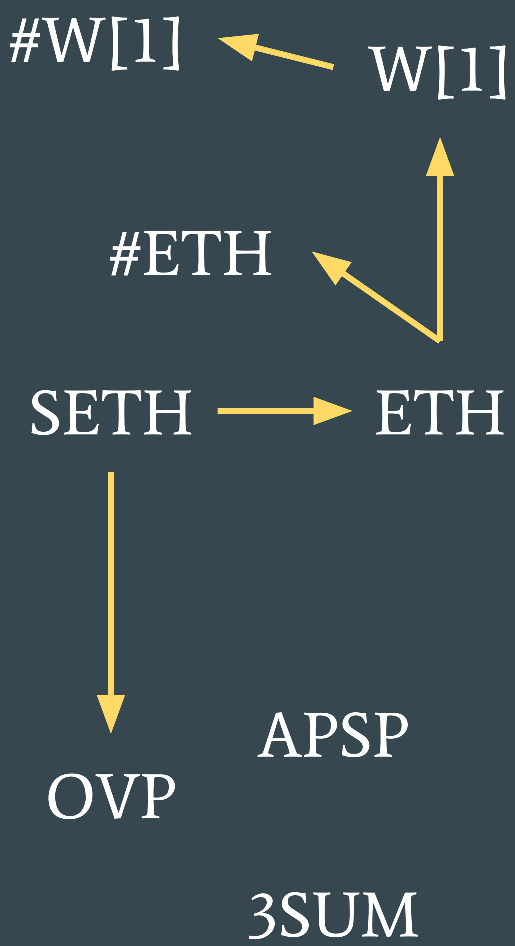
Fine-Grained Counting Complexity I

...

Holger Dell

Saarland University and Cluster of Excellence (MMCI)
&
Simons Institute for the Theory of Computing

50 Shades of Fine-Grained



fixed-parameter tractability

exponential-time “exact algorithms”

polynomial-time algorithms

Outline

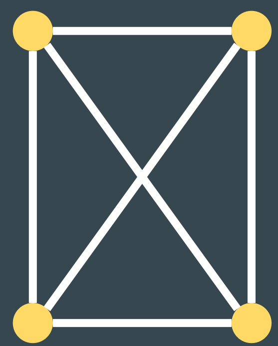
- classical counting complexity
- fine-grained lens
- specific problems
- structural results
- open problems

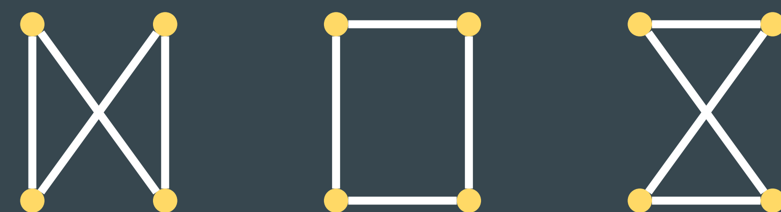
The classics

Decision vs. Counting

Decision	Counting
SAT	#SAT
Is the CNF formula satisfiable?	How many satisfying assignments are there?
NP	#P
problems $L : \{0,1\}^* \rightarrow \{0,1\}$	problems $f : \{0,1\}^* \rightarrow \mathbf{N}$
that reduce to SAT	that reduce to #SAT

Example: Counting Hamiltonian Cycles reduces to #SAT





Three Hamiltonian Cycles

Circuit C
Input variables x_e

Three satisfying assignments

$C(x)$ accepts
iff x is a length-n cycle

Parsimonious reductions and the counting version of NP

R is a **parsimonious** reduction from f to g if

$$f(\ x\ ) = g(\ R(x)\ )\ \text{for all } x.$$

$$\#P = \left\{ \begin{array}{c} \text{problems } f \text{ that parsimoniously} \\ \text{poly-time reduce to } \#SAT \end{array} \right\}$$

Counting solutions is harder than finding one

Some examples of counting problems

"Combinatorial" counting problems

f(G) = # Hamiltonian Cycles

#P-complete using existing hardness reduction

"Optimization" problems

f(G) = size of the largest clique

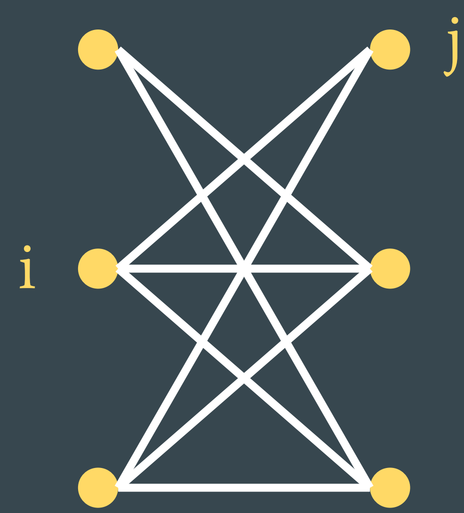
In P^{NP} and probably not #P-hard

"Algebraic" problems

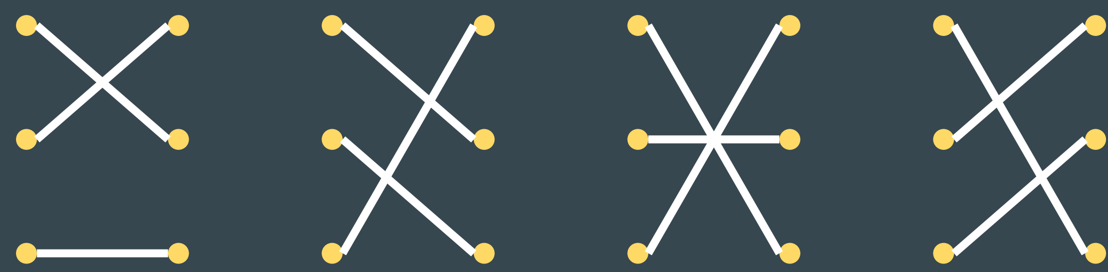
f(matrix A) = value of the determinant

in poly-time

Count Perfect Matchings in Bipartite Graphs



d + d vertices



Perfect Matchings

$= \mathrm{per}(A)$

$= \sum_{\text{permutation } \pi} \prod_{i \in \{1..d\}} A_{i\,\pi(i)}$

d×d matrix A

$A_{ij} = 1$ iff $\{i,j\}$ is an edge

Computing the permanent

Evaluation time
$\sim d! \sim 2^{d \log d}$

$\mathrm{per}(\ d\times d \text{ matrix } A\ ) = \sum_{\text{permutation } \pi} A_{1\ \pi(1)} \cdots A_{d\ \pi(d)}$

$= \sum_{\text{all functions } f : \{1..d\} \to \{1..d\}} A_{1\ f(1)} \cdots A_{d\ f(d)}$

$- \sum_j \sum_{f : \{1..d\} \to \{1..d\} \setminus \{j\}} A_{1\ f(1)} \cdots A_{d\ f(d)}$

$+ \sum_{j,k} \sum_{f : \{1..d\} \to \{1..d\} \setminus \{j,k\}} A_{1\ f(1)} \cdots A_{d\ f(d)}$

…

$\prod_{i\in\{1..d\}} \sum_{j\in\{1..d\}} A_{ij}$

Evaluation time
$O(d2^d)$

Ryser's Inclusion-Exclusion Formula (1963)

$$= \sum_{S \subseteq \{1..d\}} (-1)^{|S|} \prod_{i\in\{1..d\}} \sum_{j\in\{1..d\}\setminus S} A_{ij}$$

Permanent and Determinant

$$\mathrm{per}(A) = \sum_{\pi} \prod_{i} A_{i\,\pi(i)}$$

$$\det(A) = \sum_{\pi} (-1)^{\mathrm{sgn}(\pi)} \prod_{i} A_{i\,\pi(i)}$$

$$\det(A) \equiv \mathrm{per}(A) \pmod 2$$

Permanent: Probably not parsimoniously hard

If there was a **parsimonious** reduction R from #SAT to per:

- $\#SAT(F) \equiv per(R(F)) \equiv det(R(F)) \pmod 2$
- Distinguish $\#SAT(F) = \mathbf{1}$ from $= \mathbf{0}$
- RP = NP [Valiant-Vazirani isolation lemma]

Polynomial-time oracle reductions from f to g

R is a Turing machine with oracle access to g

such that $f(x) = R^g(x)$ holds for all x

Theorem [Valiant 79].

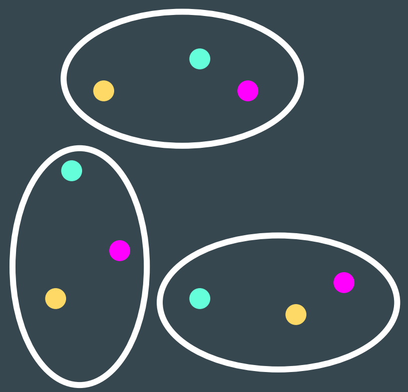
Poly-time oracle reduction from #SAT to Permanent.

→ If Permanent is poly-time, then PH=P

Proof that the Permanent is Hard I

[D and Marx 15+] Reduction from # 3-Dimensional Matching

- Sets of size 3
- 3 colors
- Every set is multicolored



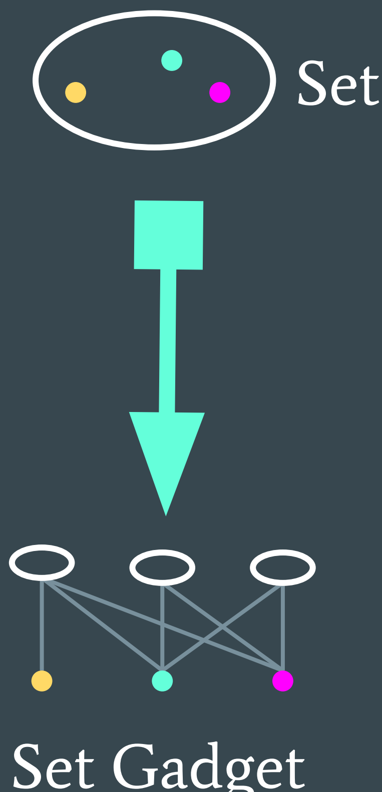
3D-matching = n/3 disjoint sets

Fact: The known reduction from 3-Sat to 3-Dimensional Matching is parsimonious.

Proof that the Permanent is Hard II

Compute: # 3-Dimensional Matching (n elements, m sets)

Oracle: # Size-n Matchings in $n + 3m$ bipartite graphs.



Higher moduli
+ Chinese Remainder Theorem

→ Recover exact number

“S

2 Matchi

“ elected”

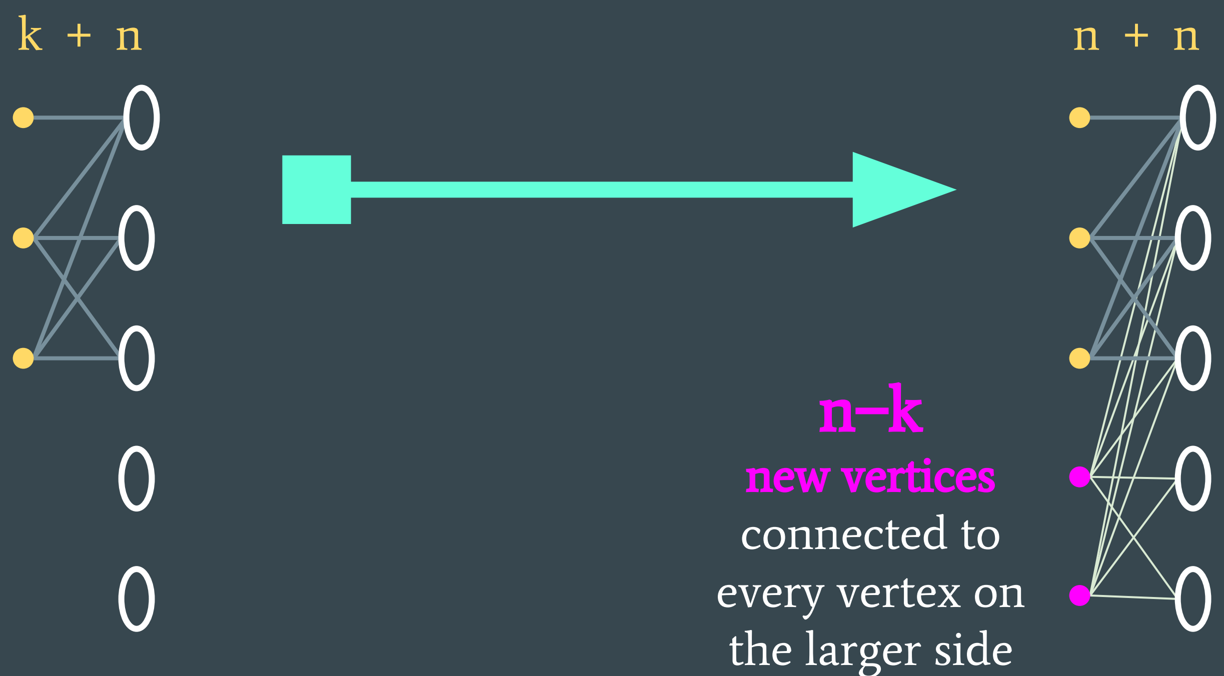
6 Mat of this type

Size-n Matchings $\equiv$ 2× #3DM (mod 3)

Proof that the Permanent is Hard III

Compute: # k-Matchings in a k+n bipartite graph

Oracle: Permanent = # Perfect Matchings in n+n bipartite graph



$(n-k)! \times$ # k-Matchings(G) = # Perfect Matchings(G')

Proof that the Permanent is Hard, Summary

3-CNF Sat

Permanent

parsimonious standard reduction

+ Chinese Remainder Theorem

simple
standard
reduction

3-dimensional matchings mod 3
5-dimensional matchings mod 5
...
p-dimensional matchings mod p

simple gadget

k-matchings in k+n graph mod p

Fine-Grained Counting Complexity

Counting Satisfying Assignments of CNFs

Theorem [Chan and Williams 15]

Deterministically compute #SAT for a CNF formula **F** in time $2^{n\,(1 - \text{savings})}$

- **F** is a k-CNF → savings ~ 1 / k
- **F** has cn clauses → savings ~ 1 / log c

Counting Exponential Time Hypotheses

#ETH

#SAT for k-CNFs does not have $\exp(o(n))$ time algorithm

Sparsification Lemma [Impagliazzo Paturi Zane 01; Calabro Impagliazzo Paturi 06]

Can assume $m \sim k^k n$

#SETH

#SAT for CNFs does not have 1.999^n time algorithm

Fine-Grained Complexity of the Permanent

Theorem [Curticapean 15; D Husfeldt Marx Taslaman Wahlén 10]

If per($d \times d$ matrix A) can be computed in $\exp(o(d))$,
then #ETH is false

Theorem [Servedio and Wan 05]

If A has cn nonzero entries,
$\mathrm{per}(A)$ can be computed in time $(2-\varepsilon)^d$ where $\varepsilon(c) < 1$

PETH (Permanent Exponential Time Hypothesis)

$\mathrm{per}(A)$ cannot be computed in time 1.999^d

Counting Solutions to 2-CNF formulas

#SAT for 3-CNFs

Sparsification Lemma

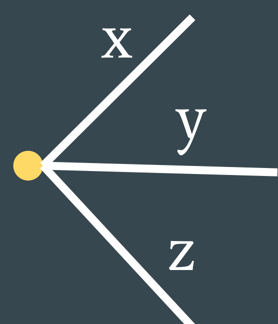
#SAT for 3-CNFs

each variable appears O(1) times

via Permanent [Curticapean 15]

All Matchings

in constant degree graph



Theorem

#SAT for 2-CNFs is #ETH-hard

#SAT for 2-CNFs

each variable appears O(1) times

$(\neg x \lor \neg y)$

$(\neg x \lor \neg z)$

$(\neg y \lor \neg z)$

Results for Various Counting Problems

Count Perfect Matchings in General Graphs

per($(n/2)\times(n/2)$ matrix)

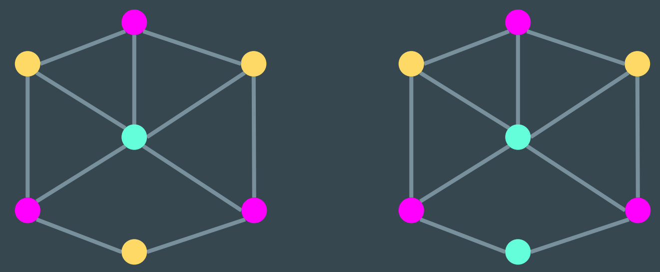
= # Perfect Matchings of bipartite graph with $n/2+n/2$ vertices

$\rightarrow 2^{n/2}$ algorithm

Theorem [Björklund 11].

Count perfect matchings in general graphs in time $2^{n/2}$.

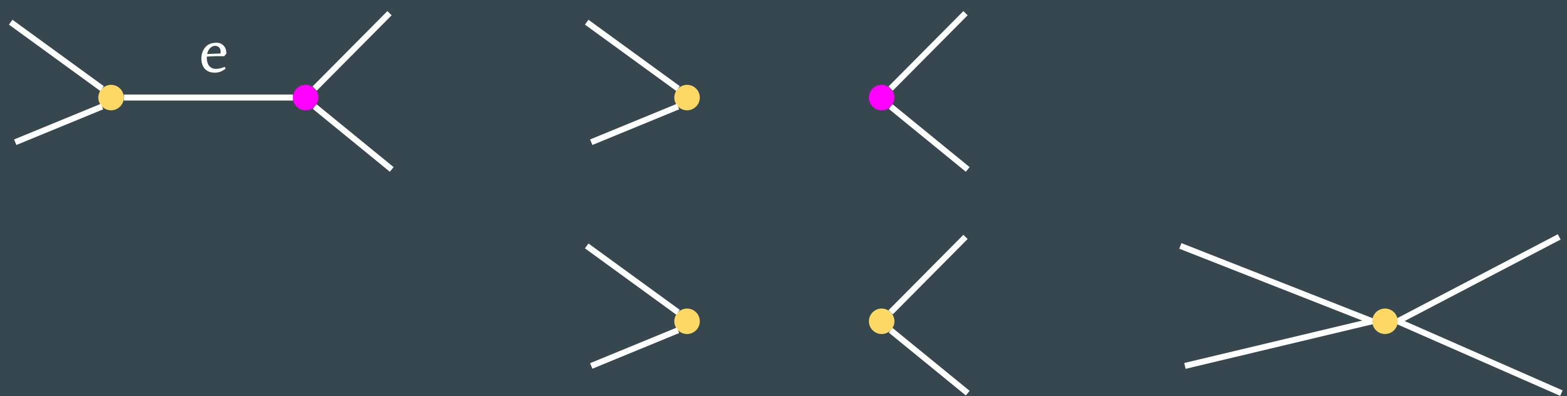
Proper q-colorings



Chromatic polynomial & Deletion-Contraction

$\chi(G, q)$ = # proper q-colorings of G

G G−e G/e



$$\chi(G, q) = \chi(G-e, q) - \chi(G/e, q)$$

$$\chi(\text{k-independent set}, q) = q^k$$

$\rightarrow$ $\chi(G, q)$ is a degree-n polynomial in q.

Compute # q-Colorings

The deletion-contraction algorithm takes time 2^m.

Theorem [Björklund Husfeldt Koivisto 09]
Compute the number of q-colorings in time 2^n.

Theorem [Impagliazzo Paturi Zane 01]
ETH $\rightarrow$ no $2^{o(n)}$ algorithm for q-coloring.

The Tutte Polynomial

$$T(G, x, y) = \sum_{A \subseteq E} (x-1)^{k(A)-k(G)} (y-1)^{k(A)+|A|-|V|}$$

Generalizes

- chromatic polynomial $\chi(G, q) = (-1)^{n-k(G)} q^{k(G)}\ T(G, 1-q, 0)$
- Ising model, q-state Potts model
- many combinatorial problems

Computing the Tutte polynomial

The trivial algorithm runs in time 2^m

Theorem [Björklund Husfeldt Kaski Koivisto 08]
It can be computed in time 2^n

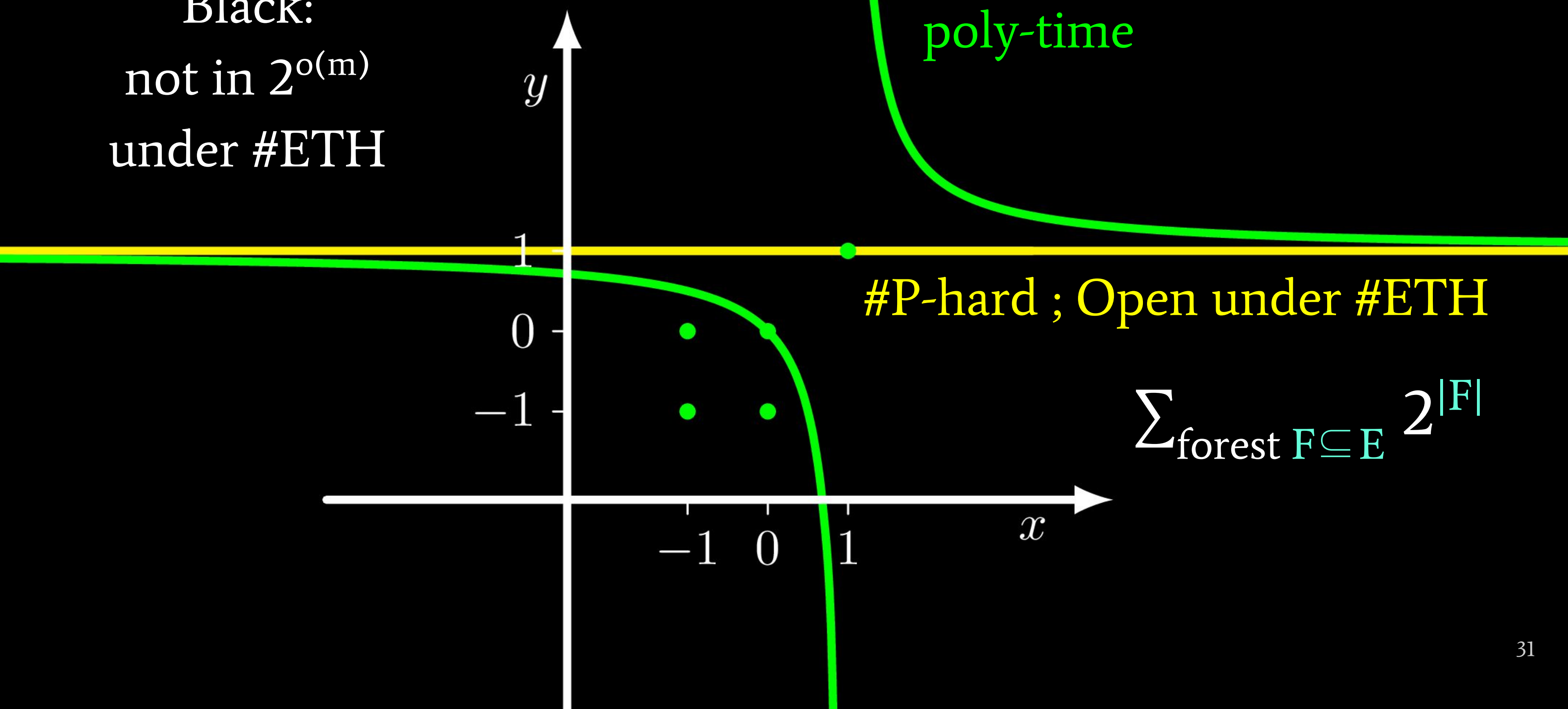
Theorem [Curticapean 15; D Husfeldt Marx Taslaman Wahlén 10; Jaeger Vertigan Welsh 1990]

#ETH → no $2^{o(m)}$ algorithm

The Tutte Plane of Computational Problems

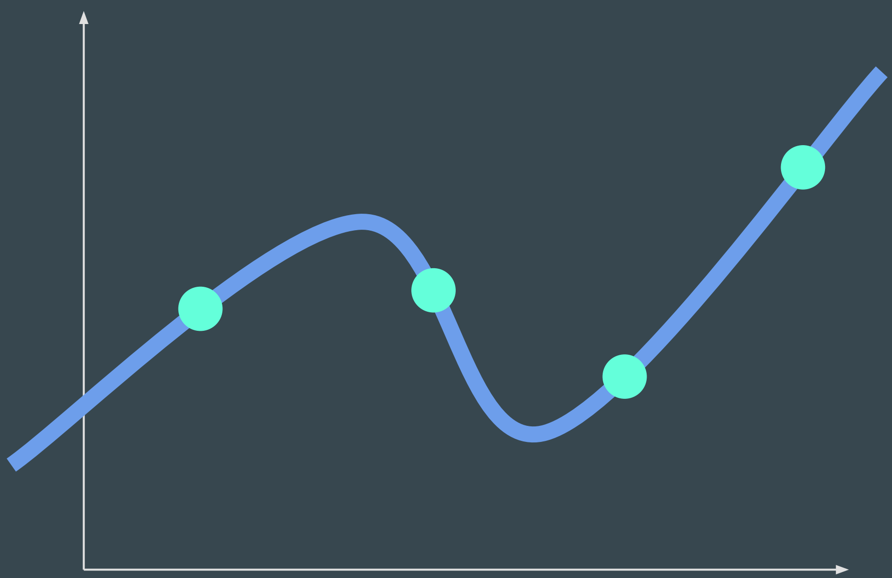
Fix $x, y \in \mathbf{Q}$ Input: G Output: $T(G, x, y)$

Black:
not in $2^{o(m)}$
under #ETH



Polynomial Interpolation

→ compute **p** in poly-time from samples



polynomial p
degree d

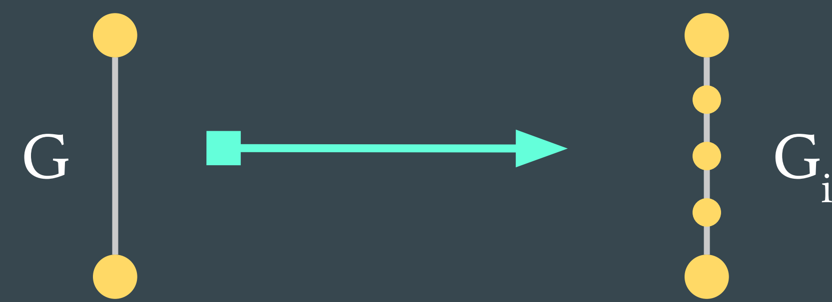
d + 1 samples
(a, **p**(a))

Interpolation in Counting Complexity

[seriously, like, every paper in the area]

$T(G, z_i) = T(G_i, z)$

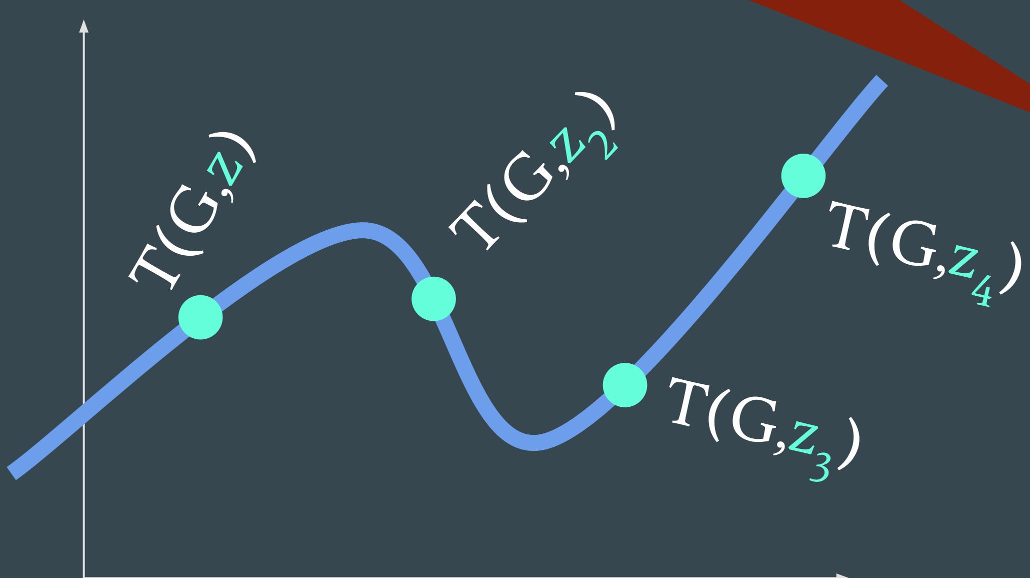
Only rules out $\mathbf{2^{o(m/\log m)}}$ time algorithms under #ETH



Need **m+1 samples**

→ m+1 different gadgets

→ $m(G_i) \sim m \log m$



Tutte polynomial T(G, z)

degree m

Block interpolation [Curticapean 15]

$$T(G, z) = \sum_{A \subseteq E} q^{k(A)} z^{|A|}$$

Can rule out $2^{o(m)}$ time algorithms under #ETH

Partition edges E into n/r blocks of size r

$$T(G, z_1, \ldots, z_{n/r}) = \sum_{A \subseteq E} q^{k(A)} z_1^{|A(1)|} \ldots z_{n/r}^{|A(n/r)|}$$

$\rightarrow$ Multivariate interpolation

$\sim r^{n/r} = \exp(\varepsilon\, n)$ samples

$r+1$ distinct gadgets per variable

Approximate Counting

$\pi \times$ 

(German idiom)

Approximate Counting

[Jerrum Sinclair Vigoda 04]

poly(n/ε)-time ($1+\varepsilon$)-approximation (FPRAS) for Permanent

[Stockmeyer 1985]

FPRAS for # Sat when given access to an NP-oracle

[Traxler 14]

If CNF-Sat is in 1.99^n time,

we can ($1+1.1^{-n}$)-approximate # CNF-Sat in time 1.99001^n

Open Problems

Dichotomy theorems

Constraint Satisfaction Problems (CSPs)

$$R_1(x_1,y_1,z_1) \wedge R_1(x_2,y_1,z_3) \wedge R_2(x_2,y_2,z_1) \wedge \ldots$$

Theorem [Bulatov 08, Dyer Richerby 10]

Dichotomy for #CSP depending on $\{R_1, R_2, .. \}$:

It's either in P or #P-complete

Is there a Dichotomy under #ETH ?

- Weighted #CSP [Cai, Chen, Lu 11]
- Planar Holant problems [Cai, Fu, Guo, Williams 15]

Is Counting really harder than Decision?

If SETH is true,

- CNF-SAT takes time 2^n
- # CNF-SAT takes time 2^n
- QBF-SAT takes time 2^n

In applications: CNF-SAT **much** easier than QBF-SAT.

Is there a tight reduction from QBF-SAT to # CNF-SAT ?

Summary

- counting is hard: $PH \subseteq P^{\#P}$
- is computing $\sum_{\text{forest } F \subseteq E} 2^{|F|}$ hard under ETH or #ETH ?
- is the permanent hard under SETH ?
- which problems are hard under PETH ?
- fine-grained inapproximability ?
- is fine-grained counting really harder than decision?
  can we tightly reduce QBF-SAT to # CNF-SAT ?