The PCP theorem for NP over the reals*

Martijn Baartse and Klaus Meer[1]

1 **Computer Science Institute, BTU Cottbus**
Platz der Deutschen Einheit 1
D-03046 Cottbus, Germany
martijnbaartse@msn.com,meer@informatik.tu-cottbus.de

Abstract

In this paper we show that the PCP theorem holds as well in the real number computational model introduced by Blum, Shub, and Smale. More precisely, the real number counterpart $\mathrm{NP}_{\mathbb{R}}$ of the classical Turing model class NP can be characterized as $\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1))$. Our proof structurally follows the one by Dinur for classical NP. However, a lot of minor and major changes are necessary due to the real numbers as underlying computational structure. The analogue result holds for the complex numbers and $\mathrm{NP}_{\mathbb{C}}$.

1998 ACM Subject Classification F.2.2 Complexity of Proof Procedures

Keywords and phrases PCP, real number computation, systems of polynomials

Digital Object Identifier 10.4230/LIPIcs.STACS.2013.104

1 Introduction

One of the major achievements in theoretical computer science within the last two decades certainly was the PCP theorem. It gave a new characterization of the complexity class NP via so called probabilistically checkable proofs and had tremendous impact on the field of approximation algorithms in combinatorial optimization.

Starting point for the present paper is the real number model of computation and its related complexity theory as introduced by Blum, Shub, and Smale, henceforth called BSS-model for short, see [6, 5]. The model focusses on algebraic aspects of computation over arbitrary structures, and here in particular the real numbers. As with the Turing model the major open question in this real number framework is whether the real complexity classes $\mathrm{P}_{\mathbb{R}}$ and $\mathrm{NP}_{\mathbb{R}}$ of problems decidable and verifiable, respectively, in polynomial time are different.

The definition of probabilistically checkable proofs makes sense as well in the real number model. Given the importance of the classical PCP theorem it is only natural to ask whether the corresponding characterization holds as well in the BSS framework for $\mathrm{NP}_{\mathbb{R}}$. The goal of this paper is to prove the PCP theorem in this setting.

1.1 Previous work and outline of proof

For the classical PCP theorem, i.e., the equality $\mathrm{PCP}(O(\log n), O(1)) = \mathrm{NP}$ in the Turing model two different proofs are known, the original proof by Arora et al. [3, 2] and, more recently, the groundbreaking new proof by Dinur [7]. The first PCP type theorem for the real number model was given in [9]. There, the existence of so called transparent long proofs

* We gratefully acknowledge support of both authors by project ME 1424/7-1 of the Deutsche Forschungsgemeinschaft DFG.

30th Symposium on Theoretical Aspects of Computer Science (STACS'13).
Editors: Natacha Portier and Thomas Wilke; pp. 104–115

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for the real counterpart $\mathrm{NP}_{\mathbb{R}}$ of NP is shown, see also [4]. Due to the reals as underlying structure the proof needs considerable technical changes compared to the analogue discrete result. The latter are mainly caused by the presence of unstructured domains of certain linear real functions; on these domains, invariants of the uniform distribution on the vector space $GF(2^n)$ are lost. Those invariants, however, play a central role in proving the Turing model result.

The first characterization of real $\mathrm{NP}_{\mathbb{R}}$ by a $\mathrm{PCP}_{\mathbb{R}}$ class is presented in [10]: $\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), \mathit{polylog}(n))$. The proof faces similar difficulties as the one in [9], this time dealing with low-degree polynomials as coding objects instead of linear functions on certain real domains. The key point here is to embed and expand a so called low-degree test for polynomials defined on (almost) arbitrary real domains given in [8] to construct a suitable verifier for an $\mathrm{NP}_{\mathbb{R}}$-complete problem.

Whereas the design of a long transparent proof goes into both existing proofs for the PCP theorem in the Turing model, the construction of real low-degree tests structurally follows the original proof by Arora and co-authors. It is, however, unclear to the authors of this paper at the time being whether one can obtain the full real PCP theorem by continuing the proof along these lines. We comment on that at the end of the paper. Thus, it is natural to ask whether Dinur's proof can be carried over to the real number model.

Dinur's proof is based on constructing a certain reduction between 3-SAT formulas in which for unsatisfiable formulas a so called gap amplification is obtained. The existence of such a reduction is known to imply the PCP theorem. Nevertheless, at a first glance it is not clear whether similar ideas could be used for an appropriate $\mathrm{NP}_{\mathbb{R}}$-complete problem over the reals. The reason is that Dinur's proof heavily works with constraint systems that are to be solved over different finite alphabets as domains. The proof constructs several transformations between such finite alphabet constraint problems. It seems unclear whether the same can be done over uncountable structures. However, it turns out that this part depends more on the combinatorial structure of the constraints involved than on the question over which domains they are to be solved. To see this, a view of real polynomial systems – the main objects involved in real number computations – is taken that seems a bit uncommon in algorithmic semi-algebraic geometry. The latter requires not only to consider the semi-algebraic solution set of such a system (as it is usually done in algebraic geometry), but to put more focus on a suitable grouping of the polynomials involved in the system. Then, it is more important to argue about common semi-algebraic solutions of some of these groups than of the entire system. It turns out that this can be accomplished as well over $\mathbb{R}$. Consequently, at the moment Dinur's proof seems more appropriate for adaptation to real computational models than the classical one. The arguments given hold as well for the complex number BSS model and its corresponding $\mathrm{PCP}_{\mathbb{C}}$ classes and $\mathrm{NP}_{\mathbb{C}}$. Since all required changes are minor below we only add short remarks on the complex model where appropriate.

The paper is organized as follows. Section 2 introduces the main notions like real number verifiers and $\mathrm{PCP}_{\mathbb{R}}$ classes as well as the central $\mathrm{NP}_{\mathbb{R}}$-complete problem to be studied. It is a particular form of deciding solvability of polynomial systems; the problem is defined in such a way that the existence of a reduction amplifying the unsatisfiability gap will imply the $\mathrm{PCP}_{\mathbb{R}}$ theorem. The construction of this reduction is given in Section 3. A discussion about future questions ends the paper. All lacking proofs are postponed to the full version.

A word concerning the style of writing: We decided to extensively explain the flow of ideas at the cost of having to omit most of the proofs due to lacking space. This hopefully clarifies what is needed for a transformation of Dinur's proof to the reals and why this indeed is possible. A description of Dinur's proof is given in the very recommendable textbook [1].

2 Basic notions

We assume the reader to be familiar with real and complex number complexity theory, see [5]. Very briefly, a BSS machine is a uniform Random Access Machine that computes with real numbers as basic entities. An input $x \in \mathbb{R}^n$ is given the algebraic size $size_{\mathbb{R}}(x) := n$, and each operation $\{+, -, *, :, \geq 0?\}$ among real numbers can be performed with (algebraic) costs 1. The complexity class $\mathrm{NP}_{\mathbb{R}}$ consists of all decision problems L for which there exists a polynomial time verification procedure that satisfies the following requirements. Given $x \in L$ there is a proof y of polynomial size in the (algebraic) size of x such that the procedure accepts (x, y). And for every $x \notin L$ the procedure rejects all tuples (x, y), no matter what y looks like. For complex computations inputs stem from some $\mathbb{C}^n$ and only equality tests are allowed. The Quadratic Polynomial Systems problem introduced below is a typical example of a problem in $\mathrm{NP}_{\mathbb{R}}$ or in $\mathrm{NP}_{\mathbb{C}}$, respectively, depending on where coefficients lie and where solutions are searched.

Usually, the natural verification procedures for $\mathrm{NP}_{\mathbb{R}}$ problems have to inspect all components of y before the decision is made. The question one studies in relation with PCPs is whether this has to be the case. It is formalized using special randomized algorithms.

▶ **Definition 1** (Verifiers). Let $r, q : \mathbb{N} \mapsto \mathbb{N}$ be two functions. An $(r(n), q(n))$-restricted verifier V in the BSS model is a particular randomized real number algorithm working in three phases. For an input $x \in \mathbb{R}^* := \bigcup_{i \geq 1} \mathbb{R}^i$ of algebraic size n and another vector $y \in \mathbb{R}^*$ representing a potential membership proof of x in a certain set $L \subseteq \mathbb{R}^*$, the verifier in a first phase produces non-adaptively a sequence ρ of $r(n)$ many random bits (under the uniform distribution on $\{0,1\}^{r(n)}$). Given x and this sequence ρ of $r(n)$ many random bits, in the next phase, V computes deterministically the indices of $q(n)$ many components of y. Finally, in the decision phase V uses the input x, the random string ρ and the values of the chosen components of y in order to perform a deterministic polynomial time algorithm in the BSS model. At the end of this algorithm V either accepts or rejects the triple (x, y, ρ). For an input x, a guess y and a sequence of random bits ρ we denote by $V(x, y, \rho) \in \{0, 1\}$ the result of V.

Though being a real number algorithm the verifier generates discrete random bits in phase 1. The use of these bits is for addressing registers of the machine in which the basic units of a proof y, i.e., real numbers are stored. Therefore it is appropriate to work with this discrete kind of randomness. We can define the real language accepted by a verifier together with complexity classes $\mathrm{PCP}_{\mathbb{R}}(r(n), q(n))$ as follows.

▶ **Definition 2** ($\mathrm{PCP}_{\mathbb{R}}$-classes). Let $r, q : \mathbb{N} \mapsto \mathbb{N}$; a real number decision problem $L \subseteq \mathbb{R}^*$ is in class $\mathrm{PCP}_{\mathbb{R}}(r(n), q(n))$ iff there exists an $(r(n), q(n))$-restricted verifier V such that conditions a) and b) below hold:

a) For all $x \in L$ there exists a $y \in \mathbb{R}^*$ such that for all randomly generated strings $\rho \in \{0,1\}^{r(size_{\mathbb{R}}(x))}$ the verifier accepts.
b) For any $x \notin L$ and for all $y \in \mathbb{R}^*$ the verifier rejects with probability at least $\frac{3}{4}$.

In both cases the probability is chosen uniformly over all random strings ρ.

We next introduce the central decision problem to consider in this paper. It deals with polynomial systems and is a variant of the Hilbert Nullstellensatz problem. However, the viewpoint under which we structure such systems is a bit unusual, so the definition at a first glance might look confusing. The reason we take this unusual point of view is that we want this decision problem to resemble the CSP problem which plays a main role in Dinur's proof. A clarifying example follows after the definition.

▶ **Definition 3.** a) Let m, k, q, s be integers. An instance of the quadratic polynomial systems decision problem $QPS(m, k, q, s)$ is defined as follows. There are m constraints, each of which is a group of at most k real polynomials. The polynomials depend on arrays of real number variables. These arrays are disjoint, different arrays do not contain the same variable. Each array has s components and thus represents a vector of variables ranging over $\mathbb{R}^s$. The single polynomials then depend on the variables that occur as components in one of the arrays. All polynomials in one constraint have degree at most 2 and depend on at most q many arrays, i.e., on at most qs many real variables altogether.

b) A constraint is satisfied by a point $x \in \mathbb{R}^{qs}$ if x is a common zero of all polynomials in the constraint. The $QPS(m, k, q, s)$-instance is solvable if there is a common real solution for all its constraints.

c) If above coefficients belong to $\mathbb{C}$ and solutions are searched in $\mathbb{C}^{qs}$ we obtain the complex $QPS(m, k, q, s)$ problem.

▶ Remark. a) Below, we usually consider the parameters k, q, s as constants, whereas m is depending on the actual instance. In that sense it would be more correct to talk about $QPS(k, q, s)$-instances; however, at many places we argue about the changes in the number of constraints, so the above seems notationally easier.

b) Over the reals parameter k in principle is not necessary. Here, using basic arguments from [6] we could always choose $k = 1$ and the corresponding constraint to be given by a single polynomial equation $f(x) = 0$ of degree at most 4 with f being non-negative on a corresponding $\mathbb{R}^n$. However, we notationally prefer to take care of k. Firstly in order to deal as well with the complex BSS model in which the above simplification does not work. Secondly, in many cases below we believe specifying k increases readability to state explicitly which polynomial equations enter into which constraints and might cause its unsatisfiability.

c) Since mqs is an upper bound for the number of variables an instance depends on at many places below we do not specify this number for concrete instances more precisely.

▶ **Example 4.** It is well known that deciding existence of a real zero of a polynomial system $\mathcal{P} := \{p_1, \ldots, p_m\}$, where each polynomial p_i depends on at most three variables and has degree at most two is $\mathrm{NP}_{\mathbb{R}}$-complete, see [5]. We give two formulations of this question in the new framework by specifying different choices of parameters. The examples show $\mathrm{NP}_{\mathbb{R}}$-completeness of the QPS problem for the given choices of k, q, and s.

i) The system $\mathcal{P}$ can be formulated as an instance in $\mathrm{QPS}(m, 1, 3, 1)$. Each constraint consists of a single polynomial p_i, thus $k = 1$; the variable arrays all have dimension $s = 1$ and each polynomial depends on at most 3 such arrays.

ii) If we take arrays of dimension $s = 3$, then in a first reduction step we consider all such arrays as depending on different variables. Then additional constraints have to be added to guarantee consistency between the same variables occurring in arrays of different polynomials. Such a constraint has a single polynomial depending on two arrays. It claims equality between variable components that have to be the same in the originally given system. We thus obtain a $\mathrm{QPS}(\tilde{m}, 1, 2, 3)$ instance where $\tilde{m}$ is a bound for m plus the number of different pairs of variable arrays.

Note that the above instances are equivalent to $\mathcal{P}$ as far as solvability is concerned. Below it will be very important to argue about the number of constraints not satisfied if a system is unsolvable. For these arguments it is crucial to group the single polynomials into constraints.

For what follows QPS-instances with parameter $q = 2$ are most important. This is due to the possibility of canonically assigning a constraint graph to such an instance that connects

arrays as vertices. It is then more important to argue about this graph than about the semi-algebraic solution set of (subsets of) the polynomials involved in the system (though the latter of course cannot not be completely disregarded).

The starting point of Dinur's proof is a simple observation which implies the PCP theorem if the existence of a very particular reduction can be established. We next recall this type of reduction and state the corresponding easy lemma for QPS and the $\mathrm{PCP}_{\mathbb{R}}$ theorem.

▶ **Definition 5.** a) For a $\mathrm{QPS}(m,k,q,s)$-instance ϕ denote by $UNSAT(\phi)$ the smallest fraction of constraints in ϕ that cannot be satisfied in common. Thus, if ϕ is satisfiable $UNSAT(\phi)=0$ and otherwise $UNSAT(\phi)\geq\frac{1}{m}$.

b) A gap reduction for QPS-instances is a polynomial time BSS algorithm that works as follows. There is a fixed $\epsilon>0$ such that given a $\mathrm{QPS}(m,k,q,s)$ instance ϕ the algorithm computes an instance ψ in $QPS(m',k,q,s)$ satisfying the following:

i) if ϕ is satisfiable so is ψ;

ii) if ϕ is not satisfiable, then at most a fraction of $(1-\epsilon)$ many of the constraints of ψ are satisfiable in common, i.e., $UNSAT(\psi)\geq\epsilon$.

Clearly, m' is polynomially bounded in m.

The following lemma is easy to prove. Note, however, that it seems unclear whether its converse holds as well as it does in the Turing model.

▶ **Lemma 6.** *Suppose there exists a gap reduction for QPS with a fixed $\epsilon>0$. Then the* $\mathrm{PCP}_{\mathbb{R}}$ *theorem follows, i.e., $NP_{\mathbb{R}}=\mathrm{PCP}_{\mathbb{R}}(O(\log n),O(1))$.*

3 The main proof

We shall now turn to the main part of the proof, the construction of a gap reduction for the $\mathrm{NP}_{\mathbb{R}}$-complete problem $\mathrm{QPS}(m,C,Q,1)$, see Example 4. The parameters $C\geq 1$ and $Q\geq 3$ are constants that will be specified later. The structure of the proof is similar to that of the classical PCP theorem by Dinur. Its basic idea is as follows. Starting from a $\mathrm{QPS}(m,C,Q,1)$-instance ϕ which is unsatisfiable an amplification step is performed. It constructs in polynomial time another QPS-instance ψ out of ϕ that has an increased unsatisfiability ratio. More precisely, if $UNSAT(\phi)=\epsilon$, then $UNSAT(\psi)\geq c\cdot\epsilon$ for a suitable constant $c>1$ and ϵ small enough. Now in principle starting with $UNSAT(\phi)\geq\frac{1}{m}$ and repeating this amplification $\log m$ times the gap is increased from at least one unsatisfied constraint in ϕ to a constant fraction of unsatisfied constraints in the finally resulting instance. However, the amplification step increases the dimension of the variable arrays too much. Thus, before repeating amplification a dimension reduction step is performed that first reduces the parameter s again. Note that dimension reduction in Dinur's proof is called alphabet reduction. Over the reals, however, parameter s refers to the dimension of variable arrays, whereas the underlying alphabet is always infinite. We thus consider the changed notion to be more appropriate here.

Amplification is performed on instances having particularly structured constraint graphs. These are related to so called expanders. Therefore, in a preprocessing step it has to be shown why it is possible to start with such particular instances.

The section is organized as follows. The first subsection collects the results necessary to do the preprocessing. Since it closely follows the classical preprocessing step omit the proofs. Next, the amplification step is described. Though basically Dinur's idea works over the reals as well, a lot of small details and calculations have to be changed. We thus include the full

proof, always pointing out where differences to the discrete setting occur. The dimension reduction step is given in subsection 3.3. It relies on the long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$, see [9, 4].

3.1 Preprocessing

In order to apply below the main steps necessary to establish the existence of a gap reduction for QPS, namely amplification and dimension reduction, we first have to preprocess a given instance. The goal of this preprocessing step is to obtain a QPS instance that has a constraint set which in a certain sense is highly structured. Such instances are called *nice* below. Niceness is modelled using expanders, a well known concept from graph theory. Throughout this section (except for the start of the first preprocessing step) we consider QPS instances whose constraints depend on two variable arrays, i.e., for which parameter $q = 2$. This allows to canonically attach a constraint graph to the instance.

▶ **Definition 7.** (Constraint graph) For a QPS$(m, k, 2, s)$ instance ϕ we define its constraint graph as the graph which uses the variable arrays of ϕ as vertices and where two of them are connected iff they occur in a common constraint of ϕ.

Before the amplification step is performed it is necessary to guarantee that this constraint graph has a particular structure.

We shall first give the necessary graph theoretical definitions and then show why without loss of generality we can start from a nice QPS instance. Expanders are regular graphs that in a certain sense exhibit properties of random regular graphs of the same degree of regularity. In this section we define algebraic expanders.

For the definition of algebraic expansion we need the random walk matrix of a graph G.

▶ **Definition 8.** (Random walk matrix) Let $G = (V, E)$ be a graph. The random walk matrix $A(G)$ of G is defined to be the $|V| \times |V|$ matrix in which the entry A_{ij} equals the probability that in a random walk on G vertex j is chosen after vertex i. Here each edge incident with node i and not being a loop is chosen with the same probability, whereas loops are chosen with twice this probability; see Remark 3.1 below.

▶ **Definition 9.** (Algebraic expansion) Let $n, d \in \mathbb{N}, \lambda < 1$, and $G = (V, E)$ a d-regular graph with $|V| = n$. Let $\lambda(G)$ be the second largest eigenvalue in absolute value of $A(G)$. The graph G is called a d-regular expander with expansion parameter λ if $\lambda(G) \leq \lambda$.

The QPS-instances that are important below are required to have a constraint graph which is an expander having additional properties. The corresponding definition is

▶ **Definition 10.** A QPS(m, k, q, s)-instance ϕ is called nice if the following conditions hold:

i) the number q of arrays on which each constraint depends is 2;
ii) the constraint graph G of ϕ is d-regular for some absolute constant $d \in \mathbb{N}$ which is in particular independent of the parameter s. We allow G to have loops (resulting from constraints that only depend on a single array); for each vertex of the graph one third of the edges incident to that vertex are loops.
iii) The constraint graph is an expander with algebraic expansion parameter $\lambda(G) \leq 0.9$.

▶ Remark. Our main purpose when considering random walks on a constraint graph is to guarantee that all edges occur with the same probability as, say, first edge of such a walk if a vertex is chosen at random. To achieve this property we consider loops as contributing one

edge which, however, in a random walk is chosen with twice the probability of edges that are no loops. There is still one constraint attached to a loop, and the loop contributes 2 to the degree of its vertex. Consequently, for the random walk matrix a loop contributes $\frac{2}{d}$ to the corresponding diagonal entry.

The following theorem summarizes preprocessing.

▶ **Theorem 11.** *There exist a constant $d \in \mathbb{N}$ and a polytime computable function from QPS instances to QPS instances which maps a QPS(m, k, q, s) instance ϕ to a nice instance ψ in QPS$(3qd^2m, k + qs, 2, qs)$ such that*
- *if ϕ is satisfiable, then ψ is satisfiable;*
- *if ϕ is not satisfiable, then $UNSAT(\psi) \geq UNSAT(\phi)/(240qd^2)$.*

It is important for what follows that above both the number of constraints is increased and the unsatisfiability factor is decreased by a constant factor only.

3.2 Amplification

Given a nice QPS-instance the all-over purpose now is to perform a logarithmic number of reduction rounds to increase the unsatisfiability gap. The first step in a round is an amplification step which increases the ratio of unsatisfied constraints by a constant factor > 1. After the amplification step a dimension reduction step reduces again the dimension of the variable arrays which has been increased during amplification. In this subsection amplification is explained.

Suppose a nice QPS$(m, k, 2, s)$-instance ψ is given. Let d denote the corresponding regularity parameter and let n be the number of variable arrays. Our final goal is to construct an instance which either is satisfiable if ψ was or in which for any assignment a constant fraction $\epsilon_{final} > 0$ (to be specified) of the constraints will not be satisfied. We will concentrate our arguments on how amplification works for unsatisfiable instances ψ which have a gap that is too small, i.e., smaller than ϵ_{final}. If $UNSAT(\psi)$ is already large enough it will stay above this ϵ_{final} after the reduction, see below.

So we assume that the input instance has a gap which is smaller than some constant which we will specify later. Our goal is to construct in polynomial time an instance ψ^t in some QPS$(m(t), k(t), 2, s(t))$ such that $UNSAT(\psi^t) \geq c \cdot UNSAT(\psi)$ for a suitable constant $c > 1$. Here, $t \in \mathbb{N}$ is a suitably chosen constant that results from the construction. Before going into details here is a brief outline of how amplification was done by Dinur, adapted to the real case. The new instance ψ^t has the same number n of variable arrays as ψ. Whereas the latter range over $\mathbb{R}^s$ the former range over an enlarged $\mathbb{R}^{s(t)}$, where $s(t) := d^{t+\sqrt{t}+1} \cdot s$. The constraints in the new instance are built on base of paths with $2t$ many edges in the old constraint graph G_ψ. Each such path results in an own constraint of ψ^t. Before defining such a constraint it is necessary to describe the role of the variable arrays in ψ^t. For a vertex $i \in \{1, \ldots, n\}$ the corresponding variable array $y_i \in \mathbb{R}^{s(t)}$ consists of $d^{t+\sqrt{t}+1}$ many blocks of dimension s each (therefore $s(t) = d^{t+\sqrt{t}+1} \cdot s$). Each block is thought of as an old variable array of ψ that corresponds to a vertex in G_ψ reachable within $t + \sqrt{t}$ steps from i. Since G_ψ is d-regular there are at most $d^{t+\sqrt{t}+1}$ many such vertices. In that sense we can say that an assignment for all new variable arrays $y_i \in \mathbb{R}^{s(t)}, 1 \leq i \leq n$, claims a value for all old arrays x_j for vertices j in a $(t + \sqrt{t})$-neighborhood of vertex i. Of course, different y_i might claim different values on the same old array.

Now let p be a path of length $2t$ in G_ψ from vertex i_1 to i_{2t+1}, say $p := (i_1, i_2, \ldots, i_{2t+1})$. For each such path a constraint is added to ψ^t as follows: The constraint depends on the two arrays y_{i_1} and $y_{i_{2t+1}}$ and expresses two requirements:

1. Consistency-between-new-variables requirement: Since y_{i_1} claims values for $x_{i_1}, x_{i_2}, \ldots, x_{i_{t+\sqrt{t}+1}}$ and $y_{i_{2t+1}}$ claims values for $x_{i_{2t+1}}, x_{i_{2t}}, \ldots, x_{i_{t-\sqrt{t}+1}}$, the old variables x_{i_j} for $j \in \{t-\sqrt{t}+1, \ldots, t+\sqrt{t}+1\}$ are covered by both new arrays. We thus include for all those j linear equations expressing that that y_{i_1} and $y_{i_{2t+1}}$ claim the same values on all their components. This contributes $(2\sqrt{t}+1) \cdot s$ linear equations to the constraint. [1]
2. Consistency-with-old-constraints requirement: As explained above edges (i_j, i_{j+1}) of G_ψ are covered by both variable arrays $y_{i_1}, y_{i_{2t+1}}$ for $j \in \{t-\sqrt{t}+1, \ldots, t+\sqrt{t}\}$; to each of these $2\sqrt{t}$ many edges there corresponds an old constraint of ψ. The new constraint requires as well that those old constraints are satisfied by the values assigned to the old variable arrays through y_{i_1} (or equivalently, because of item 1., through $y_{i_{2t+1}}$).

Requirement 2. for each j is the same as in the given instance just changing variables. So each constraint in ψ^t is made of $\leq k(t) := 2\sqrt{t}k + (2\sqrt{t}+1) \cdot s$ many polynomial equations. Since G_ψ is regular there are at most $m(t) := n \cdot d^{2t}$ many paths of length $2t$, so this bounds as well the number of constraints in ψ^t.

It is easy to see that a satisfying assignment for ψ extends to one for ψ^t. Just propagate the assignment to the y_i's according to the first consistency requirement above. The hard part to see is why an unsatisfiability ratio of a given (unsatisfiable) ψ is increased by the construction. Towards this aim we relate any assignment for the new variable arrays to a so-called plurality assignment for the old arrays. Assuming this plurality assignment (like any other) to violate a fraction of $UNSAT(\psi)$ many constraints in ψ it is then shown that the given assignment y for ψ^t violates a ratio of $\geq c \cdot UNSAT(\psi)$ many constraints of ψ^t. This reasoning closely follows Dinur's one. However, due to the fact that assignments stem from the uncountable set $\mathbb{R}^{s(t)}$ instead of a finite set some arguments have to be adjusted. One necessary change in order to perform this adjustment is the inclusion of the consistency-between-new-variables requirement above.

Now towards the details. Given an assignment $y = (y_1, \ldots, y_n) \in \mathbb{R}^{n \cdot s(t)}$ for the n variable arrays $y_i \in \mathbb{R}^{s(t)}$ of ψ^t, we first define the plurality assignment inferred from it to the old arrays $x_j \in \mathbb{R}^s$: Let $t \in \mathbb{N}$ be fixed. Consider a vertex v of G_ψ together with a random walk in G_ψ of length t starting in v. Remember the way loops are treated in such a walk, see Remark 3.1. With a certain probability the walk reaches a vertex u which obviously belongs to the t-neighborhood of v. Then for this u there is a new variable array y_u. Its assignment in particular claims a value for the old array x_v. The plurality assignment x_v^{pa} for x_v then is defined to be the assignment resulting with highest probability from y according to the above random walk process. Ties can be broken arbitrarily.

One technical difference to the discrete setting has to be pointed out here. Over the reals there is no guarantee how often the plurality assignment occurs at least. It is only clear that it occurs at least once. Contrary, if the variables take values over a finite alphabet there is a constant lower bound on the probability with which the plurality assignment occurs; this bound depends only on the alphabet size but not on t. This difference requires below a modification of the discrete arguments.

Suppose then that ψ is unsatisfiable with $UNSAT(\psi) = \epsilon > 0$. Let an arbitrary assignment y for ψ^t and the related plurality assignment x^{pa} for ψ be fixed. Every assignment x^{pa} violates $\geq m \cdot \epsilon$ constraints (where m denotes the number of constraints in ψ). Our goal is to show that y violates at least a fraction of $\epsilon(t) = c \cdot \epsilon$ constraints in ψ^t for a large enough

[1] These requirements are not included in the classical construction due to the underlying finite alphabets. It will become obvious below why it is needed over the reals.

value $c > 1$. This is achieved by analyzing two cases. Consider an edge $e = (i_j, i_{j+1})$ in G_ψ such that the plurality assignment $(x^{pa}_{i_j}, x^{pa}_{i_{j+1}})$ violates the corresponding constraint in ψ.

a) If the values $x^{pa}_{i_j}, x^{pa}_{i_{j+1}}$ have been claimed by relatively many (to be specified) endpoints of the corresponding random walks it is shown that many of the endpoints of $2t$-step paths of G_ψ in which e occurs in the middle segment claim the plurality values for $x^{pa}_{i_j}, x^{pa}_{i_{j+1}}$, i.e., y violates many constraints in ψ^t. This part is analyzed similarly as in the finite alphabet situation.
b) If at least one of the values $x^{pa}_{i_j}$ or $x^{pa}_{i_{j+1}}$ has been claimed by few endpoints only (but still represents the majority of the occurring values) we show that y violates the consistency-between-new-variables requirements in a lot of constraints. This case has to be handled because of the reals as underlying structure.

We now calculate a lower bound for the expectation of a random variable V defined as follows: V counts the number of edges e as mentioned above in a random path that cause the corresponding constraint to be unsatisfied. We also calculate an upper bound for the square $E[V^2]$ of the number of such edges in a random path. Since for any nonnegative random variable V taking integral values by an application of Chebychev's inequality it is $Pr[V > 0] \geq E[V]^2/E[V^2]$ this will then give us a lower bound on the fraction of paths for which the corresponding constraint in ψ^t is not satisfied.

Assume we have an edge $e = (i_j, i_{j+1})$ such that the corresponding constraint in ψ is violated by the plurality assignment. We start by considering case a) and assume that the plurality values of $x_{i_j}, x_{i_{j+1}}$ are claimed relatively often. At first sight this information seems pretty useless because if we look at the set of paths in which e occurs in the middle segment then it is obvious that for almost all of them the distance (along the path) from x_{i_j} and $x_{i_{j+1}}$ to the respective endpoints is not t. The plurality assignment was defined using random walks of length t, so it does not say anything directly about walks which have a different length. To solve this problem we need the many loops guaranteed to exist by the niceness condition. Their existence implies that a random walk of t steps statistically is not too different from a random walk which has a few steps more or less. If the random walk starts in u, the probability that we end in v only changes very slightly if we make our walk a few steps longer or shorter. The following lemma makes this precise.

▶ **Lemma 12.** *Let $t \in \mathbb{N}, \delta \leq 1/160$ and $j \in \{t - \delta\sqrt{t}, \ldots, t + \delta\sqrt{t}\}$. If the plurality assignments for x_{i_j} and $x_{i_{j+1}}$ both occur with probability at least $\frac{5}{8}$, then the following holds. For a fraction of at least $\frac{1}{4}$ of the paths of length $2t$ that have $e = (i_j, i_{j+1})$ as j-th edge the values that the starting point y_{i_1} and the endpoint $y_{i_{2t+1}}$ of the path claim for x_{i_j} and $x_{i_{j+1}}$, respectively, agree with the plurality assignments for those arrays.*

In order to obtain the desired lower bound for $E[V]$ next we have to deal with the case where the plurality assignment is claimed with a small probability only. The following shows that in this case the corresponding edge e leads in a large fraction of the paths to a violation of the corresponding constraint via case b).

▶ **Lemma 13.** *Let $t \in \mathbb{N}, \delta \leq 1/160$ and $j \in \{t-\delta\sqrt{t}, \ldots, t+\delta\sqrt{t}\}$. If the plurality assignment for x_{i_j} occurs with probability less than $\frac{5}{8}$ the following holds. For a fraction of at least $\frac{1}{4}$ of the paths of length $2t$ that have e as j-th edge the values that the starting point y_{i_1} and the endpoint $y_{i_{2t+1}}$ of the path claim for x_{i_j} disagree.*

The corresponding statement is true for $x_{i_{j+1}}$.

Let F denote the set of edges in the instance ψ such that the corresponding constraint is violated by the plurality assignment. Recall that our goal is to prove a lower bound for

$\Pr[V > 0]$, where V is the random variable which counts the number of edges e in a random $2t$-step path that satisfy the following: e belongs to F, it is the j-th edge in the path for a $j \in \{t - \delta\sqrt{t}, \ldots, t + \delta\sqrt{t}\}$, where $\delta = 1/160$ and causes the corresponding constraint in ψ^t to be unsatisfied by assignment y. We are now able to extract such a lower bound.

▶ **Lemma 14.** *Let $\epsilon \leq \frac{1}{d\cdot\sqrt{t}}$, let ψ be an instance with $UNSAT(\psi) = \epsilon$ and ψ^t be constructed as above. For the random variable V as defined above it is $Pr[V > 0] \geq c \cdot \epsilon$ with $c = \frac{\sqrt{t}}{3520d}$.*

The above lemmas result in

▶ **Theorem 15.** *There exists an algorithm which works in polynomial time that maps a nice $QPS(m, k, 2, s)$ 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 $UNSAT(\psi^t) \geq \frac{\sqrt{t}}{3520d} \cdot UNSAT(\psi)$.*

Note that the construction works precisely the same in the complex BSS model.

3.3 Dimension reduction

The amplification step increases an unsatisfiability ratio $\epsilon < \frac{1}{d\sqrt{t}}$ by a factor $c \geq \frac{\sqrt{t}}{3520d}$. Thus starting with an unsatisfiable instance ϕ that has m constraints it would be sufficient to repeat the amplification step $\Omega(\log m)$ number of times in order to end with an instance that has a constant unsatisfiability gap $\geq \frac{1}{d\sqrt{t}}$. However, doing it naively the dimension of arrays in the resulting instance would no longer remain constant. This would imply as well the query complexity to be not any longer constant, compare Lemma 6. Therefore, the dimension has to be reduced again each time an amplification step was applied. This has to be done in such a way that we do not lose too much of the gap-increase the amplification step gave.

To get around this problem one should first alter the instance in such a way that every constraint depends on at most Q variables only. Here, Q is an absolute constant independent of the array size. In Dinur's proof this is done using so-called transparent long proofs for NP. The corresponding construction is called alphabet-reduction there because the different amplification steps deal with satisfiability problems over finite alphabets of different cardinalities. With respect to the real number model it is more appropriate to consider it as a dimension reduction. All instances that occur during amplification are to be solved over the reals, i.e., there are no different 'alphabets' to deal with.

Transparent long proofs have been used already in the first proof of the classical PCP theorem, see [2], where they are crucial for applying a technique called verifier-composition. In the real and complex number model [9, 4] show the existence of transparent long proofs for all problems in $\mathrm{NP}_{\mathbb{R}}$ and $\mathrm{NP}_{\mathbb{C}}$, respectively. More precisely, a verifier for an $\mathrm{NP}_{\mathbb{R}}$-complete problem is designed which uses a superpolynomial number of random bits and inspects a constant number Q of proof components. As already said in the introduction proving this requires considerable additional work in comparison to the Turing setting. The main task is to design an algorithm for testing linearity of certain real number functions on unstructured finite subsets of some $\mathbb{R}^n$. Unstructured here means in particular that these domains are not closed under addition and scalar multiplication. This causes certain invariance properties of the uniform distribution to be violated. The latter, however, is crucially used in the finite alphabet framework to show existence of long transparent proofs.

Long transparent proofs provide a way to replace each constraint in ψ^t by many constraints all depending on at most Q real variables (i.e., arrays of dimension 1); Q denotes the constant

query complexity of the long transparent proof and thus is independent of the instance. If the old constraint is not satisfiable, then at least half of the new ones will not be satisfied. Thus one gets a reduction from constraints considered as QPS-instances to QPS-instances which blows up the gap to a constant. As seeming disadvantage the size of the long proof becomes superpolynomial in the size of the instance. But the verification using long proofs will be applied to instances of constant size only, namely single constraints in ψ^t. Thus the length of the transparent long proofs in fact does not matter at all. The much more important aspect is their structure which will not be explained here due to lack of space. The main result of this subsection, of which we also omit the proof, is

▶ **Theorem 16.** *There exists a reduction which works in polynomial time and maps a QPS$(m(t), k(t), 2, s(t))$-instance ψ^t to a QPS$(\widehat{m}(t), C, Q, 1)$-instance $\widehat{\psi^t}$, where C, Q are constants, $\widehat{m}(t)$ is linear in $m(t)$ (the multiplication factor being double exponential in $s(t)$) and the following holds:*

- *If ψ^t is satisfiable, then so is $\widehat{\psi^t}$ and*
- *if ψ^t is unsatisfiable, then UNSAT$(\widehat{\psi^t}) \geq$ UNSAT$(\psi^t)/(160(d+1)^2)$.*

3.4 Putting all together

Let $Q \geq 3$ be the $O(1)$-constant from long transparent proofs for QPS$(m, 1, 3, 1)$ and let $C \geq 1$ be the number of polynomials in a proof check ,i.e., the number of polynomials in the QPS-instance which the verifier computes out of the input instance and the random bits.

Given an instance ϕ of QPS$(m, C, Q, 1)$, applying preprocessing yields a nice instance of QPS$(3Qd^2m, C+Q, 2, Q)$ (Theorem 11), then applying amplification yields an instance of QPS$(3d^{2t+2}Qm, 2\sqrt{t}(C+Q)+(\sqrt{t}+1)Q, 2, d^{t+\sqrt{t}+1}Q)$, and finally applying dimension reduction yields an instance $\widehat{\psi^t}$ of QPS$(m', C, Q, 1)$ with the following properties.

- m' is linear in m, the multiplication factor is double exponential in Qd^t;
- if ϕ is satisfiable so is $\widehat{\psi^t}$;
- if ϕ is unsatisfiable and UNSAT$(\phi) \leq \frac{1}{d\sqrt{t}}$, then UNSAT$(\widehat{\psi^t}) \geq$ UNSAT$(\phi) \cdot \frac{\sqrt{t}}{10^{10}Qd^4}$.

Assume ϕ is not satisfiable. We now choose $t = (2 \cdot 10^{10}Qd^4)^2$ so that a gap which is smaller than $\frac{1}{d\sqrt{t}}$ will be amplified with a factor of at least 2 by this reduction. Thus from an instance of QPS$(m, 1, 3, 1)$, one builds in $\log m$ steps an instance with a gap of at least $\frac{1}{d\sqrt{t}}$.

Finally, since in every step the number of constraints increases linearly, after less than $\log m$ steps the number of constraints in the final instance is polynomial in m. Using Lemma 6 we thus arrive at the Main Theorem:

▶ **Theorem 17.** *It holds $NP_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1))$. The same is true in the BSS model over $\mathbb{C}$.*

4 Open questions

First, we consider it interesting to figure out whether the theorem as well can be proved along the lines of the first proof of the classical PCP theorem in [2, 3]. Its main ingredients are certain property testing procedures as well as a technique called verifier composition. Whereas the latter is very similar to the ideas behind the dimension reduction step above, the different property testing algorithms necessary probably result in more severe difficulties in the real number setting. Testing linear functions can be done similarly, as has been discussed above in relation with transparent long proofs for $\mathrm{NP}_{\mathbb{R}}$. In [10] a first characterization of $\mathrm{NP}_{\mathbb{R}}$ via $\mathrm{PCP}_{\mathbb{R}}(O(\log n), O(polylog(n)))$ was given by designing a real algorithm for testing low-degree polynomials. This algorithm is based on testing the maximal degree of a polynomial

with respect to its variables. In order to apply the verifier composition step the classical proof of the PCP theorem puts such a low-degree test into a better structure by designing a total-degree test. It is unclear whether such a test could be designed as well over the reals without loosing other important properties such as the length of a proof.

Secondly, approximation problems have not yet been studied in real number complexity to a comparable extent as in classical complexity theory. An important implication of the classical PCP theory was the non-approximability of the MAX-3-SAT optimization problem via so called polynomial time approximation schemes, see [1]. A natural problem to study in this respect is to maximize the number of commonly solvable polynomial equations in a real number polynomial system. A direct implication of the existence of a gap-reduction shows that this maximum is not efficiently approximable. More precisely, given a system and an arbitrary $\epsilon > 0$, unless $\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$ there is no real number algorithm running in polynomial time in the system's size which approximates the maximal number of commonly solvable equations within a relative factor $1 + \epsilon$. A promising direction for future research seems to get more (non)-approximability results of that type for natural real number optimization problems as consequence of the $\mathrm{PCP}_{\mathbb{R}}$ theorem.

Thirdly, in view of the BSS model having been introduced for many further structures like rings, vector spaces or groups one might ask whether the PCP theorem as well holds in such structures.

Finally, there are of course many further questions that have been studied in the Turing model as consequence of the PCP theorem which also make sense in the BSS-model. One typical such is the problem to optimize the parameters in the $\mathrm{PCP}_{\mathbb{R}}$ theorem.

References

1 S. Arora, B. Barak: Computational Complexity: A Modern Approach. Cambridge University Press, 2009.

2 S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy: Proof verification and hardness of approximation problems. Journal of the ACM 45 (3), 501–555, 1998. Preliminary version: Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 14–23, 1992.

3 S. Arora, S. Safra: Probabilistic checking proofs: A new characterization of *NP*. Journal of the ACM 45 (1), 70–122, 1998. Preliminary version: Proc. of the 33rd Annual IEEE Symposium on the Foundations of Computer Science, 2–13, 1992.

4 M. Baartse, K. Meer: Topics in real and complex number complexity theory. Submitted to: Proc. of the Santaló Summer School "Real Computation and Complexity", UIMP, Santander, 2012.

5 L. Blum, F. Cucker, M. Shub, S. Smale: Complexity and Real Computation. Springer, 1998.

6 L. Blum, M. Shub, S. Smale: On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. Bull. AMS, vol. 21, 1–46, 1989.

7 I. Dinur: The PCP theorem by gap amplification. Journal of the ACM Vol. 54 (3), 2007.

8 K. Friedl, Z. Hátsági, A. Shen: Low-degree tests. Proc. SODA, 57–64, 1994.

9 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.

10 K. Meer: Almost transparent short proofs for $\mathrm{NP}_{\mathbb{R}}$. Extended abstract in: Proc. 18th International Symposium on Fundamentals of Computation Theory FCT 2011, Oslo, Lecture Notes in Computer Science 6914, 41–52, 2011.