The Complexity of Counting Problems

Tyson Williams
(University of Wisconsin-Madison)

Joint with:
Jin-Yi Cai and Heng Guo
(University of Wisconsin-Madison)

Michael Kowalczyk
(Northern Michigan University)

Complexity Theory Review

In the beginning, there was SAT

Problem: SAT
Input: A Boolean formula (in conjunctive normal form).
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

In the beginning, there was SAT

Problem: SAT
Input: A Boolean formula (in conjunctive normal form).
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

Example input:

$$(w \vee \overline{x} \vee \overline{y} \vee z) \wedge (x \vee y) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{w} \vee x \vee \overline{y} \vee \overline{z})$$

In the beginning, there was SAT

Problem: SAT
Input: A Boolean formula (in conjunctive normal form).
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

Example input:

$$(w \vee \overline{x} \vee \overline{y} \vee z) \wedge (x \vee y) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{w} \vee x \vee \overline{y} \vee \overline{z})$$

Example output: *Yes*

In the beginning, there was SAT

Problem: SAT
Input: A Boolean formula (in conjunctive normal form).
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

Example input:

$$(w \vee \overline{x} \vee \overline{y} \vee z) \wedge (x \vee y) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{w} \vee x \vee \overline{y} \vee \overline{z})$$

Example output: *Yes*
Satisfying assignment:

$$w = x = \text{True} \qquad y = z = \text{False}$$

In the beginning, there was SAT

Problem: SAT
Input: A Boolean formula (in conjunctive normal form).
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

Example input:

$$(w \vee \overline{x} \vee \overline{y} \vee z) \wedge (x \vee y) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{w} \vee x \vee \overline{y} \vee \overline{z})$$

Example output: *Yes*
Satisfying assignment:

$$w = x = \text{True} \qquad y = z = \text{False}$$

Theorem (Cook '71, Levin '73)

SAT $\in$ NP-Complete.

And SAT begat 3SAT

Problem: 3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

And SAT begat 3SAT

Problem: 3SAT
Input: A Boolean formula (in conjunctive normal form)
such that each clause has exactly 3 literals.
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

Example input:

$$(\overline{w} \vee x \vee z) \wedge (\overline{x} \vee y \vee z) \wedge (w \vee \overline{y} \vee z) \wedge (w \vee \overline{x} \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z}) \\ \wedge (\overline{w} \vee y \vee \overline{z}) \wedge (w \vee x \vee y) \wedge (\overline{w} \vee \overline{x} \vee \overline{y})$$

And SAT begat 3SAT

Problem: 3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

Example input:

$$(\overline{w} \vee x \vee z) \wedge (\overline{x} \vee y \vee z) \wedge (w \vee \overline{y} \vee z) \wedge (w \vee \overline{x} \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z})$$
$$\wedge (\overline{w} \vee y \vee \overline{z}) \wedge (w \vee x \vee y) \wedge (\overline{w} \vee \overline{x} \vee \overline{y})$$

Example output: *No*

And SAT begat 3SAT

Problem: 3SAT
Input: A Boolean formula (in conjunctive normal form)
such that each clause has exactly 3 literals.
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

Example input:

$$(\overline{w} \vee x \vee z) \wedge (\overline{x} \vee y \vee z) \wedge (w \vee \overline{y} \vee z) \wedge (w \vee \overline{x} \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z}) \\ \wedge (\overline{w} \vee y \vee \overline{z}) \wedge (w \vee x \vee y) \wedge (\overline{w} \vee \overline{x} \vee \overline{y})$$

Example output: *No*

Theorem

3SAT $\in$ NP-Complete.

And 3SAT begat 1-in-3SAT and NAE-3SAT

Problem: 1-IN-3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.

Problem: NAE-3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.

$$(\overline{w} \vee x \vee z) \wedge (\overline{x} \vee y \vee z) \wedge (w \vee \overline{y} \vee z) \wedge (w \vee \overline{x} \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z})$$

Problem: 1-IN-3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.
Output: "*Yes*" if each clause has exactly 1 true literal
"*No*" otherwise.

Problem: NAE-3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.

$$(\overline{w} \vee x \vee z) \wedge (\overline{x} \vee y \vee z) \wedge (w \vee \overline{y} \vee z) \wedge (w \vee \overline{x} \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z})$$

And 3SAT begat 1-in-3SAT and NAE-3SAT

Problem: 1-IN-3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.
Output: "*Yes*" if each clause has exactly 1 true literal
"*No*" otherwise.

Problem: NAE-3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.
Output: "*Yes*" if the literals in each clause are not all equal
"*No*" otherwise.

$$(\overline{w} \vee x \vee z) \wedge (\overline{x} \vee y \vee z) \wedge (w \vee \overline{y} \vee z) \wedge (w \vee \overline{x} \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z})$$

And 3SAT begat 1-in-3SAT and NAE-3SAT

Problem: 1-IN-3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.
Output: "*Yes*" if each clause has exactly 1 true literal
"*No*" otherwise.

Problem: NAE-3SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has exactly 3 literals.
Output: "*Yes*" if the literals in each clause are not all equal
"*No*" otherwise.

$$(\overline{w} \vee x \vee z) \wedge (\overline{x} \vee y \vee z) \wedge (w \vee \overline{y} \vee z) \wedge (w \vee \overline{x} \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z})$$

Theorem

1-IN-3SAT, NAE-3SAT $\in$ NP-Complete.

Then came the monotone versions

Definition

A Boolean formula is monotone if the formula contains no negations.

Then came the monotone versions

Definition

A Boolean formula is monotone if the formula contains no negations.

What is the complexity of...

- MON-3SAT?
- MON-1-IN-3SAT?
- MON-NAE-3SAT?

Then came the monotone versions

Definition

A Boolean formula is monotone if the formula contains no negations.

What is the complexity of...

- MON-3SAT?
- MON-1-IN-3SAT $\in$ NP-Complete
- MON-NAE-3SAT $\in$ NP-Complete

Then came the monotone versions

Definition

A Boolean formula is monotone if the formula contains no negations.

What is the complexity of...

- $\textsc{Mon}\text{-}3\mathrm{SAT} \in \mathrm{P}$
- $\textsc{Mon}\text{-}1\text{-}\textsc{in}\text{-}3\mathrm{SAT} \in$ NP-Complete
- $\textsc{Mon}\text{-}\mathrm{NAE}\text{-}3\mathrm{SAT} \in$ NP-Complete

What else is tractable?

Problem: Horn-SAT
Input: A Boolean formula (in conjunctive normal form)
such that each clause has at most 1 positive literal.
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

What else is tractable?

Problem: Horn-SAT
Input: A Boolean formula (in conjunctive normal form)
such that each clause has at most 1 positive literal.
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

Example input:

$$(\overline{w} \lor \overline{y} \lor z) \land (w \lor \overline{x} \lor \overline{z}) \land (x \lor \overline{y} \lor \overline{z}) \land (\overline{w} \lor y \lor \overline{z}) \land (\overline{w} \lor \overline{x} \lor \overline{y})$$

What else is tractable?

Problem: HORN-SAT
Input: A Boolean formula (in conjunctive normal form) such that each clause has at most 1 positive literal.
Output: "*Yes*" if there is a satisfying assignment
"*No*" otherwise.

Example input:

$$(\overline{w} \vee \overline{y} \vee z) \wedge (w \vee \overline{x} \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{w} \vee y \vee \overline{z}) \wedge (\overline{w} \vee \overline{x} \vee \overline{y})$$

Theorem

HORN-SAT $\in$ P.

Constraint Satisfaction Problems

These types of problems called Constraint Satisfaction Problems (CSPs).

These types of problems called Constraint Satisfaction Problems (CSPs).

Definition

The CSP defined by a set of constraints $\mathcal{F}$ is denoted by $\mathrm{CSP}(\mathcal{F})$.

These types of problems called Constraint Satisfaction Problems (CSPs).

Definition

The CSP defined by a set of constraints $\mathcal{F}$ is denoted by $\mathrm{CSP}(\mathcal{F})$.

Examples:

$$
\begin{array}{rll}
\mathrm{SAT} & \text{has} & \mathcal{F} = \{\mathsf{OR}_k \mid k \in \mathbb{N}\} \cup \{\mathsf{NOT}_2\} \\
\mathrm{3SAT} & \text{has} & \mathcal{F} = \{\mathsf{OR}_3, \mathsf{NOT}_2\} \\
\mathrm{1\text{-}\textsc{in}\text{-}3SAT} & \text{has} & \mathcal{F} = \{\mathsf{EXACTLY\text{-}ONE}_3, \mathsf{NOT}_2\} \\
\mathrm{NAE\text{-}3SAT} & \text{has} & \mathcal{F} = \{\mathsf{NOT\text{-}All\text{-}EQUAL}_3, \mathsf{NOT}_2\} \\
\\
\textsc{Mon}\mathrm{\text{-}3SAT} & \text{has} & \mathcal{F} = \{\mathsf{OR}_3\} \\
\textsc{Mon}\mathrm{\text{-}1\text{-}\textsc{in}\text{-}3SAT} & \text{has} & \mathcal{F} = \{\mathsf{EXACTLY\text{-}ONE}_3\} \\
\textsc{Mon}\mathrm{\text{-}NAE\text{-}3SAT} & \text{has} & \mathcal{F} = \{\mathsf{NOT\text{-}All\text{-}EQUAL}_3\}
\end{array}
$$

One theorem to rule them all

One theorem to rule them all

Theorem (Schaefer's dichotomy theorem '78)

*For any set of constraint functions $\mathcal{F}$,
the problem $\mathrm{CSP}(\mathcal{F})$ is NP-Complete
unless one of the following conditions holds,
in which case the problem is in P:*

One theorem to rule them all

Theorem (Schaefer's dichotomy theorem '78)

For any set of constraint functions $\mathcal{F}$,
the problem $\mathrm{CSP}(\mathcal{F})$ is NP-Complete
unless one of the following conditions holds,
in which case the problem is in P:

All constraints in $\mathcal{F}$...

1. *that are not constantly false are true when all its arguments are true;*
2. *that are not constantly false are true when all its arguments are false;*
3. *are equivalent to a conjunction of binary clauses;*
4. *are equivalent to a conjunction of Horn clauses;*
5. *are equivalent to a conjunction of dual-Horn clauses;*
6. *are equivalent to a conjunction of affine formula.*

One theorem to rule them all

Theorem (Schaefer's dichotomy theorem '78)

For any set of constraint functions $\mathcal{F}$,
the problem $\mathrm{CSP}(\mathcal{F})$ is NP-Complete
unless one of the following conditions holds,
in which case the problem is in P:

All constraints in $\mathcal{F}$...

1. *that are not constantly false are true when all its arguments are true;*
2. *that are not constantly false are true when all its arguments are false;*
3. *are equivalent to a conjunction of binary clauses;*
4. *are equivalent to a conjunction of Horn clauses;*
5. *are equivalent to a conjunction of dual-Horn clauses;*
6. *are equivalent to a conjunction of affine formula.*

One theorem to rule them all

Theorem (Schaefer's dichotomy theorem '78)

For any set of constraint functions $\mathcal{F}$,
the problem $\mathrm{CSP}(\mathcal{F})$ is NP-Complete
unless one of the following conditions holds,
in which case the problem is in P:

All constraints in $\mathcal{F}$...

1. *that are not constantly false are true when all its arguments are true;*
2. *that are not constantly false are true when all its arguments are false;*
3. *are equivalent to a conjunction of binary clauses;*
4. *are equivalent to a conjunction of Horn clauses;*
5. *are equivalent to a conjunction of dual-Horn clauses;*
6. *are equivalent to a conjunction of affine formula.*

Three takeaways from Schaefer's dichotomy theorem

Three takeaways from Schaefer's dichotomy theorem

- Theorist: describes the line between easy problems and hard problems

Three takeaways from Schaefer's dichotomy theorem

- Theorist: describes the line between easy problems and hard problems
- Practitioner (i.e. employee of Google): a complexity dictionary

Three takeaways from Schaefer's dichotomy theorem

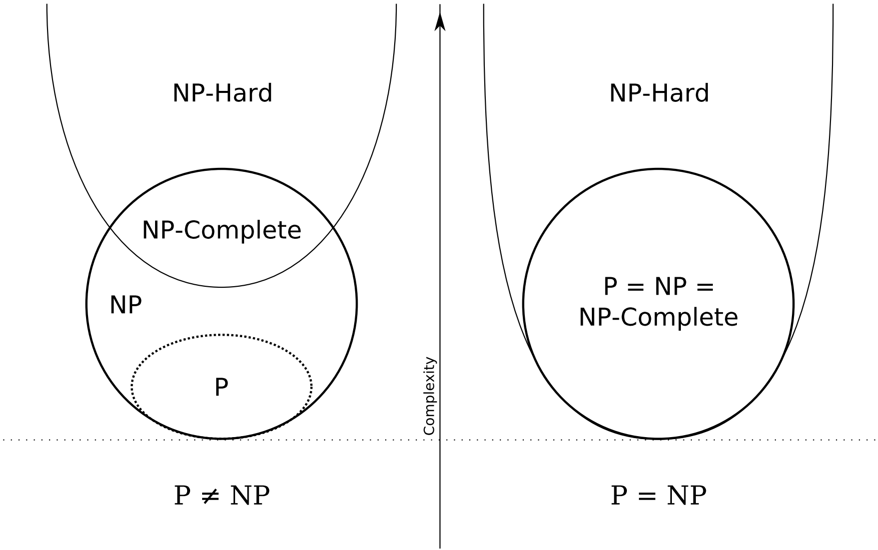
- Theorist: describes the line between easy problems and hard problems
- Practitioner (i.e. employee of Google): a complexity dictionary
- Observation: no problems of intermediate complexity

Three takeaways from Schaefer's dichotomy theorem

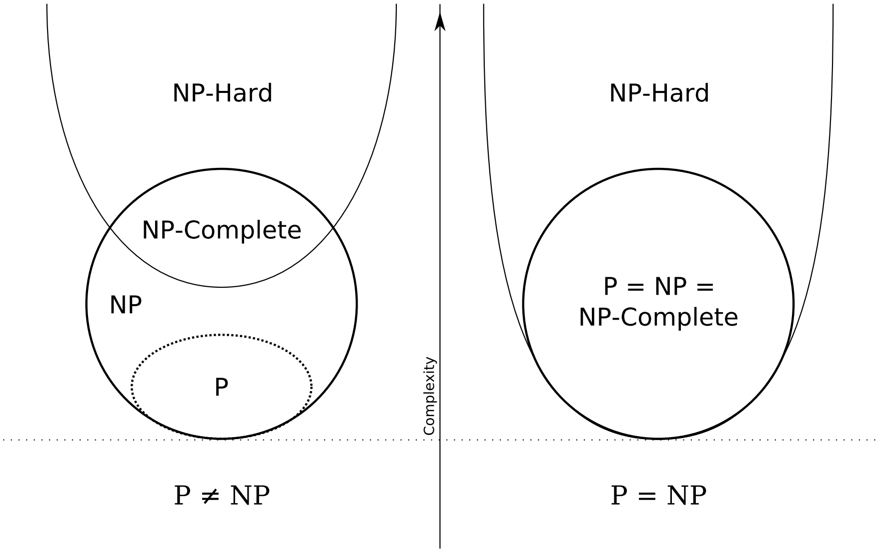
- Theorist: describes the line between easy problems and hard problems
- Practitioner (i.e. employee of Google): a complexity dictionary
- Observation: no problems of intermediate complexity

Theorem (Ladner's theorem '75)

If $\mathrm{P} \neq \mathrm{NP}$, then there exists problems in NP of intermediate complexity.



$P \neq NP$ $P = NP$

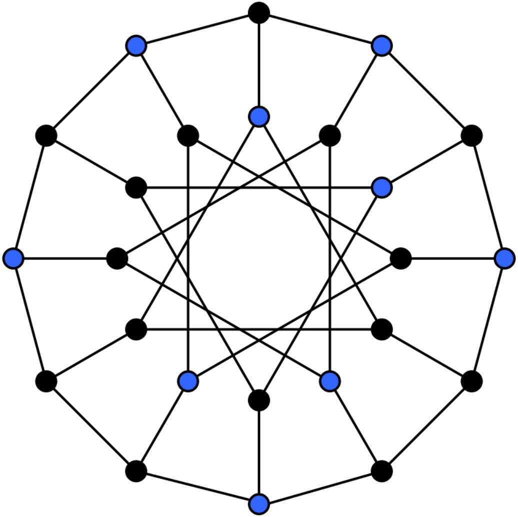


http://commons.wikimedia.org/wiki/File:P_np_np-complete_np-hard.svg

A Motivation for Counting

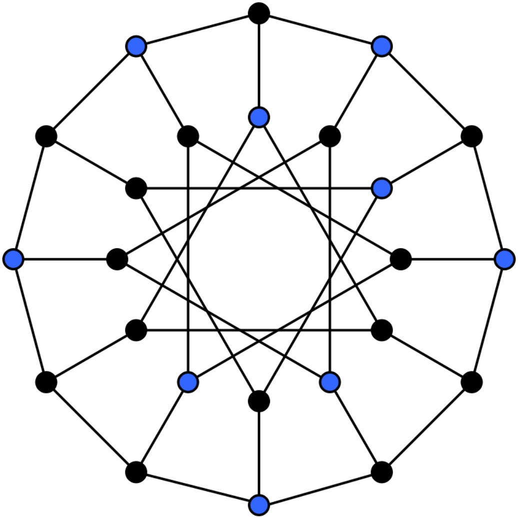
Independent Set

Want large independent set.



Independent Set

Want large independent set.



http://commons.wikimedia.org/wiki/File:Independent_set_graph.svg

Problem: IndependentSet
Input: A graph G and $k \in \mathbb{N}$.
Output: "*Yes*" if G contains an independent set of size at least k
"*No*" otherwise.

The Facts

Problem: IndependentSet
Input: A graph G and $k \in \mathbb{N}$.
Output: "*Yes*" if G contains an independent set of size at least k
"*No*" otherwise.

IndependentSet $\in$ NP-Complete

Problem: INDEPENDENTSET
Input: A graph G and $k \in \mathbb{N}$.
Output: "*Yes*" if G contains an independent set of size at least k
"*No*" otherwise.

INDEPENDENTSET $\in$ NP-Complete

Next question: how close to optimal can we get?

A Different Approach

Let $\mathcal{I}(G)$ be the set of independent sets in G.

A Different Approach

Let $\mathcal{I}(G)$ be the set of independent sets in G.

Want to randomly sample I from $\mathcal{I}(G)$ such that

$$\Pr(I) \propto w(I).$$

A Different Approach

Let $\mathcal{I}(G)$ be the set of independent sets in G.

Want to randomly sample I from $\mathcal{I}(G)$ such that

$$\Pr(I) \propto w(I).$$

Then must have

$$\Pr(I) = \frac{w(I)}{Z(G)},$$

where

$$Z(G) = \sum_{I \in \mathcal{I}(G)} w(I).$$

A Different Approach

Let $\mathcal{I}(G)$ be the set of independent sets in G.

Want to randomly sample I from $\mathcal{I}(G)$ such that

$$\Pr(I) \propto w(I).$$

Then must have

$$\Pr(I) = \frac{w(I)}{Z(G)},$$

where

$$Z(G) = \sum_{I \in \mathcal{I}(G)} w(I).$$

Know as the partition function in statistical physics.

Weight functions

$$\Pr(I) = \frac{w(I)}{Z(G)} \qquad \text{where} \qquad Z(G) = \sum_{I \in \mathcal{I}(G)} w(I).$$

Weight functions

$$\Pr(I) = \frac{w(I)}{Z(G)} \qquad \text{where} \qquad Z(G) = \sum_{I \in \mathcal{I}(G)} w(I).$$

If $w(I) = 1$, then $Z(G) = |\mathcal{I}(G)|$ is the number of independent sets.

Weight functions

$$\Pr(I) = \frac{w(I)}{Z(G)} \qquad \text{where} \qquad Z(G) = \sum_{I \in \mathcal{I}(G)} w(I).$$

If $w(I) = 1$, then $Z(G) = |\mathcal{I}(G)|$ is the number of independent sets.

Statistical physicists considered $w(I) = \lambda^{|I|}$ from some nonnegative $\lambda \in \mathbb{R}$.

Weight functions

$$\Pr(I) = \frac{w(I)}{Z(G)} \qquad \text{where} \qquad Z(G) = \sum_{I \in \mathcal{I}(G)} w(I).$$

If $w(I) = 1$, then $Z(G) = |\mathcal{I}(G)|$ is the number of independent sets.

Statistical physicists considered $w(I) = \lambda^{|I|}$ from some nonnegative $\lambda \in \mathbb{R}$.

- If $\lambda = 1$, then $Z(G) = |\mathcal{I}(G)|$ again.

Weight functions

$$\Pr(I) = \frac{w(I)}{Z(G)} \qquad \text{where} \qquad Z(G) = \sum_{I \in \mathcal{I}(G)} w(I).$$

If $w(I) = 1$, then $Z(G) = |\mathcal{I}(G)|$ is the number of independent sets.

Statistical physicists considered $w(I) = \lambda^{|I|}$ from some nonnegative $\lambda \in \mathbb{R}$.

- If $\lambda = 1$, then $Z(G) = |\mathcal{I}(G)|$ again.
- If $\lambda = 0$, then $Z(G) = 1$.

Weight functions

$$\Pr(I) = \frac{w(I)}{Z(G)} \qquad \text{where} \qquad Z(G) = \sum_{I \in \mathcal{I}(G)} w(I).$$

If $w(I) = 1$, then $Z(G) = |\mathcal{I}(G)|$ is the number of independent sets.

Statistical physicists considered $w(I) = \lambda^{|I|}$ from some nonnegative $\lambda \in \mathbb{R}$.

- If $\lambda = 1$, then $Z(G) = |\mathcal{I}(G)|$ again.
- If $\lambda = 0$, then $Z(G) = 1$.

Theorem (Sly,Sun '12)

For $d \geq 3$ and $\lambda > \lambda_c(d) = \frac{(d-1)^{d-1}}{(d-2)^d}$, unless $NP = RP$ there is no approximation algorithm for the partition function with $w(I) = \lambda^{|I|}$ on d-regular graphs.

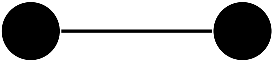
Local Constraints

Definition

A vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex in the set.

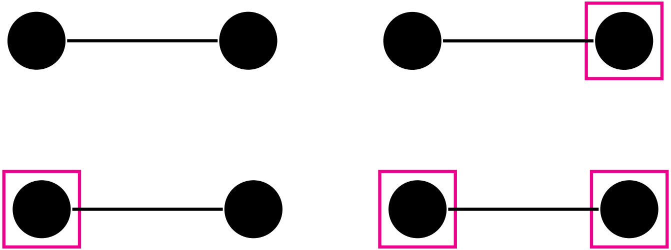
Definition

A vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex in the set.



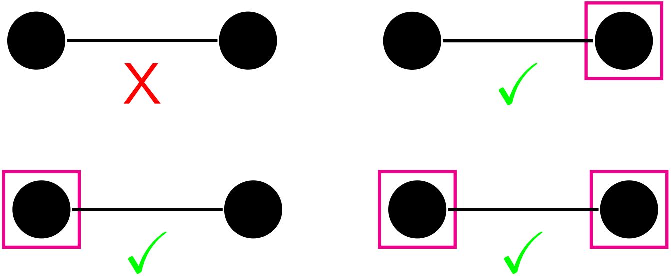
Definition

A vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex in the set.



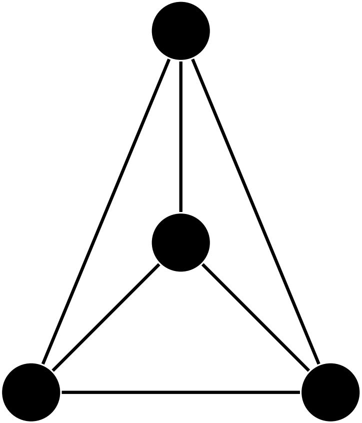
Definition

A vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex in the set.



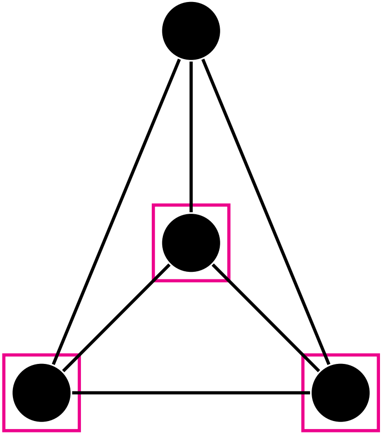
Systematic Approach to #VertexCover

- $G = (V, E)$

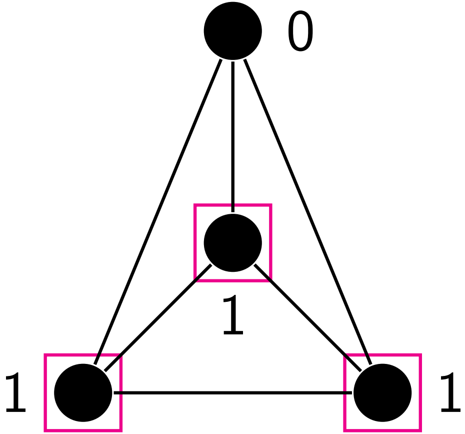


Systematic Approach to #VertexCover

- $G = (V, E)$

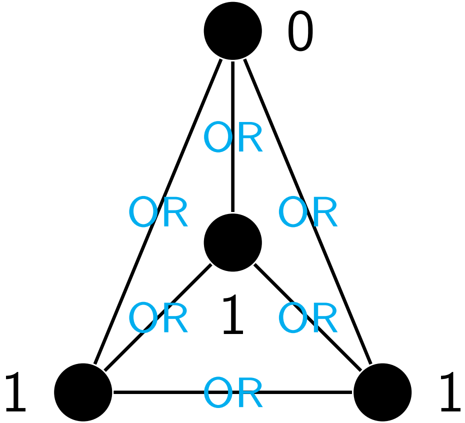


- $G = (V, E)$
- $\sigma : V \to \{0, 1\}$



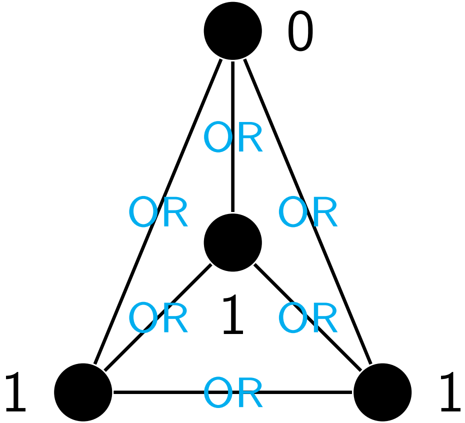
Systematic Approach to #VertexCover

- $G = (V, E)$
- $\sigma : V \to \{0, 1\}$



Systematic Approach to #VertexCover

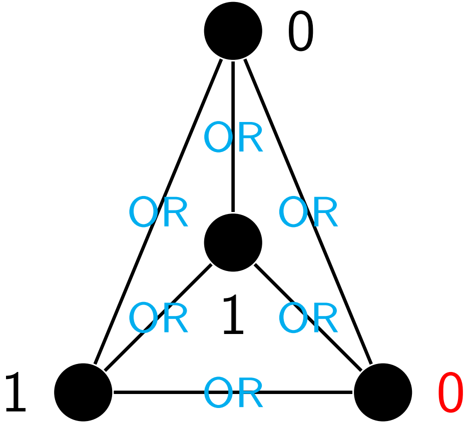
- $G = (V, E)$
- $\sigma : V \to \{0, 1\}$



$$\prod_{(u,v)\in E} \mathrm{OR}(\sigma(u), \sigma(v)) = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Systematic Approach to #VertexCover

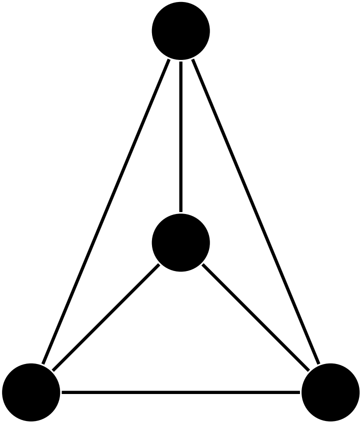
- $G = (V, E)$
- $\sigma : V \to \{0, 1\}$



$$\prod_{(u,v) \in E} \mathrm{OR}(\sigma(u), \sigma(v)) = 1 \cdot 1 \cdot 0 \cdot 1 \cdot 1 \cdot 1 = 0$$

Systematic Approach to #VertexCover

- $G = (V, E)$
- $\sigma : V \to \{0, 1\}$



$$\#\textsc{VertexCover}(G) = \sum_{\sigma : V \to \{0,1\}} \prod_{(u,v) \in E} \mathrm{OR}(\sigma(u), \sigma(v))$$

Generalize

$$\sum_{\sigma: V \to \{0,1\}} \prod_{(u,v) \in E} \mathrm{OR}(\sigma(u), \sigma(v))$$

$$\sum_{\sigma: V \to \{0,1\}} \prod_{(u,v) \in E} \mathrm{OR}\left(\sigma(u), \sigma(v)\right)$$

Input		**Output**
p	q	$\mathrm{OR}(p, q)$
0	0	0
0	1	1
1	0	1
1	1	1

Generalize

$$\sum_{\sigma:V\to\{0,1\}} \prod_{(u,v)\in E} f(\sigma(u),\sigma(v))$$

Input		Output
p	q	$\mathrm{OR}(p,q)$
0	0	0
0	1	1
1	0	1
1	1	1

Input		Output
p	q	$f(p,q)$
0	0	w
0	1	x
1	0	y
1	1	z

where $w, x, y, z \in \mathbb{C}$

Generalize

Partition Function: $Z(\cdot)$

$$Z(G) = \sum_{\sigma: V \to \{0,1\}} \prod_{(u,v) \in E} f(\sigma(u), \sigma(v))$$

Input		Output
p	q	$\mathrm{OR}(p, q)$
0	0	0
0	1	1
1	0	1
1	1	1

Input		Output
p	q	$f(p, q)$
0	0	w
0	1	x
1	0	y
1	1	z

where $w, x, y, z \in \mathbb{C}$

Theorem (Cai, Kowalczyk, W '12)

Over 3-regular graphs G, the exact counting problem for any (binary) complex-weighted function f

$$Z(G) = \sum_{\sigma : V \to \{0,1\}} \prod_{(u,v) \in E} f(\sigma(u), \sigma(v))$$

is either computable in polynomial time or $\#\mathrm{P}$-hard.

Problem: HamiltonianCycle
Input: A graph G.
Output: "*Yes*" if G contains an Hamiltonian cycle
"*No*" otherwise.

Nonlocal Examples

Problem: HamiltonianCycle
Input: A graph G.
Output: "*Yes*" if G contains an Hamiltonian cycle
"*No*" otherwise.

Problem: Connected
Input: A graph G.
Output: "*Yes*" if G is connected
"*No*" otherwise.

Problem: HAMILTONIANCYCLE
Input: A graph G.
Output: "*Yes*" if G contains an Hamiltonian cycle
"*No*" otherwise.

Problem: CONNECTED
Input: A graph G.
Output: "*Yes*" if G is connected
"*No*" otherwise.

Confessions of a theorists:

Nonlocal Examples

Problem: HamiltonianCycle
Input: A graph G.
Output: "*Yes*" if G contains an Hamiltonian cycle
"*No*" otherwise.

Problem: Connected
Input: A graph G.
Output: "*Yes*" if G is connected
"*No*" otherwise.

Confessions of a theorists:

- Some proofs of this depending on definition of "local".

Problem: HAMILTONIANCYCLE
Input: A graph G.
Output: "*Yes*" if G contains an Hamiltonian cycle
"*No*" otherwise.

Problem: CONNECTED
Input: A graph G.
Output: "*Yes*" if G is connected
"*No*" otherwise.

Confessions of a theorists:

- Some proofs of this depending on definition of "local".
- Formally, just think of these as conjectures.

Symmetric Function

Definition

A function is symmetric if invariant under any permutation of its inputs.

Definition

A function is symmetric if invariant under any permutation of its inputs.

Examples:

$$\begin{aligned} \mathrm{OR}_2 &= [0, 1, 1] \\ \mathrm{AND}_3 &= [0, 0, 0, 1] \\ \text{EVEN-PARITY}_4 &= [1, 0, 1, 0, 1] \\ \mathrm{MAJORITY}_5 &= [0, 0, 0, 1, 1, 1] \\ (=_6) = \mathrm{EQUALITY}_6 &= [1, 0, 0, 0, 0, 0, 1] \end{aligned}$$

Constraint Graph for $\#\text{CSP}(\mathcal{F})$ Instance

$\mathcal{F} = \{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_3\}$

Constraint Graph for $\#\text{CSP}(\mathcal{F})$ Instance

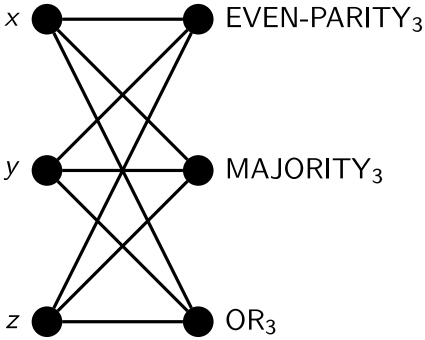
$\mathcal{F} = \{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_3\}$

$$\text{EVEN-PARITY}_3(x, y, z) \wedge \text{MAJORITY}_3(x, y, z) \wedge \text{OR}_3(x, y, z)$$

Constraint Graph for $\#\text{CSP}(\mathcal{F})$ Instance

$\mathcal{F} = \{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_3\}$

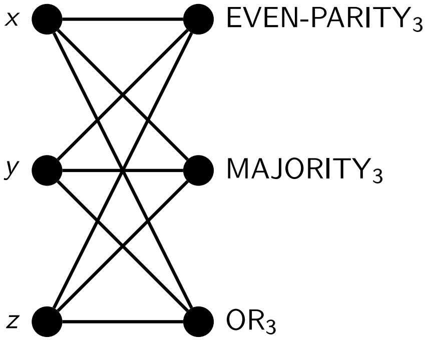
$$\text{EVEN-PARITY}_3(x, y, z) \wedge \text{MAJORITY}_3(x, y, z) \wedge \text{OR}_3(x, y, z)$$



Constraint Graph for #CSP($\mathcal{F}$) Instance

$\mathcal{F} = \{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_3\}$

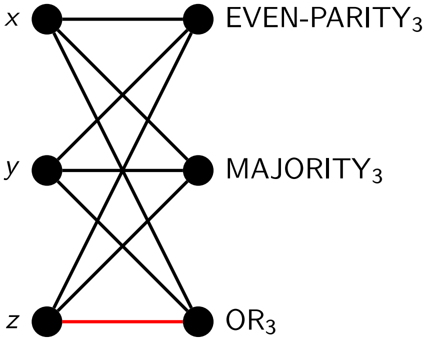
$$\text{EVEN-PARITY}_3(x, y, z) \wedge \text{MAJORITY}_3(x, y, z) \wedge \text{OR}_3(x, y, z)$$



NOT planar, so **NOT** an instance of PI-#CSP($\{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_3\}$)

$\mathcal{F} = \{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_3\}$

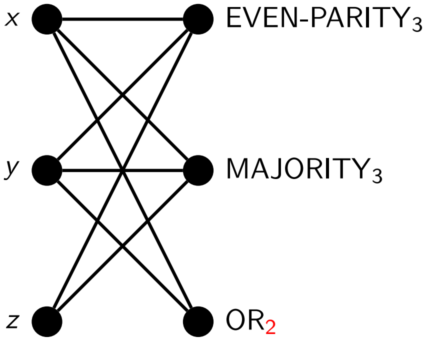
$$\text{EVEN-PARITY}_3(x, y, z) \wedge \text{MAJORITY}_3(x, y, z) \wedge \text{OR}_3(x, y, z)$$



NOT planar, so **NOT** an instance of
PI-#CSP($\{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_3\}$)

$\mathcal{F} = \{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_2\}$

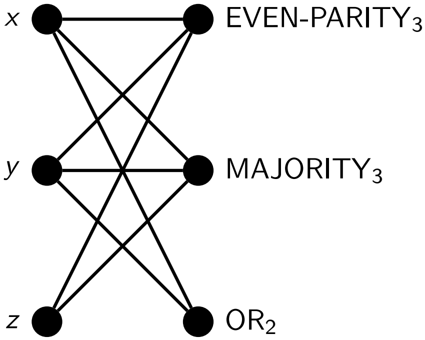
$$\text{EVEN-PARITY}_3(x, y, z) \wedge \text{MAJORITY}_3(x, y, z) \wedge \text{OR}_2(x, y)$$

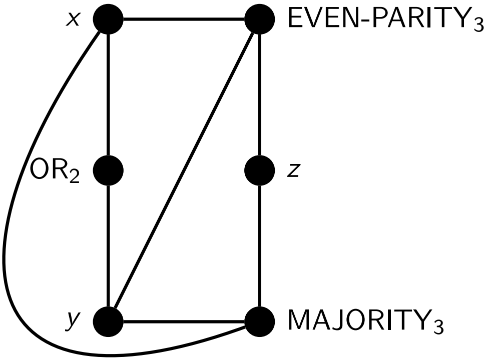


Constraint Graph for $\#\text{CSP}(\mathcal{F})$ Instance

$\mathcal{F} = \{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_2\}$

$$\text{EVEN-PARITY}_3(x, y, z) \wedge \text{MAJORITY}_3(x, y, z) \wedge \text{OR}_2(x, y)$$





VALID instance of PI-$\#\text{CSP}(\{\text{EVEN-PARITY}_3, \text{MAJORITY}_3, \text{OR}_2\})$

More Counting Dichotomies

Theorem (Cai, Lu, Xia '09)

Let $\mathcal{F}$ be any set of complex-valued constraints in Boolean variables. Then $\#\mathsf{CSP}(\mathcal{F})$ is either $\#\mathrm{P}$-hard or computable in polynomial time.

More Counting Dichotomies

Theorem (Cai, Lu, Xia '09)

Let $\mathcal{F}$ be any set of complex-valued constraints in Boolean variables. Then $\#\text{CSP}(\mathcal{F})$ is either $\#\text{P}$-hard or computable in polynomial time.

Theorem (Cai, Xia '12)

Let $\mathcal{F}$ be any set of complex-valued constraints. Then $\#\text{CSP}(\mathcal{F})$ is either $\#\text{P}$-hard or computable in polynomial time.

More Counting Dichotomies

Theorem (Cai, Lu, Xia '09)

Let $\mathcal{F}$ be any set of complex-valued constraints in Boolean variables. Then $\#\text{CSP}(\mathcal{F})$ is either $\#\mathrm{P}$-hard or computable in polynomial time.

Theorem (Cai, Xia '12)

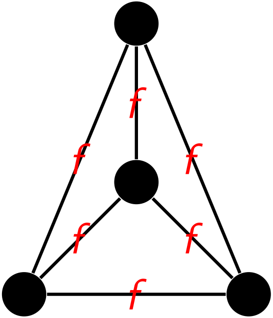
Let $\mathcal{F}$ be any set of complex-valued constraints. Then $\#\text{CSP}(\mathcal{F})$ is either $\#\mathrm{P}$-hard or computable in polynomial time.

Theorem (Guo, W '13)

Let $\mathcal{F}$ be any set of symmetric, complex-valued constraints in Boolean variables. Then PI-$\#\text{CSP}(\mathcal{F})$ is either $\#\mathrm{P}$-hard or computable in polynomial time.

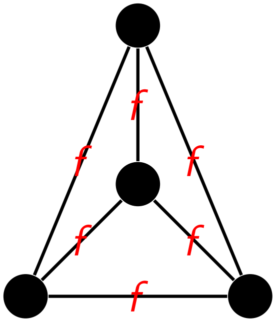
Definition of Holant Function

- Partition Function



$$\sum_{\sigma:V\to\{0,1\}} \prod_{(u,v)\in E} f(\sigma(u), \sigma(v))$$

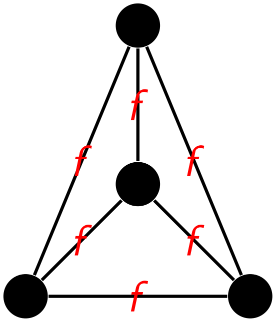
- Partition Function
  - Assignments to vertices
  - Functions on edges



$$\sum_{\sigma:V\to\{0,1\}} \prod_{(u,v)\in E} f\left(\sigma(u),\sigma(v)\right)$$

Definition of Holant Function

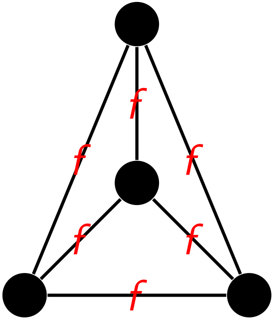
- Partition Function
  - Assignments to vertices
  - Functions on edges



$$\sum_{\sigma: V \to \{0,1\}} \prod_{(u,v) \in E} f(\sigma(u), \sigma(v))$$

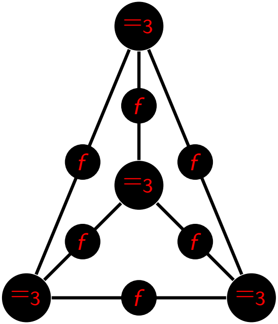
- Holant Function
  - Assignment to edges
  - Functions on vertices

- Partition Function
  - Assignments to vertices
  - Functions on edges



$$\sum_{\sigma:V\to\{0,1\}} \prod_{(u,v)\in E} f(\sigma(u),\sigma(v))$$

- Holant Function
  - Assignment to edges
  - Functions on vertices

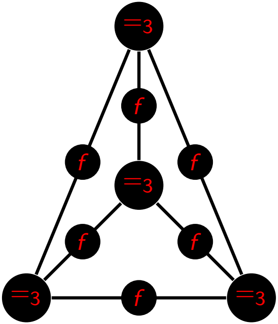


$$\sum_{\sigma:E\to\{0,1\}} \prod_{v\in V} g_v\left(\sigma\mid_{E(v)}\right)$$

Definition of Holant Function

- Holant($\{f\} \mid \{=_3\}$) is a counting problem defined over (2,3)-regular bipartite graphs.

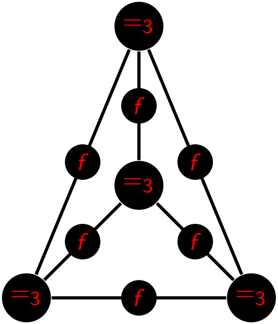
- Holant Function
  - Assignment to edges
  - Functions on vertices



$$\sum_{\sigma: E \to \{0,1\}} \prod_{v \in V} g_v \left(\sigma \mid_{E(v)} \right)$$

- Holant($\{f\} \mid \{=_3\}$) is a counting problem defined over (2,3)-regular bipartite graphs.
- Degree 2 vertices take f.
- Degree 3 vertices take $=_3$.

- Holant Function
  - Assignment to edges
  - Functions on vertices



$$\sum_{\sigma:E\to\{0,1\}} \prod_{v\in V} g_v\left(\sigma \mid_{E(v)}\right)$$

Example Holant Problems

- $\text{Holant}(\{\text{OR}_2\} \mid \{=_3\})$ is $\#\text{VertexCover}$ on 3-regular graphs.

Example Holant Problems

- Holant($\{\text{OR}_2\} \mid \{=_3\}$) is $\#\text{VertexCover}$ on 3-regular graphs.
- Holant($\{\text{NAND}_2\} \mid \{=_3\}$) is $\#\text{IndependentSet}$ on 3-regular graphs.

Example Holant Problems

- Holant($\{\text{OR}_2\} \mid \{=_3\}$) is $\#\textsc{VertexCover}$ on 3-regular graphs.
- Holant($\{\text{NAND}_2\} \mid \{=_3\}$) is $\#\textsc{IndependentSet}$ on 3-regular graphs.
- $\left.\begin{array}{l}\text{Holant}(\{=_2\} \mid \{\text{AT-MOST-ONE}\}) \\ \text{Holant}(\text{AT-MOST-ONE})\end{array}\right\}$ is $\#\textsc{Matching}$.

Example Holant Problems

- Holant($\{\mathrm{OR}_2\} \mid \{=_3\}$) is #VertexCover on 3-regular graphs.
- Holant($\{\mathrm{NAND}_2\} \mid \{=_3\}$) is #IndependentSet on 3-regular graphs.
- Holant($\{=_2\} \mid \{$AT-MOST-ONE$\}$), Holant(AT-MOST-ONE) } is #Matching.
- Holant($\{=_2\} \mid \{$EXACTLY-ONE$\}$), Holant(EXACTLY-ONE) } is #PerfectMatching.

Theorem (Cai, Guo, W '13)

Let $\mathcal{F}$ be any set of symmetric, complex-valued constraints in Boolean variables.
Then Holant($\mathcal{F}$) *is either $\#\mathrm{P}$-hard or computable in polynomial time.*

A Proof Technique: Polynomial Interpolation

Polynomial Interpolation

- 2 (distinct) points defines a (unique) line

Polynomial Interpolation

- 2 (distinct) points defines a (unique) line
- 3 (distinct) points defines a (unique) quadratic
  (actually: polynomial of degree at most 2)

Polynomial Interpolation

- 2 (distinct) points defines a (unique) line
- 3 (distinct) points defines a (unique) quadratic
  (actually: polynomial of degree at most 2)

Lemma

Given $n + 1$ distinct points (x_i, y_i), there is a unique polynomial $p(\cdot)$ of degree at most n such that $p(x_i) = y_i$.

- 2 (distinct) points defines a (unique) line
- 3 (distinct) points defines a (unique) quadratic
  (actually: polynomial of degree at most 2)

Lemma

Given $n+1$ *distinct points* (x_i, y_i)*, there is a unique polynomial* $p(\cdot)$ *of degree at most* n *such that* $p(x_i) = y_i$.

Furthermore, the coefficients of p *can be computed in polynomial time.*

#PerfectMatching $\leq_T$ #Matching [Valiant '79]

#PerfectMatching $\leq_T$ #Matching [Valiant '79]

- Given a graph G with n vertices.

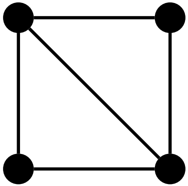
#PerfectMatching $\leq_T$ #Matching [Valiant '79]

- Given a graph G with n vertices.
- Assume we can compute the number of matchings in any graph.

#PerfectMatching $\leq_T$ #Matching [Valiant '79]

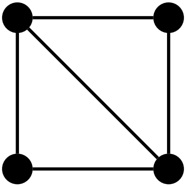
- Given a graph G with n vertices.
- Assume we can compute the number of matchings in any graph.
- Goal is to compute the number of perfect matchings in G.

- Given a graph G with n vertices.
- Assume we can compute the number of matchings in any graph.
- Goal is to compute the number of perfect matchings in G.



#PerfectMatching $\leq_T$ #Matching [Valiant '79]

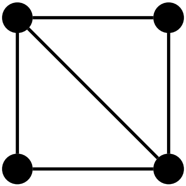
- Given a graph G with n vertices.
- Assume we can compute the number of matchings in any graph.
- Goal is to compute the number of perfect matchings in G.



- Let m_k be the number of matchings that omit k vertices.

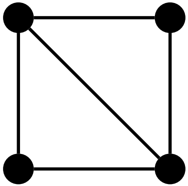
#PerfectMatching $\leq_T$ #Matching [Valiant '79]

- Given a graph G with n vertices.
- Assume we can compute the number of matchings in any graph.
- Goal is to compute the number of perfect matchings in G.



- Let m_k be the number of matchings that omit k vertices.
- Let G_ℓ be the graph G after adding, for each vertex v, ℓ vertices incident only to v.

- Given a graph G with n vertices.
- Assume we can compute the number of matchings in any graph.
- Goal is to compute the number of perfect matchings in G.



- Let m_k be the number of matchings that omit k vertices.
- Let G_ℓ be the graph G after adding, for each vertex v, ℓ vertices incident only to v.

$$\#\text{Matching}(G_\ell) = \sum_{k=0}^{n} m_k (\ell + 1)^k.$$

Thank You