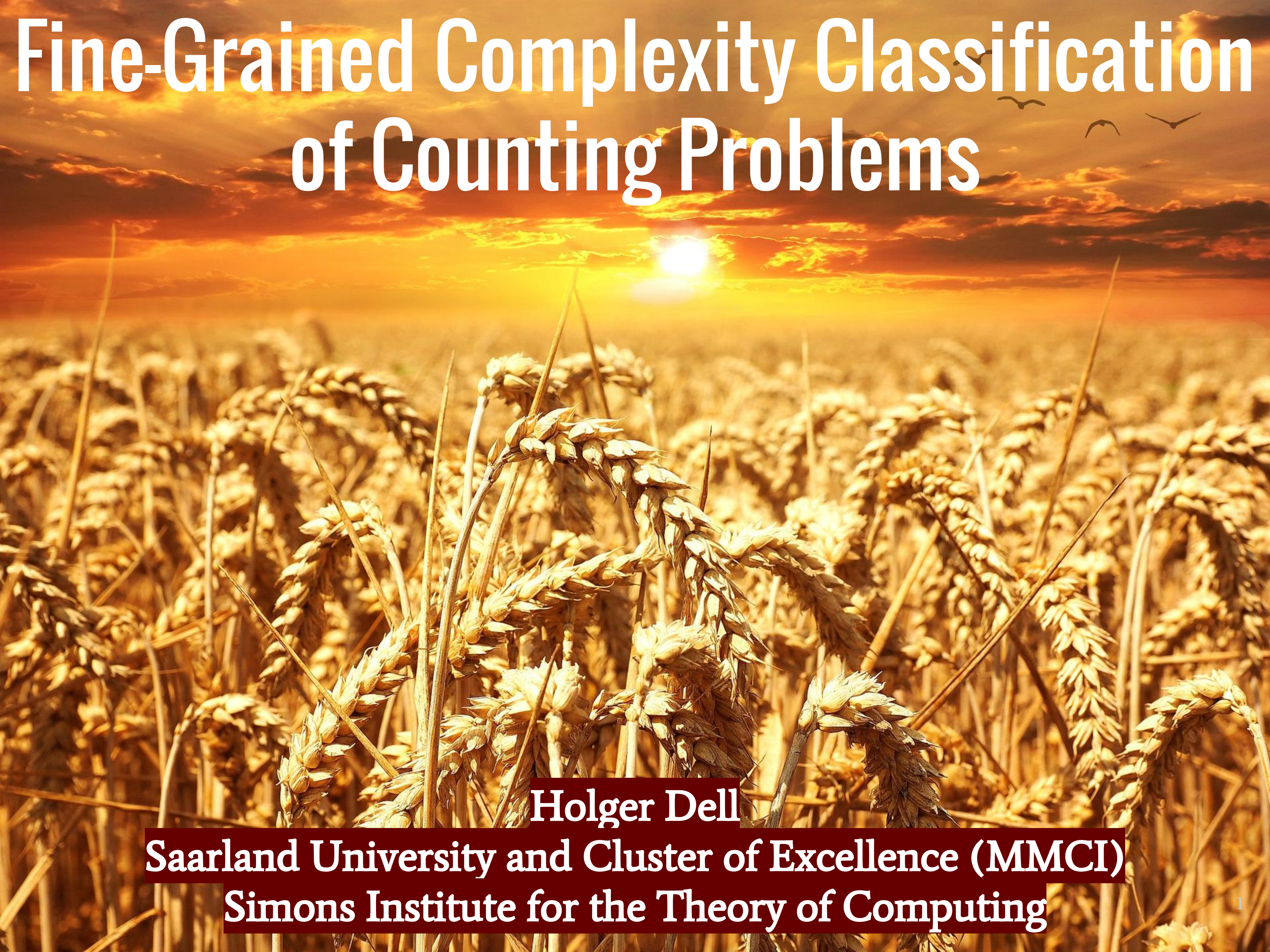


Fine-Grained Complexity Classification of Counting Problems

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

In P or not in P ?

Exact counting

P | #P-hard

Approximate counting

FPRAS | #BIS-complete | #SAT-hard

FPRAS | no FPRAS unless RP=NP

Motivation for fine-grained complexity

#P-hard problems could have algorithms running in time:

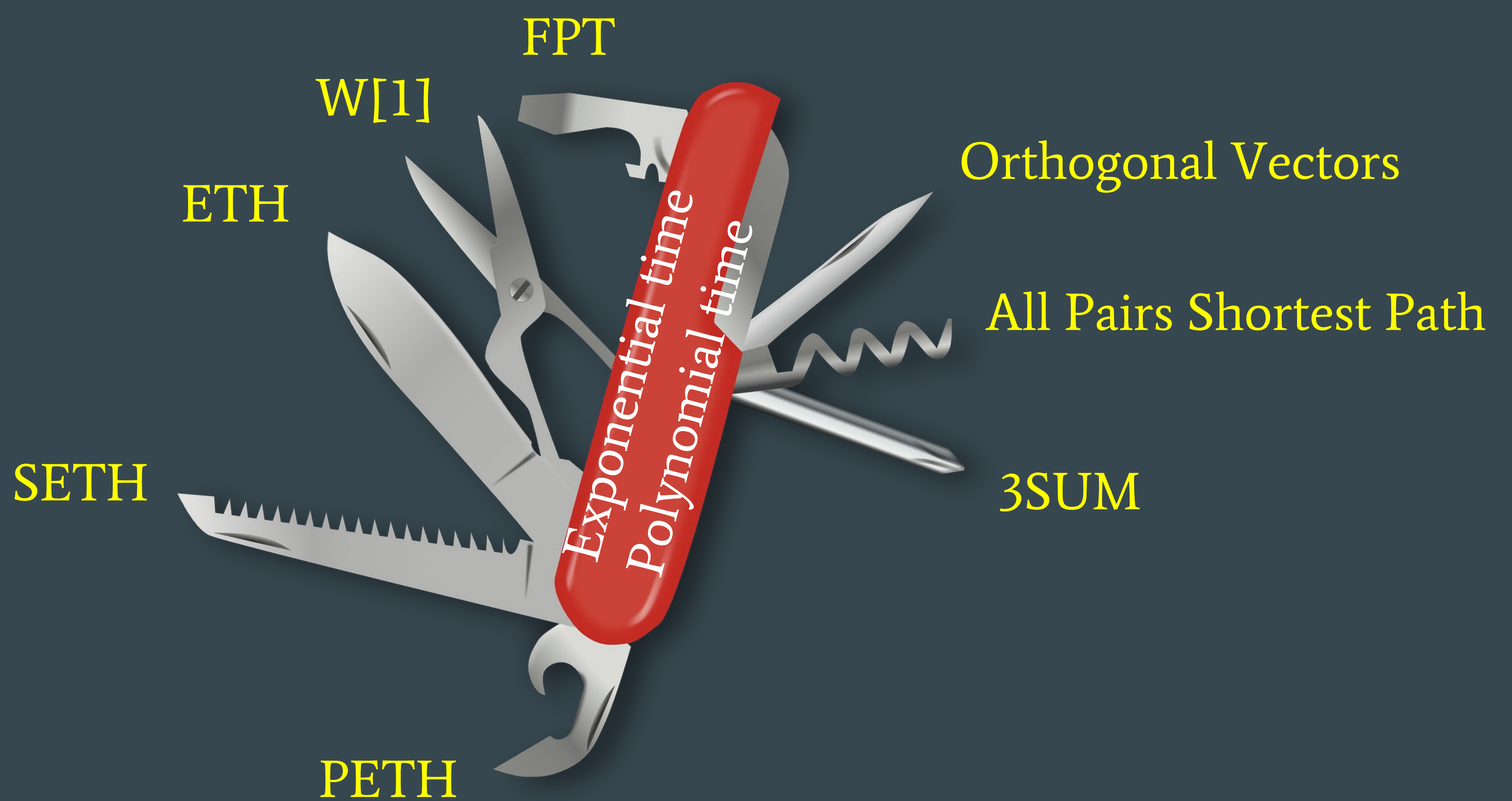
$O(2^n)$ $O(1.01^n)$ $O(2^{\sqrt{n}})$ $O(n^k)$ $O(2^k\, n)$

Problems in P could have algorithms running in time:

$O(n)$ $O(n^2)$ $O(n^{100})$

Fine-grained complexity is a toolbox
to pin down best running time more precisely

Available conjectures, problems, and classes

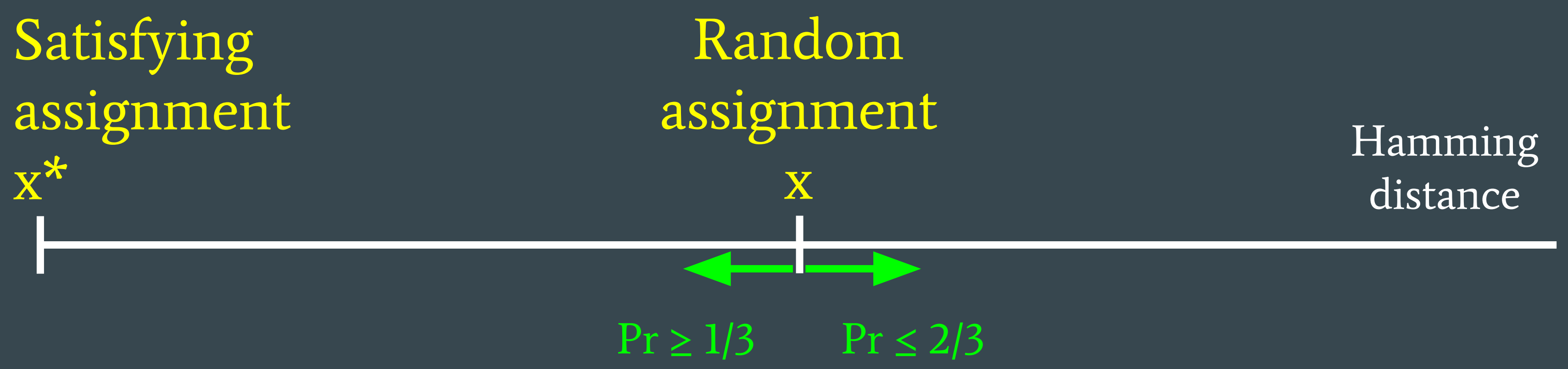


3-CNF-SAT faster than exhaustive search

Schöning's algorithm

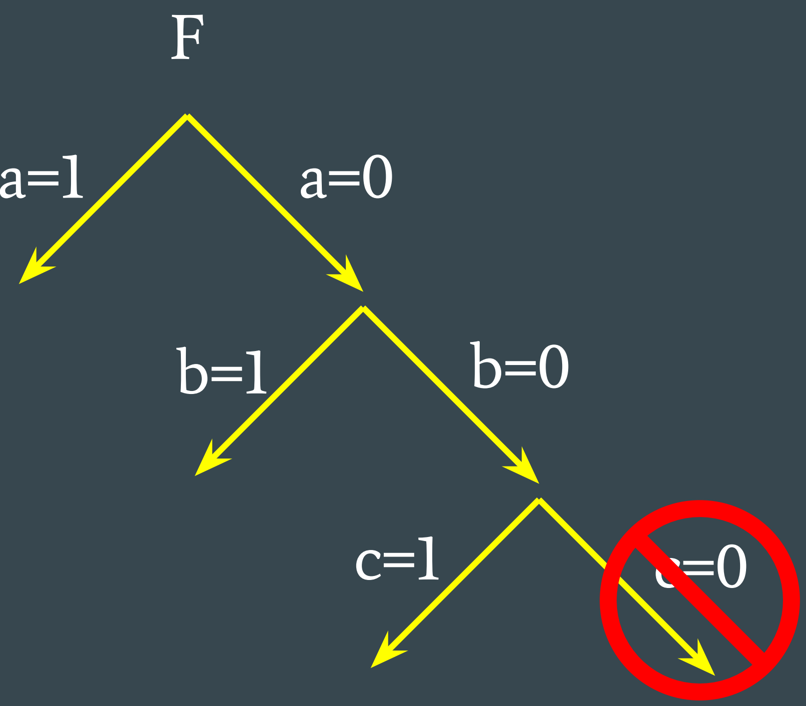
Expected running time: $\mathbf{(4/3)^n}$

- Sample random assignment $x \in \{0,1\}^n$
- While there is clause $(a \lor b \lor c)$ not satisfied by x:
  - Choose random literal in {a,b,c}
  - Flip its value in x
- Restart process if too long



Branching algorithms

Formula F
contains clause
$(a \lor b \lor c)$



For every clause $(a \lor b \lor c)$, only need to look at 7⁄8 of all assignments

→ **Time $7^{n/3} \sim 1.913^n$**

Counting Satisfying Assignments of 3-CNFs

Kutzkov 07: exact counting in time $O(1.6423^n)$

Thurley 12: ε-approximation in time $O(\varepsilon^{-2} \; 1.5366^n)$

Exponential time hypothesis (#ETH)

$\exists\, \delta>0.$ #SAT for 3-CNFs is not in time $(1+\delta)^n$

Sparsification Lemma

(Impagliazzo Paturi Zane 01; Calabro Impagliazzo Paturi 06; D, Husfeldt, Wahlén 10)

Can assume #clauses ~ $(1/\delta)^3\, n$

Sparsification Lemma

(Impagliazzo Paturi Zane 01; Calabro Impagliazzo Paturi 06; D Husfeldt Wahlén 10)

Input:

- k-CNF formula F with n variables and m clauses
- $\varepsilon>0$

Output:

$F_1 \dots F_t$ such that

- $\mathrm{sat}(F) = \mathrm{sat}(F_1) \sqcup \dots \sqcup \mathrm{sat}(F_t)$
- each F_i has $(k/\varepsilon)^k\, n$ clauses
- $t = 2^{\varepsilon n}$

General CNFs

Chan and Williams 15:

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

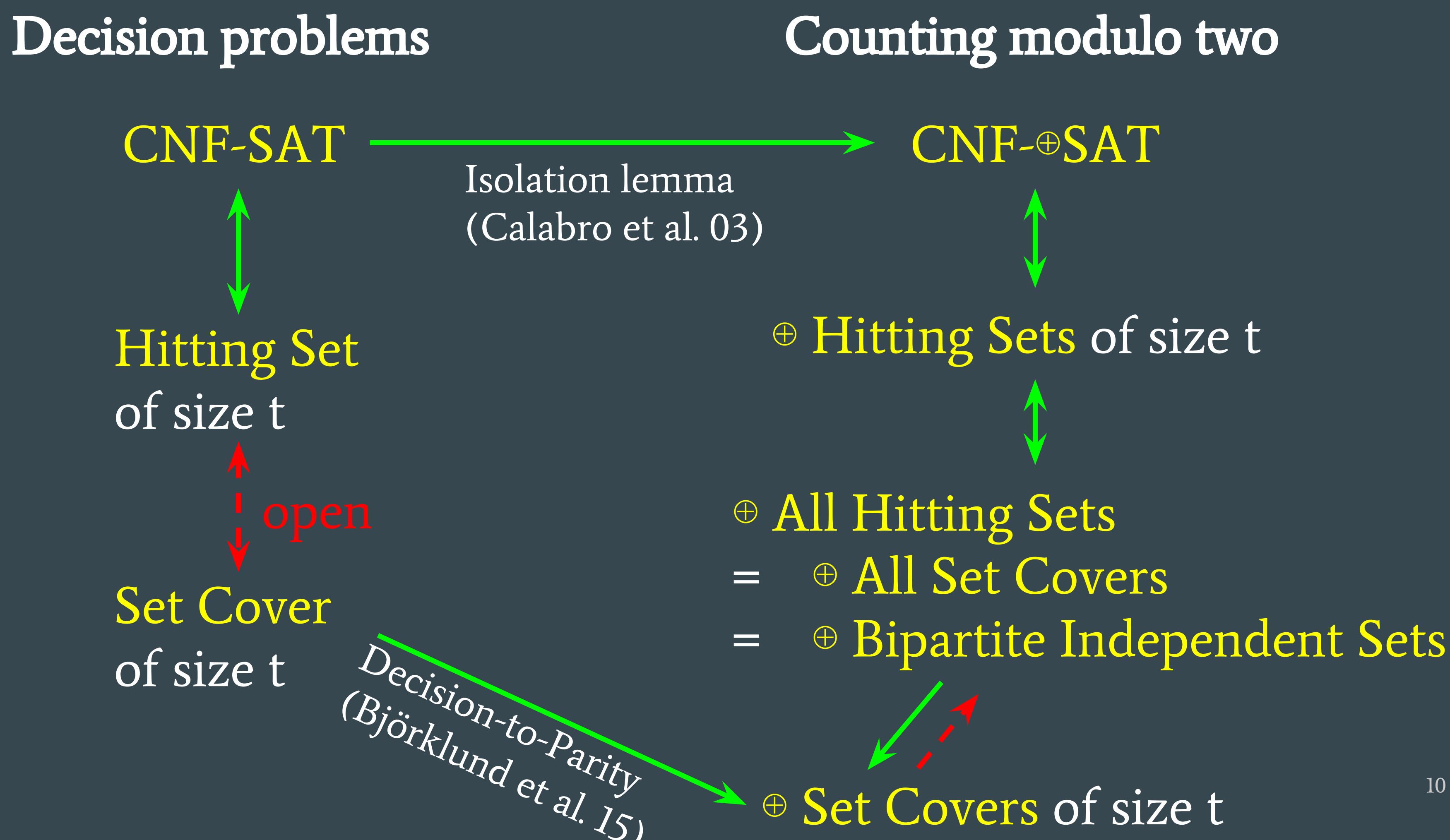
- **F** is a k-CNF → savings ~ 1 / k
- **F** has cn clauses → savings ~ 1 / log c

Strong exponential time hypothesis (#SETH)

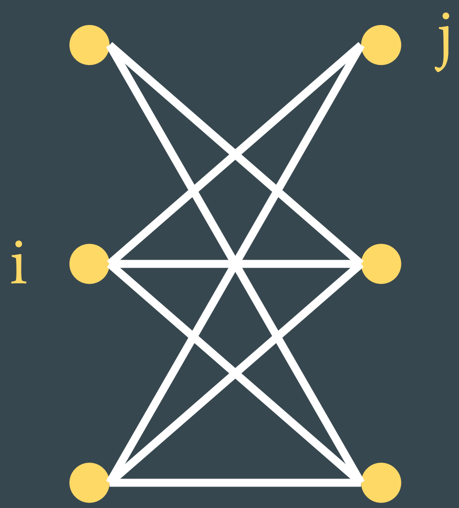
$\forall \delta > 0 \ \exists k.$ #SAT for k-CNFs cannot have savings $\geq \delta$

Problems equivalent under SETH

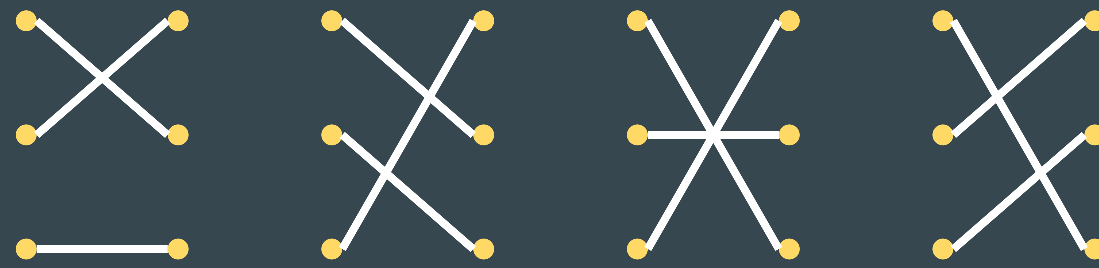
Cygan et al. 2012



Perfect Matchings in Bipartite Graphs



d + d vertices



$d \times d$ matrix A

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

\# Perfect Matchings

$= \mathrm{per}(A)$

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

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)

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

Fine-Grained Complexity of the Permanent

Curticapean 15; D Husfeldt Marx Taslaman Wahlén 10

If per($d \times d$ matrix A) can be computed in time $2^{o(d)}$, then #ETH is false

Servedio and Wan 05

If A has $\leq cd$ nonzero entries, per(A) can be computed in time $(2-\delta)^d$ where $\delta(c) < 1$

Permanent Strong Exponential Time Hypothesis (PETH)

$\forall \delta > 0 \; \exists c.$ per(c-sparse A) not in time $(2-\delta)^d$

Count Perfect Matchings in General Graphs

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

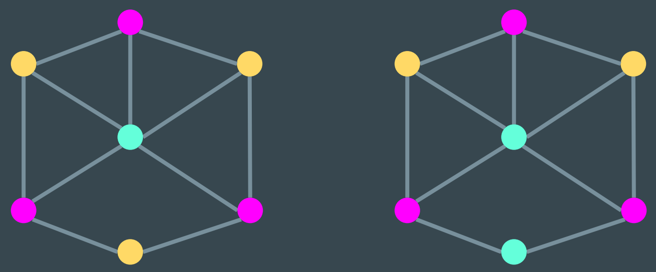
= # Perfect Matchings of bipartite graph with $n/2+n/2$ vertices

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

Björklund 11

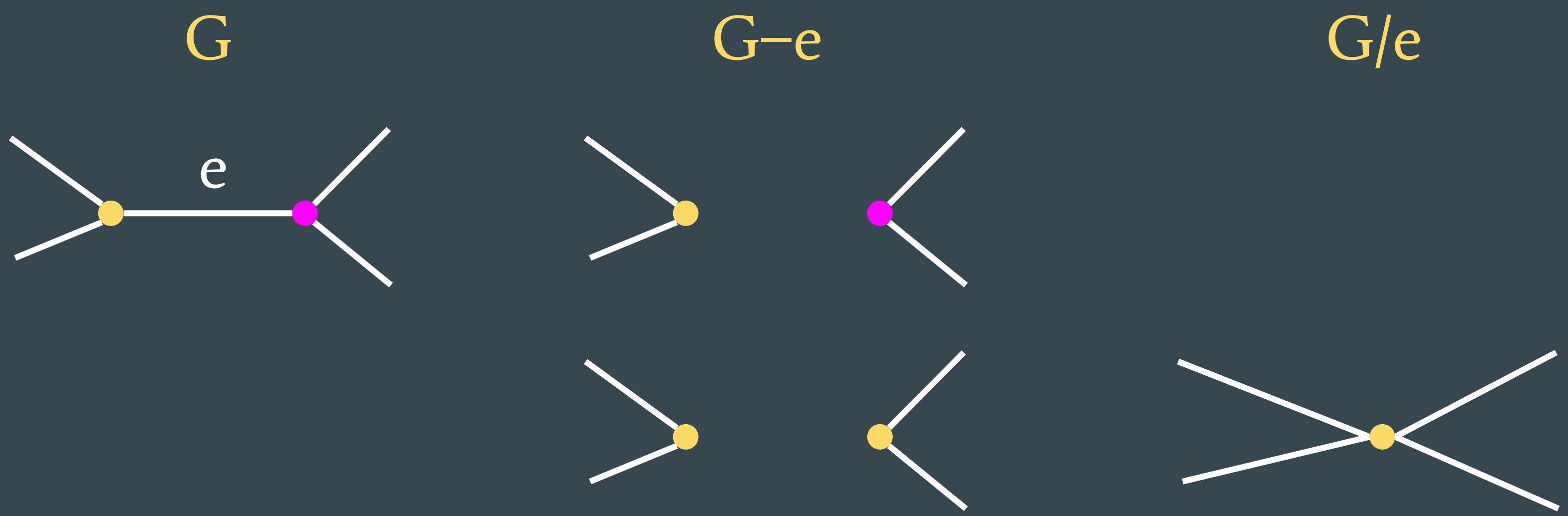
Count perfect matchings in general graphs in time $2^{n/2}$.

Proper q-colorings



Chromatic polynomial & Deletion-Contraction

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



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

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

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

Compute # q-Colorings

The deletion-contraction algorithm takes time 2^m.

Björklund Husfeldt Koivisto 09

Compute the number of q-colorings in time 2^n.

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

Björklund Husfeldt Kaski Koivisto 08

Time 2^n algorithm

Curticapean 15; D Husfeldt Marx Taslaman Wahlén 10; Jaeger Vertigan Welsh 90

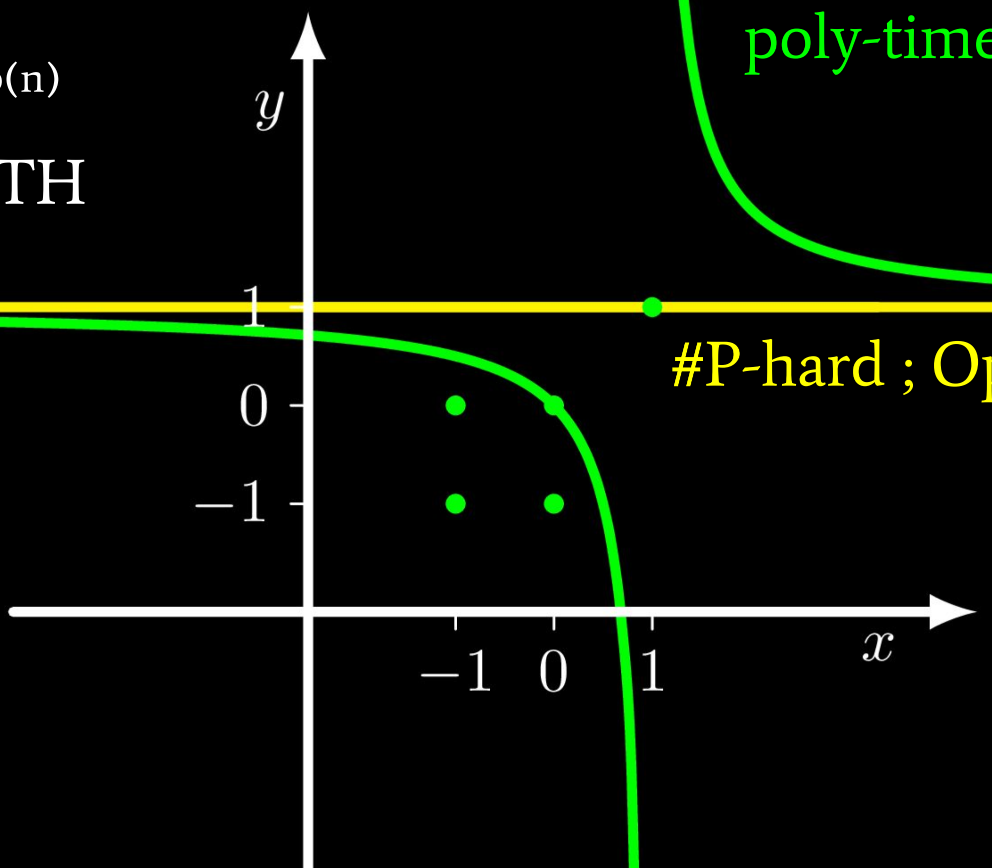
#ETH → no $2^{o(n)}$ algorithm

The Tutte Plane of Computational Problems

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

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

poly-time

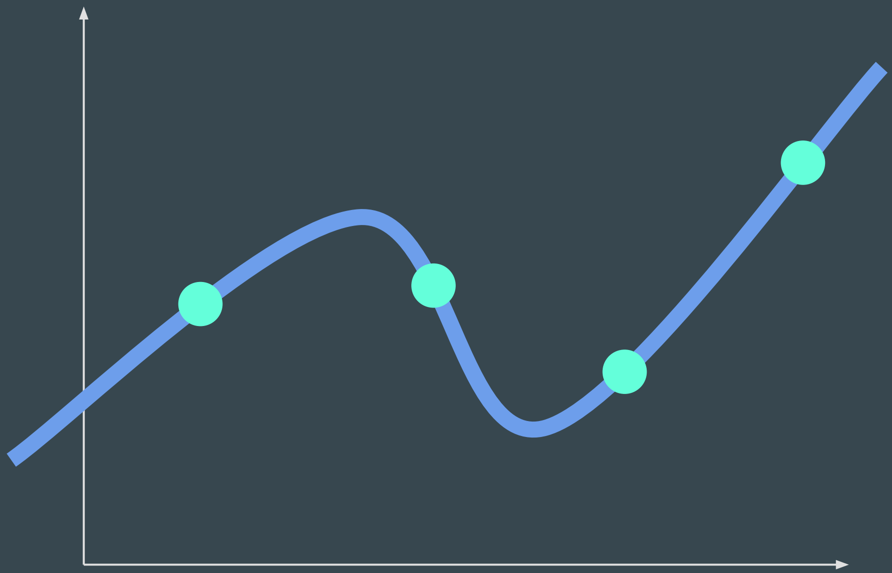


#P-hard ; Open under #ETH

$\sum_{\text{forest } F \subseteq E} 2^{|F|}$

Polynomial Interpolation

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

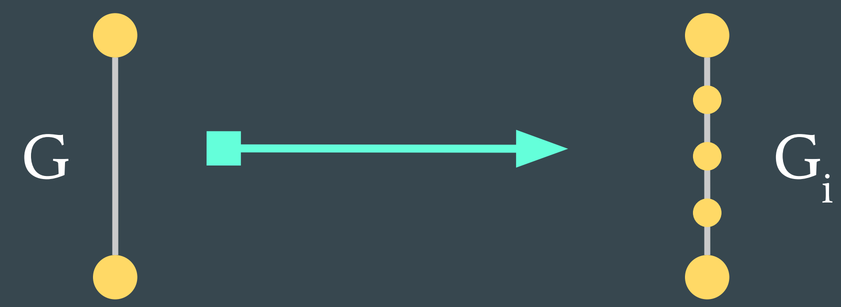


univariate polynomial p
degree m

m + 1 samples
(a, p(a))

Interpolation in Counting Complexity

$p(G, a_i) \quad = \quad p(G_i, a)$

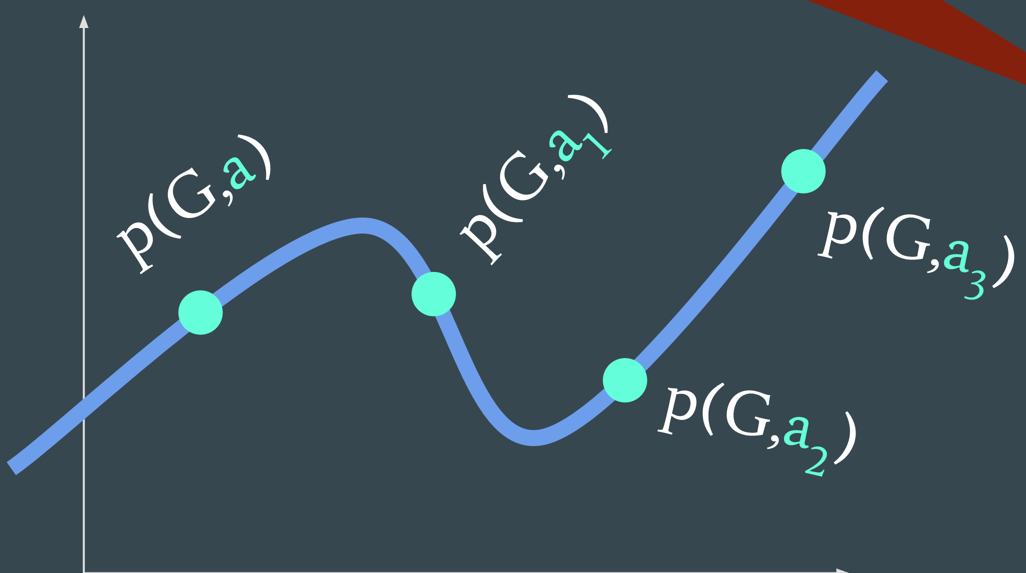


Only rules out $2^{o(m/\log m)}$ time algorithms under #ETH

Need **m+1 samples**

→ m+1 different gadgets

→ $m(G_i) \sim m \log m$



Graph polynomial p(G, z)

degree m

Block interpolation
Curticapean 15

Univariate $\mathbf{p}(G, z) \rightarrow$ Multivariate $\mathbf{q}(G, z_1, \ldots, z_{m/r})$
such that

- $\mathbf{p}(G, z) = \mathbf{q}(G, z_1, \ldots, z_{m/r})$
- Each z_i has degree $r=O(1)$

$\rightarrow$ Multivariate interpolation
$\sim r^{m/r} = \exp(\varepsilon\, m)$ **samples**
$r+1=O(1)$ distinct gadgets per variable

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

Approximate Counting

Valiant Vazirani 86

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

Traxler 14

If CNF-Sat is in $c^n \cdot poly(m)$ time,

we can $(1+1.1^{-n})$-approximate # CNF-Sat in time $(c + .00001)^n$

Is Counting really harder than Decision?

If SETH is true,

- CNF-SAT takes time 2^n
- # CNF-SAT takes time 2^n (even to approximate)
- 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 ?

Open problems

- 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 for # CNF-SAT ?

- is counting really harder than decision?
  can we tightly reduce QBF-SAT to # CNF-SAT ?