P versus NP over Various Structures

Klaus Meer

Brandenburg University of Technology, Cottbus, Germany

26th ESSLLI Summer School 2014, Tübingen, August 2014

PART 5: Probabilistically Checkable Proofs

jointly with J.A. Makowsky

Outline for today

1. Introduction
2. Verifiers and real PCP classes
3. The classical PCP theorem
4. Long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$
5. The real PCP theorem

1. Introduction

Inspiring source of interesting questions in BSS framework:

which form do classical theorems (Turing model) take?

Examples:

- decidability of problems in $\mathrm{NP}_{\mathbb{R}}$

1. Introduction

Inspiring source of interesting questions in BSS framework:

which form do classical theorems (Turing model) take?

Examples:

- decidability of problems in $\mathrm{NP}_{\mathbb{R}}$
- transfer theorems between structures

1. Introduction

Inspiring source of interesting questions in BSS framework:

which form do classical theorems (Turing model) take?

Examples:

- decidability of problems in $\mathrm{NP}_{\mathbb{R}}$
- transfer theorems between structures
- complexity separations: $\mathrm{P}_{\mathbb{R}} \neq \mathrm{NC}_{\mathbb{R}}$ (Cucker)

1. Introduction

Inspiring source of interesting questions in BSS framework:

which form do classical theorems (Turing model) take?

Examples:

- decidability of problems in $\mathrm{NP}_{\mathbb{R}}$
- transfer theorems between structures
- complexity separations: $\mathrm{P}_{\mathbb{R}} \neq \mathrm{NC}_{\mathbb{R}}$ (Cucker)
- Toda's theorem (Basu & Zell)

1. Introduction

Inspiring source of interesting questions in BSS framework:

which form do classical theorems (Turing model) take?

Examples:

- decidability of problems in $\mathrm{NP}_{\mathbb{R}}$
- transfer theorems between structures
- complexity separations: $\mathrm{P}_{\mathbb{R}} \neq \mathrm{NC}_{\mathbb{R}}$ (Cucker)
- Toda's theorem (Basu & Zell)
- Ladner's theorem

1. Introduction

Inspiring source of interesting questions in BSS framework:

which form do classical theorems (Turing model) take?

Examples:

- decidability of problems in $\mathrm{NP}_{\mathbb{R}}$
- transfer theorems between structures
- complexity separations: $\mathrm{P}_{\mathbb{R}} \neq \mathrm{NC}_{\mathbb{R}}$ (Cucker)
- Toda's theorem (Basu & Zell)
- Ladner's theorem
- ...

Today: PCP theorem

Suppose you work at a university and have to grade a Master's thesis; you know the student is bad and you do not want to read the entire text in order to prove it.

However, you have to write a report which should not state there are faults if everything is correct.

Today: PCP theorem

Suppose you work at a university and have to grade a Master's thesis; you know the student is bad and you do not want to read the entire text in order to prove it.

However, you have to write a report which should not state there are faults if everything is correct.

Question: Can you read only a very small fraction (less than one page) of the thesis and nevertheless be almost sure to detect a fault?

Today: PCP theorem

Suppose you work at a university and have to grade a Master's thesis; you know the student is bad and you do not want to read the entire text in order to prove it.

However, you have to write a report which should not state there are faults if everything is correct.

Question: Can you read only a very small fraction (less than one page) of the thesis and nevertheless be almost sure to detect a fault?

Yes, if thesis is written according to the PCP-theorem

Today: PCP theorem

Suppose you work at a university and have to grade a Master's thesis; you know the student is bad and you do not want to read the entire text in order to prove it.

However, you have to write a report which should not state there are faults if everything is correct.

Question: Can you read only a very small fraction (less than one page) of the thesis and nevertheless be almost sure to detect a fault?

Yes, if thesis is written according to the PCP-theorem

(and for some students also without it ...)

Probabilistically Checkable Proofs give a new surprising characterization of class NP

One of the most important results in Theoretical Computer Science in last 20 years; important as well for questions about approximation algorithms

So far two different proofs exist:

1. Arora & Safra 1992 / 1998

   Arora & Lund & Motwani & Sudan & Szegedy 1992 / 1998

2. Dinur 2005

Example (NP-verification for NP-complete problem 3-SAT)

Given $\phi(x_1, \ldots, x_n) = C_1 \wedge \ldots \wedge C_m$ formula in Conjunctive Normal Form, each C_i with at most 3 literals, is there a satisfying assignment $y \in \{0,1\}^n$ for ϕ? $(C_i = x_1 \vee \bar{x}_2 \vee x_4)$

Example (NP-verification for NP-complete problem 3-SAT)

Given $\phi(x_1, \ldots, x_n) = C_1 \wedge \ldots \wedge C_m$ formula in Conjunctive Normal Form, each C_i with at most 3 literals, is there a satisfying assignment $y \in \{0,1\}^n$ for ϕ? $(C_i = x_1 \vee \bar{x}_2 \vee x_4)$

NP-verification algorithm requires

- polynomial running time in $size(\phi)$ on input (ϕ, y)
- for each satisfiable ϕ there is a guess y^* such that algorithm accepts (ϕ, y^*)
- for all unsatisfiable ϕ and all guesses y algorithm rejects

Example (NP-verification for NP-complete problem 3-SAT)

Given $\phi(x_1, \ldots, x_n) = C_1 \wedge \ldots \wedge C_m$ formula in Conjunctive Normal Form, each C_i with at most 3 literals, is there a satisfying assignment $y \in \{0,1\}^n$ for ϕ? $(C_i = x_1 \vee \bar{x}_2 \vee x_4)$

NP-verification algorithm requires

- polynomial running time in $size(\phi)$ on input (ϕ, y)
- for each satisfiable ϕ there is a guess y^* such that algorithm accepts (ϕ, y^*)
- for all unsatisfiable ϕ and all guesses y algorithm rejects

Easy: Guess assignment y^*, check by plugging into ϕ

Central for above verification: the algorithm has to inspect all components of the potential satisfying assignment.

Can we design other verification algorithms that have to inspect less many parts of a potential proof, may be paying something for it?

Central for above verification: the algorithm has to inspect all components of the potential satisfying assignment.

Can we design other verification algorithms that have to inspect less many parts of a potential proof, may be paying something for it?

Surprising result: Less many above turns out to be constantly many only; we pay by including randomization, i.e., false proofs might be accepted with very small probability.

Proofs must code assignments completely differently!

Example

Suppose as part of a verification proof you want to check whether two vectors $a, b \in \{0,1\}^n$ are the same.

You may ask an oracle information about a, b, but separately (so oracle question 'is $a = b$' forbidden; scenario at the moment sounds a bit strange, but reoccurs later on)

Example

Suppose as part of a verification proof you want to check whether two vectors $a, b \in \{0,1\}^n$ are the same.

You may ask an oracle information about a, b, but separately (so oracle question 'is $a = b$' forbidden; scenario at the moment sounds a bit strange, but reoccurs later on)

1. Easy way: Write down a, b and compare componentwise; each component has to be read

Example

Suppose as part of a verification proof you want to check whether two vectors $a, b \in \{0,1\}^n$ are the same.

You may ask an oracle information about a, b, but separately (so oracle question 'is $a = b$' forbidden; scenario at the moment sounds a bit strange, but reoccurs later on)

1. Easy way: Write down a, b and compare componentwise; each component has to be read

2. A bit more tricky: Expect proof to contain all results $a^t \cdot r$ and $b^t \cdot r$; pick randomly an $r \in \{0,1\}^n$ and read only the two corresponding results in your proof.

Example (cntd.)

With probability $\frac{1}{2}$ test detects if $a \neq b$.

This probability can be made arbitrarily small by constantly many repetitions, i.e., still reading constantly many components only.

Example (cntd.)

With probability $\frac{1}{2}$ test detects if $a \neq b$.

This probability can be made arbitrarily small by constantly many repetitions, i.e., still reading constantly many components only.

Disadvantage:

Example (cntd.)

With probability $\frac{1}{2}$ test detects if $a \neq b$.

This probability can be made arbitrarily small by constantly many repetitions, i.e., still reading constantly many components only.

Disadvantage: Table of values the proof expects is exponentially large

Example (cntd.)

With probability $\frac{1}{2}$ test detects if $a \neq b$.

This probability can be made arbitrarily small by constantly many repetitions, i.e., still reading constantly many components only.

Disadvantage: Table of values the proof expects is exponentially large

What about $\mathrm{NP}_{\mathbb{R}}$-verification algorithms in the real BSS model?

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

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

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

Previous question makes perfect sense:

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

Formalization using real number verifiers, i.e., particular probabilistic BSS machines running in polynomial time

2. Verifiers and real PCP classes

Definition (Real verifiers)

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

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

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

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

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

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

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

$\mathrm{PCP}_{\mathbb{R}}(poly, 0) = \mathrm{P}_{\mathbb{R}}$ based on co-$\mathrm{RP}_{\mathbb{R}} = \mathrm{P}_{\mathbb{R}}$ (Cucker et al)

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

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

$\mathrm{PCP}_{\mathbb{R}}(poly, 0) = \mathrm{P}_{\mathbb{R}}$ based on co-$\mathrm{RP}_{\mathbb{R}} = \mathrm{P}_{\mathbb{R}}$ (Cucker et al)

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

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

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

$\mathrm{PCP}_{\mathbb{R}}(poly, 0) = \mathrm{P}_{\mathbb{R}}$ based on co-$\mathrm{RP}_{\mathbb{R}} = \mathrm{P}_{\mathbb{R}}$ (Cucker et al)

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

$\mathrm{PCP}_{\mathbb{R}}(O(\log n), 1)$: leads to questions about real zeros of systems of univariate polynomials given by algebraic circuits; already difficult for a single polynomial given by straight line program.

Example on blackboard

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

more precisely proof of

Theorem (Baartse & M. 2013)

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

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

Ultimate goal: Characterizations of $\mathrm{NP}_{\mathbb{R}}$ via real number PCPs, more precisely proof of

Theorem (Baartse & M. 2013)

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:

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

3. The classical PCP theorem

Theorem (The classical PCP theorem, Arora et al. 1992)

$$NP = PCP(O(\log n), O(1))$$

PCP classes defined similarly as above, Turing model of computation; inclusion $NP \subseteq PCP(O(\log n), O(1))$ is the hard part to prove

3. The classical PCP theorem

Theorem (The classical PCP theorem, Arora et al. 1992)

$$NP = PCP(O(\log n), O(1))$$

PCP classes defined similarly as above, Turing model of computation; inclusion $NP \subseteq PCP(O(\log n), O(1))$ is the hard part to prove

Note: $PCP(O(\log n), O(1))$ closed under polynomial time reductions; thus sufficient to show that a fixed NP-complete problem has an $(O(\log n), O(1))$-verifier; consider 3-SAT;

Same argument for $\mathrm{NP}_{\mathbb{R}}$-complete problems and BSS model

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

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

Existence of long transparent proofs for 3-SAT:

Theorem

$$3\text{-}SAT \in PCP(O(n^3), O(1))$$

Existence of long transparent proofs for 3-SAT:

Theorem

$$3\text{-}SAT \in PCP(O(n^3), O(1))$$

A satisfying assignment a of a given formula ϕ is coded as follows:

i) arithmetization of formula together with randomization leads to polynomial P_r of degree 3 such that

- if $a \in \{0,1\}^n$ satisfies ϕ, then $P_r(a) = 0$
- if a is not satisfying, then $P_r(a) = 0$ only with small probability w.r.t. r

ii) P_r can be decomposed as

$$P_r(a) = f_0(r) + A_a(f_1(r)) + B_a(f_2(r)) + C_a(f_3(r))$$

where $A_a : \{0,1\}^n \mapsto \{0,1\}$, $B_a : \{0,1\}^{n^2} \mapsto \{0,1\}$, $C_a : \{0,1\}^{n^3} \mapsto \{0,1\}$ are linear functions canonically attached to a and

ii) P_r can be decomposed as

$$P_r(a) = f_0(r) + A_a(f_1(r)) + B_a(f_2(r)) + C_a(f_3(r))$$

where $A_a : \{0,1\}^n \mapsto \{0,1\}$, $B_a : \{0,1\}^{n^2} \mapsto \{0,1\}$, $C_a : \{0,1\}^{n^3} \mapsto \{0,1\}$ are linear functions canonically attached to a and the $f_i(r)$ can be computed efficiently from r.

ii) P_r can be decomposed as

$$P_r(a) = f_0(r) + A_a(f_1(r)) + B_a(f_2(r)) + C_a(f_3(r))$$

where $A_a : \{0,1\}^n \mapsto \{0,1\}$, $B_a : \{0,1\}^{n^2} \mapsto \{0,1\}$, $C_a : \{0,1\}^{n^3} \mapsto \{0,1\}$ are linear functions canonically attached to a and the $f_i(r)$ can be computed efficiently from r.

Thus evaluating $P_r(a)$ for given r requires only to look up one function value of A_a, B_a, and C_a

The proof the verifier expects thus contains the function values of A_a, B_a, and C_a; it has exponential size.

Introduced new difficulties to be circumvented:

1. Does a table of function values correspond to an almost linear function:

(self-)testing linear functions

Introduced new difficulties to be circumvented:

1. Does a table of function values correspond to an almost linear function:

   (self-)testing linear functions

2. If yes, how can we compute the correct values if table contains small errors:

   (self-)correcting linear functions

Introduced new difficulties to be circumvented:

1. Does a table of function values correspond to an almost linear function:

   (self-)testing linear functions

2. If yes, how can we compute the correct values if table contains small errors:

   (self-)correcting linear functions

3. Even if the tables for A_a, B_a, and C_a represent linear functions are they coming from the same a:

   consistency

Main idea of Dinur's proof: construction of gap creating reduction for NP-complete Constraint Satisfiability Problem; problem defined (and needed) over arbitrary finite alphabets;

Main idea of Dinur's proof: construction of gap creating reduction for NP-complete Constraint Satisfiability Problem; problem defined (and needed) over arbitrary finite alphabets;

instances consist of constraints, important quantity for unsatisfiable instances is fraction of violated constraints;

Main idea of Dinur's proof: construction of gap creating reduction for NP-complete Constraint Satisfiability Problem; problem defined (and needed) over arbitrary finite alphabets;

instances consist of constraints, important quantity for unsatisfiable instances is fraction of violated constraints;

Starting from a small gap depending on $\frac{1}{\text{input size}}$ a tricky gap amplification construction is invoked $O(\log n)$ times to increase the gap to be constant.

Amplification increases size of underlying finite alphabets; second step performs alphabet reduction by using long transparent proofs.

Main problems to face in real number model:

- linearity over $\mathbb{R}$ more difficult:

  domains of candidate functions not any longer well structured like $GF(2^n)$; even unclear which domains to choose

  invariance properties of uniform distribution lost in such domains

Main problems to face in real number model:

- linearity over $\mathbb{R}$ more difficult:

  domains of candidate functions not any longer well structured like $GF(2^n)$; even unclear which domains to choose

  invariance properties of uniform distribution lost in such domains

- CSP problems over finite alphabets not appropriate since underlying alphabet is always $\mathbb{R}$
- find an appropriate problem that could replace CSP
- ... and for everything a lot of details to be checked ...

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

Theorem (M. 2005)

$\mathrm{NP}_{\mathbb{R}} \subseteq PCP_{\mathbb{R}}(O(f(n)), O(1))$, *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 magnitude of f)

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

Theorem (M. 2005)

$\mathrm{NP}_{\mathbb{R}} \subseteq PCP_{\mathbb{R}}(O(f(n)), O(1))$, *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 magnitude of f)

Proof complements 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 not a 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

$$L_A, L_B : \{0,1\}^m \mapsto \mathbb{R}^n, \mathbb{R}^{n^2} \text{ and } L_0 : \{0,1\}^m \mapsto \mathbb{R}$$

such that

$$P(a,r) = L_0(r) + A \circ L_A(r) + B \circ L_B(r)$$

Moreover, L_A, L_B and L_0 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

$$P(a, r) = L_0(r) + A \circ L_A(r) + B \circ L_B(r)$$

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

Example

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

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

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

results in

$$L_0(r) \;=\; (\pi, 0, 1) \cdot \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \quad , \quad L_A(r) \;=\; \begin{pmatrix} 1 & 0 & \pi \\ 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}$$

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;

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

no constants beside $0, 1$

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 a doubly 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

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

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

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

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

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

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 $\mathrm{PCP}_{\mathbb{R}}$ theorem in a situation where n is constant; so size of $f(n)$ does not matter; more crucial: structure of verification proof!

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

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

Changed viewpoint of QPS problem

5. 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 $\mathbb{R}^s$

5. 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 $UNSAT(\psi) \geq \epsilon$, i.e., at least a fraction of $\epsilon > 0$ constraints in ψ are 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. $\square$

Digression / naive question: why not designing a gap-reduction by collecting all constraints in a single one?

Certainly polynomial time computable: squaring and summing of all polynomials does not change solvability;

new instance ψ has a single constraint, so either $UNSAT(\psi) = 0$ or $UNSAT(\psi) = 1$; in the latter case the gap is constant.

Digression / naive question: why not designing a gap-reduction by collecting all constraints in a single one?

Certainly polynomial time computable: squaring and summing of all polynomials does not change solvability;

new instance ψ has a single constraint, so either $UNSAT(\psi) = 0$ or $UNSAT(\psi) = 1$; in the latter case the gap is constant.

But: constraint in new system depends on all variables, i.e., verification needs to read all components of an assignment.

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

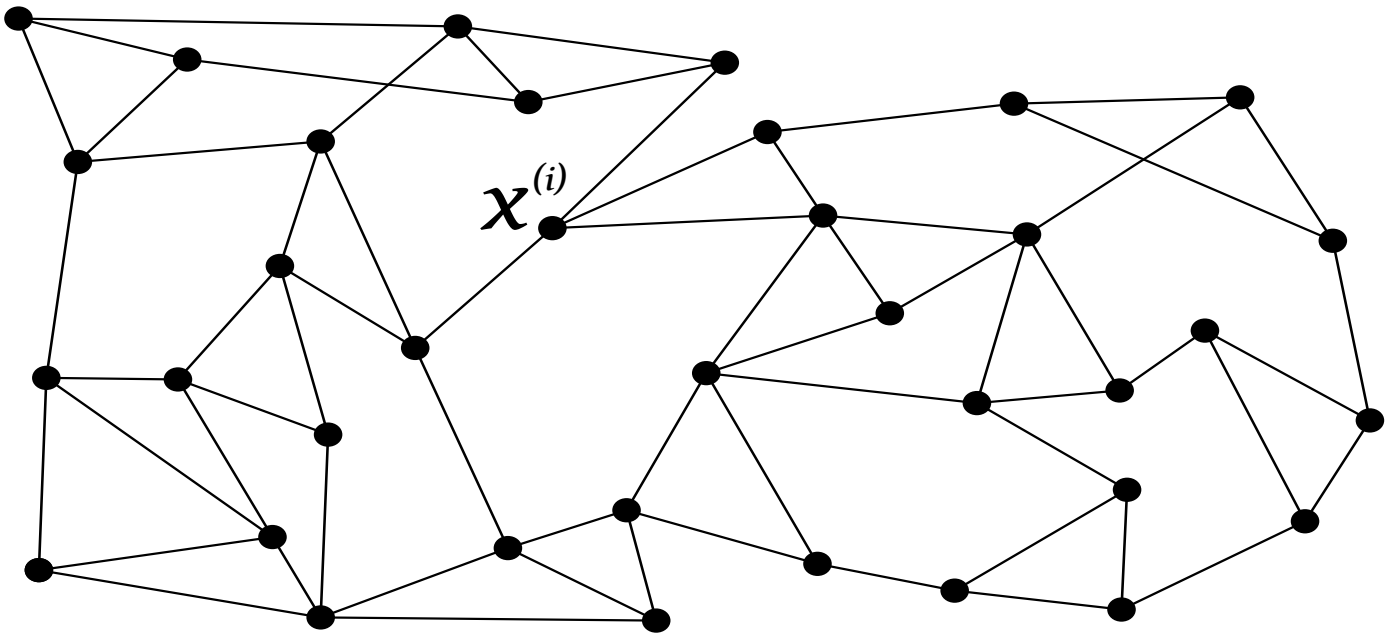
Basic idea is to transform QPS$(m, k, 2, s)$-instance to new one such that violated constraints in old instance occur in significantly more constraints of new instance and violate it;

Step 1: Preprocessing, technical, no major difficulties

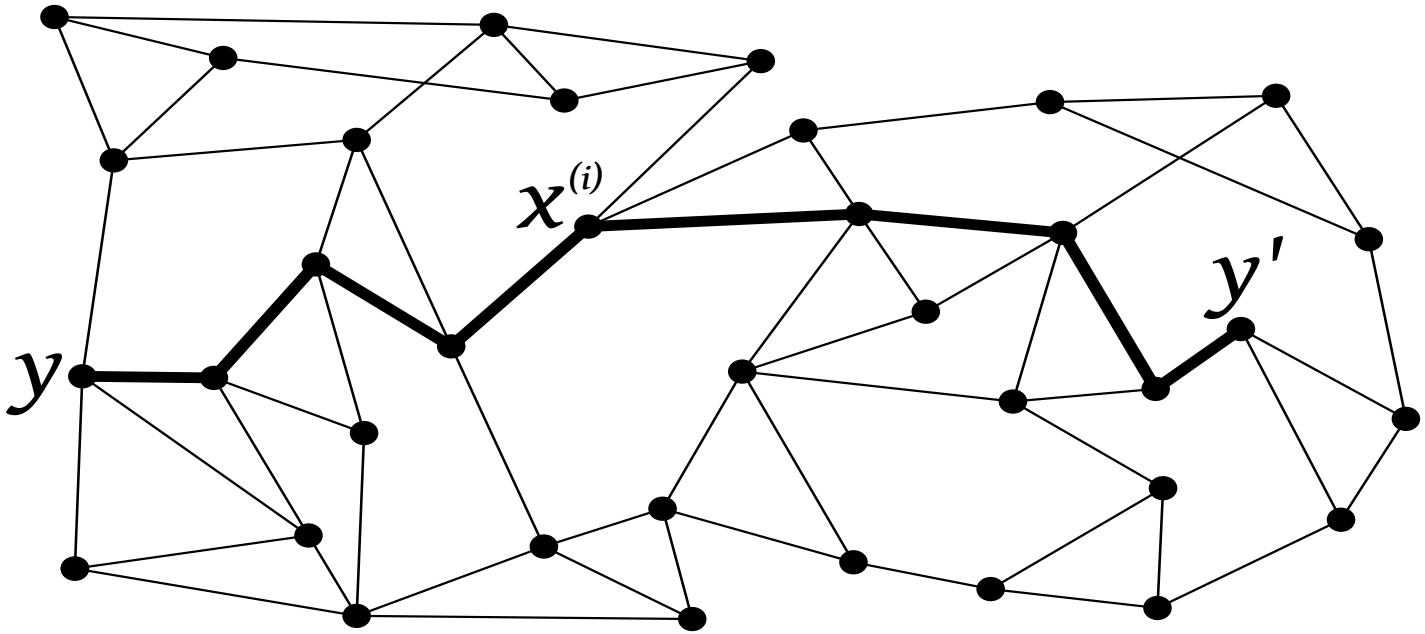
Step 2: Amplification, similar to Dinur, minor changes necessary

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 walks in old graph; due to expander structure violated constraints = edges occur in many walks.

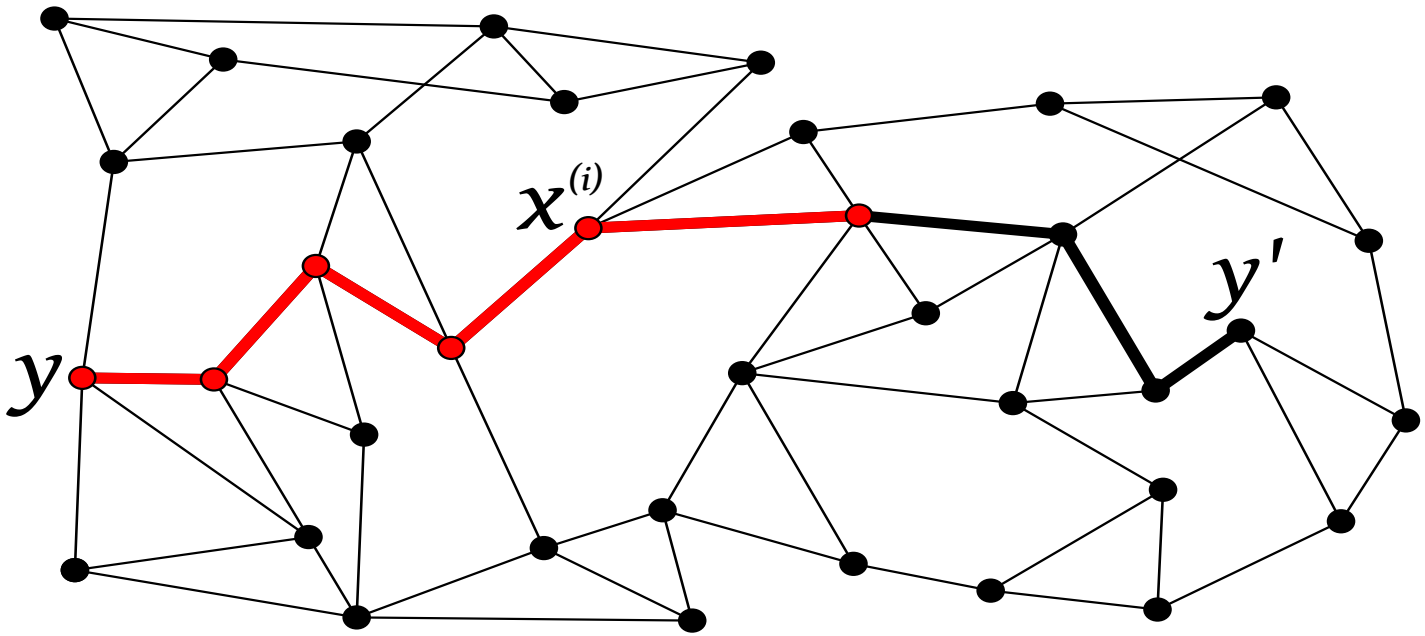


Old constraint graph and variable arrays $x^{(i)} \in \mathbb{R}^s$

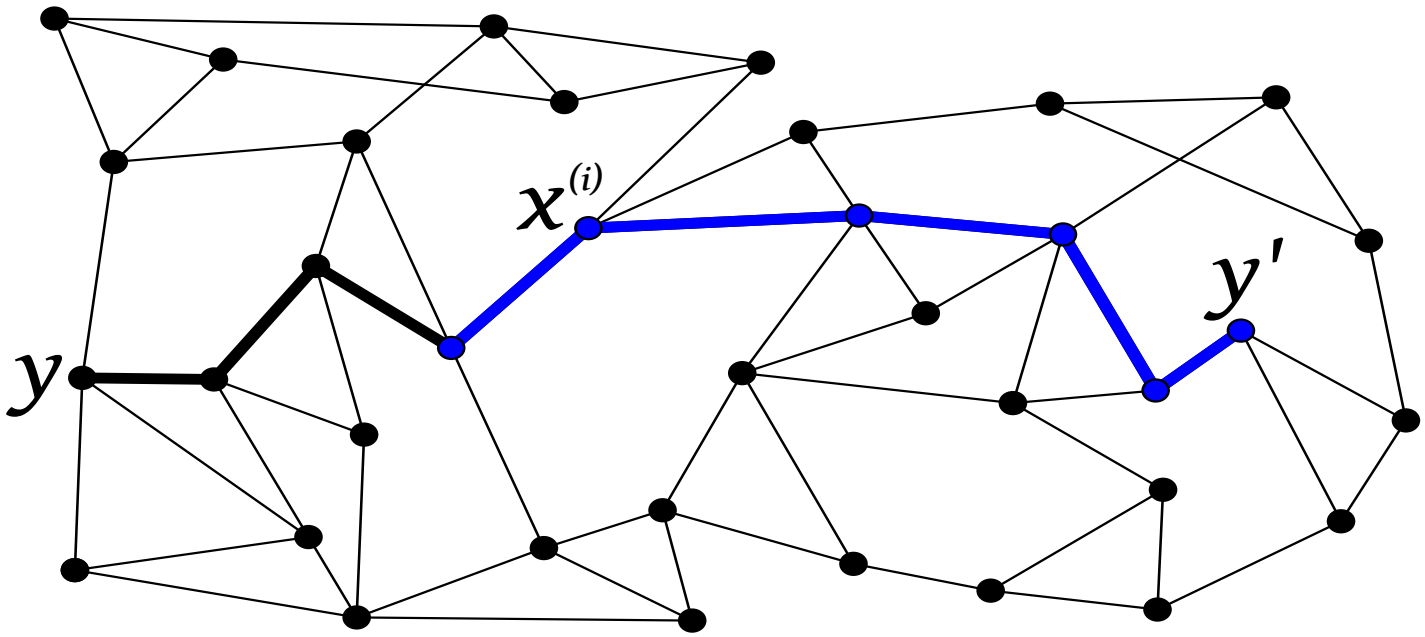


New constraint for each walk of length $2t$, t constant;

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



New array y claims values on old arrays in t-neighborhood



similarly for y' and its t-neighborhood

The new constraint in ψ^t attached to this walk depends on variables y, y' attached to the start- and endpoint of the walk;

it requires for all edges (i, j) in the walk such that $x^{(i)}$ gets assignment from y and $x^{(j)}$ gets assignment from y' that the old constraint given by edge (i, j) is satisfied by the new assignments.

The new constraint in ψ^t attached to this walk depends on variables y, y' attached to the start- and endpoint of the walk;

it requires for all edges (i, j) in the walk such that $x^{(i)}$ gets assignment from y and $x^{(j)}$ gets assignment from y' that the old constraint given by edge (i, j) is satisfied by the new assignments.

Amplification of unsatisfiability gap results from fact that edges of violated constraints occur in many walks!

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 $UNSAT(\psi^t) \geq \frac{\sqrt{t}}{3520d} \cdot UNSAT(\psi)$.*

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 neither will 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 $\mathrm{QPS}(\widehat{m}(t), K, Q, 1)$-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 $\mathrm{QPS}(\widehat{m}(t), K, Q, 1)$-instances

Reduction in gap factor is harmless!

Theorem (Baartse & M. 2013)

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. 2013)

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:

Theorem (Baartse & M. 2013)

Unless $P_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$ 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?

A weaker version can be shown using algebraic low-degree polynomials as coding objects

Theorem (M. 2012)

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

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

A weaker version can be shown using algebraic low-degree polynomials as coding objects

Theorem (M. 2012)

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

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.

Ongoing work (together with M. Baartse):

design a better structure of these tests by using particular trigonometric low-degree polynomials; they combine finite fields as domain and the reals as range, thus giving back some of the structural advantages of the classical proof.

Ongoing work (together with M. Baartse):

design a better structure of these tests by using particular trigonometric low-degree polynomials; they combine finite fields as domain and the reals as range, thus giving back some of the structural advantages of the classical proof.

But: as usual, new problems enter, for example univariate restrictions of such trigonometric polynomials have different degree properties.

Ongoing work (together with M. Baartse):

design a better structure of these tests by using particular trigonometric low-degree polynomials; they combine finite fields as domain and the reals as range, thus giving back some of the structural advantages of the classical proof.

But: as usual, new problems enter, for example univariate restrictions of such trigonometric polynomials have different degree properties.

Current status: Low-degree test can be done with nice structure.

Ongoing work (together with M. Baartse):

design a better structure of these tests by using particular trigonometric low-degree polynomials; they combine finite fields as domain and the reals as range, thus giving back some of the structural advantages of the classical proof.

But: as usual, new problems enter, for example univariate restrictions of such trigonometric polynomials have different degree properties.

Current status: Low-degree test can be done with nice structure.

Interesting in its own right: Property testing over real and complex numbers

Thanks for your audience!

References real number PCP theorem

1. M. Baartse, K. Meer: The PCP theorem for NP over the reals. In: Foundations of Computational Mathematics. Springer.

   Extended abstract: Proc. STACS 2013, LIPIcs series Vol. 20, Schloss Dagstuhl, 104–115, 2013

2. M. Baartse, K. Meer: Topics in Real and Complex Number Complexity Theory. In: Recent Advances in Real Complexity and Computation, Contemporary Mathematics Vol. 604, American Mathematical Society, 1–53, 2013.

3. K. Meer: Transparent long proofs: A first PCP theorem for $\mathrm{NP}_{\mathbb{R}}$. Foundations of Computational Mathematics, Springer, vol. 5 nr.3, 231-255, 2005.

References classical PCP theorem

1. Arora, Lund, Motwani, Sudan, Szegedy: Proof verification and intractability of approximation problems. JACM 45: 501–555, 1998
2. Arora, Safra: Probabilistically checkable proofs: a new characterization of NP. JACM 45: 70–122, 1998
3. Ausiello et al: Approximation Algorihtms. Springer, 1998
4. Hougardy, Prömel, Steger: PCPs and their consequences for approximation algorithms. DAM 136, 175–223, 1994.
5. Radhakrishnan & Sudan, Bulletin AMS 44 (1), 19-61, 2007
6. Webpage of M. Sudan contains a lot of survey papers