Real Number PCPs

Klaus Meer

Brandenburg University of Technology, Cottbus, Germany

Santaló's Summer School, Part 3, July, 2012

joint work with Martijn Baartse

(work supported by DFG, GZ:ME 1424/7-1)

Outline

1. Introduction

First two talks:

- Real number complexity with important complexity class $\mathrm{NP}_{\mathbb{R}}$
- Probabilistically checkable proofs PCPs in Turing model and PCP theorem to characterize Turing class NP

Today: PCPs in real number complexity

Example (Quadratic Polynomial Systems QPS)

Input: $n, m \in \mathbb{N}$, real polynomials in n variables

$p_1, \ldots, p_m \in \mathbb{R}[x_1, \ldots, x_n]$ of degree at most 2; each p_i depending on at most 3 variables;

Do the p_i's have a common real zero?

$\mathrm{NP}_{\mathbb{R}}$ verification for solvability of system

$$p_1(x) = 0 \ , \ \ldots, p_m(x) = 0$$

guesses solution $y^* \in \mathbb{R}^n$ and plugs it into all p_i's ; obviously all components of y^* have to be inspected

PCP question makes perfect sense:

Can we stabilize a verification proof, e.g., for QPS, and detect faults with high probability by inspecting constantly many (real) components only?

Real verifiers: particular probabilistic BSS machines running in polynomial time

Definition (Real verifiers)

$r, q : \mathbb{N} \mapsto \mathbb{N}$; a real verifier $V(r, q)$ is a polynomial time probabilistic BSS machine working as follows: V gets as input vectors $x \in \mathbb{R}^n$ (the problem instance) and $y \in \mathbb{R}^s$ (the verification proof)

i) V generates non-adaptively $r(n)$ random bits;

ii) from x and the $r(n)$ random bits V determines $q(n)$ many components of y;

iii) using x, the $r(n)$ random bits and the $q(n)$ components of y V deterministically produces its result (accept or reject)

Acceptance condition for a language $L \subseteq \mathbb{R}^*$:

A real verifier V accepts a language L iff

- for all $x \in L$ there is a guess y such that

$$\Pr_{\rho \in \{0,1\}^{r(n)}} \{V(x, y, \rho) = \text{'accept'}\} = 1$$

- for all $x \notin L$ and for all y

$$\Pr_{\rho \in \{0,1\}^{r(n)}} \{V(x, y, \rho) = \text{'reject'}\} \geq \frac{1}{2}$$

Important: probability aspects still refer to discrete probabilities.

Real verifiers as well produce random bits.

Definition

$\mathcal{R}, \mathcal{Q}$ function classes.

$L \in \mathrm{PCP}_{\mathbb{R}}(\mathcal{R}, \mathcal{Q})$: L is accepted by a real verifier $V(r, q)$ with

$r \in \mathcal{R}, q \in \mathcal{Q}$

Definition

$\mathcal{R}, \mathcal{Q}$ function classes.

$L \in \mathrm{PCP}_{\mathbb{R}}(\mathcal{R}, \mathcal{Q})$: L is accepted by a real verifier $V(r, q)$ with

$r \in \mathcal{R}, q \in \mathcal{Q}$

Example

$\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(0, poly)$

$\mathrm{NP}_{\mathbb{R}} \supseteq \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1))$

$\mathrm{PCP}_{\mathbb{R}}(O(\log n), 1)$: leads to questions about zeros of univariate polynomials given by straight line program

Goal: Characterizations of $\mathrm{NP}_{\mathbb{R}}$ via real number PCPs

Goal: Characterizations of $\mathrm{NP}_{\mathbb{R}}$ via real number PCPs

Recall the two proofs in Turing model by Arora et al. and Dinur:

- both need long transparent proofs for NP, i.e.,

$$\mathrm{NP} \subseteq PCP(poly(n), O(1))$$

- Arora et al. prove existence of short, almost transparent proofs:

$$\mathrm{NP} \subseteq PCP(O(\log n), poly \log n)$$

and use verifier composition to get full PCP theorem

- Dinur constructs gap reduction from 3-SAT to 3-SAT

Goal: Characterizations of $\mathrm{NP}_{\mathbb{R}}$ via real number PCPs

Recall the two proofs in Turing model by Arora et al. and Dinur:

- both need long transparent proofs for NP, i.e.,

$$\mathrm{NP} \subseteq PCP(poly(n), O(1))$$

- Arora et al. prove existence of short, almost transparent proofs:

$$\mathrm{NP} \subseteq PCP(O(\log n), poly \log n)$$

  and use verifier composition to get full PCP theorem
- Dinur constructs gap reduction from 3-SAT to 3-SAT

Here: more on first and third item

2. Long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$

Theorem

$\mathrm{NP}_{\mathbb{R}} \subseteq PCP_{\mathbb{R}}(O(f(n)), O(1))$, *where* f *is superpolynomial*

i.e., $\mathrm{NP}_{\mathbb{R}}$ has long transparent proofs.

Note: Most important wrt application in full PCP theorem is structure of the long transparent proofs (more than parameter values)

2. Long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$

Theorem

$\mathrm{NP}_{\mathbb{R}} \subseteq PCP_{\mathbb{R}}(O(f(n)), O(1))$, *where* f *is superpolynomial*

i.e., $\mathrm{NP}_{\mathbb{R}}$ has long transparent proofs.

Note: Most important wrt application in full PCP theorem is structure of the long transparent proofs (more than parameter values)

As by-product proof gives generalization of results by Rubinfeld & Sudan on self-testing and -correcting linear functions on finite subsets of $\mathbb{R}^n$: property testing

Problem setting; new difficulties

Sufficient: produce $(O(f(n)), O(1))$-verifier for $\mathrm{NP}_{\mathbb{R}}$-complete problem; we take QPS;

Problem setting; new difficulties

Sufficient: produce $(O(f(n)), O(1))$-verifier for $\mathrm{NP}_{\mathbb{R}}$-complete problem; we take QPS; what to use as more stable verification?

Problem setting; new difficulties

Sufficient: produce $(O(f(n)), O(1))$-verifier for $\mathrm{NP}_{\mathbb{R}}$-complete problem; we take QPS; what to use as more stable verification?

Consider QPS input $p_1, \ldots, p_m$, guess $y \in \mathbb{R}^n$; for $r \in \{0,1\}^m$ define

$$P(y, r) := \sum_{i=1}^{m} p_i(y) \cdot r_i$$

Observations:

- if $a \in \mathbb{R}^n$ is a zero, then $P(a, r) = 0 \ \forall \ r$;
- if $a \in \mathbb{R}^n$ is no zero, then $\Pr_r[P(a, r) > 0] \geq \frac{1}{2}$

Minor technical difference to classical setting: structure of P

Important: Separate dependence of P on guessed zero a from that on real coefficients of p_i's

Lemma

There are real linear functions A, B of n, n^2 variables, respectively, depending on a only, as well as linear functions

$$L_A, L_B : \{0,1\}^m \mapsto \mathbb{R}^n, \mathbb{R}^{n^2} \text{ and } C : \{0,1\}^m \mapsto \mathbb{R}$$

such that

$$P(a, r) = C(r) + A \circ L_A(r) + B \circ L_B(r)$$

Moreover, L_A, L_B and C depend on the coefficients of the p_i's.

More precisely: A, B depend on a guessed zero $a \in \mathbb{R}^n$ as follows:

$$A : \mathbb{R}^n \mapsto \mathbb{R}, A(w_1, \ldots, w_n) = \sum_{i=1}^{n} a_i \cdot w_i$$

$$B : \mathbb{R}^{n^2} \mapsto \mathbb{R}, B(w_{11}, \ldots, w_{nn}) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i \cdot a_j \cdot w_{ij}$$

More precisely: A, B depend on a guessed zero $a \in \mathbb{R}^n$ as follows:

$$A : \mathbb{R}^n \mapsto \mathbb{R}, A(w_1, \ldots, w_n) = \sum_{i=1}^{n} a_i \cdot w_i$$

$$B : \mathbb{R}^{n^2} \mapsto \mathbb{R}, B(w_{11}, \ldots, w_{nn}) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i \cdot a_j \cdot w_{ij}$$

Important: In order to evaluate

$$P(a, r) = C(r) + A \circ L_A(r) + B \circ L_B(r)$$

one needs to know only two values, one for A and one for B.

Example

$$p_1(a) = \pi + 1 \cdot a_1 + 2 \cdot a_2, \qquad p_2(a) = 3 \cdot a_1 a_3 - 1 \cdot a_4^2,$$

$$p_3(a) = 1 + \pi a_1 + 7 a_2 a_3$$

$$\begin{aligned} P(a,r) := \sum_{i=1}^{3} p_i \cdot r_i \;\; = \;\; & \pi r_1 + 1 \cdot r_3 + \\ & + a_1 \cdot (1 \cdot r_1 + \pi r_3) + a_2 \cdot 2 r_1 + \\ & + a_1 a_3 \cdot 3 r_2 + a_4^2 \cdot 1 \cdot r_2 + a_2 \cdot a_3 \cdot 7 r_3 \end{aligned}$$

results in

$$C(r) \;=\; (\pi, 0, 1) \cdot \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \quad , \quad L_A(r) \;=\; \begin{pmatrix} 1 & 0 & \pi \\ 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}$$

Similarly for L_B!

Idea to stabilize verification proofs (classical):

Instead of potential zero a guess function tables for A, B

Then probabilistically check that with high probability

- functions are linear (linearity test)
- functions do result from the same a (consistency test)
- this a is a zero (satisfiability test)

Problems with this idea

Main problems occur because of new domains to be considered:

Turing model:

domains where to guess function values obvious: $\mathbb{Z}_2^n$

each evaluation like $A(a+b)$, $a, b \in \mathbb{Z}_2^n$ remains in $\mathbb{Z}_2^n$;

BSS model:

domains for A, B not obvious: L_A, L_B give real values for each $r \in \mathbb{Z}_2^n$ (and different ones for each new input); evaluations like $A \circ L_A(r_1 + r_2)$ once more enlarge domain

uniform distribution over $\mathbb{Z}_2^n$

invariant under shifts;

uniform distribution on potential domains far from invariant, domains not even closed under shifting;

uniform distribution over $\mathbb{Z}_2^n$

invariant under shifts;

no constants beside $0, 1$

uniform distribution on potential domains far from invariant, domains not even closed under shifting;

arbitrary reals as constants, so linearity check also requires $A(\lambda \cdot x) = \lambda \cdot A(x)$. Again: what domain for λ's?

Sketch of what verifier will do:

i) Expect verification proof to contain function tables for A, B on appropriate domains; tables will have an double exponential size

ii) Check: both functions linear on their domains with high probability;

iii) Check: both functions arise from same $a \in \mathbb{R}^n$ with high probability;

iv) Check: a is a zero of input polynomials with high probability.

Appropriate domain for map A (I)

Outline of construction:

test domain: $\mathcal{X}_1$, proof should provide $A(x)$ for all $x \in \mathcal{X}_1 \oplus \mathcal{X}_1$

safe domain: $\mathcal{X}_0 \subset \mathcal{X}_1$ and Λ; if all tests succeed function A is almost surely linear on $\mathcal{X}_0$ with scalar factors from Λ

Appropriate domain for map A (I)

Outline of construction:

test domain: $\mathcal{X}_1$, proof should provide $A(x)$ for all $x \in \mathcal{X}_1 \oplus \mathcal{X}_1$

safe domain: $\mathcal{X}_0 \subset \mathcal{X}_1$ and Λ; if all tests succeed function A is almost surely linear on $\mathcal{X}_0$ with scalar factors from Λ

Goal for defining domains: obtain as far as possible shift-invariance

- for fixed $x \in \mathcal{X}_0$ it is $\Pr_{y \in \mathcal{X}_1}(x + y \in \mathcal{X}_1) \geq 1 - \epsilon$
- for fixed $\lambda \in \Lambda$ it is $\Pr_{y \in \mathcal{X}_1}(\lambda y \in \mathcal{X}_1) \geq 1 - \epsilon$

Appropriate domain for map A (I)

Outline of construction:

test domain: $\mathcal{X}_1$, proof should provide $A(x)$ for all $x \in \mathcal{X}_1 \oplus \mathcal{X}_1$

safe domain: $\mathcal{X}_0 \subset \mathcal{X}_1$ and Λ; if all tests succeed function A is almost surely linear on $\mathcal{X}_0$ with scalar factors from Λ

Goal for defining domains: obtain as far as possible shift-invariance

- for fixed $x \in \mathcal{X}_0$ it is $\Pr_{y \in \mathcal{X}_1}(x + y \in \mathcal{X}_1) \geq 1 - \epsilon$
- for fixed $\lambda \in \Lambda$ it is $\Pr_{y \in \mathcal{X}_1}(\lambda y \in \mathcal{X}_1) \geq 1 - \epsilon$

Clear: $\mathcal{X}_0$ must contain $L_A(\mathbb{Z}_2^m)$

Appropriate domain for map A (II)

$\Lambda := \{\lambda_1, \ldots, \lambda_K\} \subset \mathbb{R}$ multiset of entries in $L_A, K := O(n)$

(w.l.o.g. $m = O(n)$) and as safe domain

$\mathcal{X}_0 := \{\sum_{i=1}^{K} s_i \cdot \lambda_i \mid s_i \in \{0,1\}\}^n$.

Then

- $\mathbb{Z}_2^n \subseteq \mathcal{X}_0$ (i.e., a basis of $\mathbb{R}^n$) and
- $L_A(\mathbb{Z}_2^m) \subseteq \mathcal{X}_0$

Appropriate domain for map A (III)

The test domain then is

$$\mathcal{X}_1 := \left\{ \frac{1}{\alpha} \sum_{\beta \in M^+} s_\beta \cdot \beta \mid s_\beta \in \{0, \ldots, n^3\}, \alpha \in M \right\}^n,$$

where $M := \{\prod_{i=1}^K \lambda_i^{t_i} | t_i \in \{0, \ldots, n^2\}\}$, and

$M^+ := \{\prod_{i=1}^K \lambda_i^{t_i} | t_i \in \{0, \ldots, n^2 + 1\}\}$

Appropriate domain for map A (III)

The test domain then is

$$\mathcal{X}_1 := \left\{ \frac{1}{\alpha} \sum_{\beta \in M^+} s_\beta \cdot \beta \mid s_\beta \in \{0, \ldots, n^3\}, \alpha \in M \right\}^n,$$

where $M := \{\prod_{i=1}^K \lambda_i^{t_i} | t_i \in \{0, \ldots, n^2\}\}$, and

$M^+ := \{\prod_{i=1}^K \lambda_i^{t_i} | t_i \in \{0, \ldots, n^2 + 1\}\}$

Lemma

$\mathcal{X}_1$ is almost invariant under additive shifts with fixed $x \in \mathcal{X}_0$ and multiplicative shifts with fixed $\lambda \in \Lambda$. It has doubly exponential cardinality in n.

Testing linearity

Test Linearity

Choose $k \in \mathbb{N}$ large enough; perform k rounds of the following:

- uniformly and independently choose random x, y from $\mathcal{X}_1$ and random α, β from M;
- check if $A(x+y) = \frac{1}{\alpha}A(\alpha x) + \frac{1}{\beta}A(\beta y)$?

If all k checks were correct accept, otherwise reject.

Testing linearity

Test Linearity

Choose $k \in \mathbb{N}$ large enough; perform k rounds of the following:

- uniformly and independently choose random x, y from $\mathcal{X}_1$ and random α, β from M;
- check if $A(x+y) = \frac{1}{\alpha}A(\alpha x) + \frac{1}{\beta}A(\beta y)$?

If all k checks were correct accept, otherwise reject.

In k rounds linearity test requires to read $3k$ proof components.

Theorem

If A passes k rounds of linearity test for large enough k, then there is a function f_A such that

a) f_A is defined via

$$f_A(x) := \text{majority}_{y \in \mathcal{X}_1, \alpha \in M} \quad \frac{1}{\alpha} \cdot (A(\alpha(x+y)) - A(\alpha x))\}$$

and is linear on $\mathcal{X}_0$ wrt scalars from Λ;

Theorem

If A passes k rounds of linearity test for large enough k, then there is a function f_A such that

a) f_A *is defined via*

$$f_A(x) := \text{majority}_{y \in \mathcal{X}_1, \alpha \in M} \quad \frac{1}{\alpha} \cdot (A(\alpha(x+y)) - A(\alpha x))\}$$

and is linear on $\mathcal{X}_0$ wrt scalars from Λ;

b) f_A *is the unique linear function close to A, i.e., both differ in at most a given arbitarily small fraction $0 < \epsilon < \frac{1}{2}$ of points from $\mathcal{X}_1$;*

Theorem

If A passes k rounds of linearity test for large enough k, then there is a function f_A such that

a) f_A *is defined via*

$$f_A(x) := \text{majority}_{y \in \mathcal{X}_1, \alpha \in M} \quad \frac{1}{\alpha} \cdot (A(\alpha(x+y)) - A(\alpha x))\}$$

and is linear on $\mathcal{X}_0$ wrt scalars from Λ;

b) f_A *is the unique linear function close to A, i.e., both differ in at most a given arbitarily small fraction $0 < \epsilon < \frac{1}{2}$ of points from $\mathcal{X}_1$;*

c) A can be self-corrected, i.e., for any $x \in \mathcal{X}_0$ the correct value $f_A(x)$ can be computed with high probability from finitely many entries in the table for A.

Consistency, solvability

Do the above as well for function table B and f_B;

suppose both functions are linear with high probability, then:

- Check consistency, i.e., whether f_A, f_B result from a single assignment a; uses self-correction and easy test to check whether coefficient vector $\{b_{ij}\}$ of f_B satisfies $b_{ij} = a_i \cdot a_j$;
- check solvability: see beginning of talk

Theorem (Existence of long transparent proofs)

$\mathrm{NP}_{\mathbb{R}} \subseteq \mathrm{PCP}_{\mathbb{R}}(f(n), O(1))$, *where* $f = n^{O(n)}$.

The same holds for BSS model over complex numbers.

Proof.

Tests use constantly many values stored in doubly exponentially large tables.

All arguments the same over $\mathbb{C}$.

Theorem (Existence of long transparent proofs)

$\mathrm{NP}_{\mathbb{R}} \subseteq \mathrm{PCP}_{\mathbb{R}}(f(n), O(1))$, *where* $f = n^{O(n)}$.

The same holds for BSS model over complex numbers.

Proof.

Tests use constantly many values stored in doubly exponentially large tables.

All arguments the same over $\mathbb{C}$.

IMPORTANT: The theorem is applied in the full PCP theorem in a situation where n is constant; so size of $f(n)$ does not matter; more crucial: structure of verification proof!

3. The PCP theorem over $\mathbb{R}$

Goal: Adaption of Dinur's proof to show real PCP theorem

Changed viewpoint of QPS problem

3. The PCP theorem over $\mathbb{R}$

Goal: Adaption of Dinur's proof to show real PCP theorem

Changed viewpoint of QPS problem

QPS(m, k, q, s): system with m constraints, each consisting of k polynomial equations of degree 2; polynomials depend on at most q variable arrays having s components, i.e., ranging over $\mathbb{R}^s$

3. The PCP theorem over $\mathbb{R}$

Goal: Adaption of Dinur's proof to show real PCP theorem

Changed viewpoint of QPS problem

QPS(m, k, q, s): system with m constraints, each consisting of k polynomial equations of degree 2; polynomials depend on at most q variable arrays having s components, i.e., ranging over $\mathbb{R}^s$

Example

Way we considered QPS instances so far, i.e., m single equations of quadratic polynomials, each with at most 3 variables, can be changed easily to

Example (cntd.)

QPS$(m, 1, 3, 1)$-instances: each constraint is single equation, arrays are single variables (dimension 1), at most 3 arrays per constraint

Example (cntd.)

QPS$(m, 1, 3, 1)$-instances: each constraint is single equation, arrays are single variables (dimension 1), at most 3 arrays per constraint

or as

QPS$(\tilde{m}, 1, 2, 3)$-instances: arrays have dimension 3, original constraints depend on 1 such array, consistency between components expressed in further constraints depending on 2 arrays.

Example (cntd.)

QPS$(m, 1, 3, 1)$-instances: each constraint is single equation, arrays are single variables (dimension 1), at most 3 arrays per constraint

or as

QPS$(\tilde{m}, 1, 2, 3)$-instances: arrays have dimension 3, original constraints depend on 1 such array, consistency between components expressed in further constraints depending on 2 arrays.

Below we always try to work with $q = 2$ arrays per constraint in order to define constraint graphs between arrays: edge between two arrays represents constraint depending on those arrays.

Example

$p_1(x_1, x_2, x_3) = 0, p_2(x_2, x_3, x_4) = 0, p_3(x_4, x_5) = 0$

Example

$p_1(x_1, x_2, x_3) = 0, p_2(x_2, x_3, x_4) = 0, p_3(x_4, x_5) = 0$

Variable arrays of dimension 3:

$\chi^{(1)} = (z_1, z_2, z_3), \chi^{(2)} = (z_4, z_5, z_6), \chi^{(3)} = (z_7, z_8)$

Example

$p_1(x_1, x_2, x_3) = 0, p_2(x_2, x_3, x_4) = 0, p_3(x_4, x_5) = 0$

Variable arrays of dimension 3:

$\chi^{(1)} = (z_1, z_2, z_3), \chi^{(2)} = (z_4, z_5, z_6), \chi^{(3)} = (z_7, z_8)$

Old constraints depending on a single array:

$p_1(\chi^{(1)}) = 0, p_2(\chi^{(2)}) = 0, p_3(\chi^{(3)}) = 0$

Example

$p_1(x_1, x_2, x_3) = 0, p_2(x_2, x_3, x_4) = 0, p_3(x_4, x_5) = 0$

Variable arrays of dimension 3:

$\chi^{(1)} = (z_1, z_2, z_3), \chi^{(2)} = (z_4, z_5, z_6), \chi^{(3)} = (z_7, z_8)$

Old constraints depending on a single array:

$p_1(\chi^{(1)}) = 0, p_2(\chi^{(2)}) = 0, p_3(\chi^{(3)}) = 0$

Consistency constraints depending on two arrays:

$$\left.\begin{array}{l} z_2 - z_4 = 0 \\ z_3 - z_5 = 0 \end{array}\right\} \text{ consistency constraint for } (\chi^{(1)}, \chi^{(2)})$$

$z_6 - z_7 = 0$ consistency constraint for $(\chi^{(2)}, \chi^{(3)})$

Crucial: gap-reduction between QPS-instances, i.e., polynomial time transformation of QPS(m, k, q, s)-instance ϕ into QPS(m', k', q, s)-instance ψ such that:

- if ϕ is satisfiable so is ψ ;

Crucial: gap-reduction between QPS-instances, i.e., polynomial time transformation of QPS(m, k, q, s)-instance ϕ into QPS(m', k', q, s)-instance ψ such that:

- if ϕ is satisfiable so is ψ ;
- if ϕ is unsatisfiable, then at least a fraction of $\epsilon > 0$ constraints in ψ is violated by each assignment.

  Here ϵ is fixed constant

Proposition

If a gap-reduction exists, then $\mathrm{NP}_{\mathbb{R}} = PCP_{\mathbb{R}}(O(\log n), O(1))$.

Proposition

If a gap-reduction exists, then $\mathrm{NP}_{\mathbb{R}} = PCP_{\mathbb{R}}(O(\log n), O(1))$.

Proof.

Verifier for instance ϕ computes reduction result ψ and expects proof to provide satisfying assignment for ψ. Randomly choose a constraint in ψ and evaluate.

Proposition

If a gap-reduction exists, then $\mathrm{NP}_{\mathbb{R}} = PCP_{\mathbb{R}}(O(\log n), O(1))$.

Proof.

Verifier for instance ϕ computes reduction result ψ and expects proof to provide satisfying assignment for ψ. Randomly choose a constraint in ψ and evaluate. If ϕ is unsatisfiable the chosen constraint is violated with probability $\geq \epsilon$. Verifier reads *qs* proof components. Finitely many repetitions increase probability sufficiently. □

Thus the goal is to design a gap-reduction as follows:

1. Preprocessing puts QPS-instances into highly structured form; constraints depend on $q = 2$ many arrays of fixed dimension s and constraint graph is particular d-regular expander graph

Thus the goal is to design a gap-reduction as follows:

1. Preprocessing puts QPS-instances into highly structured form; constraints depend on $q = 2$ many arrays of fixed dimension s and constraint graph is particular d-regular expander graph
2. Amplification step increases unsatisfiability ratio of an instance by a constant factor > 1;

Thus the goal is to design a gap-reduction as follows:

1. Preprocessing puts QPS-instances into highly structured form; constraints depend on $q = 2$ many arrays of fixed dimension s and constraint graph is particular d-regular expander graph
2. Amplification step increases unsatisfiability ratio of an instance by a constant factor > 1;

Thus the goal is to design a gap-reduction as follows:

1. Preprocessing puts QPS-instances into highly structured form; constraints depend on $q = 2$ many arrays of fixed dimension s and constraint graph is particular d-regular expander graph
2. Amplification step increases unsatisfiability ratio of an instance by a constant factor > 1; disadvantage: parameters q, s get too large if applied several times, i.e., query complexity too large;

Thus the goal is to design a gap-reduction as follows:

1. Preprocessing puts QPS-instances into highly structured form; constraints depend on $q = 2$ many arrays of fixed dimension s and constraint graph is particular d-regular expander graph
2. Amplification step increases unsatisfiability ratio of an instance by a constant factor > 1; disadvantage: parameters q, s get too large if applied several times, i.e., query complexity too large;
3. Dimension reduction scales parameters q and s down again at price of small lost in unsatisfiability ratio.

Step 1: Preprocessing, technical, no major difficulties

Step 1: Preprocessing, technical, no major difficulties

Step 2: Amplification, similar to Dinur, minor changes necessary

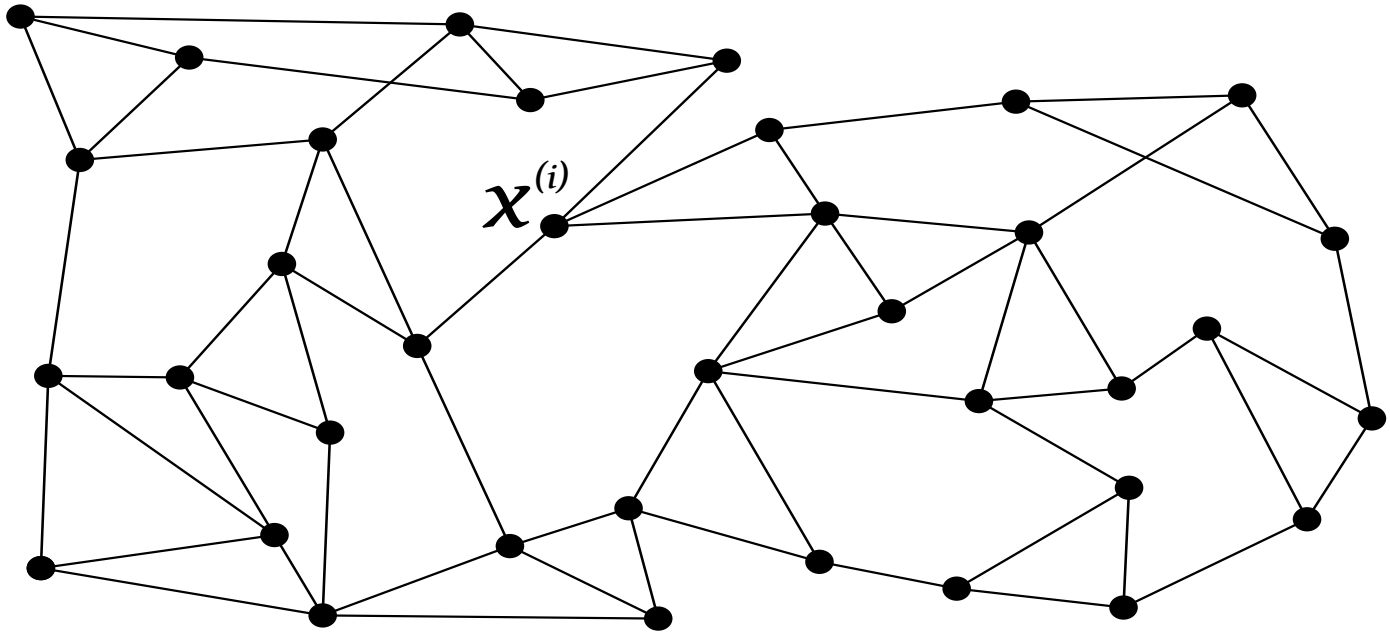
Basic idea is to transform QPS$(m, k, 2, s)$-instance to new one such that violated constraints in old instance occur in significantly more constraints of new instance and violate it;

Step 1: Preprocessing, technical, no major difficulties

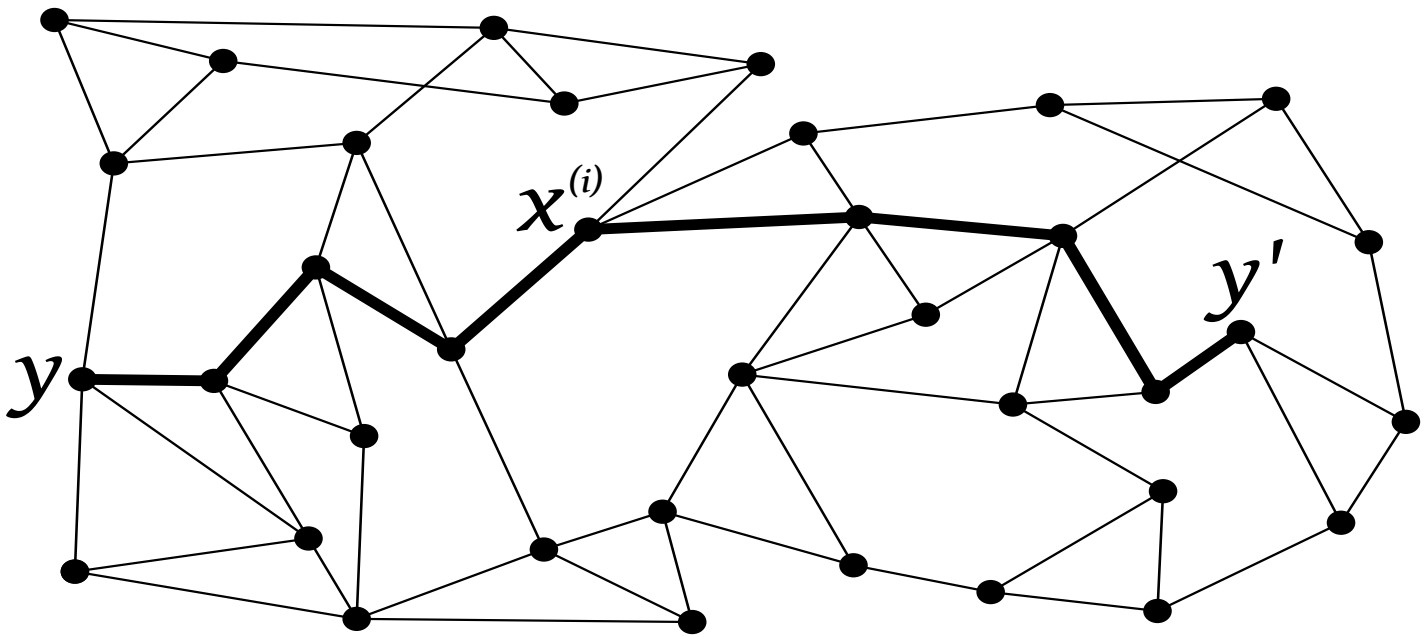
Step 2: Amplification, similar to Dinur, minor changes necessary

Basic idea is to transform QPS$(m, k, 2, s)$-instance to new one such that violated constraints in old instance occur in significantly more constraints of new instance and violate it;

this is achieved using random walks of constant length t on constraint graph; new constraints are made of paths in old graph; due to expander structure violated constraints = edges occur in many paths.

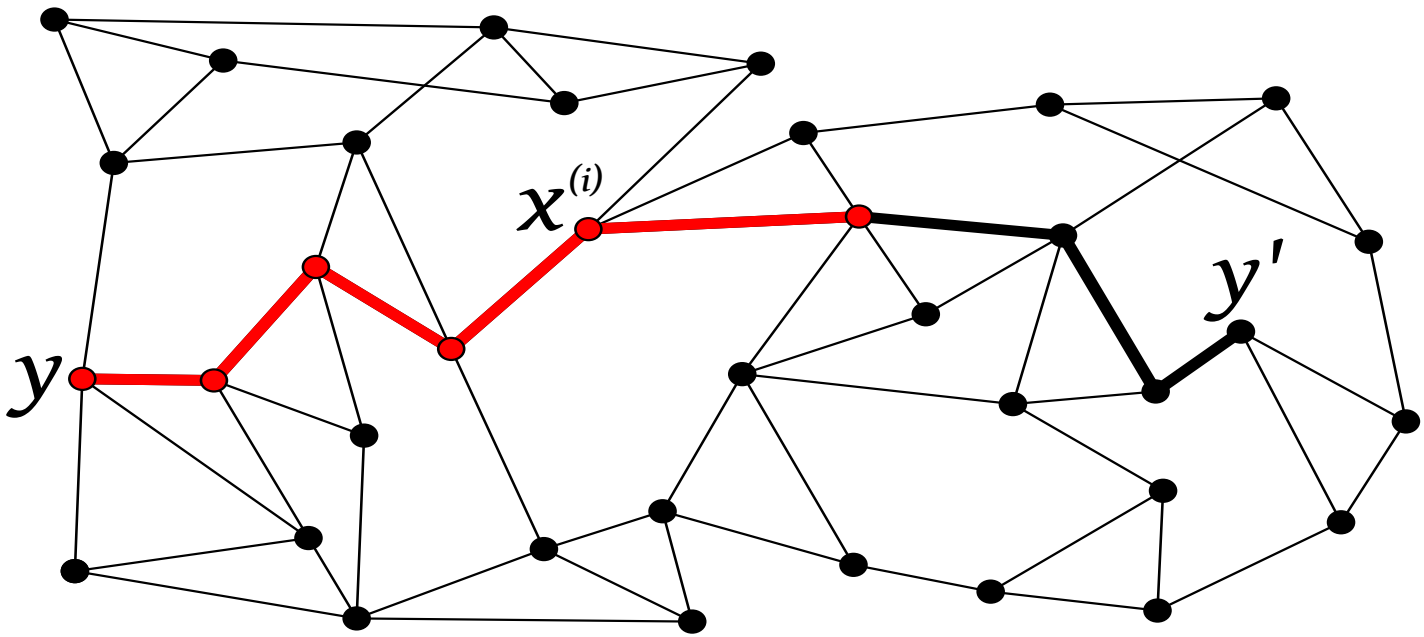


Old constraint graph and variable arrays $x^{(i)} \in \mathbb{R}^s$

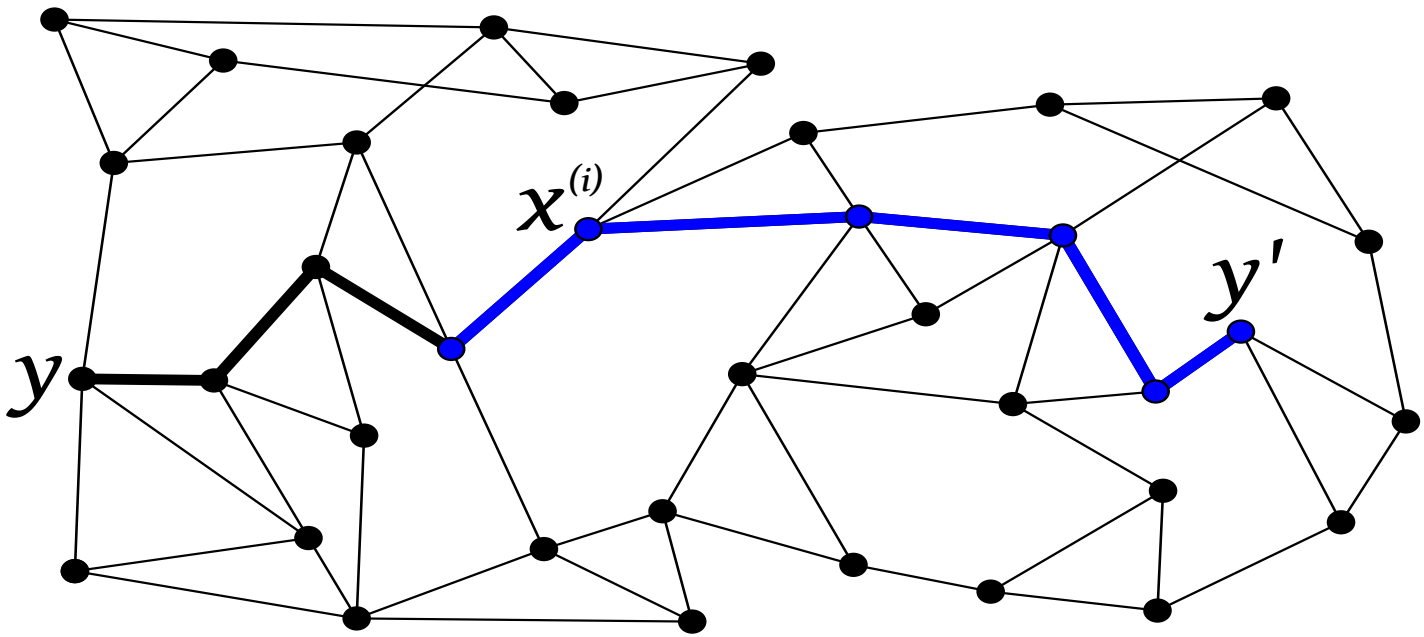


New constraint for each path of length $2t$, t constant;

new variable arrays $y \in \mathbb{R}^{s(t)}$ with $s(t) \leq d^{t+\sqrt{t}+1} \cdot s$



New arrays y claim values on old arrays in t-neighborhood



similarly for y'

Theorem

There exists a polynomial time algorithm that maps a (preprocessed) QPS$(m, k, 2, s)$ instance ψ to a QPS$(d^{2t}m, 2\sqrt{t}k + (2\sqrt{t}+1)s, 2, d^{t+\sqrt{t}+1}s)$-instance ψ^t and has the following properties:

- *If ψ is satisfiable, then ψ^t is satisfiable.*
- *If ψ is not satisfiable and UNSAT$(\psi) < \frac{1}{d\sqrt{t}}$, then UNSAT$(\psi^t) \geq \frac{\sqrt{t}}{3520d} \cdot$ UNSAT(ψ).*

We choose t, d such that $\frac{\sqrt{t}}{3520d} \geq 2$.

Note: if initially ψ has m constraints, then $UNSAT(\psi) \geq \frac{1}{m}$,

thus $O(\log m)$ rounds of amplification increase gap to a constant.

Why not done?

Note: if initially ψ has m constraints, then $UNSAT(\psi) \geq \frac{1}{m}$,

thus $O(\log m)$ rounds of amplification increase gap to a constant.

Why not done?

Problem: array size will not be constant any longer, and so will either query complexity

Step 3: Dimension reduction

use long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$ to reduce array dimension

while not decreasing gap too much;

Step 3: Dimension reduction

use long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$ to reduce array dimension

while not decreasing gap too much;

consider constraint C in instance ψ^t obtained after amplification;

C depends on two arrays $u, v \in \mathbb{R}^{s(t)}$; checking whether C is satisfied by concrete assignment for (u, v) can be expressed by algebraic circuit of size $poly(s(t))$, i.e., constant size

Step 3: Dimension reduction

use long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$ to reduce array dimension

while not decreasing gap too much;

consider constraint C in instance ψ^t obtained after amplification;

C depends on two arrays $u, v \in \mathbb{R}^{s(t)}$; checking whether C is satisfied by concrete assignment for (u, v) can be expressed by algebraic circuit of size $poly(s(t))$, i.e., constant size

use long transparent proofs to replace C by

$\mathrm{QPS}(\widehat{m}(t), K, Q, 1)$-instance, where K is constant and Q is the constant query complexity of a long transparent proof

Transformation works as follows:

new constraints express what verifier expects from long transparent proof to show that circuit for C accepts assignment (u, v);

Transformation works as follows:

new constraints express what verifier expects from long transparent proof to show that circuit for C accepts assignment (u, v);

this gives correct result with high probability and needs only Q components to be read instead of $poly(s(t))$ many.

Transformation works as follows:

new constraints express what verifier expects from long transparent proof to show that circuit for C accepts assignment (u, v);

this gives correct result with high probability and needs only Q components to be read instead of $poly(s(t))$ many.

Problem: Same variables in different constraints must get consistent assignment

Transformation works as follows:

new constraints express what verifier expects from long transparent proof to show that circuit for C accepts assignment (u, v);

this gives correct result with high probability and needs only Q components to be read instead of $poly(s(t))$ many.

Problem: Same variables in different constraints must get consistent assignment

Solution: Structure of long transparent proofs also guarantees this to be expressible via QPS$(\widehat{m}(t), K, Q, 1)$-instances

Transformation works as follows:

new constraints express what verifier expects from long transparent proof to show that circuit for C accepts assignment (u, v);

this gives correct result with high probability and needs only Q components to be read instead of $poly(s(t))$ many.

Problem: Same variables in different constraints must get consistent assignment

Solution: Structure of long transparent proofs also guarantees this to be expressible via QPS$(\widehat{m}(t), K, Q, 1)$-instances

Reduction in gap factor is harmless!

Theorem (Baartse & M.)

The PCP theorem holds for the real Blum-Shub-Smale model, i.e.,

$$\mathrm{NP}_{\mathbb{R}} = PCP_{\mathbb{R}}(O(\log n), O(1))$$

Theorem (Baartse & M.)

The PCP theorem holds for the real Blum-Shub-Smale model, i.e.,

$$\mathrm{NP}_{\mathbb{R}} = PCP_{\mathbb{R}}(O(\log n), O(1))$$

The same is true for the complex BSS model:

$$NP_{\mathbb{C}} = PCP_{\mathbb{C}}(O(\log n), O(1))$$

Final remarks

Theorem implies non-approximability result for following optimization problem:

Given a system of polynomial equations over $\mathbb{R}$, find the maximum number of equations that commonly can be satisfied.

Final remarks

Theorem implies non-approximability result for following optimization problem:

Given a system of polynomial equations over $\mathbb{R}$, find the maximum number of equations that commonly can be satisfied.

Existence of gap-reduction implies:

Unless $\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$ there is no polynomial time algorithm (in the system's size) which, given the system and an $\epsilon > 0$, approximates the above maximum within a factor $1 + \epsilon$.

Can the PCP theorem be proved along the lines of the first classical proof by Arora et al?

Can the PCP theorem be proved along the lines of the first classical proof by Arora et al?

Currently not clear; a weaker version can be shown using low-degree polynomials as coding objects

Theorem (M.)

$$\mathrm{NP}_{\mathbb{R}} = PCP_{\mathbb{R}}(O(\log n), poly \log n)$$

Can the PCP theorem be proved along the lines of the first classical proof by Arora et al?

Currently not clear; a weaker version can be shown using low-degree polynomials as coding objects

Theorem (M.)

$$\mathrm{NP}_{\mathbb{R}} = PCP_{\mathbb{R}}(O(\log n), poly \log n)$$

Classical proof constructs final verifier by composing long transparent proofs with low-degree proofs; needs better structure than the one sufficient to show above theorem; existence over $\mathbb{R}$ unclear.

Thanks for your audience and

thanks again to L.M. Pardo and P. Montaña!