Approximate Counting II: Constraint Satisfaction Problems

David Richerby

University of Oxford

Counting Complexity and Phase Transitions Boot Camp

Approximate counting (revision)

FPRAS: randomized algorithm for a function $f : \Sigma^* \to \mathbb{R}$.

Input: Instance x and error tolerance ϵ.

Output: A value z such that

$$\Pr[\mathrm{e}^{-\epsilon} f(x) \leq z \leq \mathrm{e}^{\epsilon} f(x)] \geq \tfrac{3}{4}\,.$$

Running time polynomial in $|x|$ and $1/\epsilon$.

AP-reducibility: if there's an FPRAS for f and g is AP-reducible to f, there's an FPRAS for g.

Constraint Satisfaction Problems

Fix a finite domain D and a set Γ of named relations over D.

Instance: A set V of variables and a set of constraints, each of the form $(R, x_i, \dots, x_k)$ where R is a k-ary relation in Γ and each $x_i \in V$.

Constraint Satisfaction Problems

Fix a finite domain D and a set Γ of named relations over D.

Instance: A set V of variables and a set of constraints, each of the form $(R, x_i, \dots, x_k)$ where R is a k-ary relation in Γ and each $x_i \in V$.

An assignment $\sigma\colon V \to D$ is *satisfying* if $(\sigma(x_1), \dots, \sigma(x_k)) \in R$ for each constraint.

$\#\mathrm{CSP}(\Gamma)$: given an instance with constraints from Γ, how many satisfying assignments are there? (Γ is a parameter.)

#3-SAT is a counting CSP

Domain $D = \{0,1\}$; $\Gamma = \{R_0, R_1, R_2, R_3\}$, where

$$R_0 = \{0,1\}^3 \setminus \{000\} \qquad \text{clause } x \vee y \vee z$$
$$R_1 = \{0,1\}^3 \setminus \{100\} \qquad \text{clause } \neg x \vee y \vee z$$
$$R_2 = \{0,1\}^3 \setminus \{110\} \qquad \text{clause } \neg x \vee \neg y \vee z$$
$$R_3 = \{0,1\}^3 \setminus \{111\} \qquad \text{clause } \neg x \vee \neg y \vee \neg z\,.$$

#3-SAT is a counting CSP

Domain $D = \{0,1\}$; $\Gamma = \{R_0, R_1, R_2, R_3\}$, where

$$R_0 = \{0,1\}^3 \setminus \{000\} \qquad \text{clause } x \vee y \vee z$$
$$R_1 = \{0,1\}^3 \setminus \{100\} \qquad \text{clause } \neg x \vee y \vee z$$
$$R_2 = \{0,1\}^3 \setminus \{110\} \qquad \text{clause } \neg x \vee \neg y \vee z$$
$$R_3 = \{0,1\}^3 \setminus \{111\} \qquad \text{clause } \neg x \vee \neg y \vee \neg z\,.$$

Bad news: no FPRAS for #3-SAT unless $\mathrm{NP} = \mathrm{RP}$.

General theme: this talk will be about computational hardness, rather than approximation algorithms.

CSP – alternative view

Can view an instance I and a constraint language Γ as relational structures, and a satisfying assignment for I is a homomorphism $I \to \Gamma$.

If there are homomorphisms $I_1 \to I_2$ and $I_2 \to \Gamma$, then I_1 and I_2 must both be "yes" instances.

CSP – alternative view

Can view an instance I and a constraint language Γ as relational structures, and a satisfying assignment for I is a homomorphism $I \to \Gamma$.

If there are homomorphisms $I_1 \to I_2$ and $I_2 \to \Gamma$, then I_1 and I_2 must both be "yes" instances.

Non-example: Hamiltonicity is not a CSP.

Corollary: not every $\#\mathrm{P}$ problem is a $\#\mathrm{CSP}(\Gamma)$, so Ladner-like hierarchies doesn't necessarily apply to exact or approx $\#\mathrm{CSP}$.

Complexity of exact $\#\mathrm{CSP}(\Gamma)$

Theorem. For every constraint language Γ, $\#\mathrm{CSP}(\Gamma)$ is either in FP or is $\#\mathrm{P}$-complete.

Xi Chen's second talk will have details.

Approximation trichotomy for Boolean $\#\mathrm{CSP}(\Gamma)$

Theorem. Let Γ be a constraint language over domain $\{0, 1\}$.

- If every relation in Γ is affine, then $\#\mathrm{CSP}(\Gamma) \in \mathrm{FP}$.
- Otherwise, if every relation in Γ can be defined by a conjunction of predicates $x_i = 0$, $x_i = 1$ and $x_i \to x_j$, then $\#\mathrm{CSP}(\Gamma) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$.
- Otherwise, $\#\mathrm{CSP}(\Gamma) \equiv_{\mathrm{AP}} \#\mathrm{SAT}$.

$\#\mathrm{BIS}$ and $\#\mathrm{CSP}(\mathrm{IMP})$

Let $\mathrm{IMP} = \{00, 01, 11\}$.

Lemma. $\#\mathrm{BIS} \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathrm{IMP})$.

Proof. Replace every edge $(x, y) \in V_{\mathrm{L}} \times V_{\mathrm{R}}$ with the constraint (IMP, x, y). This forbids $x = 1$, $y = 0$ so 1 means "in" on left and 0 means "in" on right. $\square$

#BIS and #CSP(IMP)

Let $\text{IMP} = \{00, 01, 11\}$.

Lemma. $\#\text{BIS} \leq_{\text{AP}} \#\text{CSP}(\text{IMP})$.

Proof. Replace every edge $(x, y) \in V_{\text{L}} \times V_{\text{R}}$ with the constraint (IMP, x, y). This forbids $x = 1$, $y = 0$ so 1 means "in" on left and 0 means "in" on right. □

Lemma. $\#\text{CSP}(\text{IMP}, {=}0, {=}1) \leq_{\text{AP}} \#\text{BIS}$.

Proof. (IMP, x, y) enforces $x \leq y$ so imposes a partial order on the variables' values. Satisfying assignments correspond to downsets and $\#\text{DOWNSETS} \leq_{\text{AP}} \#\text{BIS}$. □

Gadgets

Knowing that $\#\mathrm{BIS} \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathrm{IMP})$ isn't enough.

Our constraint language contains relations built from IMP, $=_0$, $=_1$ but might not actually contain IMP.

Gadgets

Knowing that $\#\mathrm{BIS} \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathrm{IMP})$ isn't enough.

Our constraint language contains relations built from IMP, $=$0, $=$1 but might not actually contain IMP.

Suppose a relation $R \in \Gamma$ contains n_0 tuples t with $t_1 = 0$ and n_1 with $t_0 = 1$. If $n_0 > n_1$, simulate the constraint $x = 0$ by many constraints $R(x, v_2, \ldots, v_k)$.

Can implement IMP from a relation defined using it.

Gadgets

Knowing that $\#\mathrm{BIS} \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathrm{IMP})$ isn't enough.

Our constraint language contains relations built from IMP, $=$0, $=$1 but might not actually contain IMP.

Suppose a relation $R \in \Gamma$ contains n_0 tuples t with $t_1 = 0$ and n_1 with $t_0 = 1$. If $n_0 > n_1$, simulate the constraint $x = 0$ by many constraints $R(x, v_2, \dots, v_k)$.

Can implement IMP from a relation defined using it.

Can implement OR or NAND from any relation that's not affine or defined from IMP, $=$0 and $=$1.

Approximation trichotomy for Boolean $\#\mathrm{CSP}(\Gamma)$

Theorem. Let Γ be a constraint language over domain $\{0, 1\}$.

- If every relation in Γ is affine, then $\#\mathrm{CSP}(\Gamma) \in \mathrm{FP}$.
- Otherwise, if every relation in Γ can be defined by a conjunction of predicates $x_i = 0$, $x_i = 1$ and $x_i \to x_j$, then $\#\mathrm{CSP}(\Gamma) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$.
- Otherwise, $\#\mathrm{CSP}(\Gamma) \equiv_{\mathrm{AP}} \#\mathrm{SAT}$.

Weighted #CSP

Extend constraint languages from sets of relations to sets $\mathcal{F}$ of functions $D^k \to \mathbb{R}_{\geq 0}$.

Goal is to compute the partition function of an instance I, given by

$$Z(I) = \sum_{\sigma\colon V \to D} \prod_i f_i(\sigma(x_{i,1}), \dots, \sigma(x_{i,k}))\,.$$

Unweighted case corresponds to functions $D^k \to \{0, 1\}$.

Example – Ising model

Ising with no external field is just $\#\mathrm{CSP}(\{f\})$ where

$$f(0,0) = f(1,1) = \lambda$$
$$f(0,1) = f(1,0) = 1\,.$$

(Mixed) external fields can be implemented by adding unary functions to the constraint language.

Complexity of exact weighted $\#\mathrm{CSP}(\mathcal{F})$

Theorem. For every weighted constraint language $\mathcal{F}$, $\#\mathrm{CSP}(\mathcal{F})$ is either in FP or is $\mathrm{FP}^{\#\mathrm{P}}$-complete.

Xi Chen's second talk will have details, again.

Functional clones

The computational complexity of $\#\mathrm{CSP}(\mathcal{F})$ depends on what functions can be defined from the functions in $\mathcal{F}$.

- Constraints $f(x, y)$, $g(x, y)$ together define the function $h(x, y) = f(x, y) \cdot g(x, y)$.

Functional clones

The computational complexity of $\#\mathrm{CSP}(\mathcal{F})$ depends on what functions can be defined from the functions in $\mathcal{F}$.

- Constraints $f(x, y)$, $g(x, y)$ together define the function $h(x, y) = f(x, y) \cdot g(x, y)$.
- If z appears in no other constraint, $f(x, z)$ corresponds to the function $h(x) = \sum_{z \in D} f(x, z)$.

Functional clones

The computational complexity of $\#\mathrm{CSP}(\mathcal{F})$ depends on what functions can be defined from the functions in $\mathcal{F}$.

- Constraints $f(x, y)$, $g(x, y)$ together define the function $h(x, y) = f(x, y) \cdot g(x, y)$.
- If z appears in no other constraint, $f(x, z)$ corresponds to the function $h(x) = \sum_{z \in D} f(x, z)$.
- The constraint $f(x, y)$ can be considered to define a function of any arity ≥ 2 (e.g., $h(x, y, z) = f(x, y)$).

Functional clones

The computational complexity of $\#\mathrm{CSP}(\mathcal{F})$ depends on what functions can be defined from the functions in $\mathcal{F}$.

- Constraints $f(x,y)$, $g(x,y)$ together define the function $h(x,y) = f(x,y) \cdot g(x,y)$.
- If z appears in no other constraint, $f(x,z)$ corresponds to the function $h(x) = \sum_{z \in D} f(x,z)$.
- The constraint $f(x,y)$ can be considered to define a function of any arity ≥ 2 (e.g., $h(x,y,z) = f(x,y)$).
- Limits.

Functional clones

The computational complexity of $\#\mathrm{CSP}(\mathcal{F})$ depends on what functions can be defined from the functions in $\mathcal{F}$.

- Constraints $f(x, y)$, $g(x, y)$ together define the function $h(x, y) = f(x, y) \cdot g(x, y)$.
- If z appears in no other constraint, $f(x, z)$ corresponds to the function $h(x) = \sum_{z \in D} f(x, z)$.
- The constraint $f(x, y)$ can be considered to define a function of any arity ≥ 2 (e.g., $h(x, y, z) = f(x, y)$).
- Limits.

A *clone* is a set of functions closed under these operations.

Log-supermodular functions

A function $\{0,1\}^k \to \mathbb{R}_{\geq 0}$ is *log-supermodular* (LSM) if

$$f(\mathbf{x} \vee \mathbf{y})\, f(\mathbf{x} \wedge \mathbf{y}) \geq f(\mathbf{x})\, f(\mathbf{y}) \text{ for all } \mathbf{x}, \mathbf{y} \in \{0,1\}^k .$$

All nullary and unary functions are trivially LSM.

Log-supermodular functions

A function $\{0,1\}^k \to \mathbb{R}_{\geq 0}$ is *log-supermodular* (LSM) if

$$f(\mathbf{x} \vee \mathbf{y})\, f(\mathbf{x} \wedge \mathbf{y}) \geq f(\mathbf{x})\, f(\mathbf{y}) \text{ for all } \mathbf{x}, \mathbf{y} \in \{0,1\}^k\,.$$

All nullary and unary functions are trivially LSM.

IMP is LSM, since, e.g.,

$$\begin{aligned} \mathrm{IMP}(01 \vee 10)\,\mathrm{IMP}(01 \wedge 10) = \mathrm{IMP}(11)\,\mathrm{IMP}(00) = 1 \\ \geq \mathrm{IMP}(01)\,\mathrm{IMP}(10) = 0\,. \end{aligned}$$

Log-supermodular functions

A function $\{0,1\}^k \to \mathbb{R}_{\geq 0}$ is *log-supermodular* (LSM) if

$$f(\mathbf{x} \vee \mathbf{y})\, f(\mathbf{x} \wedge \mathbf{y}) \geq f(\mathbf{x})\, f(\mathbf{y}) \text{ for all } \mathbf{x}, \mathbf{y} \in \{0,1\}^k\,.$$

All nullary and unary functions are trivially LSM.

IMP is LSM, since, e.g.,

$$\begin{aligned}\mathrm{IMP}(01 \vee 10)\,\mathrm{IMP}(01 \wedge 10) = \mathrm{IMP}(11)\,\mathrm{IMP}(00) = 1 \\ \geq \mathrm{IMP}(01)\,\mathrm{IMP}(10) = 0\,.\end{aligned}$$

Ferromagnetic Ising is also LSM, e.g.,

$$f(00)\, f(11) = \lambda^2 > f(01)\, f(10) = 1\,.$$

Properties of clones

Notation: $\langle \mathcal{F} \rangle$ is the smallest clone containing $\mathcal{F}$.

A clone is called *conservative* if it contains $\mathcal{U}$, the set of all unary functions $\{0, 1\} \to \mathbb{R}_{\geq 0}$.

Properties of clones

Notation: $\langle \mathcal{F} \rangle$ is the smallest clone containing $\mathcal{F}$.

A clone is called *conservative* if it contains $\mathcal{U}$, the set of all unary functions $\{0, 1\} \to \mathbb{R}_{\geq 0}$.

If $f \notin \langle \mathrm{NEQ}, \mathcal{U} \rangle$, then $\mathrm{IMP} \in \langle f, \mathcal{U} \rangle$.

If, also, f is not LSM, then $\langle f, \mathcal{U} \rangle$ is all Boolean functions.

Approximating weighted Boolean $\#\mathrm{CSP}$

Approximate theorem. Let $\mathcal{F}$ be a conservative weighted Boolean constraint language.

- If $\mathcal{F} \subseteq \langle \mathrm{NEQ}, \mathcal{U} \rangle$, then there is an FPRAS for $\#\mathrm{CSP}(\mathcal{F})$.
- Otherwise, $\#\mathrm{CSP}(\mathcal{F})$ is $\#\mathrm{BIS}$-hard.
- If, in addition, there is a non-LSM $f \in \mathcal{F}$, then $\#\mathrm{SAT} \equiv_{\mathrm{AP}} \#\mathrm{CSP}(\mathcal{F})$.

Approximating weighted Boolean $\#\mathrm{CSP}$ properly

Actual theorem. Let $\mathcal{F}$ be a conservative weighted Boolean constraint language.

- If $\mathcal{F} \subseteq \langle \mathrm{NEQ}, \mathcal{U} \rangle$ then, for every finite $\mathcal{G} \subset \mathcal{F}$, there is an FPRAS for $\#\mathrm{CSP}(\mathcal{G})$.
- Otherwise, there is a finite $\mathcal{G} \subset \mathcal{F}$ such that $\#\mathrm{CSP}(\mathcal{G})$ is $\#\mathrm{BIS}$-hard.
- If, in addition, there is a non-LSM $f \in \mathcal{F}$, then $\#\mathrm{SAT} \equiv_{\mathrm{AP}} \#\mathrm{CSP}(\mathcal{G})$ for some finite $\mathcal{G} \subset \mathcal{F}$.

Extending the domain

We can extend to arbitrary finite domains D.

Classification depends on definable binary functions and their behaviour on 2-element subdomains.

Extending the domain

We can extend to arbitrary finite domains D.

Classification depends on definable binary functions and their behaviour on 2-element subdomains.

A constraint langauge is *weakly log-supermodular* if, for all definable binary functions f and all $a, b \in D$,

$$\begin{aligned} & f(a,a)\, f(b,b) \geq f(a,b)\, f(b,a) \\ \text{or } & f(a,a)\, f(b,b) = 0\,. \end{aligned}$$

Approximating conservative weighted $\#\mathrm{CSP}$

Theorem. For any conservative weighted constraint language $\mathcal{F}$,

- If $\mathcal{F}$ is weakly log-modular, then $\#\mathrm{CSP}(\mathcal{G}) \in \mathrm{FP}$ for every finite $\mathcal{G} \subset \mathcal{F}$.
- Otherwise, if $\mathcal{F}$ is weakly log-supermodular then,
  - for every finite $\mathcal{G} \subset \mathcal{F}$, $\#\mathrm{CSP}(\mathcal{G}) \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathcal{G}')$ for some finite set $\mathcal{G}'$ of log-supermodular functions, and
  - $\#\mathrm{BIS} \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathcal{G})$ for some finite $\mathcal{G} \subset \mathcal{F}$.
- Otherwise, $\#\mathrm{SAT} \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathcal{G})$ for some finite $\mathcal{G} \subset \mathcal{F}$.

Why so conservative?

- Unary functions used in reductions.
- Structure of non-conservative clones not as well understood.
- Current area of research.