariants f(H) admit a recurrence that expresses f(H) in terms of f(H\e) and f(H/e), with three cases:
  - e is a loop
  - e is a cut-edge
  - e is neither of the above

Example 1: #Spanning Trees

- Let H be connected and let $\tau(H)$ be the number of spanning trees in H
- Then,
  - $\tau(H) = 1$ if H has no edges
  - $\tau(H) = \tau(H \backslash e)$ if e is a loop
  - $\tau(H) = \tau(H / e)$ if e is a cut-edge
  - $\tau(H) = \tau(H \backslash e) + \tau(H / e)$ otherwise

#Spanning Trees

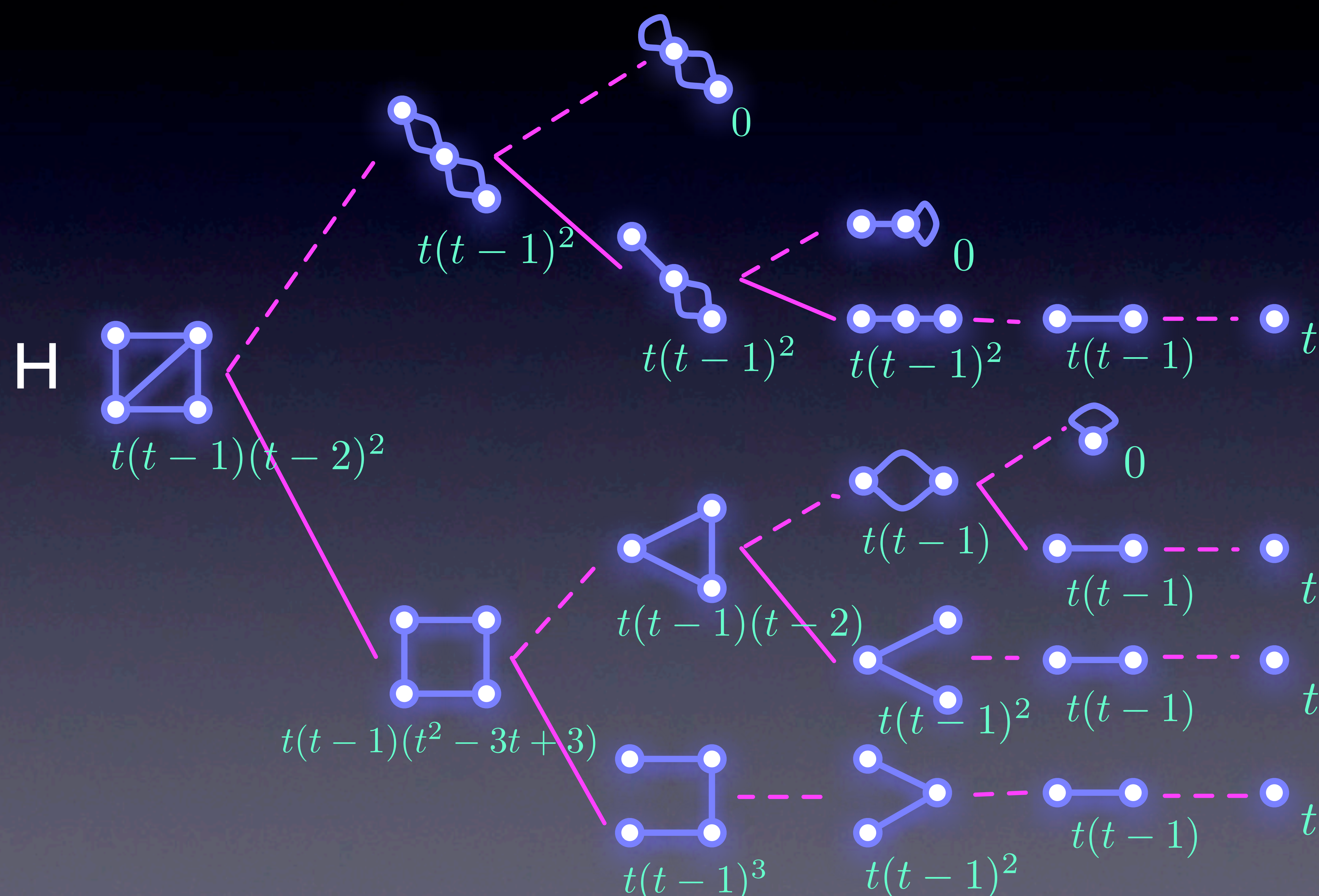


Example 2: #Graph Coloring

- Let $P_H(t)$ be the number of proper colorings of the vertices of *H* with t colors
- Then,

$$P_H(t) = \begin{cases} t^n & \text{if } H \text{ has no edges,} \\ 0 & \text{if } e \text{ has a loop,} \\ (t-1)P_{H/e}(t) & \text{if } e \text{ is a cut-edge,} \\ P_{H\backslash e}(t) - P_{H/e}(t) & \text{otherwise} \end{cases}$$

#Graph Coloring



The Tutte polynomial

Every undirected multigraph H has an associated polynomial in two indeterminates *x*, *y*

$$T_H(x,y) = \begin{cases} 1 & \text{if } H \text{ has no edges,} \\ yT_{H\setminus e}(x,y) & \text{if } e \text{ has a loop,} \\ xT_{H/e}(x,y) & \text{if } e \text{ is a cut-edge,} \\ T_{H\setminus e}(x,y) + T_{H/e}(x,y) & \text{otherwise} \end{cases}$$

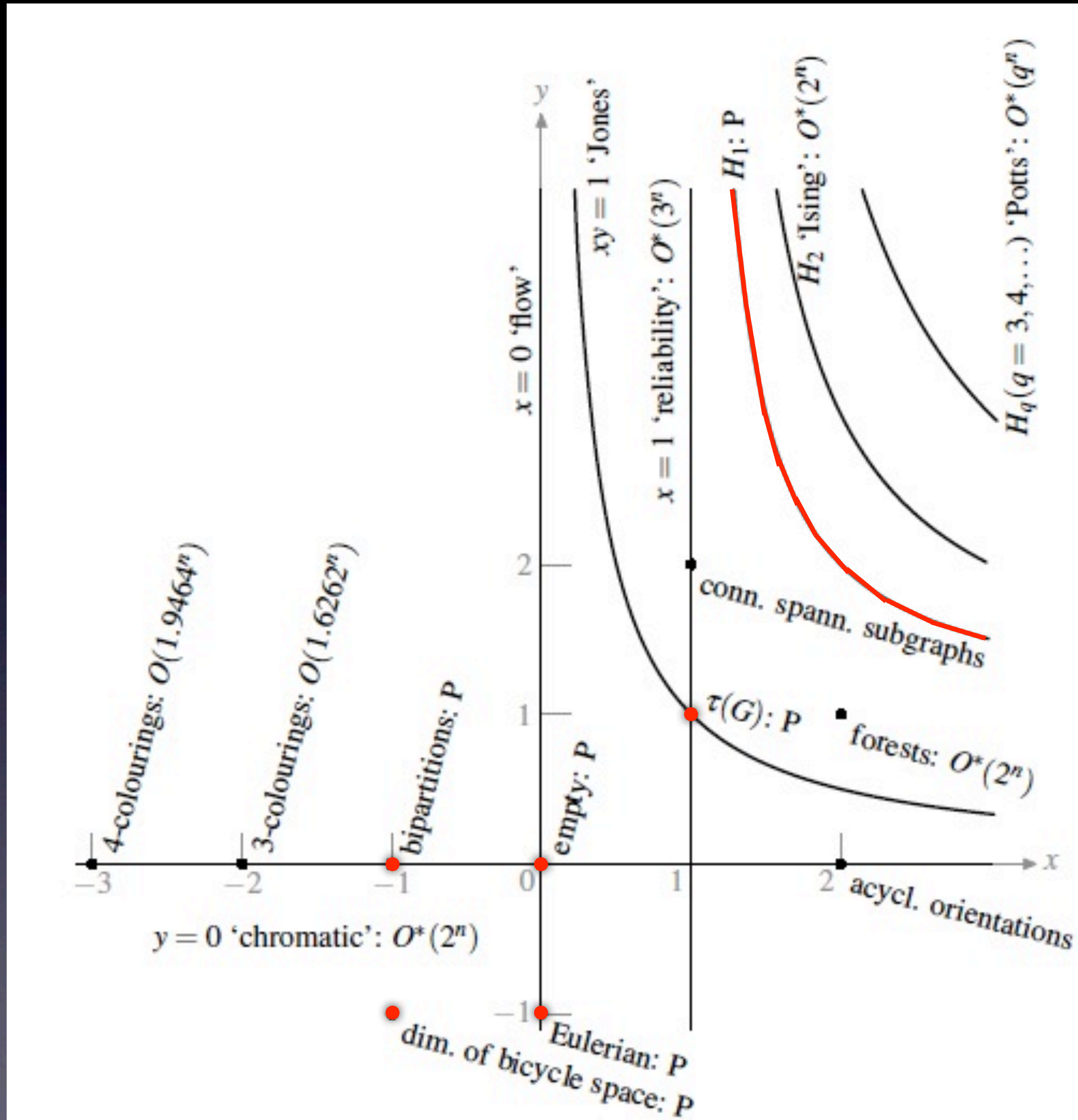
The Tutte polynomial is a universal invariant

- “Recipe Theorem” (Oxley & Welsh 1979)

  Every constant-coefficient deletion-contraction recurrence is (*) an evaluation of the Tutte polynomial at a specific point (x,y)

- (*) up to an “easily computable” multiplicative constant

An atlas of the Tutte plane



LONDON MATHEMATICAL SOCIETY
LECTURE NOTE SERIES

Edited by Professor J. W. S. Cassels

Department of Pure Mathematics and Mathematical Statistics
16 Mill Lane, Cambridge, CB2 1SB, England

with the assistance of
G. R. Allan (*Cambridge*)
P. M. Cohn (*London*)
C. B. Thomas (*Cambridge*)

The London Mathematical Society is incorporated under Royal Charter

Complexity: Knots, Colourings and Counting

D. J. A. Welsh
University of Oxford

These notes are based on a series of lectures given at the Advanced Research Institute of Discrete Applied Mathematics held at Rutgers University. Their aim is to link together algorithmic problems arising in knot theory, statistical physics and classical combinatorics. Apart from the theory of computational complexity concerned with enumeration problems, introductions are given to several of the topics treated, such as combinatorial knot theory, randomised approximation algorithms, percolation and random cluster models.

To researchers in discrete mathematics, computer science and statistical physics, this book will be of great interest, but any non-expert should find it an appealing guide to a very active area of research.

CAMBRIDG[E]
UNIVERSITY PRE[SS]

ISBN 0-521-45740-8

For a list of books available in this s

WELSH Complexity: Knots, Colourings and Counting

London Mathematical Society Lecture Note Series 186

Complexity: Knots, Colourings and Counting

D. J. A. Welsh

CAMBRIDGE UNIVERSITY PRESS

Computing $T_H(x,y)$?

- Problem:
  Given H as input, compute $T_H(x,y)$
- The problem is #P-hard
- Solvable by deletion–contraction
  in time $\exp(O(n \log n))$ ~ spanning trees in H
- *But can we go faster ?*

Tutte polynomial (equivalent formulation)

$$T_H(x,y) = \sum_{F \subseteq E} (x-1)^{c(F)-c(E)}(y-1)^{c(F)+|F|-|V|}$$

$$= \sum_{d=1}^{n} \sum_{k=0}^{m} s_{d,k}(x-1)^{d-c}(y-1)^{d+k-n}$$

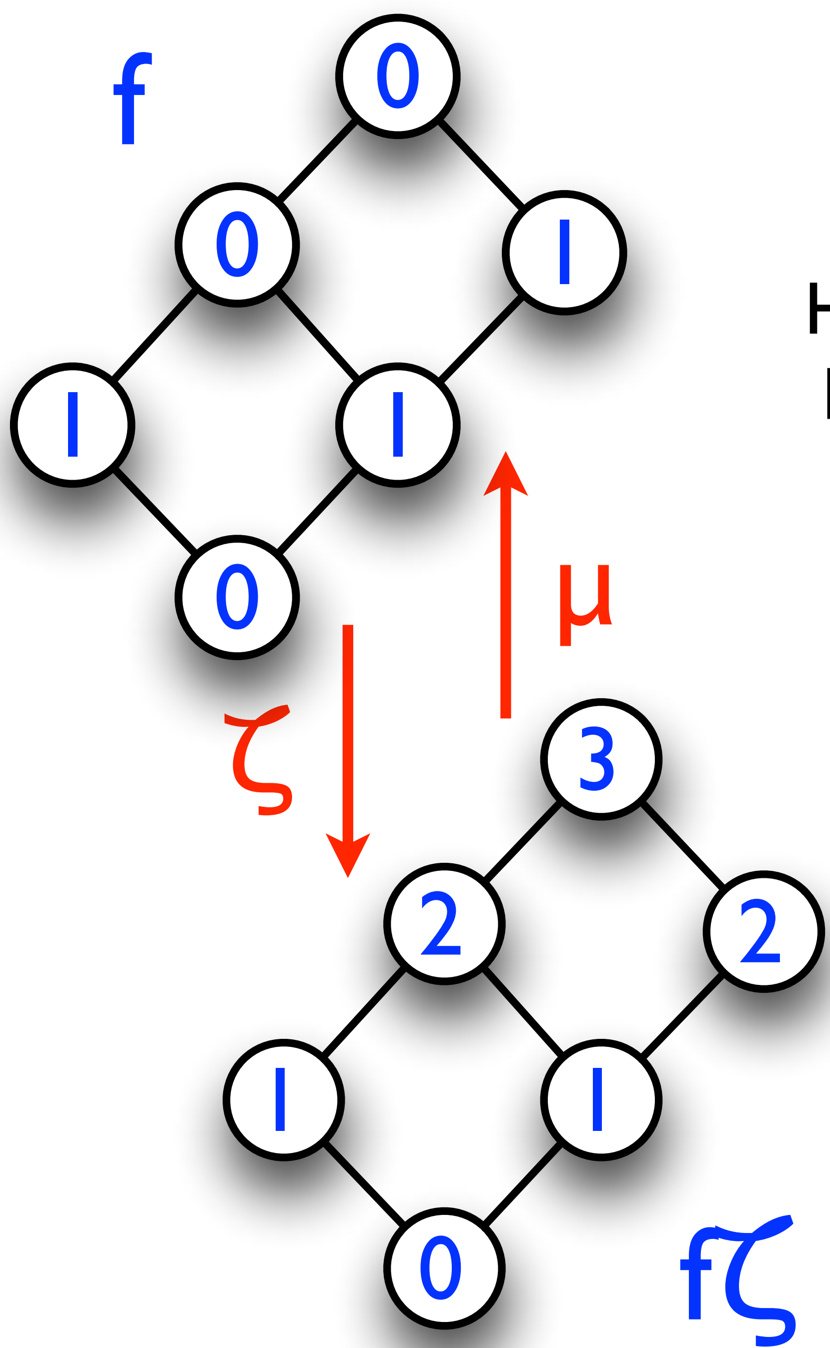
= #Spanning subgraphs of H
with exactly d connected components
and exactly k edges

The good, the bad, ...

The Tutte polynomial in $O^*(2^n)$ time
(Björklund, K., Koivisto, Husfeldt 2008)

- Universe = all spanning subgraphs
- Bad = disconnected spanning subgraphs
- Good = connected spanning subgraphs
- Bad objects *partition* into *strictly smaller* good objects (=connected components)
- |Good| = |Universe| – |Bad|

Fast Möbius inversion with applications



Petteri Kaski

Helsinki Institute for Information Technology HIIT &
Department of Information and Computer Science
Aalto University, Helsinki

Joint Estonian–Latvian
Theory Days at Medzābaki, Lidaste
30 September 2012

Aalto University
School of Science

A more detailed introduction/invitation (Husfeldt 2011) arXiv:1105.2942

Invitation to Algorithmic Uses of Inclusion–Exclusion

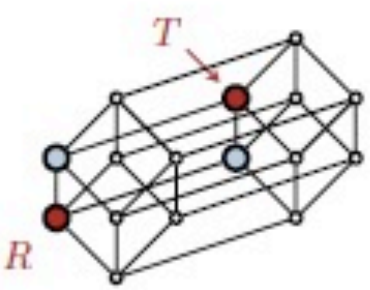
Thore Husfeldt

IT University of Copenhagen, Denmark
Lund University, Sweden

Abstract. I give an introduction to algorithmic uses of the principle of inclusion–exclusion. The presentation is intended to be be concrete and accessible, at the expense of generality and comprehensiveness.

1 The principle of inclusion–exclusion. There are as many odd-sized as even-sized subsets sandwiched between two different sets: For $R \subseteq T$,

$$\sum_{R \subseteq S \subseteq T} (-1)^{|T \setminus S|} = [R = T]\,. \qquad (1)$$



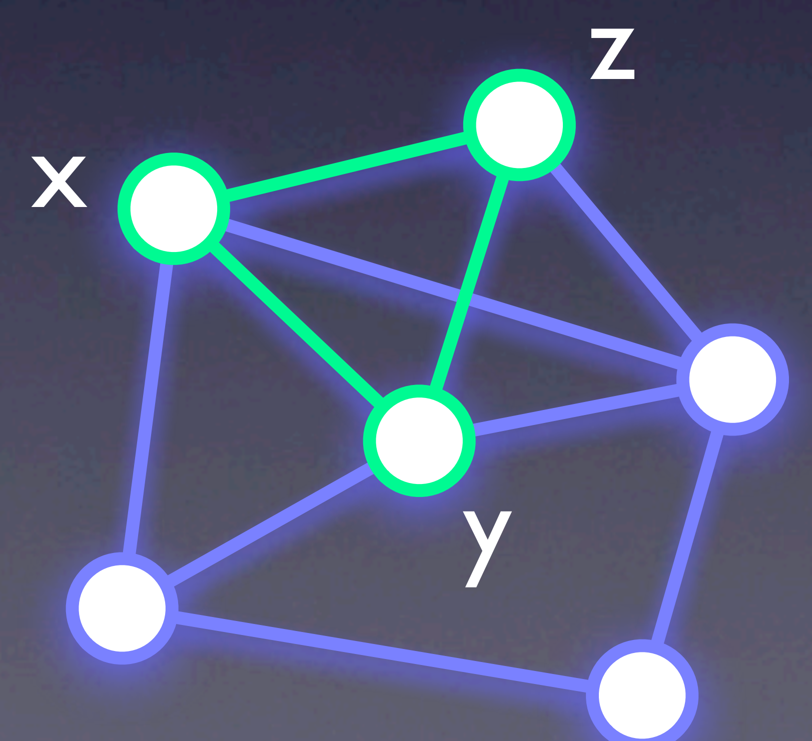
We use Iverson notation $[P]$ for proposition P, meaning $[P] = 1$ if P and $[P] = 0$ otherwise.

Proof of (1). If $R = T$ then there is exactly one sandwiched set, namely $S = T$. Otherwise we set up a bijection between the odd- and even-sized subsets as

Surprise 3:
#k-Matchings

#Triangles in $O(n^\omega)$ time (Itai & Rodeh 1978)

- Input: Loopless undirected multigraph H
- Let A(x,y) be the number of edges that join x and y
- The number of triangles through x,y,z is A(x,y)A(y,z)A(z,x)



#Triangles in $O(n^{\omega})$ time (Itai & Rodeh 1978)

- The total number of triangles in H is

$$N = \frac{1}{3!} \sum_{x,y,z} A(x,y)A(y,z)A(z,x)$$

$$= \frac{1}{3!} \sum_{x,y} A(x,y) \underbrace{\sum_{z} A(y,z)A(z,x)}_{\text{Matrix product } A^2(y,x)}$$

$2 \leq \omega < 2.3727$ (Vassilevska Williams 2012)

“Split-and-list”

- **Split** the problem (= either the instance or the solution) into two or more parts
- **List** (or count) the solutions of each part
- **Join** solutions of the parts in all possible ways to solutions of the original problem

#k-Matchings by splitting to three parts

(Nešetřil & Poljak 1985 -- for #k-Clique)

Time

$n^{\omega k/3+O(1)}$

$\geq 2k/3$

- Suppose (for simplicity) that 3 divides k
- Construct a graph H' where
  each vertex is a k/3 -subset S of vertices of H
- The weight of vertex S is #Perfect Matchings
  in the induced subgraph H[S]
- Join vertices S,T by an edge iff S and T are disjoint
- #Weighted Triangles in H' = f(k) * #k-Matchings in H

#k-Matchings by splitting to two parts

- Vassilevska & Williams 2009
- Koutis & Williams 2009
- Björklund, Husfeldt, K. & Koivisto 2009

Time
$n^{k/2+O(1)}$

#k-Matchings by splitting to two parts

(Björklund, Husfeldt, K. & Koivisto 2009)

$$f : \binom{V}{k/2} \to R \quad ,$$ f(A) = #(k/2)-Matchings in H[A]

#k-Matchings in H $$= \binom{k/2}{k/4}^{-1} \sum_{\substack{A,B \in \binom{V}{k/2} \\ A \cap B = \emptyset}} f(A) f(B)$$

Resource bottleneck

Weighted disjoint pairs

(Björklund, Husfeldt, K. & Koivisto 2009)

Input:

$$f : \binom{V}{k/2} \to R \qquad g : \binom{V}{k/2} \to R$$

|V| = n

Task: Evaluate

$$\Delta(f,g) = \sum_{\substack{A,B \in \binom{V}{k/2} \\ A \cap B = \emptyset}} f(A)g(B)$$

Time

$n^{k/2+O(1)}$

#k-Matchings by splitting to three parts

(Björklund, K. & Kowalik 2014)

$f : \binom{V}{k/3} \to R$, f(A) = #(k/3)-Matchings in H[A]

#k-Matchings in H $= \binom{k/2}{k/6, k/6, k/6}^{-1} \sum_{\substack{A,B,C \in \binom{V}{k/3} \\ A \cap B = \emptyset \\ A \cap C = \emptyset \\ B \cap C = \emptyset}} f(A) f(B) f(C)$

Resource bottleneck

Weighted disjoint triples

(Björklund, K. & Kowalik 2014)

Input:

$$f : \binom{V}{k/3} \to R \qquad g : \binom{V}{k/3} \to R \qquad h : \binom{V}{k/3} \to R$$

Task: Evaluate

$$\Delta(f,g,h) = \sum_{\substack{A,B,C \in \binom{V}{k/3} \\ A \cap B = \emptyset \\ A \cap C = \emptyset \\ B \cap C = \emptyset}} f(A)g(B)h(C)$$

Time

$n^{0.4547k+O(1)}$

Proof/algorithm idea: "Method of linear equations"

- We want to compute a quantity $\Delta(f, g, h)$
- Let $x_k = \Delta(f, g, h)$ be an indeterminate
- Set up "related indeterminates" $x_0, x_1, \ldots, x_k$
- Set up a system of linear equations

$$A\vec{x} = \vec{b}$$

- Solve for x_k "indirectly" via "easier" equations and/or indeterminates in the system

Thank you!

- Selected techniques in subgraph counting
  - Inclusion–exclusion & linear equations ("the good, the bad, and the universe")
  - Split-and-list & fast matrix multiplication
  - Zeta and Möbius transforms
- Applications ("selected surprises"):
  - #Perfect Matchings
  - Deletion–contraction & Tutte polynomial
  - #k-Matchings