

# Exponential Time Complexity of Weighted Counting of Independent Sets

Christian Hoffmann

Donghua University, Shanghai

December 15th, 2010

# Main Theme

reduce SAT (satisfiability) to IS (independent set)

- ▶ counting problems
- ▶ preserve subexponential time  
 $2^{o(n)}$

# Independent Set Problem – Counting Versions

## counting – #IS

Input: graph  $G$ ,  $n$  vertices

Output: number of all independent sets

## weighted counting – $w\#IS$

Input: graph  $G = (V, E)$ ,  $n$  vertices,  $x \in \mathbb{Q}$

Output:  $\sum x^{|A|}$  over all independent sets  $A \subseteq V$

## counting ind. sets of size $\frac{1}{3}n$ – $\#\frac{1}{3}\text{-IS}$

Input: graph  $G$ ,  $n$  vertices

Output: number of independent sets of size exactly  $\frac{1}{3}n$

# Counting Complexity of IS

#IS, w#IS,  $\#\frac{1}{3}$ -IS

- ▶ fastest algorithms: time  $2^{O(n)}$

## Question

Is there a  $2^{o(n)}$  time algorithm for counting independent sets?

- ▶ #P-hard  
     $\rightsquigarrow$  no polynomial time algorithm unless  $\text{FP} = \#\text{P}$

## This work's goal

No  $2^{o(n)}$  time algorithm unless ...

# SAT – Counting Version

## # $d$ -SAT

Input:  $d$ -CNF formula  $\phi$  in  $n$  variables,  $m$  clauses

Output: number of satisfying assignments of  $\phi$

- ▶ fastest algorithms: time  $2^{O(n)}$
- ▶ #P-hard
  - ↪ no polynomial time algorithm unless  $FP = \#P$

# Exponential Time Hypothesis

## #ETH

$\exists c > 0$  such that

no deterministic algorithm can solve #3-SAT in time  $2^{c \cdot n}$

( $n$  = number of variables)

- ▶ Decision Version: Impagliazzo, Paturi, Zane (CCC 1999, J. Comp. Syst. Sc. 2001)
- ▶ Counting Version: Dell, Husfeldt, Wahlén (ICALP 2010)

# New Results

## Theorem (1)

*$\#_{\frac{1}{3}}IS$  requires time  $2^{\Omega(n)}$  unless  $\#ETH$  fails*

## Theorem (2)

*$w\#IS$  requires time  $2^{\Omega(n/\log^3 n)}$  unless  $\#ETH$  fails*

## Corollary

*$\#IS$  requires time  $2^{\Omega(n/\log^3 n)}$  unless  $\#ETH$  fails*

# First Insight

## Theorem (1)

*$\#_{\frac{1}{3}}$ -IS requires time  $2^{\Omega(n)}$  unless  $\#ETH$  fails*

Why is this conditional lower bound on  $\#_{\frac{1}{3}}$ -IS helpful?

- ▶ Many actual approaches to find/count IS (all ind. s., maximum ind. s., ...) can be used to solve  $\#_{\frac{1}{3}}$ -IS.
- ▶ Every approach to IS that runs in subexponential time must be
  - ▶ either powerful enough to solve SAT in subexponential time
  - ▶ or specialized enough to not be applicable to  $\#_{\frac{1}{3}}$ -IS.

# Proof of Theorem 1

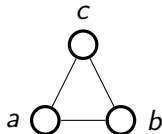
## Theorem (1)

$\# \frac{1}{3}$ -IS requires time  $2^{\Omega(n)}$  unless  $\#ETH$  fails

Proof:

- ▶ sparsify  $\#3SAT$  (Impagliazzo, Paturi, Zane; Dell, Husfeldt, Wahlén):  $2^{\Omega(\#variables)}$  iff  $2^{\Omega(\#clauses)}$
- ▶ reduce  $\#3SAT$  to  $\#X3SAT$  (from Schaefer 1978)  
 $\#X3SAT$ : clause satisfied iff *exactly* one literal true
- ▶ reduce  $\#X3SAT$  to  $\# \frac{1}{3}$ -IS

$a \vee b \vee c \mapsto$



similar to standard  
reduction  $3SAT \leq IS$   
– but *not* the same!!



# Proof of Theorem 2

Reduce  $\#\frac{1}{3}$ -IS to  $w\#IS$

- ▶ independent set **polynomial**

$$\mathcal{I}(G; x) = \sum_{\substack{A \subseteq V(G) \\ A \text{ independent}}} x^{|A|}$$

**coefficients:**  $\#\text{ind sets of size } 0, 1, 2, \dots, n$   
(includes  $\#\frac{1}{3}$ -IS)

**evaluation:** (weighted) counting of *all* ind sets

- ▶ reduce **computing all coefficients** to **evaluation** at 1 (or another fixed point)  
 $n + 1$  oracle calls, instance size:  $n \mapsto O(n \log^3 n)$
- ▶ conditional lower bounds:  $2^{\Omega(n)} \rightsquigarrow 2^{\Omega(n/\log^3 n)}$

## Usual Reduction “Computation $\leq$ Evaluation at $\xi$ ”

- ▶ Input: graph  $G$  with  $n$  vertices
- ▶ **construct**  $G_1, G_2, \dots, G_n$  such that

$$\mathcal{I}(G_i; \xi) = p_i \mathcal{I}(G; \xi_i)$$

for easy computable  $p_i$  and pairwise distinct  $\xi_i$

- ▶ **evaluate**  $(\mathcal{I}(G_1; \xi), \mathcal{I}(G_2; \xi), \dots, \mathcal{I}(G_n; \xi))$   
 $= (\mathcal{I}(G; \xi_1), \mathcal{I}(G; \xi_2), \dots, \mathcal{I}(G; \xi_n))$
- ▶ **interpolate**  $x$ -coefficients of  $\mathcal{I}(G; x)$

How to do the “construct” step?

# Old approach: vertex cloning

vertex cloning:

- ▶  $k$ -clone-transformation at vertex  $v$  with neighbors  $N(v)$ :  
 $v \mapsto$  vertices  $v_1, \dots, v_k$ , each with neighbors exactly  $N(v)$
- ▶ do this for every vertex  $v$  of  $G \rightsquigarrow G_k$

$$\mathcal{I}(G_k; x) = \mathcal{I}(G; (1+x)^k - 1)$$

Reduction via vertex cloning:

- ▶ 1-clone, 2-clone,  $\dots$ ,  $n$ -clone –  $n$  different evaluation point
- ▶ bad: size of oracle instances:  $|G|, 2|G|, \dots, n|G|$   
 $2^{\Omega(n)} \rightsquigarrow 2^{\Omega(n/n)}$

# Our approach: $S$ -cloning

$S$ -cloning:

- ▶  $S = \{s_1, \dots, s_\ell\}$  multiset of positive integers
- ▶  $v \mapsto v_1, \dots, v_\ell$  as with vertex cloning
- ▶ at  $v_i$  append path of length  $s_i$
- ▶ do this for every vertex  $v$  of  $G \rightsquigarrow G_S$

$$\mathcal{I}(G_S; x) = p_S \mathcal{I}(G; x_S)$$

for some  $x_S$  and some  $p_S$

reduction via  $S$ -cloning:

## Lemma

There are  $S_1, \dots, S_n$  such that

- ▶  $x_{S_i}$  are pairwise distance
- ▶  $|S_i| = O(\log n)$  and  $\sum_{S \in S_i} = O(\log^3 n)$

# Reduction via S-cloning

- ▶ size of oracle instances:  $O(\log^3 n)|G|$   
 $2^{\Omega(n)} \rightsquigarrow 2^{\Omega(n/\log^3 n)}$

# Insights

- ▶ **actual algorithms** for ind set are exponential time unless
  - ▶ also improve SAT to subexponential *or*
  - ▶ too special to solve  $\#\frac{1}{3}$ -IS
- ▶ **interpolation** techniques for counting problems from  $\#P$ -hardness theory
  - ▶ can be extended for considering **(sub)exponential time** complexity,
  - ▶ but leave a gap: only  $2^{\Omega(n/\log^3 n)}$  for  $\#IS$  (though  $2^{\Omega(n)}$  for  $\#\frac{1}{3}$ -IS and  $\#d$ -SAT)
- ▶ open problems:
  - ▶ close this gap
  - ▶ (sub)exponential time complexity of concrete problems
    - ▶ Is there a reduction  $\#\frac{1}{3}$ -IS to  $\#$ maximum ind set?
    - ▶ ...