Probabilistic Verification and Non-approximability

Mario Szegedy and Kashyap Kolipaka

Contents

M. Szegedy (✉) • K. Kolipaka
Department of Computer Science, Rutgers, The State University of New Jersey, Piscataway, NJ, USA
e-mail: szegedy@cs.rutgers.edu; kolipaka@cs.rutgers.edu

Abstract

This chapter describes some of the highlights of the theory of Probabilistically Checkable Proofs, from the milestone discovery of Feige et. al. through Dinur's proof of the PCP theorem to the lately discovered consequences of the Unique Games Conjecture.

Our goal is to illuminate the major themes that run through the entire theory: probabilistic verification, proof recursion, gap enlarging reductions, code checking, constraint satisfaction problems. We have also discussed specific problems: set cover, independent set, graph coloring, etc. Since a completely comprehensive picture of this vast subject is not possible, we give pointers to a generous amount of literature.

1 Introduction

In 1991, Feige, Goldwasser, Lovász, Safra, and Szegedy [38] showed that results about multi-prover systems or equivalently PCPs imply the relative hardness of approximating the maximum clique number of a graph. Their discovery has joined the subjects of probabilistic verification and the theory of NP optimization, and a new subject emerged, called PCP theory. The goal of this chapter is to give an introduction and an overview of this theory. Since the key results alone would span several hundred pages, we have restricted ourselves to a representative selection.

Undoubtedly, the most fundamental part of the theory, with its numerous consequences, is the PCP theorem, which asserts that MAX3SAT is NP hard to approximate within a factor of $1 + \epsilon$ (for some $\epsilon > 0$). In Sect. 3 we sketch the short proof of it by Irit Dinur [33].

Reductions play a key part in proving hardness of approximation. They fall roughly into two types. Originally and traditionally relative hardness of approximating optimal values of NP optimization problems (NPO) was proven by gap-preserving reductions. In the more modern theory, we almost exclusively use Karp reduction between gap problems. We explain both types, as well as various amplification techniques, such as the parallel repetition.

We introduce the reader to probabilistic verification. The prover-verifier intuition has been the powerful force that exponentially accelerated the theory's development and even today, when alternative interpretations are available, still proves to be indispensable when trying to obtain stronger results.

We shall talk about the non-approximability of classic NPO such as Max Clique, graph coloring, and set cover.

We shall explain the long code of Bellare, Goldreich, and Sudan, as it receives multiple applications throughout the theory.

A well-studied class of optimization problems is the class of Constraint Satisfaction Problems (CSP), which also happens to be the class, for which our non-approximability theory is the most successful. We devote a lot of attention to this class and sketch optimal non-approximability for its members.

Many of the optimal non-approximability results come from unique games, in particular, a beautiful and general result due to Raghavendra [83]. Unique games-based hardness assumptions, initiated by Khot [61], might transform our views on complexity, by providing an alternative to the P versus NP paradigm. We devote two sections to unique games.

As we have indicated, we have made several omissions. We do not discuss several known non-approximability results. We do not mention efforts and techniques to optimize parameters of PCPs, such as proof size. Our chapter is concerned only with NPO. Probabilistic debate systems [26] and non-approximability of #P problems are out of the scope of this survey.

1.1 Probabilistically Checkable Proofs

The idea of probabilistic verification builds on the idea of deterministic verification. Recall that every language $L \in NP$ is deterministically verifiable in polynomial time, meaning that for every $L \in NP$, there is a polynomial-time machine V such that if $x \in L$ then there exists a polynomial size membership such that $V(x, P) = 1$ and if $x \notin L$ then for every P, $V(x, P) = 0$ (Fig. 1). The concept of probabilistic verification is analogous.

Definition 1 (Probabilistic Verification) For functions $f, g : \mathbf{N} \to \mathbf{N}$, a probabilistic polynomial-time verifier $V(x, P, r)$ is (f, g)*-restricted* if, for every input x of size n, it uses at most $f(n)$ random bits and examines at most $g(n)$ bits in the membership proof while checking it. Let Σ be an alphabet and $L \subseteq \Sigma^*$. V is an (f, g)-restricted polynomial-time probabilistic verifier for L with completeness q and soundness p if for every input x:

- If $x \in L$, then there exists a membership proof P such that

$$Prob_r(V(x, P, r) = 1) \geq q$$

- If $x \notin L$, then for any membership proof P,

$$Prob_r(V(x, P, r) = 1) \leq p$$

If q and p are not stated explicitly, then, by definition, $q = 1$, and $p = 1 - \epsilon$ for some fixed $\epsilon > 0$.

Here we have to make some comments. First, it is best to think of V as a random access machine. Since Turing machines and random access machines are polynomial-time equivalent, we may also view V as a Turing machine, but even in the latter case, we need to assume that V has random access to the bits of P. Another issue is the *adaptivity* of V. Are the bits of P which V accesses determined in advance from x and r, or the location of the second bit read from P may depend on value of the first bit read from P, etc.? In the first case, we call the verifier *nonadaptive*, and in the second case, we call the verifier *adaptive*. Most verifier constructions in the PCP theory are nonadaptive.

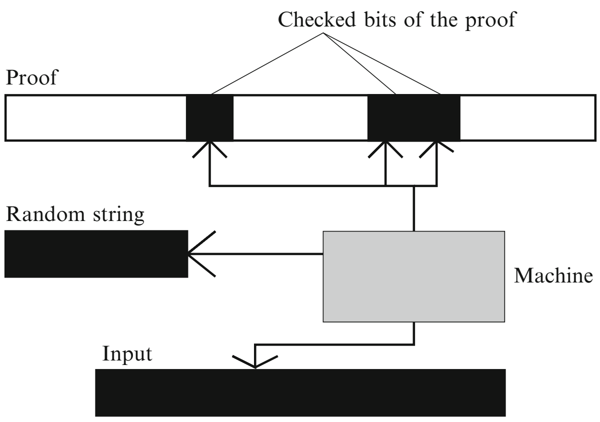

Fig. 1 A schematic picture of a probabilistic verifier. Only the black bits are read for a fixed random choice r

Definition 2 (PCP Classes[5]) A language L is in the class $PCP(f, g)$ iff there exists a constant $C > 0$ and an $(f(|x|^C), g(|x|^C))$-restricted polynomial-time probabilistic verifier for L. For completeness and soundness values other than 1 and $1 - \epsilon$, we denote the corresponding PCP class by $PCP_{q,p}(f, g)$.

Clearly, $NP = PCP(0, |x|)$.

Lemma 1 *If $L \in PCP_{q,p}(f, g)$ and L_1 polynomial time many–one reduces to L, then $L_1 \in PCP_{q,p}(f, g)$.*

Proof Since L_1 many–one reduces to L, there is a polynomially computable T such that for every $x \in \Sigma^*$, $T(x) \in L$ iff $x \in L_1$. Let $V(x, P, r)$ be a probabilistic verifier for L. Then $V(T(x), P, r)$ is a probabilistic verifier for L_1 with the required parameters. □

The PCP notation allows for a compact description of many known results:

Arora=A, Babai=B, Feige=Fe, Fortnow=F, Goldwasser=G Levin=Le, Lovasz=Lo, Lund=Lu, Motwani=M, Safra=Sa Sudan=Su, Szegedy=Sz	
$\text{NEXP} = \text{PCP}(n, n)$	BFLu[11]
$\text{NP} \subseteq \text{PCP}(\log n \log\log n, \log n \log\log n)$	BFLeSz [12]
$\text{NP} \subseteq \text{PCP}(\log n \log\log n, \log n \log\log n)$	FeGLoSaSz [38]
$\text{NP} = \text{PCP}(\log n, \sqrt{\log n})$	ASa [5]
$\text{NP} = \text{PCP}(\log n, 3)$	ALuMSuSz [4]

In Sect. 1.4 we are going to prove the last one [4] which we also call the basic PCP theorem because it is the basis of many further improvements. To describe these improvements we need to introduce new parameters such as free bit complexity, amortized complexity, and their combinations. The various parameters will be discussed in details in Sect. 2.4.

1.1.1 Alphabet Size

Besides the four parameters of a probabilistic verifier (randomness, query size, completeness, soundness), sometimes its alphabet size may be an issue. Actually, for a machine $V(x, P, r)$, we have two different alphabets we may care about: that of input x and that of proof y. We will see cases, when the alphabet size of P is particularly important. We assume that alphabet of the input is either binary or it is understood from the context.

Definition 3 A language L is in the class $[\Sigma]PCP(f, g)$ iff there exists a constant $C > 0$ and an $(f(|x|^C), g(|x|^C))$-restricted polynomial-time probabilistic verifier $V(x, P, r)$ for L, where $P \in \Sigma^*$.

Perhaps interestingly, $[\{0, 1\}]PCP_{1,p}(\log n, 2) = P$ for any $p < 1$.

1.2 A Brief History of Probabilistic Verification

The usual definition of NP uses the notion of a deterministic verifier that checks membership proofs of languages in polynomial time. Goldwasser, Micali, and Rackoff [49] and Babai [9, 13] were the first who allowed the verifier to be a probabilistic polynomial-time Turing machine that interacts with a "prover," which is an infinitely powerful Turing machine trying to convince the verifier that the input x is in the language. A surprising result due to Lund, Fortnow, Karloff, and Nisan [10, 73] and Shamir [90] has shown that every language in PSPACE—which is suspected to be a much larger class than NP—admits such "interactive" membership proofs. Another variant of proof verification, due to Ben-Or, Goldwasser, Kilian, and Wigderson [17], involves a probabilistic polynomial-time verifier interacting with more than one mutually noninteracting provers. The class of languages with such interactive proofs is called MIP (for multi-prover interactive proofs). Fortnow, Rompel, and Sipser [43] gave an equivalent definition of MIP as languages that have a probabilistic polynomial-time oracle verifier that checks membership proofs (possibly of exponential length) using *oracle access* to the proof.

Babai, Fortnow, and Lund [11] showed that MIP is exactly NEXP, the class of languages for which membership proofs can be checked deterministically in exponential time. This result is surprising because NEXP is just the exponential analogue of NP, and its usual definition involves no notion of randomness or interaction. Therefore, researchers tried to discover if the MIP = NEXP result can be "scaled down" to say something interesting about NP. Babai, Fortnow, Levin, and Szegedy [12] introduced the notion of *transparent* membership proofs, namely, membership proofs that can be checked in polylogarithmic time, provided the input is encoded with some error-correcting code. They showed that NP languages have such proofs. Feige et al. [38] showed a similar result, but with a somewhat more efficient verifier. Arora and Safra [5] further improved the efficiency of checking membership proofs for NP languages up to the point that they were able to give a new definition for NP this way.

They define a parameterized hierarchy of complexity classes called PCP (for probabilistically checkable proofs). This definition uses the notion of a "probabilistic oracle verifier" of Fortnow et al. [43] and classifies languages based on how efficiently such a verifier can check membership proofs for them. The notion of "efficiency" refers to the number of random bits used by the verifier as well as the number of bits it reads in the membership proof. Note that we count only the bits of the *proof* that are read—not the bits of the *input* which the verifier is allowed to read fully. The definition of a class very similar to PCP was implicit in the work of Feige et al. [38].

The paper of Arora, Lund, Motwani, Sudan, and Szegedy [4] in 1992 building on [5] was the first to decrease the number of check bits to constant. It is not just philosophically important that every transparently written proof can be checked with high accuracy by looking only at a constant number of places in the proof, but the construct of ALMSS also serves as a building block for nearly every construct that comes after it.

1.3 The Language of Cryptography

Articles in cryptography often describe mathematical objects such as information, knowledge, and secret through interactions between human characters. While the framework originally was used almost exclusively by cryptographers, it has become an integral part of the entire computer science culture.

In the remaining part of the text, we are going to need expressions such as "true prover," "verifier," and "cheating prover." But what do they mean exactly? When we prove that a probabilistic verifier V recognizes a language L, our argument consists of two parts. In the first part we show that if $x \in L$, the verifier accepts some proof P with probability 1. In the second part we show that if $x \notin L$ then the verifier rejects every proof with probability at least ϵ.

The first part of the argument features the *true prover* who is responsible to provide a string P which is accepted by the verifier with probability one. In this formula, $\text{Prob}_r(V(x, P, r) = 1) = 1$. The presumed structure of P is expressed in terms of the actions of the true prover. For instance, when we say that "the good prover encodes string a using an encoding E," it means that the presumed structure of the proof is a code word $E(a)$.

When we investigate the $x \notin L$ case, the main character of the story is the *cheating prover*. By assumption he is an ultimately devious creature whose proof does not adhere to the structure of the true prover. He has to cheat somewhere, since no correct proof exists when $x \notin L$. This property is required by the axioms of a proof system for L. The goal of the cheating prover is to maximize the probability with which the verifier accepts P, the "proof" he presents. It is important to emphasize that the proof of the cheating prover is an arbitrary string formed from the letters of the given alphabet, typically $\Sigma = \{0, 1\}$. When we argue about the $x \in L$ case, we never need to talk about the cheating prover.

Like in the case of other formal proof systems in the case of PCPs, the proof itself is just a vehicle to indicate that a statement holds and itself is not interesting for the verifier at all. Only the mere existence or non-existence of its corrected version matters. The verifier does not want to learn the entire proof or to verify that the proof adheres to the structure of the good prover *everywhere*. It is sufficient for us to show that if $x \notin L$, no matter what string the cheating prover may write down, it is so far from being consistent either with the structure of the true prover or with other constraints defined by input x that the verifier is able to catch the error with probability ϵ.

1.4 The PCP Theorem

Theorem 1 (PCP Theorem) $NP \subseteq PCP(\log n, 3)$. *Moreover, we can also assume that the verifier is nonadaptive and his accepting criterion is a disjunct (i.e., "OR") of the literals made from the three bits he looks at.*

Notice that by Definition 2, the factor $\log n$ scales by a constant, but not the second argument. The following seemingly weaker variant of the above theorem is also often called the PCP theorem:

Theorem 2 $NP \subseteq [\Sigma]PCP(\log n, c)$ *for some fixed constant* c *and for some fixed alphabet* Σ.

This statement was first proven by Arora, Lund, Motwani, Sudan, and Szegedy [4], based on several previous works [11, 12, 38], in particular on a simultaneous work of Arora and Safra [5]. The original proof was based on several algebraic and combinatorial techniques. The proof had several variations, but they all had the basic design. We call this the classic PCP proof design. A new type of proof was given by Irit Dinur in 2005 [33]. She has proved he following:

Theorem 3 $NP \subseteq [\{0,1\}^3]PCP(\log n, 2)$.

We give a sketch of her proof in Sect. 3. In the rest of this section, we show that Theorem 2 implies Theorem 1 (the converse is straightforward).

Lemma 2 *Let* $\ell > 3$ *be a constant and* $f(n)$ *be an arbitrary function from the positive integers to the positive integers. If there is a nonadaptive* $(f(n), \ell)$*-restricted verifier* V *that checks a proof* P *with alphabet* Σ *and recognizes a language* L *with completeness* 1 *and soundness* $1 - \epsilon$*, then there is a non-adaptive* $(f(n) + O(1), 3)$*-restricted verifier* V' *that checks a proof* P' *with alphabet* $\{0,1\}$ *and recognizes* L *such that its checking criterion is always a disjunct of three literals made from the three bits it reads. Furthermore,* V' *has completeness* 1 *and soundness* ϵ/K*, where* $K > 0$ *is some constant dependent only on* ℓ *and* Σ.

Proof We prove this lemma with a technique called *proof composition*. Proof composition is a very general principle stating that if we prove the existence of a proof for $x \in L$, then $x \in L$ follows (thus, instead of a proof, it suffices to have a proof that a proof exists). A proof composition technique in the PCP context was developed in [5]. Here we need a very simple version of it.

Let us fix x. For every fixed random choice r, the accepting criterion of V is a Boolean-valued function φ_r of the ℓ places of P examined by V. The composed proof contains the original proof and some extra information that helps the verifier. Since we want the composed proof to have a binary alphabet, we also encode the alphabet of the original proof in binary via some (arbitrary) injection $\Sigma \rightarrow \{0,1\}^{\lceil \log |\Sigma| \rceil}$. The part of the new proof that is identical to P, with its letters encoded in binary is called the *core*. For every $r \in \{0,1\}^{f(n)}$, we then create a constant size proof β_r that helps to convince the verifier that φ_r holds. The entire new proof is now the union of the core and $\bigcup_r \beta_r$. The details are as follows:

First recall from logic that every Boolean function $\varphi(\alpha)$ on ν variables can be written in the form $(\exists \beta)\ \psi(\alpha,\beta)$, where ψ is a $3CNF$ formula. β is a sequence of at most $(\nu-1)2^{\nu}$ *auxiliary variables*, and ψ has at most $2^{\nu}+(\nu-1)2^{\nu+1}$ clauses, but for our purposes, we only care that these quantities are constants in terms of ν.

Let $(\exists \beta_r)\ \psi_r(\alpha_r,\beta_r)$ be the $3CNF$-equivalent of φ_r. Here α_r is the $\ell\lceil \log |\Sigma| \rceil$ bits of information that machine V reads from P under random bit r. The help β_r (of the true prover) is any evaluation of the auxiliary variables for which $\psi_r(\alpha_r,\beta_r)=1$.

The new verifier, V', picks a random r and a random clause of ψ_r and accepts if the selected clause evaluates to 1 under α_r and β_r of the composed proof (α_r is part of the core, β_r is the sequence of auxiliary bits that the prover provides for query r). Since ψ_r has constant number of clauses, the composed verifier needs at most constant number of additional random bits to perform its query.

If V accepts a proof P with probability 1, then for every r there exists a β_r such that $g_r(\alpha_r,\beta_r)=1$, so there exists a composed proof which V' accepts with probability 1.

Assume now that $x \notin L$. We claim that no composed proof will be accepted with probability greater than $1-\frac{\epsilon}{K}$ by V', where K is the uniform upper bound on the number of clauses in $\psi_r(\alpha_r,\beta_r)=1$ for all rs.

To see this, assume that a composed proof with core P is accepted with probability greater than $1-\frac{\epsilon}{K}$. Then the fraction of rs for which the predicate $(\exists \beta_r)\ \psi_r(\alpha_r,\beta_r)$ is false is less than $K \times \frac{\epsilon}{K}=\epsilon$. Then V accepts P with probability greater than $1-\epsilon$, a contradiction. □

2 Non-approximability Results

NPO (see [29, 30, 56]) have one of the following forms:

- Find $\max_{|y| \leq q(|x|)} P(x,y)$ given input x. [maximization]
- Find $\min_{|y| \leq q(|x|)} P(x,y)$ given input x. [minimization]

Here $P(x, y)$ is a polynomially computable function that assigns non-negative rationals to pairs of strings, and q is a polynomial. (In the literature witness y is selected from an arbitrary polynomial-time computable set of strings not necessarily only of the form $\{y \mid |y| \leq q(|x|)\}$. Our definition is sufficient to describe all relevant problems and enables us to avoid anomalies that arise from the more general definition.)

While one can construct optimization problems that are not in NPO, the class plays a primary role in optimization theory with such well-known problems as coloring, allocation, scheduling, Steiner tree problems, TSP, linear and quadratic programming, knapsack, and vertex cover. The consequences of the PCP theory almost solely concern NPO problems. There are exceptions [26, 27], but we do not discuss them here.

Most NPO problems are NP-hard to compute exactly. A polynomial-time algorithm A is said to *approximate* an NP maximization problem P to within a factor $r(x) \geq 1$, if A outputs a witness y such that the optimal solution for input x lies within $\max_{|y|\leq q(|x|)} P(x, y)/r(x)$ and $\max_{|y|\leq q(|x|)} P(x, y)$. Here $r(x)$ is called *approximation ratio* and can be generalized also to minimization problems in an obvious way.

Below we give three examples to NPO problems:

MAX3SAT: Let x represent a 3CNF formula, y represent an assignment, and $P(x, y)$ be the number of clauses in x satisfied by y. [Maximization]

Set cover: Let x represent a polynomial size set system, y represent a selection of sets, and $P(x, y)$ be the function, which equals to the number of selected sets if the selected sets cover the universe, otherwise, the size of the entire set system. [Minimization]

PCP: Let V be an $(\log n, g(n))$-restricted probabilistic verifier for a language L. x represents an input, y represents a proof, and $P(x, y)$ is the acceptance probability of proof y for input x. Since the verifier uses $\log n$ randomness, we can run V on all random strings and compute this probability in polynomial time. [Maximization]

If in the latest example, we take L to be an NP-complete language and V to be the verifier of Theorem 1, then it is easy to see that if we could efficiently approximate the acceptance probability of V to within a factor better than $1 - \epsilon$ (for the ϵ of the theorem), then we could also solve the membership problem for L. This is an example to an argument about non-approximability. What is special about the optimization problem coming from Theorem 1 is that the value of $P(x, y)$ is proportional to a number of very simple predicates that y satisfies, namely, each predicate depends only on three bits of y. This makes this NPO problem a *CSP*. Before the PCP theorem there were no non-approximability results for any CSP. The theory of PCP exploits the fact that probabilistic verifiers for NP-complete languages translate to provably hard approximation problems.

2.1 A Brief History of Approximating NPO Problems

The NP-completeness theory of Cook [28] and Levin [70] puts NPO problems into two categories: NP-hard or not NP-hard. The first alarm that this characterization is too coarse was sent off in an early paper of D. Johnson [56] entitled "Approximation Algorithms for Combinatorial Problems."

Motivated by exact bounds on the performance of various bin packing heuristics of [45], Johnson gave algorithms for the subset sum, the set cover, and the MAX k-SAT problems with guarantees on their performances ($1 + o(1)$, $O(\log |S|)$, $2^k/(2^k - 1)$, respectively). He also gave non-approximability results, but unfortunately they referred only to specific algorithms. Nevertheless, he has brought up the issue of classifying NPO problems by the best approximation ratio achievable for them in polynomial time.

For years advances were very slow. Sahni and Gonzales [88] proved the non-approximability of the nonmetric traveling salesman and other problems, but their proofs were very similar to standard NP-completeness proofs, and they effected only problems where approximation algorithms were not explicitly sought for. A more promising approach was found by Garey and Johnson [46] who used graph products to show that the chromatic number of a graph cannot be approximated to within a factor of $2-\epsilon$ unless $P = NP$ and an approximation algorithm for the Max Clique within *some* constant factor could be turned into an algorithm which approximates Max Clique within *any* constant factor.

The old landscape of approximation theory of NPO radically changed when in 1991 Feige, Goldvasser, Lovász, Safra, and Szegedy [38] for the first time used Babai, Fortnow, and Lund's characterization of NEXP in terms of multi-prover interactive proof systems [11] to show that approximating the clique within any constant factor is hard to NTIME($n^{1/\log\log n}$). Simultaneously Papadimitriou and Yannakakis defined a subclass of NPO what they called MAXSNP, in which problems have an elegant logical description and can be approximated within a constant factor. They also showed that if MAX3SAT,vertex cover, Max Cut, and some other problems in the class could be approximated with an arbitrary precision, then the same would hold for all problems in MAXSNP. They established this fact by reducing MAXSNP to these problems in an *approximation preserving* manner. Their original intention was most likely to build a theory of non-approximability via defining suitable reductions, analogous to those in the theory of NP, but new non-approximability results in [4] let them achieve much more. The theory of NP and the theory of non-approximability met, and research developed rapidly in three directions:

1. Non-approximability results for NPO problems
2. Construction of approximation algorithms achieving optimal or near-optimal ratios
3. Development of approximation preserving reductions and discovery of new (non)-approximability classes

Today, we have a precise idea about the non-approximability status of many NPO problems. The on-line compendium of Pierluigi Crescenzi and Viggo Kann [29] keeps track of the growing number of results and continuously updates data about most known NPO problems.

2.2 The FGLSS Graph

The first paper to construct a combinatorial object from a PCP was that of Feige et al. [38]. For deterministic verifiers an analogous construction cannot be meaningfully made (Fig. 2). The combinatorial object was simply a graph G, and the construction implied hardness of approximating the optimization problem

$$MaxClique(G) = \max_{S \subseteq V(G)} \{|S| \mid \text{ for all } v, w \in S \text{ we have } (v, w) \in E(G)\}$$

Let V be an (f, g)-restricted verifier. The evaluation of the entries of the proof the verifier sees is called a *view*. An *accepting view* is a view the verifier accepts for some input x and random coin flip r.

By definition, an accepting view contains stars at positions where the verifier never looks at (for a particular random run) and exactly at these positions. Thus, an accepting view can be thought of as a partial proof that causes the verifier to accept. When the verifier is adaptive, the first bit read by him determines the location of the second bit, etc., so even for a fixed r, the locations of the non-stars may vary from view to view. Since the query size is upper bounded by $g(n)$, every accepting view contains at most $g(n)$ entries which are not stars. We call two accepting views consistent if they agree in locations where both are non-stars.

Let $ACC_V(x, r)$ be the set of all accepting views for a verifier V for input x and random string r, and let

$$ACC_V(x) = \{(r, W) \mid |r| = f(n), W \in ACC_V(x, r)\}.$$

Definition 4 (FGLSS Graph $G_V(x)$) Feige et al. [38] define a graph $G_V(x)$ on vertex set $ACC_V(x)$ such that (r, W) and (r', W') are connected if and only if $r \neq r'$ and W and W' are consistent.

Lemma 3 *Let V be an (f, g)-restricted verifier. Then:*
1. *$G_V(x)$ has at most $2^{f(n)+g(n)}$ vertices.*
2. *$G_V(x)$ can be constructed in time polynomial in $\max\{n, 2^{f(n)+g(n)}\}$.*
3. *$MaxClique(G_V(x))$ is equal to $\max_P |\{r \mid V(r, P, x) = 1\}|$.*

Proof One can easily check that even for an adaptive V, there are at most $2^{g(n)}$ different accepting views for a fixed r. Thus, the total number of accepting views is at most $2^{f(n)}2^{g(n)}$. In order to check whether a pair (r, W) is an accepting view or not, we can run V, which is by definition a polynomial-time machine. By comparing two views bit-wise, we can check in linear time if they form an edge in $G_V(x)$. As a result $G_V(x)$ can be constructed in time polynomial in $\max\{n, 2^{f(n)+g(n)}\}$.

The main observation of FGLSS is that there is a natural many–one map from the set of proofs (or proof candidates) to the set of cliques of $G_V(x)$ such that all maximal clique has at least one inverse image. Indeed, if we map a proof P to the set of all (r, W) pairs, with the property that $V(x, P, r) = 1$ and W is the verifier's view of P when reads r, then these pairs form a clique in $G_V(x)$, since their view

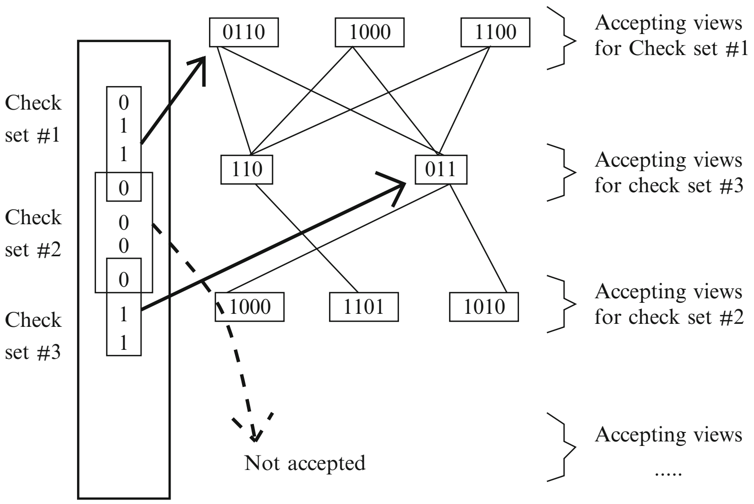

Fig. 2 Construction of the FGLSS graph. The set of all accepted views of a proof map to the set of vertices of a clique (the map is represented by the *arrows*). The reverse also holds as follows: every maximal clique of the graph corresponds to some proof

components are consistent with each other. Conversely, let C be a maximal clique of $G_V(x)$. There is a string (i.e., a proof) which is consistent with all elements of C. This can be easily seen, since a location is either star in all elements of C or uniquely defined by some elements of C. This string cannot be consistent with any view that is not in C, since C was maximal. Notice that in this correspondence, the size of C is exactly the number of those rs for which there is a W such that $(r, W) \in C$, since W, if exists, is unique for a fixed r. This number further equals to the number of different coin flips for which the verifier accepts any proof that maps to C. Point 3 of the lemma follows. $\square$

We shall apply Lemma 3 in Sect. 2.8.

2.3 The Gap-3SAT Instance

The gap-3SAT instance is a 3SAT formula, which is either satisfiable or at most $1-\epsilon$ of its clauses are satisfiable.

Theorem 4 ([4]) *There exists an $\epsilon > 0$ such that for every $L \in NP$, there is polynomial-time computable map from Σ^* to 3SAT instances (mapping $x \in \Sigma^*$ to ϕ_x) such that:*
1. *If $x \in L$, then ϕ_x is satisfiable.*
2. *If $x \notin L$, then every assignment leaves at least ϵ clauses of ϕ_x unsatisfied.*

Indeed, build a PCP for L as in Theorem 1, and define

$$\phi_x = \wedge_r c_r,$$

where c_r is the verifier's accepting criterion for random string r and which by the assumption of Theorem 1 is a disjunct of three literals. The variables are the bits of the proof. If $x \in L$ then all c_is evaluate to 1, but if $x \notin L$ then ϵ fraction of the clauses must remain unsatisfied for every assignment, i.e., for every proof of the prover.

The theorem has the immediate consequence that $MAX3SAT$ cannot be approximated in polynomial time to within a factor better than $1/(1-\epsilon)$; otherwise, we could use this algorithm to efficiently compute if $x \in L$. Since MAX3SAT is in MAXSNP (Papadimitriou and Yannakakis [80]), we get

Theorem 5 ([4]) *No MAXSNP-complete problem can be efficiently approximated to within a factor arbitrarily close to 1 unless P=NP.*

Theorem 5 does not give explicit non-approximability gaps. For explicit gaps see Sect. 2.7.

2.4 Refined Parameters of PCPs

In Sect. 1.1 we parameterized probabilistic verifiers with the number of random bits, $f(n)$, and the number of check bits $g(n)$ (see Definition 1). When we make a reduction from a PCP to an NPO problem, the exact non-approximability gap of the latter may depend on several other parameters of the PCP. Here we list some, starting with the most important ones:

1. The *completeness probability*, $q(n)$, which is the least probability of acceptance, when $x \in L$. Originally $q(n) = 1$, but Håstad [53, 54] proved that we can improve on other parameters if we allow $q(n)$ to be slightly less than 1.
2. The *soundness probability*, $p(n)$, which is the greatest probability of acceptance, when $x \notin L$.
3. The *free bit complexity*. In Sect. 2.2 we defined $ACC_V(x, r)$. Using this notation,

$$free(n) = \max_{|x|=n} \log_2 \max_r |ACC_V(x, r)|.$$

4. The *average free bit complexity*:

$$free_{av}(n) = \max_{|x|=n} \log_2 \left(\frac{1}{2^{f(n)}} \sum_r |ACC_V(x, r)| \right).$$

Note that $\sum_r |ACC_V(x, r)|$ is the size of the FGLSS graph.

5. The alphabet size $|\Sigma|$. In most cases we assume that the witness $\mathbf{w}$ is a string over the binary alphabet $\{0, 1\}$. Raz and Safra [87] showed the relevance of using alphabets which may contain up to n symbols.

The parameters we have discussed so far were defined from the model of PCP. We also consider *derived parameters* that can be expressed in terms of the parameters we have defined so far:

1. The *gap* function $gap(n) = q(n)/p(n)$.
2. The *amortized query bit complexity* defined as $\frac{g(n)}{\log_2 gap(n)}$.
3. The *amortized free bit complexity* defined as $\frac{free(n)}{\log_2 gap(n)}$, sometimes as $\frac{free_{av}(n)}{\log_2 gap(n)} = \max_{|x|=n} \log_{q/p} \left(\frac{1}{2^{f(n)}} \sum_r |ACC_V(x,r)| \right)$ [40].

The following notations for PCP classes with refined parameters are more or less standard in the literature:

$FPCP_{q,p}(f, free)$ denotes the class, where the query complexity is replaced with the free bit complexity. $\overline{FPCP}(f, \overline{free})$ is $\cup FPCP_{q,p}(f, free)$, where the union runs through for all choices of $\underline{q(n)}$, $p(n)$, and $free(n)$ such that the amortized free bit complexity is at most $\overline{free}(n)$.

2.5 Amplification Techniques

Amplification techniques are transformations on probabilistically checkable proofs or on the FGLSS graph that improve on their parameters. An ancestor of these techniques was used in [46] to prove that the chromatic number cannot be approximated to within a factor of $2 - \epsilon$. In this section we review the naive repetition, the parallel repetition, the proof composition, and the randomized graph product (see also in Fig. 3). All of them are playing important roles in the theory of non-approximability.

2.5.1 Naive Repetition

In the case of naive repetition, the new verifier's query is the conjunct of k independently chosen queries of the original verifier. It increases the gap in between the completeness and the soundness probabilities as shown in the table below:

Naive repetition	Original verifier	Repeated verifier
Number of random bits	$f(n)$	$kf(n)$
Number of check-bits	$g(n)$	$kg(n)$
Completeness probability	$q(n)$	$q(n)^k$
Soundness probability	$p(n)$	$p(n)^k$

Perhaps the most troublesome aspect of the naive repetition is that it increases the number of random bits the verifier needs to use. Zuckerman [98] suggests a "less naive" repetition, where the verifier chooses its k-tuple from a set S of polynomially

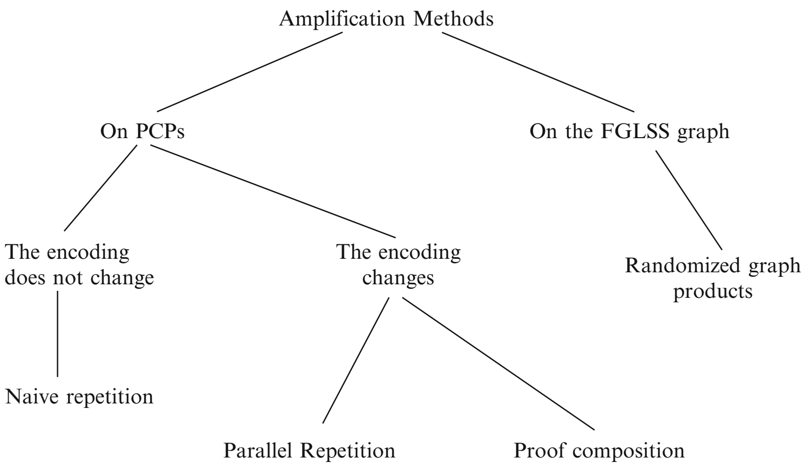

Fig. 3 Various types of amplifications

many k-tuples. Unfortunately, Zuckerman constructs S randomly. A PCP^S verifier is a PCP verifier that works with a (random) advise S. With high probability over a random S with $|S| = 2^{F(n)}$, the following holds for Zuckerman's verifier:

	Original verifier	Zuckerman's verifier
Number of random bits	$f(n)$	$F(n)$
Number of check bits	$g(n)$	$g(n)(F(n)+2)$
Completeness probability	1	1
Soundness probability	$1/2$	$2^{f(n)-F(n)}$

Lars Engebretsen and Jonas Holmerin [36, Lemma 14] have computed the performance of the Zuckerman's verifier for all values of the completeness and soundness probabilities. This is what they got:

	Original verifier	EH-verifier
Number of random bits	$f(n)$	$F(n)$
Number of check bits	$g(n)$	$g(n)\frac{F(n)+2}{-\log p(n)} = g(n)D$
Completeness probability	$q(n)$	$q^D(n)/2$
Soundness probability	$p(n)$	$2^{f(n)-F(n)}$

In the case of perfect completeness, the EH verifier also has perfect completeness.

2.5.2 Parallel Repetition

Parallel repetition of PCPs, as opposed to naive repetition, requires to change not only the verifier's behavior but also the encoding. It increases the alphabet size in order to keep the query size only two and works as follows.

Assume that an (f, g)-restricted verifier V expects P to be segmented, i.e., P consists of blocks (usually of some fixed block size). Note that the segmentation is something that is only in the verifier's mind, so a cheating prover has no influence on this: segmented proof simply refers to the verifier's reading behavior. Assume that the verifier always accesses data only from two different blocks of the proof. A further restriction is that P consists of two sets of blocks, P_1 and P_2, such that for every random string, V accesses one block from each set. A PCP with this structural restriction is called a *two-prover system*, since P is being thought to be provided by two different provers whose answers the verifier compares one with another.

Any naive repetition with $k > 1$ would alter the two-prover nature of the system, but parallel repetition keeps it. We shall denote a verifier V repeated k times in parallel by V^k. The true prover for V^k writes down all possible ordered k-tuples of segments from P_1 and all possible ordered k-tuples of segments from P_2, and these k-tuples are the segments of the repeated provers. The verifier uses $kf(n)$ randomness to simulate k independent checks $(c_{1,1}, c_{2,1})$, … $(c_{1,k}, c_{2,k})$ of V by querying segments $(c_{1,1}, \ldots, c_{1,k}) \in P_1^k$ and $(c_{2,1}, \ldots, c_{2,k}) \in P_2^k$. V^k accepts iff V would accept all pairs $(c_{1,i}, c_{2,i})$ for $1 \leq i \leq k$.

It is easy to compute the acceptance probabilities if the proof has the P_1^k, P_2^k structure, but there is no guarantee that cheating provers adhere to this structure. Surprisingly we can still say something about the maximum acceptance probability for V^k.

Theorem 6 (Raz's Parallel Repetition Theorem [86]) *For every verifier V of a 2-prover PCP, there is a constant $0 \leq c \leq 1$:*

$$\max_{P'} Prob_{r_1,\ldots,r_k}(V^k(x, P', r_1, \ldots, r_k) = 1) \leq c^k, \qquad (1)$$

where c depends only on $p(V, x) = \max_P Prob_r(V(x, P, r) = 1)$ and the maximum segment size for V and is strictly less than 1 if $p(V, x) < 1$. (In Eq. (1) P' ranges through all (possibly cheating) proofs for V^k.)

The properties of the repeated proof are summarized in the table below:

	Original verifier	With k repetitions
Number of random bits	$f(n)$	$kf(n)$
Number of check bits	$g(n)$	$kg(n)$
Completeness probability	$q(n)$	$q(n)^k$
Soundness probability	$p(n)$	c^k (see Theorem 6)

The price we pay for keeping the two-prover structure is a weaker bound on the soundness probability than in the case of naive repetition.

Two-prover systems were introduced by Ben-Or, Goldwasser, Kilian, and Wigderson [17] and have been studied by Rompel and Sipser [43], Feige and Lovász [41], and by many others.

In many PCP constructions, we use a certain folklore two-prover protocol. The verifier of this protocol, which we call V_{3SAT}, comes from the gap-3SAT instance of Sect. 2.3 and works as below.

Let ϕ_x (associated with $x \in \Sigma^*$) be a 3SAT instance, which is either completely satisfiable or at most $1 - \epsilon$ fraction of its clauses are satisfiable for some $\epsilon > 0$. By Theorem 4 we can assume that the language $L = \{x \mid \phi_x \text{ is satisfiable}\}$ is NP-complete and that ϕ_x is polynomial-time computable from x. The verifier V_{3SAT} computes ϕ_x, picks a random clause in it, and asks the first prover to evaluate the three literals in the clause. If all the three literals evaluate to false, V_{3SAT} rejects outright. Otherwise, he picks a random variable in the clause and asks the second prover to evaluate this variable (the two provers do not hear each other's answers, but they can agree in a strategy in advance, and they also know x). If the two provers evaluate the variable differently, the verifier rejects, otherwise accepts.

In order to see that the above informal description indeed defines a two-prover system in our former sense, identify P_i with the concatenation of all answers to all possible questions of the verifier to prover i. Each answer is one segment, and the number of segments in P_i equals to the number of different questions the verifier may ask from prover i. The segment size of P_1 is three bits, and the segment size of P_2 is one bit.

Next we show that the above protocol has a gap at least $\epsilon/3$. Clearly, when $x \in L$, i.e., when ϕ_x is satisfiable, both provers play consistently with the lexicographically first satisfying assignment of ϕ_x, and V_{3SAT} accepts with probability 1. If $x \notin L$, every assignment fails to satisfy at least ϵ fraction of the clauses of ϕ_x. Observe that the strategy of P_2 corresponds to an assignment to the variables of ϕ_x. For the sake of simplicity, we call this assignment P_2. This assignment must fail to satisfy at least ϵ fraction of the clauses of ϕ_x, and the verifier has probability ϵ that he finds one of these clauses. The first prover of course will lie and will not give the same evaluation to one of the variables in that clause as P_2 (to avoid sure rejection), but then, conditioned on the event that P_2 gets the clause in question, the verifier notices this inconsistency with probability at least $1/3$. Thus, in case $x \notin L$, V_{3SAT} accepts with probability at most $1 - \epsilon/3$.

Feige [37] shows that we may also assume that in ϕ_x each variable appears exactly in 5 clauses and that each clause contains exactly 3 variables. By Feige's restriction, we may assume that V_{3SAT} first selects a random variable and then chooses a clause randomly out of the five that contains this variable. Observe that the sequence in which the verifier asks the provers does not matter, and in the sequel we call the prover who provides the clauses the "second prover." We may also assume that the verifier decides at his output after he has heard both provers.

In all applications V_{3SAT} is going to be repeated in parallel ℓ times, where ℓ depends on the application and ranges from a large constant to $\log n$. The lemma below, which is a consequence of the PCP theorem and the parallel repetition theorem, summarizes the properties of $V_{3\mathrm{SAT}}^{\ell}$.

Lemma 4 *Let $\ell = \ell(|x|)$ be an arbitrary function of $|x|$ (but most often a constant), and let $b = \{b_1, \ldots, b_N\}$ be the set of variables of Feige's ϕ_x and $c = \{c_1, \ldots, c_{5N/3}\}$ be the clauses of ϕ_x. (We say $b_i \in c_j$ if b_i or $\overline{b_i}$ appears in c_j). Then $V_{3\mathrm{SAT}}^{\ell}$ works as follows:*

1. *It randomly picks an ℓ tuple $U = (b_{i_1}, \ldots, b_{i_\ell})$ and checks it against an ℓ tuple $W = (c_{j_1}, \ldots, c_{j_\ell})$ such that $b_{i_\nu} \in c_{j_\nu}$ for $1 \le \nu \le \ell$. W is called a* companion *of U, and it is randomly picked from the set of all companions of U. The provers provide evaluations of all variables involved in the queries to them (in particular, the second prover gives the values of all variables in all clauses occurring in W)* without listening to each other. *The verifier accepts if $V_{3\mathrm{SAT}}$ would accept (b_{i_ν}, c_{j_ν}) for every $1 \le \nu \le \ell$, i.e., all c_{j_ν} are satisfied, and the assignments of the variables in U and W are consistent.*
2. *It has perfect completeness.*
3. *It has soundness at most c^{ℓ} for some fixed constant $c < 1$.*
4. *It uses $O(\ell \log n)$ random bits and $\ell + 3\ell$ check bits. (Here parameter n is the length of x, and it is in polynomial relation with N.) More importantly, the verifier reads only two segments of the proof. With segment sizes ℓ (bits) for prover one and 3ℓ (bits) for prover two, this is a two-query PCP.*

The above lemma amplifies the PCP theorem and implies approximation hardness of the *label cover* problem:

Label Cover Problem: An instance of the *label cover* problem is a tuple, $\mathcal{P} = (G, \Sigma_1, \Sigma_2, \Pi)$, where:

1. $G = (X, Y, E)$ is a bipartite graph with $V(G) = X \cup Y$ and $E = E(G) = E(X, Y)$.
2. Nodes in X, Y are assigned labels from Σ_1, Σ_2, respectively.
3. $\Pi = \{\pi_{xy} \mid (x, y) \in E\}$ is a set of functions, $\pi_{xy} : \Sigma_1 \to \Sigma_2$.

The variables of $\mathcal{P}$, by definition, are the labels assigned to nodes in $X \cup Y$. For each edge $(x, y) \in E$, $\pi_{xy}(l(x)) = l(y)$ defines a binary "projection" constraint. Here, $l(x), l(y)$, are the labels assigned to x, y, respectively. The value of a label assignment is the fraction of constraints that are satisfied.

A $V_{3\mathrm{SAT}}^{\ell}$ instance is a special label cover problem $\mathcal{P}$, where the vertices of X are the ℓ-tuples of clauses of a $V_{3\mathrm{SAT}}$ instance Φ_x and the vertices right partition are the ℓ-tuples of the variables of Φ_x. Therefore, $\Sigma_1 = \{0, 1\}^{3\ell}$ and $\Sigma_2 = \{0, 1\}^{\ell}$. Lemma 4 implies:

Theorem 7 *For any positive $\epsilon < 1$, there exist sufficiently large, but constant size alphabets $\Sigma_1 = \Sigma_1(\epsilon)$, $\Sigma_2 = \Sigma_2(\epsilon)$, such that it is NP-hard to approximate label cover instances $\mathcal{P}(G, \Sigma_1, \Sigma_2, \Pi)$ within a factor smaller than $1/\epsilon$.*

Proof Choosing $\ell \in O(\log \frac{1}{\epsilon})$, the soundness of $V^{\ell}_{3\text{SAT}}$ can be made smaller than ϵ, giving the required hardness. □

2.5.3 Proof Composition

The idea of composing probabilistic verifiers first appears in the paper of Arora and Safra [5] and is aimed at reducing the number of check bits of a probabilistic proof system.

To describe proof composition, consider a proof system with an (f, g)-restricted verifier V that recognizes a language L. From now on we call V the *outer verifier*. We would like to construct a new verifier V' which recognizes the exact same language as V but uses less check bits. Let $x \in L$, $|x| = n$. The composed proof for x consists of the old proof P, which we call the *core*, and a proof P_r for every choice of the random string r with $|r| = f(n)$ which we describe later. We shall compose proof systems only if their verifiers are non-adaptive. Let Q_r be the set of bits that V reads from P when we run it with random string r. The set of accepting views $ACC_V(x, r)$ can be viewed as a language L_r consisting of words only of length $|Q_r| = g(n)$. Let V_r be an (f', g')-restricted probabilistic verifier that recognizes L_r, where $f'(g(n))$ and $g'(g(n))$ are the number of random bits and check bits V_r needs for its verification. The f' and g' bounds must uniformly hold for each L_r (a more elegant formalism for composition was worked out in [92]). V_r is called the inner verifier, and it verifies that the view of the original verifier is accepting, with the help of some extra advise string $\wp_r$, provided separately for every r in addition to the view.

The compound verifier works as follows: it picks a random r with $|r| = g(n)$ and a random string r' with $|r'| = f'(|Q_r|)$. It then runs V_r with random string r' on input Q_r, proof $\wp_r$ (reading $g'(|Q_r|)$ bits from the latter), and outputs $V_r(Q_r, \wp_r, r')$.

Let us compute the parameters of the composed proof system. Assume that V has completeness probability $q(n)$ and soundness probability $p(n)$ and that each V_r has completeness probability $p'(n)$ and soundness probability $q'(n)$. Here is what we get:

	Original proof	Composed proof
Completeness probability	$q(n)$	$q(n)q'(n)$
Soundness probability	$p(n)$	$(1 - p(n))p'(n) + p(n)$
Number of check-bits	$g(n)$	$g(n) + g'(g(n))$

As we see, all parameters are getting worse. Why? The composed verifier uses $g(n) + g'(g(n))$ check bits. The second term is small, but the first term, $g(n)$, is there, because although Q_r is input from the inner verifier's view point, it is in fact part of the composed proof. The situation would be different if V_r *checked* its input Q_r together with the associated proof $\wp_r$ instead of reading its entire input.

Babai, Fortnow, Levin, and Szegedy [12] have introduced a probabilistic verifier, which reads only a few bits from its input. The price of this efficiency is that the BFLS verifier can recognize a language only if it receives its input encoded. If now the outer verifier is *segmented*, i.e., the outer verifier reads k chunks of the proof, each of length $g(n)/k$, where k is a constant, we may use the BFLS verifier as the inner verifier if we replace the proof of the outer verifier with a one where each segment is encoded. (See Sect. 2.5.2 for an example to a segmented outer verifier.) In [12] it is shown:

Theorem 8 *For every language $L \in NP$ and every polynomial-time encoding function $E_n : \{0,1\}^n \rightarrow \{0,1\}^{\mathrm{poly}(n)}$ that defines a code with relative distance ϵ, and associated decoding function D, there is a verifier V^{BFLS} that uses at most $f'(n) = O_\epsilon(\log n)$ random bits, reads at most $e(n) = (\log n)^{O_\epsilon(1)}$ bits of the input and $g'(n) = (\log n)^{O_\epsilon(1)}$ bits of the proof, and has the following properties:*
1. *If the input of the verifier is of the form $z = E(x)$ for some $x \in L$, then the verifier accepts with probability at least* 1.
2. *If V^{BFLS} accepts an input z with probability at most $\frac{1}{2}$, then z decodes to some $x \in L$.*

Assume now that the outer verifier's proof is segmented and that the verifier reads at most k segments for every r, where k is a constant. The composed prover encodes the segments (each individually) with an efficient error-correcting code E'. This will be the core of the composed proof. If we now apply the BFLS verifier of Theorem 8 as the inner verifier, then the last line of our earlier table will change to

	Original proof	Composed proof
Number of check bits	$g(n)$	$e(g(n)) + g'(g(n))$

The above composition technique was employed in [5] and in [4]. A serious effort in these articles had to be made to create segmented outer verifiers. Dinur's proof of the PCP theorem does not use the above exact composition technique, although it also uses proof composition.

2.5.4 Randomized Graph Products

Graph products amplify gaps in sizes of independent sets [47] and chromatic numbers [71]. There are many different types of graph products, but the one that will work for us here is the *inclusive graph product*.

Definition 5 (Inclusive Graph Product) Let G_1 and G_2 be graphs with vertex sets V_1, resp., V_2, and edge sets E_1, resp. E_2. We define the inclusive graph product of $G_1 \times G_2$ on the vertex set $V_1 \times V_2$ such that (x_1, x_2) is connected with (y_1, y_2) if and only if x_1 is connected with y_1 in G_1 or x_2 is connected with y_2 in G_2.

Let $\alpha(G)$ be the maximum independent set size of G and $\omega(G)$ be the maximum clique size of G. Independent set sizes are multiplicative with respect to the inclusive graph product. For G^k, the k-wise inclusive graph product of G with itself, means

$$\alpha(G^k) = \alpha(G)^k$$
$$\omega\left(\overline{G}^k\right) = \omega(G)^k.$$

To obtain the behavior of powers of G with respect to the chromatic number, we first define the fractional chromatic number of a graph:

Definition 6 (Fractional Chromatic Number) Let G be an undirected graph and I be a set of G. The indicator function X_I of I is map from $V(G)$ to the reals which assigns 1 to the elements of I and 0 to the elements of $V(G) \setminus I$. Let $Ind(G)$ be the collection of all independent sets of G. We define the fractional chromatic number of G by

$$\chi_f(G) = \min \sum_{I \in Ind(G)} \lambda_I$$

subject to

$$\lambda_I \geq 0 \ \text{ for all } I \in Ind(I)$$
$$\sum_{I \in Ind(G)} \lambda_I X_I \geq X_{V(G)} \ \text{ coordinate-wise.}$$

Clearly,

$$\chi(G) \geq \chi_f(G) \geq \max\left(\frac{V(G)}{\alpha(G)}, \omega(G)\right). \tag{2}$$

With respect to powering we have

$$\chi(G^k) \leq \chi(G)^k \tag{3}$$
$$\chi_f(G^k) = \chi_f(G)^k. \tag{4}$$

Equation (4) is observed by Lovász [72]. We wish to amplify constant gaps in the independence number and the fractional chromatic number to polynomial gaps by setting $k = c \log n$, but the size of the resulting graph becomes super-polynomial. In order to overcome this problem, Berman and Schnitger [18] developed a randomized version of this graph product. Instead of taking G^k, they randomly, select an induced subgraph of it. Feige and Kilian [40] give the following lemma:

Lemma 5 *Vertex-induced subgraphs G' of G^k have the following properties:*

1. $\chi_f(G') \leq \chi_f^k(G)$.
2. *If $\alpha(G) \leq C$, then $\alpha(G') \leq k|V(G)|$ for nearly all subgraphs G' with $\frac{|V(G)|^k}{C^k}$ vertices.*
3. $\chi_f(G') \geq \frac{|V(G')|}{k|V(G)|}$, *for all G' satisfying $\alpha(G') \leq k|V(G)|$.*
4. $\alpha(G') \geq \frac{|V(G')|}{c^k}$ *with high probability over a fixed sized subgraph G', where $c = |V(G)|/\alpha(G)$.*

The lemma lets us amplify the PCP theorem in a simple way to prove polynomial (n^ϵ) gap for the clique approximation problem. The reduction is randomized, however.

2.6 The Long Code

The long code was invented by Bellare, Goldreich, and Sudan [14], and it has become a standard tool to prove exact non-approximability bounds for various CSP. It is also used to prove the non-approximability of Max Clique to within a factor of $|V(G)|^{1-\epsilon}$ in [55].

Let U be an fixed finite set of so-called "labels." The coordinates of the long code correspond to all possible functions $f : U \rightarrow \{0, 1\}$. When we encode an $\mathbf{x} \in U$, the coordinate corresponding to f takes the value $f(\mathbf{x})$:

$$long_U(\mathbf{x})_f = f(\mathbf{x}).$$

Since there are $2^{|U|}$ Boolean-valued functions on n inputs, the long code is a string of length $2^{|U|}$. Let A be a function that every $f : U \rightarrow \{0, 1\}$ associates an element $A_f \in \{0, 1\}$. Then A is a word of the long code if and only if there is an $\mathbf{x}$ such that for every f: $A_f = f(\mathbf{x})$.

Alternatively, we can think of the long code as the composition of two encodings: first we encode $\mathbf{x} \in U$ to unary($\mathbf{x}$) $\in \{0, 1\}^{|U|}$, where unary($\mathbf{x}$) is one only at a single position, indexed with $\mathbf{x}$, and zero everywhere else. Then we encode unary($\mathbf{x}$) with the *Hadamard encoding*. The latter contains all subset sums of the binary string it encodes, modulo two.

$\mathbf{x}$		unary($\mathbf{x}$)	Hadamard	
1	$\rightarrow$	0001	$\rightarrow$	0000000011111111
2	$\rightarrow$	0010	$\rightarrow$	0000111100001111
3	$\rightarrow$	0100	$\rightarrow$	0011001100110011
4	$\rightarrow$	1000	$\rightarrow$	0101010101010101

Sometimes we need a technique that is due to Bellare et. al. [14] called *folding*. If $A = long_U(\mathbf{x})$, then for every f, we have $A_{\neg f} = (\neg f)(\mathbf{x}) = \neg(f(\mathbf{x})) = \neg A_f$. Let $\mathbf{x}_0 \in U$ be a fixed (but otherwise arbitrarily chosen) element of U. For every f exactly one of $f(\mathbf{x}_0)$, $\neg f(\mathbf{x}_0)$ is zero. The folded long code $long'_U$ is obtained by puncturing $long_U$, keeping only those entries of every word that belong to index f with $f(\mathbf{x}_0) = 0$. This is exactly half of all entries. We, however, think of the omitted entries that they are implicitly present, and for any f with $f(\mathbf{x}_0) = 1$, we define

$$A_f = \neg A_{\neg f}. \tag{5}$$

Unfolding of A makes sense even when A is not in $long'_A$, but rather an arbitrary 0–1 vector over the index set $\{f : U \to \{0,1\} \mid f(\mathbf{x}_0) = 0\}$, if we "read" the entries that are not explicitly present by an application of Eq. (5). PCP constructs sometimes (or oftentimes) contain the folded variation.

2.6.1 Long Code Tests

Long code tests try to find out if in a prospective long code word A, there exists an $\mathbf{x}$ such that A is close to $long_U(\mathbf{x})$, by the means of checking only a few randomly chosen entries of A. The checking procedure should have the following properties:

1. If $A = long_U(\mathbf{x})$ for some $\mathbf{x}$, then the test accepts with probability q ($q = 1$ or it is close to one, depending on the setting).
2. If no such $\mathbf{x}$ is decodable from A and not even a short list of candidates, then the test accepts with probability at most p (in general, $p < q$, typically p is close to zero).

Perhaps the following arguments explain why we want to use such a wasteful encoding in our constructions:

- The long code is very efficiently testable, giving rise to efficient inner verifiers.
- From a long code word or from any word that is close to a long code word, we can efficiently decode the value that *any* Boolean-valued function takes over the encoded label.

Long code tests have either 1. perfect completeness or 2. imperfect completeness. Below we describe important examples to both ones. Our first example is a test with perfect completeness that checks $long'_U$ and is due to Dinur [33]. In the test we

Dinur's Folded Long Code Check [33].

1. Randomly select f_1, f_2, f_3, conditioned on $f_1 \vee f_2 \vee f_3 = 1$. Retrieve $b_1 = A_{f_1}$, $b_2 = A_{f_2}$, $b_3 = A_{f_3}$.
2. Accept if $b_1 \vee b_2 \vee b_3 = 1$.

Remark 1 Note that for any word A of the long code, if $f_1 \vee f_2 \vee f_3 = 1$, then $A_{f_1} \vee A_{f_2} \vee A_{f_1} = 1$.

Lemma 6 *For the above checking procedure, the following holds:*
1. *If $A = long'_U(\mathbf{x})$, then A is accepted with probability* 1.
2. *If $dist(A, long'_U) > \delta$, then A is rejected with probability $\Omega(\delta)$.*

The other test is a test with imperfect completeness, and it is due to Håstad [54]. It has historical significance too: the first exact non-approximability results were obtained using this code. We also assume that the input is folded, i.e., Eq. (5) holds. Below $\mathcal{F}$ denotes all functions of the form $f: U \to \{0, 1\}$.

Hastad's Folded Long Code Check [54]
1. Choose f_0 and f_1 from $\mathcal{F}$ with uniform probability.
2. Choose a function $\mu \in \mathcal{F}$ by setting $\mu(\mathbf{x}) = 1$ with probability $1 - \epsilon$ and $\mu(\mathbf{x}) = 0$ otherwise, independently for every $\mathbf{x} \in U$.
3. Set $f_2 = f_0 \oplus f_1 \oplus \mu$, i.e., define f_2 for each $\mathbf{x} \in U$ by $f_2(\mathbf{x}) = f_0(\mathbf{x}) \oplus f_1(\mathbf{x}) \oplus \mu(\mathbf{x})$.
4. Accept if $A_{f_0} \oplus A_{f_1} \oplus A_{f_2} = 0$.

Lemma 7 *If A is the table of a word in the folded long code, the verifier accepts with probability at least $1 - \epsilon$. If a folded string A passes the test with probability at least $\frac{1+p}{2}$, then there is a B which is the modulo two sum of at most $\epsilon^{-1} \log 1/p$ words of the long code, and A and B agree in at least $\frac{1+p}{2}$ fraction of the $2^{|U|}$ bit positions.*

Remark 2 Notice the difference between the soundness conditions of Dinur's test and Hastad's test. If Dinur's test succeeds with high probability, $\mathbf{x}$ is decodable from the code. In the case of Håstad, it is only list-decodable. Of course, in the case of Håstad "high probability" means $\frac{1+\text{tiny}}{2}$, while in the case of Dinur, it means $1-\text{tiny}$. Dinur's objective was not to prove sharp results, but to prove the PCP theorem. Her long code test is part of her alphabet reduction step.

When we prove sharp results, the outer verifier plays a role too. Some generic outer verifiers, like the label cover instance of Sect. 2.5.2, lead to a variety of optimal non-approximability results, but the inner verifier is slightly more involved than the verifier of the long code alone. The inner verifier for the label cover problem needs to check two long codes at the same time together with a relation between them. We do not describe the test here, which is very similar to Håstad's long code test, and also appears in [54]. If we replace the label cover problem with the so-called unique game problem (Sect. 4), we get sharper results, but this outer verifier is not proven to be NP-hard.

2.7 Constraint Satisfaction Problems

A general CSP instance is a set

$$\Phi = \{f_i(x_{i,1}, \ldots, x_{i,k})\}_{1 \le i \le m}$$

of k-variate functions acting on k-tuples of n variables, $x_1, \ldots, x_n$, each ranging in a domain D. Often $D = \{0, 1\}$, but in general it can be any fixed finite set. The goal is to find an assignment to the variables such that all or in the maximization version the maximum number of f_is is satisfied. Specific CSP have restrictions on the f_is, namely, they have to come from a constraint family Γ, which is a subset of k-variate functions on D (in the most general setting, "k-variate" has to be replaced with "at most k-variate"), where k is a fixed constant. CSP problems are in MAXSNP and therefore approximable within some constant factor. Here we discuss MAXkLIN, MAXkSAT and MAXkCSP. All of these problems are over the Boolean domain, i.e., $|D| = 2$.

Problem Name: MAXkLIN:
Instance: A set of linear equations over $\mathbf{F}_2$. $L = \{x_{i,1} \oplus x_{i,2} \oplus x_{i,3} = c_i \mid 1 \leq i \leq m\}$ such that for $1 \leq i \leq m$, $1 \leq j \leq 3$: $x_{i,j} \in \{x_i\}_{1 \leq i \leq n}$, where x_i are variables in $\mathbf{F}_2$ and c_i $(1 \leq m)$ are constants in $\mathbf{F}_2$.
Witness: An assignment to the variables x_i $(1 \leq i \leq n)$.
Objective: Maximize the number of equations set true in L.

Obviously, a random replacement on expectation satisfies half of the constraints. On the other hand, Håstad constructs a PCP reduction [55] which shows as follows:

Theorem 9 ([55]) *For any $\epsilon > 0$, $k \geq 3$ it is hard to approximate MAXkLIN to within a factor of* $2 - \epsilon$.

Håstad's reduction [54] briefly works as follows: The label cover problem from Sect. 2.5.2 is the outer verifier (one may view it as a starting point of the reduction). A random string of the outer verifier leads to checking an edge (x, y) of the label cover instance for the relation $\pi_{xy}(l(x)) = l(y)$, where $l(x)$ and $l(y)$ are the labels assigned to these nodes. The compound prover encodes each label with the folded long code (with $long'_{\Sigma_1}$ or with $long'_{\Sigma_2}$ depending on the node), and upon the outer verifier picking edge (x, y), the inner verifier checks the long codes associated with *both* nodes and *also* if the desired relation holds between the labels that these long codes encode. This seems like a lot of checking, but Håstad does the checking very economically, only with three query bits. It becomes important that π_{xy} is a projection. The verification process is very similar to the long code test of Håstad in Sect. 2.6, with a little extra twist, which we do not describe here. The composed verifier accepts with probability $1 - p$ for positive inputs and with probability at least $\frac{1-p}{2}$ for negative inputs, where p can be made arbitrarily small. Moreover, each checking criterion is a linear equation over $\mathbf{F}_2$ involving three variables. Theorem 9 follows.

Problem Name: MAXkSAT:
Instance: A set of disjuncts $\Phi = \{(t_{i,1} \vee t_{i,2} \vee t_{i,3}) \mid 1 \leq i \leq m\}$ such that for $1 \leq i \leq m$, $1 \leq j \leq 3$ $t_{i,j} \in \{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$, where x_i $1 \leq i \leq n$ are Boolean variables.
Witness: A Boolean assignment to the variables x_i $1 \leq i \leq n$.
Objective: Maximize the number of disjuncts set true in Φ.

Johnson [56] in 1974 showed that the MAXkSAT is approximable to within a factor of $\frac{1}{1-2^{-k}}$ in polynomial time as long as each clause contains exactly k literals. Trevisan, Sorkin, Sudan, and Williamson [95] show that MAX3SAT is approximable within 1.249 without any restriction, and Karloff and Zwick [57] show that it is approximable within a factor of $8/7$ for satisfiable instances. Here we prove the non-approximability of MAX3SAT.

Theorem 10 ([55]) *For any $\epsilon > 0$, it is hard to approximate MAX3SAT to within a factor of* $8/7 - \epsilon$.

Proof We reduce the problem to the non-approximability of MAX3LIN. Let L be an instance of MAX3LIN such that either at least $1 - \epsilon_1$ fraction of its equations are satisfied or at most $1/2 + \epsilon_2/2$ fraction of its equations are satisfied. We replace every equation $x \oplus y \oplus z$ of L with a set of four clauses $(x \wedge y \wedge \overline{z})$, $(x \wedge \overline{y} \wedge z)$, $(\overline{x} \wedge y \wedge z)$, and $(\overline{x} \wedge \overline{y} \wedge \overline{z})$. An assignment that satisfies the linear equation satisfies all the clauses, while an assignment that does not satisfies the linear equation satisfies three of the four clauses. The MAX3SAT instance Φ we construct for L is the multiset union of these replacements. It is easy to see that either $1 - \epsilon_1$ fraction of the clauses of Φ are satisfied or at most $7/8(1 + \epsilon_2)$ fraction. □

Håstad has proven the more general statement that MAXkSAT is not approximable within $\frac{1}{1-2^{-k}} - \epsilon$ for any k and $\epsilon > 0$, even if every clause consists of exactly k literals.

Theorem 10 is proven via a specific *gadget reduction*, i.e., where each constraint of a CSP is replaced with a set of constraints from another CSP, called *gadget*. These in general may contain auxiliary variables. Trevisan, Sorkin, Sudan, and Williamson [97] use linear programs to systematically construct gadgets that give optimal gap-CSPs starting from gap-E3LIN2. For a large class of CSPs, they prove that optimal gadgets exist. They arrive at these gadgets via linear programming.

Problem Name: MAXkCSP:
Instance: A set $S = \{f_i \mid 1 \le i \le n\}$ of Boolean expressions (constraints), each depending on at most k of the Boolean variables x_i $1 \le i \le k$.
Witness: A Boolean assignment to the variables x_i $(1 \le i \le n)$.
Objective: Maximize the number of expressions set true in S.

The best known algorithm for the MAXkCSP problem has an approximation ratio 2^{k-1} [94]. Samorodnitsky and Trevisan [89] showed that for any $\delta > 0$ and any NP-complete language L, there is a PCP with amortized query complexity $1 + \delta$ (!). Their proof leads to the following:

Theorem 11 ([89]) *For every k the MAXkCSP problem is NP-hard to approximate to within* $2^{k-O(\sqrt{k})}$.

2.8 The Max Clique

Instance: Undirected graph $G = (V, E)$.
Witness: A clique $C \subseteq V$ in G: $(x, y \in C) \wedge (x \neq y) \rightarrow (x, y) \in E(G)$
Objective: $\omega(G) = \max\{|C| : C \text{ is a clique in } G\}$.

Boppana and Halldórsson [20] in 1992 proved that the maximum clique problem can be approximated within a factor of $O(|V|/(\log|V|)^2)$. The hope for a big improvement faded away as non-approximability results arose. Feige, Goldwasser, Lovász, Safra and Szegedy [38] in 1991 made the following connection between clique approximation and proof checking:

Theorem 12 (FGLSS) *Let $NP \subseteq PCP_{q,p}(f, g)$. Then if $MaxClique$ of a graph of size $h(n) = 2^{f(n)+g(n)}$ can be approximated within a factor better than $q(n)/p(n)$ in time polynomial in $h(n)$, then $NP \subseteq DTIME(poly(h(n)))$.*

Proof Let L be an NP-complete language. Let $V(r, P, x)$ be a probabilistic verifier that recognizes L using $f(n)$ query bits, and $g(n)$ random bits, and which accepts for at least $q(n)2^{f(n)}$ choices of r for *some* P, if $x \in L$ and accepts for at most $p(n)2^{f(n)}$ choices of r for *every* P, if $x \notin L$.

Consider the FGLSS graph $G_V(x)$ defined in Sect. 2.2. Conditions 1–3 of Lemma 3 imply the theorem, since if we could approximate the Max Clique of $G_V(x)$ within a factor better than $q(n)/p(n)$ in time $poly(h(n))$, then we would be able to use this algorithm to tell whether $x \in L$ or not. □

Let $f(n) = \log n \log\log n$. Feige et al. show that $NP \subseteq PCP_{1,1/2}(f, f)$. This argument used the scaled-down version of the two-prover interactive protocol by Babai Fortnow and Lund [11] and provided the first rudimentary PCP construction. If we apply $\log^k n$ independent naive repetitions on the above verifier (see Sect. 2.5.1) we obtain

$$NP \subseteq PCP_{1,1/2^{\log^k n}}(f(n)\log^k n, f(n)\log^k n).$$

By Theorem 12 this implies that for every fixed $\epsilon > 0$, Max Clique cannot be approximated within a factor of $2^{\log^{1-\epsilon}|V|}$ unless NP has quasi-polynomial-time algorithms.

The next important step was made by Arora and Safra [5] who showed the NP-hardness of Max Clique up to an approximation factor $2^{\sqrt{\log|V|}}$. This result was the first to show the non-approximability of Max Clique under the natural $P \neq NP$ assumption and provided the first PCP that characterized NP.

Arora, Lund, Motwani, Sudan, and Szegedy [4] proved that $NP = PCP_{1,1/2}(\log n, O(1))$ (see Theorem 1). An amplification method [25] based on the "expander walk" technique of Ajtai, Komlós, and Szemeredi [1] yields that $NP = PCP_{1,1/n}(\log n, \log n)$. Then Theorem 12 gives that there is an ϵ that it is NP-hard to approximate $MaxClique$ up to a factor of $|V|^\epsilon$. Alon, Feige, Wigderson, and Zuckerman [3] showed how to obtain the same result starting with FGLSS graph constructed from Theorem 1 by derandomizing the graph product of Berman and Schnitger [18].

The constant ϵ was made explicit by Bellare, Goldwasser, Lund, and Russel [15]. They also slightly improved on the proof methods of [4] and coupled it with a new amplification technique of [18, 98] to show that Max Clique is hard to approximate within a factor of $|V|^{1/25}$ unless NP is contained in $co - R\tilde{P}$. Here $\tilde{P}$ stands for the quasi-polynomial time.

Notice that in the last result the hardness condition is different than before, because it involves the randomized class, $co{-}R\tilde{P}$. BGLR replace the condition $NP \subseteq PCP_{q(n),p(n)}(f(n), g(n))$ of Theorem 12 with $NP \le_R PCP_{q(n),p(n)}(f(n), g(n))$, where $\le_R$ stands for "randomly reducible." Although this weakens the hardness condition, it allows better amplification methods and thus better parameters.

Feige and Kilian [39] have observed that Theorem 12 can be replaced with the following lemma (for the definitions see Sect. 2.4):

Lemma 8 **[39]** *Let $NP \subseteq FPCP_{q,p}(f, free)$. Then if Max Clique of a graph of size $h(n) = 2^{f(n)+free(n)}$ can be approximated within a factor better than $q(n)/p(n)$ in time polynomial in $h(n)$, then $NP \subseteq DTIME(poly(h(n)))$.*

From the above lemma they showed that Max Clique cannot be approximated to within a factor of $|V|^{1/15-\epsilon}$ unless $NP = coRP$. Bellare and Sudan in [16] introduced the amortized free bit complexity, $\overline{free}$ (see Sect. 2.4), and notice that the best available amplification techniques give that

Lemma 9 *If there is a PCP which uses $O(\log n)$ randomness and has amortized free bit complexity $\overline{free}$, then Max Clique cannot be approximated within a factor better than $n^{\frac{1}{1+\overline{free}}}$ unless $NP \subseteq coR\tilde{P}$.*

They prove that Max Clique cannot be approximated to within a factor better than $|V|^{1/4}$ unless $NP \subseteq coR\tilde{P}$. The next major step was made by Bellare, Goldreich, and Sudan [14], who discovered the long code (see Sect. 2.6), and building it into their PCP immediately could reduce the free bit complexity. They prove the non-approximability of Max Clique to within a factor better than $|V|^{1/3}$ unless $NP \subseteq coRP$.

Although the constant in the exponent gradually improved, it seemed that theoretical limitations bar the improvement of the non-approximability exponent beyond $1/2$. It has turned out that the limitations could be overcome by dropping the perfect completeness property of the PCP. Relying on a new test for the long code of Bellare et. al., Håstad was able to show the following:

Theorem 13 **[54]** *Max Clique cannot be approximated to within a factor of $|V|^{1-\epsilon}$ for any $\epsilon > 0$ unless $NP \subseteq ZPP$.*

Arora=A, Babai=B, Bellare = Be, Feige=Fe,Goldwasser=G,
Goldreich = Go, Håstad =H, Kilian=K, Khot = Kh,
Lovász=Lo, Lund=Lu, Motwani=M, Russel = R,
Safra=Sa, Sudan=Su, Szegedy=Sz, Zuckerman = Z
Engebretsen=En, Holmerin=Ho

Due to	Factor	Assumption
FeGLoSaSz	$2^{\log^{1-\epsilon}\|V\|}$ for any $\epsilon > 0$	$NP \not\subseteq \tilde{P}$
ASa	$2^{\sqrt{\log\|V\|}}$	$NP \neq P$
ALuMSuSz	$\|V\|^{\epsilon}$ for some ϵ	$NP \neq P$
BeGLuR	$\|V\|^{1/30}$	$NP \neq coRP$
BeGLuR	$\|V\|^{1/25}$	$NP \not\subseteq coR\tilde{P}$
FeK	$\|V\|^{1/15}$	$NP \neq coRP$
BeSu	$\|V\|^{1/6}$	$NP \neq P$
BeSu	$\|V\|^{1/4}$	$NP \not\subseteq coR\tilde{P}$
BeGoSu	$\|V\|^{1/4}$	$NP \neq P$
BeGoSu	$\|V\|^{1/3}$	$NP \neq coRP$
H	$\|V\|^{1-\epsilon}$ for any $\epsilon > 0$	$NP \not\subseteq ZPP$
Kh	$\|V\|/2^{\log^{1-\epsilon}\|V\|}$ for any $\epsilon > 0$	$NP \not\subseteq ZPP$
EnHo	$\|V\|^{1-O(\sqrt{\log\log n})}$	$NP \subseteq ZP(2^{O(\log n(\log\log n)^{3/2})})$
Z	$\|V\|^{1-\epsilon}$ for any $\epsilon > 0$	$NP \neq P$
Z	$\|V\|/2^{\log^{1-\epsilon}\|V\|}$ for any $\epsilon > 0$	$\tilde{NP} \neq \tilde{P}$

Fig. 4 A summary table on the history of clique non-approximability

In the proof of this theorem, Håstad uses the same probabilistically checkable proof construction as he does for CSP, but the verifier works differently, and the analysis is different. He is able to achieve amortized free bit complexity ϵ for any $\epsilon > 0$.

Sudan and Trevisan [91] and Samorodnitsky and Trevisan [89] simplified the involved analysis of Håstad and constructed a PCP which has the additional benefit of having amortized query bit complexity $1 + \epsilon$. Building on [91] and [89] Lars Engebretsen and Jonas Holmerin [36] showed that unless $NP \subseteq ZPTIME(2^{O(\log n(\log\log n)^{3/2})})$, Max Clique cannot be approximated to within a factor of $|V|^{1-O(\sqrt{\log\log n})}$. Under the same assumption, Khot [60] was able to increase the non-approximability ratio to $|V|/2^{\log^{1-\epsilon}|V|}$. Using new construction of dispersers, Zuckerman de-randomized the PCP constructions of Håstad and Khot [99]. Fig. 4 summarizes developments on the Clique non-approximation problem.

2.9 The Chromatic Number

Instance: Undirected graph $G = (V, E)$.
Witness: An integer k and a k-coloring F of $V(G)$ with the property that $F : V(G) \to [1, k]$ and for every $(a, b) \in E(G) : F(a) \neq F(b)$.
Objective: $\chi(G) = \min k$ such that a k-coloring F of $V(G)$ exists.

The best-known lower bound for approximating the chromatic number was obtained by Halldórsson in 1993 [52]. Given a graph G on n nodes, he finds a coloring of G in polynomial time with at most

$$\chi(G) \times O\left(n\frac{(\log\log n)^2}{\log^3 n}\right)$$

colors. From the other direction, an early result of Garey and Johnson [46] shows that $\chi(G)$ is NP-hard to approximate to within any constant less than 2. In 1992 Lund and Yannakakis found a reduction from approximating the Max Clique of the FGLSS graph of the ALMSS verifier to the problem of approximating the chromatic number.

Theorem 14 (Lund and Yannakakis [74]) *There is an ϵ such that it is NP-hard to approximate $\chi(G)$ within a factor of $|V(G)|^{\epsilon}$.*

The proof of Lund and Yannakakis works in three steps:

1. Construct the FGLSS graph $G_V(x)$ for the ALMSS verifier V of an NP-complete language L. Observe that $\overline{G}_V(x)$ has bounded degree and:
   a. If $x \in L$ then $\alpha(\overline{G}_V(x)) = R$ (for some $R = n^c$).
   b. If $x \notin L$ then $\alpha(\overline{G}_V(x)) \leq R(1 - \epsilon)$.
2. Use the randomized graph products of Berman and Schnitger [18] and Blum [19] to obtain a graph G_1 from $\overline{G}_V(x)$, which has maximum independence number n^{ϵ_1} for positive instances and maximum independence number n^{ϵ_2} for negative instances, for some $\epsilon_1 > \epsilon_2 > 0$.
3. Apply another transformation which turns G_1 into a graph G_2 such that $\chi(G_2) \geq n^{\epsilon_3}$ for positive instances and $\chi(G_2) \leq n^{\epsilon_4}$ for negative instances, for some $\epsilon_3 > \epsilon_4 > 0$.

Improving upon this result and its subsequent sharpening [16, 58], Fürer [44] gives a randomized reduction which shows that if $\omega(G)$ cannot be approximated to within $|V(G)|^{\frac{1}{free+1}}$, then $\chi(G)$ cannot be approximated to within $|V(G)|^{\delta}$, where $\delta = \min\left(\frac{1}{2}, \frac{1}{2free+1}\right) - o(1)$. His reduction assumes the FGLSS graph structure. Feige and Kilian [40] have taken a different route and building on the Max Clique result of Håstad [54] show:

Theorem 15 [40] *Unless $NP \subseteq ZPP$, it is hard to approximate $\chi(G)$ to within $|V(G)|^{1-\epsilon}$ for any constant $\epsilon > 0$. Here ZPP denotes the class of languages that are* *solvable with a randomized algorithm that makes no error and on expectation runs in polynomial time.*

The proof of this theorem gives more, namely, that it is NP-hard (under randomized reductions) to distinguish between graphs with $\alpha(G) \leq |V(G)|^{\epsilon}$ and graphs with $\chi(G) \leq |V(G)|^{\epsilon}$. To understand their proof scheme, we need to introduce a new parameter for a PCP which requires the notion of fractional chromatic number of Sect. 2.5.4 (Definition 6).

Definition 7 (Covering Parameter) The covering parameter of a PCP with verifier V is

$$\rho = \min_{x \in L} \frac{1}{\chi_f(\overline{G}_V(x))}. \tag{6}$$

Here $G_V(x)$ is the FGLSS graph for V and x (see Definition 4).

Since for every graph G we have $\chi_f(G) \geq \omega(G)$, the covering parameter of a PCP for input x is at most $2^{-free(|x|)}$. The notion of covering parameter first appears in [40] in the context of RPCPs. RPCPs are PCPs such that for every $x \in L$, there is a probability distribution (possibly different for different inputs) on all the proofs. For a fixed $x \in L$, this distribution gives rise to a probability distribution on $Ind(\overline{G}_V(x))$, which in turn can be used to give a lower bound on the covering parameter of the PCP. This lower bound is what Feige and Kilian call the covering parameter of the RPCP. Whether we talk about RPCPs or immediately define the covering parameter of a PCP is a matter of taste. RPCPs were introduced because their definition suggests an elegant connection to zero knowledge proof systems.

Feige and Kilian do their crucial transformation directly on the FGLSS graph of the PCP theorem, and *then* they amplify it with the randomized graph product. This is the reverse of how Lund and Yannakakis do it. Here is what Feige and Kilian rely on.

Assume that for an NP-complete language L, there is a polynomial-time transformation that sends a word x into a graph $G(x)$ such that for some constants $0 < c_2 < c_1 < 1$:

$$\text{If } x \in L \text{ then } \quad \chi_f(G(x)) \leq 1/c_1. \tag{7}$$

$$\text{If } x \notin L \text{ then } \quad \alpha(G(x)) \leq c_2 |V(G(x))|. \tag{8}$$

Then, if we apply the randomized graph product construction of [18] on $G(x)$ with $k = c_3 \log_{c_2^{-1}} |V(G(x))|$, where $c_3 > 1$ is a constant, and select a random subgraph G' of G^k with size c_2^{-k}, then from Lemma 5 of Sect. 2.5.4,

$$\chi_f(G') \leq |V(G')|^{\frac{\log c_1}{\log c_2}} \qquad \text{if } x \in L, \tag{9}$$

$$\alpha(G') \leq |V(G')|^{\epsilon} \qquad \text{if } x \notin L, \tag{10}$$

where ϵ can be made arbitrarily small if c_3 is large enough. We immediately obtain the following:

Lemma 10 *Assume that for an NP-complete language L there is a polynomial-time transformation that sends a word x into a graph $G(x)$ such that for some constants $0 < c_2 < c_1 < 1$ conditions (7) and (8) hold. Then for every $\epsilon > 0$, we can construct a graph G' in randomized polynomial time such that*

$$\chi(G') \leq |V(G')|^{\frac{\log c_1}{\log c_2}+\epsilon} \text{ if } x \in L, \tag{11}$$

$$\chi(G') \geq \quad |V(G')|^{1-\epsilon} \quad \text{if } x \notin L. \tag{12}$$

Indeed the Inequality (11) follows from Inequality (10) and Inequality (12) is implied by Inequality (9) and by the following result of Lovász [72]:

$$\chi(G) \leq \chi_f(G)\log(1+\alpha(G)). \tag{13}$$

If we combine Lemma 10 with Definition 7 of the covering parameter, we obtain the following:

Lemma 11 *Suppose that one can generate a PCP for NP with $f(n)$ random bits, soundness probability $p = p(n)$, average free bit complexity $f_{av} = free_{av}(n)$, and $\rho = \rho(n)$. Assume that p, f_{av}, and ρ are constants and that the size of the FGLSS graph, $2^{f(n)+f_{av}}$, is polynomially bounded as n grows to infinity. Then it is hard to approximate $\chi(G)$ to within $|V(G)|^{\frac{f_{av}-\log p+\log\rho}{f_{av}-\log p}-\epsilon}$, where ϵ is an arbitrarily small positive constant, assuming $NP \not\subseteq ZPP$.*

Starting from the construction of Håstad [55], Feige and Kilian show the existence of PCPs for an NP-complete problem such that $\log\rho/(f_{av} - \log p)$ is arbitrarily small. Theorem 15 is implied now by Lemma 11.

We can also use Lemma 10 to derive the Lund Yannakakis result from the non-approximability result from the MAXSNP-hardness of the MAX-3-coloring problem proved by Petrank [82]. This example was given by Feige and Kilian, and it is so easy that we can give their entire proof here:

Proof (Theorem 14) Let H be the graph of Petrank's construction such that it has m edges and either it has a valid three-coloring or every three-coloring of its vertices miscolor at least qm edges, where $q > 0$ is some fixed constant. From H we construct a graph G, which has $6m$ vertices of the form (e, c), where e ranges over the m edges of H and c ranges over the 6 valid 3-colorings of the two endpoints of an edge. Two vertices (e_1, c_1) and (e_2, c_2) are connected by an edge if e_1 and e_2 intersect at a vertex of H and c_1 and c_2 disagree on the coloring of this vertex. A valid 3-coloring C of H induces an independent set of size m in G. Let π be a permutation of the three colors. As π ranges over its 6 possibilities, the composition of C with π induces 6 independent sets in G of size m. Furthermore, these independent sets are disjoint and cover $V(G)$. Hence, G has a chromatic number of 6. If H is not 3-colorable (and hence qm of its edges are miscolored) then the largest independent set of G is of size at most $m(1-q)$. If we apply Lemma 10 for G with $c_1 = 1/6$, $c_2 = (1-q)/6$, we get Theorem 14. $\square$

The above non-approximability of $\chi(G)$ was derandomized by Zuckerman [99] to get the same non-approximability ratio conditioned on $P \neq NP$.

Finally we mention another type of non-approximability result for $\chi(G)$. S. Khanna, N. Linial, and S. Safra [58] have shown that it is NP-hard to tell apart 3-chromatic graphs from 5 chromatic graphs. But the following question is open and would be very interesting to resolve:

Problem 1 Prove that for any fixed $k > 5$, it is NP-hard to tell apart a k-chromatic graph from a three-chromatic graph.

Remark 3 Recently, Dinur, Mossel, and Regev have shown [35]: For any two fixed integers $Q > q > 2$, it is hard to decide whether for an input graph G we have $\chi(G) \leq q$ or $\chi(G) \geq Q$. The hardness is under the 2-1 conjecture, if $q \geq 4$, and under a similar, but slightly more complicated "fish shaped" variant of this conjecture, if $q = 3$ (see the definition of 2-1 conjecture in Sect. 4).

2.10 Set Cover

Instance: A collection $F = \{S_1, \ldots, S_s\}$ of subsets of $S = \{1, \ldots, n\}$.
Witness: A subcollection F' of F such that $\cup_{S_i \in F'} S_i = S$.
Objective: $\nu(F) = \min |F'|$.

In 1974 D. Johnson [56] showed that the greedy algorithm finds a collection F' for the problem such that $|F'|$ is to within $\log n$ factor optimal (here the log is based on $e = 2.71..$). Chvatal [24] extended this algorithm to the weighted version, and Lovász [72] has studied a linear programming relaxation with logarithmic (in terms of the hypergraph degree) integrality gap.

The first hardness result is that of Lund and Yannakakis [74]. They show using a construction coming from the PCP theory that set cover cannot be approximated to within a factor of $\frac{\log n}{4}$ unless $NP \subseteq TIME(n^{\mathrm{polylog} n})$ and within a factor of $\frac{\log n}{2}$ unless $NP \subseteq ZIME(n^{\mathrm{polylog} n})$. Here $ZTIME(t)$ denotes the class of problems for which there is a probabilistic algorithm that makes no error and runs in expected time t.

Subsequent works went in two different directions. The best results up to date that represent these directions are that of Feige [37] which proves that set cover cannot be approximated efficiently to within a factor of $(1 - o(1)) \log n$ unless $NP \subseteq TIME(n^{O(\log \log n)})$ and that of Raz and Safra [87] which proves that set cover is NP-hard to approximate within a factor of $\epsilon \log n$ for some fixed $\epsilon > 0$. Until now there is no result which would achieve both optimal non-approximability ratio and hardness condition $P \neq NP$.

A predecessor of [37] was the result of Bellare et.al. [15], which proved that the set cover cannot be approximated within any constant ratio unless $P = NP$ and that it cannot be approximated within a factor of $\frac{\log n}{4}$ unless $NP \subseteq TIME(n^{O(\log \log n)})$. Arora and Sudan [6] followup on [87] and achieve the same bound using elegant and difficult algebraic techniques such as Hilbert's Nullstellensatz.

We cannot give here the technically very involved result of Raz and Safra, but we present a complete proof of Feige's result. The proof is a modification of a simplified argument due to Håstad (A Rewriting of Feige's Proof for the Setcover, Unpublished). At many places we use his original wording. We ought to remark that simplification is formal to a large extent: the key components of the proof are the same as those in [37].

Theorem 16 **[37]** *The set cover cannot be approximated efficiently to within a factor of* $(1 - o(1)) \log n$ *unless* $NP \subseteq TIME(n^{O(\log \log n)})$. *The logarithm is* $e = 2.71\ldots$ *based.*

Proof A partition system $B(m, p, k, d)$ [37] is a system of p partitions, $(p_i)_{i=1}^{p}$, on a basis set B with $|B| = m$ such that:

1. Each p_i $(1 \leq i \leq p)$ consists of k parts. The parts of p_i are denoted $p_{i,j}$ $(1 \leq j \leq k)$.
2. No collection $(p_{i_l,j_l})_{l=1}^{d}$ of d partition segments with all i_l distinct covers the universe.

In other words if we only use sets from different partitions to cover, we need at least d sets, while a small cover is given by the k sets of one p_i. □

Lemma 12 (Feige [37]) *For any* $c > 0$*, there exists a* $B(m, p, k, d)$ *partition system with* $p \leq (\log m)^c$*,* k *an arbitrary constant, and* $d = (1 - f(k))k \ln m$*, where* $f(k)$ *tends to 0 when* k *tends to infinity and* m *is large enough compared to* k *and* c*.*

Feige shows that a random partition system satisfies Properties 1–2 with high probability. Since $d = O(\log m)$, we can quickly check if the Properties 1–2 hold. Deterministic constructions are also known. Next we describe the PCP reduction to the set-cover problem.

Our starting point is the probabilistic verifier V_{3SAT}^{ℓ} of Lemma 4 in Sect. 2.5.2 for an NP-complete language L with the choice of $\ell = c_0 \log \log n$, where we shall choose $c_0 > 0$ to be large enough. Recall that this verifier is associated with a gap-3SAT instance ϕ_x with N Boolean variables. Note that $|x| = n$ is in polynomial relationship with N. On the other hand, the instance of set cover we construct will have slightly super-polynomial size in n.

The verifier sends a subset U of variables with $|U| = \ell$ to the first prover who answers with an assignment σ_U to these variables. Independently, he sends a W companion of U to the second prover, i.e., a set of ℓ clauses containing these variables. The second prover answers with the evaluation of the 3ℓ variables in W. The verifier rejects if not all clauses are satisfied in W or if σ_U is not equal to the projection $\pi_U(\sigma_W)$, i.e., the assignment that σ_W gives to U. What is important for our purposes is that if $x \notin L$, the verifier accepts with probability at most c_1^ℓ for some fixed $0 < c_1 < 1$. The structure of ϕ_x determines all companion pairs U, W and all the projections π_U, and we can compute these in $DTIME(n^{O(\ell)})$ from x. From now on, when we talk about a U, W pair, we always mean a pair, where W is a companion of U.

Let $k > 0$ be a constant such that $f(k)$ of Lemma 12 is less than ϵ. For each U and each k-tuple $W_1, W_2 \ldots W_k = \vec{W}$ of possible W companions of this U, we construct a separate partition system on a new set of m points. We have N^ℓ different U and, given the choice of U, each W_i can be chosen in 5^ℓ ways. We thus have $R = N^\ell 5^{k\ell}$ partition systems, and with the choice of $m = R^{\frac{1}{\epsilon}}$, the total size of the universe is $R^{1+\frac{1}{\epsilon}} = m^{1+\epsilon}$. We set the further parameters of the partition systems as $p = 2^\ell, k$, and $d = (1-f(k))k \ln m$. Since $p = 2^{c_0 \log\log n} = (\log n)^{c_0}$ and $\log m > \log n$, the conditions of Feige's lemma hold, and the existence of such partition systems is guaranteed. Notice that $m = n^{O(\log\log n)}$ for a fixed ϵ. A particular set in one of the partition systems has an index given by $U, \vec{W}, \alpha \in \{0,1\}^\ell$ and $i \in [k]$, and we denote it by $S_{U,\vec{W},\alpha,i}$.

The sets in our set-cover instance are indexed by W and β where $\beta \in \{0,1\}^W$ should be thought of as an answer from the second prover on the question W. We denote it by $T_{W,\beta}$ and it is a union of $S_{U,\vec{W},\alpha,i}$ which satisfy

$$(W_i = W) \wedge (\pi_U(\beta) = \alpha),$$

where we of course assume that β satisfies the clauses used to construct W.

Let $Q = (5N/3)^\ell$ be the number of different W. We first have the simple lemma telling us what happens when $x \in L$.

Lemma 13 *If $x \in L$, the associated set system has a cover of size Q.*

Proof We cover the universe with $\{T_{W,\sigma_W}\}_W$, where σ_W is the answer of the second prover to question W. Since the answers are consistent with those of the first prover, for each U and $\vec{W}$, there is some σ_U so that the chosen sets contain $S_{U,\vec{W},\sigma_U,i}$ for all i. This σ_U is simply the answer of the first prover to question U. By definition, the system $\{S_{U,\vec{W},\sigma_U,i}\}_{1\le i\le k}$ covers the m points of the associated $U, \vec{W}$ pair. □

Thus, we just need to prove the lemma below (and choose c_0).

Lemma 14 *If $x \notin L$, the associated set system has a no cover of size $dQ/(k(1+\epsilon))$.*

Proof Suppose we have a cover of size rQ. For each W there is a set A_W of assignments β on W such that $T_{W,\beta}$ belong to the cover. By assumption

$$\sum_W |A_W| = rQ. \tag{14}$$

We note that none of the A_W is empty since we need to cover the partition systems with $W_i = W$ for all i. Let us call a W *good* if $|A_W| \le r(1+\epsilon)$. We claim that for appropriate choice of c_0 either $r > \log m$ or there is $U, \vec{W}$ such that

1. $\pi_U(A_{W_i})$ $(1 \le i \le k)$ are pairwise disjoint.
2. W_i $(1 \le i \le k)$ are good.

Let us recall a simple lemma in set theory:

Lemma 15 *Let $A_1, \dots, A_q$ be sets of size at most s with the property that no k of them are pairwise disjoint. Then there is an element which is contained in at least $\frac{1}{(k-1)s}$ fraction of the sets.*

Proof Without loss of generality we may assume that $A_1, \dots, A_l$ are pairwise disjoint, but for all $j > l$, the set A_j intersects $A = \cup_{i=1}^{l} A_i$. Note that by our assumption $l \le k-1$; hence, $|A| \le (k-1)s$. Since every A_i intersects A, there must be an element of A which is contained at least $\frac{1}{(k-1)s} \le \frac{1}{|A|}$ of the sets. □

Let

$$S_U = \{\pi_U(A_W) \mid \ W \text{ is a companion of } U, \text{ and } W \text{ is good}\}$$

be a set system on the assignments for U, and let σ_U be an assignment which is contained in as large fraction of elements of S_U as possible (If S_U is empty then σ_U is arbitrary). Consider the strategy of the two provers when to question U the first prover answers σ_U, and to question W the second prover answers with a random $\sigma_W \in A_W$. (So the strategy of the second prover is randomized. An averaging argument shows that there is an equally good deterministic strategy.)

If for a fixed U, there is no $\vec{W}$ such that both Conditions 1 and 2 hold, then among the good companions of U, there are no $W_1, \dots, W_k$ such that $\pi_U(A_{W_i})$ $(1 \le i \le k)$ are pairwise disjoint. By applying Lemma 15 to S_U, we obtain that there is an assignment to U which occurs in at least $\frac{1}{(k-1)r(1+\epsilon)}$ fraction of the sets in S_U, and if fact σ_U is such. In what follows we assume that Conditions 1 and 2 do not hold for any $U, \vec{W}$.

By Eq. (14) the expected value of $|A_W|$ for a random W is r, so for at least $1-(1+\epsilon)^{-1}$ fraction of Ws, we have $|A_W| \le r(1+\epsilon)$. The verifier chooses such a W with probability at least $1-(1+\epsilon)^{-1}$. Conditioned on the intersection of this event (i.e., that W is good) and the event that the verifier picks some fixed U, by our earlier remark, the verifier chooses a W such that $\pi_U(A_W)$ contains σ_U with probability at least $\frac{1}{(k-1)r(1+\epsilon)}$. Hence, the probability of the event that the verifier picks a U, W pair such that W is good and $\sigma_U \in \pi_U(A_W)$ is at least

$$\frac{1-(1+\epsilon)^{-1}}{(k-1)r(1+\epsilon)}. \tag{15}$$

In the case of this event, there is a $\frac{1}{|A_W|} \geq \frac{1}{r(1+\epsilon)}$ chance over $\sigma_W \in A_W$ that $\pi_U(\sigma_W) = \sigma_U$. Combining this with (15), we get that $V^{\ell}_{3\mathrm{SAT}}$ accepts the described strategy of the two provers with probability at least

$$\frac{1-(1+\epsilon)^{-1}}{(k-1)r(1+\epsilon)} \frac{1}{r(1+\epsilon)} = \Omega(r^{-2}). \tag{16}$$

Either $r > \log m$ or since $\log m = O(c_0 \log n \log\log n)$, we can choose a large enough c_0 that depends on ϵ, k, c_1, and $\log_n N = O(1)$, such that $2^{-c_1\ell} = 2^{-c_1 c_0 \log\log n}$ becomes smaller than Expression (16), and we arrive at a contradiction.

It follows from the contradiction that there exist an U and $\vec{W}$ such that Conditions 1 and 2 hold. Fix this choice of U and $\vec{W}$ and consider how the elements from the corresponding partition system are covered. Since $|W_i| \leq r(1+\epsilon)$ for $1 \leq i \leq k$, the cover system has at most $kr(1+\epsilon)$ sets. By the disjointness of $\pi_U(A_{W_i})$s, these sets all come from distinct partitions. Using the property of the cover system, we conclude that $kr(1+\epsilon) > d$.

In the case $r > \log m$, since we have $d \leq (1 - f(k))kr$, we conclude the same. We are done, since the cover size $|rQ| > dQ/k(1+\epsilon)$, as claimed. □

Since the universe for the set-cover instance has total size $t = m^{1+\frac{1}{\epsilon}}$, we get that the quotient of the minimum cover sizes when $x \notin L$ versus when $x \in L$ is

$$\frac{1-f(k)}{1+\epsilon} \log m = \frac{1-f(k)}{(1+\epsilon)^2} \log t,$$

which tends to $\log t$ when $k \to \infty$ and $\epsilon \to 0$.

2.11 Some More Non-approximability Results

The label cover problem from Sect. 2.5.2 and its variants have served as the starting point for several known optimal non-approximability results assuming $P \neq NP$. The table below describes some further known approximation hardness results based on the NP-hardness of the label cover. Let γ_p be the p^{th} moment of the Gaussian random variable with expectation 0 and variance 1.

Problem	Known approx.	Inapprox.	Ref.
Max Cut	$\approx 1.1389.$	$\frac{17}{16} - \epsilon$	[48, 55]
L_p Groth. Problem	γ_p^2	$\gamma_p^2 - \epsilon$	[51, 67]
L_p Subspace approx.	γ_p	$\gamma_p - \epsilon$	[32, 51]

The L_p Grothendieck problem is a maximization problem where the input is a symmetric $n \times n$ matrix $A = (a_{ij})$ with zero diagonal entries. The objective is to maximize the quadratic multilinear function $\sum_{i,j\in[n]} a_{ij}x_i x_j$ subject to $\sum_{i=1}^{n}|x_i|^p \leq 1$ for $x_i \in \mathbb{R}$. The L_p subspace approximation problem is a generalization of the well-known *least squares regression* problem. The input to the problem is a set of m points $a_1,\ldots,a_m$ in $\mathbb{R}^n$. The objective of the problem is to find a k-dimensional subspace H such that $\sum_{i=1}^{m} dist(H,a_i)^p$ is minimized. Here $dist(H,a_i)$ is the ℓ_2 distance from a_i to H. The non-approximability results for the L_p Grothendieck problem and subspace approximation problem are from a recent manuscript by Guruswami, Raghavendra, Saket, and Wu [51] who use a stronger, *smooth* variant of the label cover from [42].

3 A Short Proof of the PCP Theorem

Before [33] and [34], several different proofs have been made for the PCP theorem, but their rough structure did not differ much. In 2005, Dinur published a combinatorial proof for the PCP theorem. One feature of Dinur's proof is that it can be fully explained without mentioning PCPs and talking only about gap-CSPs. The only hint to the old PCP ideas is the use of the long code.

Definition 8 An instance of a $[k,\Sigma]$CSP problem, where Σ is a constant size alphabet, consists of a set $x_1,\ldots,x_n$ of variables that take their values from Σ and a set of m constraints, where each constraint is a k-ary relation for a subset of k variables. The instance is satisfiable if there is an assignment to the variables which satisfies all constraints. We identify $[k,\Sigma]$CSP with the language of satisfiable instances.

The gap version of the problem is analogous to the gap-k-SAT problem:

Definition 9 $Gap([k,\Sigma]CSP,p,q)$ for $0\leq p<q\leq 1$ is the problem, where we output 1 on $[k,\Sigma]CSP$ instances, that have an assignment satisfying at least q fraction of the constraints and 0 on those instances with no assignment satisfying more than p fraction of the constraints. On any other instances the output is arbitrary.

Fact 1 *Every PCP for a language L, with completeness q, soundness p, query size $k=O(1)$, and witness-alphabet Σ can be interpreted as a Karp reduction from L to $Gap([k,\Sigma]\text{CSP},p,q)$ and vice-versa.*

Proof Recall that a Karp reduction from a language $L\subseteq\Sigma^*$ to the gap version of a maximization problem OPT, with lower and upper thresholds p,q, respectively, is a polynomial-time computable function f from Σ^* to instances of OPT such that:
1. If $x\in L$, then $OPT(f(x))\geq q$
2. If $x\notin L$, then $OPT(f(x))\leq p$.

Let now $V(x, P, r)$ be the verifier of a PCP for L, which queries at most k bits nonadaptively. For fixed x and r, there is a k-ary relation $\varphi_{x,r}$ expressing if the verifier accepts or rejects the entries of the proof P it views. We have

$$\max_y Prob(V(x, P, r) = 1) \;=\; \max_P |\{r|\varphi_{x,r}(P) = 1\}|/2^{|r|}. \tag{17}$$

The problem on the right-hand side of (17) is a gap $[k, \Sigma]$CSP problem, and the completeness and soundness conditions of the PCP translate to the gap requirements. The converse is also straightforward: Let Φ_x be a $[k, \Sigma]$CSP instance that we get from x by the Karp reduction for L that we have assumed to exist. This is a PCP when we treat the assignment to the variables of Φ_x as proof P. The verifier of this PCP checks if P satisfies a random clause of Φ_x. The completeness of this system is q and the soundness is p. □

Till the end of the section we are going to focus only on gap-CSP instances.

Theorem 17 (Dinur's Theorem) *Let* $\Sigma = \{0, 1\}^3$. *Then there exists* $\epsilon > 0$ *such that* $[2, \Sigma]$*CSP Karp reduces to Gap* $([2, \Sigma]CSP, 1 - \epsilon, 1)$.

Remark 4 Let $\Sigma = \{0, 1\}^3$. Then the $[2, \Sigma]$CSP is NP-complete, and the PCP theorem follows.

For the rest of the section, we fix $\Sigma = \{0, 1\}^3$.

Instance Size and Satisfiability Gap

For a $[2, \Sigma]$CSP instance $\Phi = \bigwedge_{i=1}^{m} \Phi_i$ (in Sect. 2.7 Φ_i was denoted by f_i), we define the *instance size* as $|\Phi| = m$. The *satisfiability gap* of Φ is $\overline{\text{sat}}(\Phi) = \min_P |\{i|\Phi_i(P) = 0\}|/m$ (one minus the maximal fraction of the simultaneously satisfiable constraints by any assignment P).

Let Φ be a $[2, \Sigma]$CSP instance. Either $\overline{\text{sat}}(\Phi) = 0$ (the formula is satisfiable) or $\overline{\text{sat}}(\Phi) \geq 1/|\Phi|$ (the formula is not satisfiable). For unsatisfiable instances, the satisfiability gap cannot be smaller than $1/|\Phi|$, since if the formula is not satisfiable, then at least one component of Φ is not satisfied under any assignment. Dinur constructs a reduction that enlarges this tiny gap to a constant. The same reduction, if applied to a satisfiable instance, leaves the resulting instance satisfiable. The latter property will be straightforward, so we shall focus on unsatisfiable instances. The reduction proceeds in $O(\log_2 m)$ steps, making small progresses at a time.

It is sufficient to show that there exists a (sufficiently large) constant $C > 0$ and an $\epsilon > 0$ such that any $[2, \Sigma]$CSP instance Φ can be reduced in polynomial time to a $[2, \Sigma]$CSP instance Φ' that has the following properties:

1. If Φ is satisfiable, then Φ' is satisfiable.
2. $|\Phi'| \leq C|\Phi|$.
3. $\overline{\text{sat}}(\Phi') \geq \min\{2\,\overline{\text{sat}}(\Phi),\ \epsilon\}$.

To get Theorem 17, we start from the original instance and apply the above reduction on it $\log_2 m$ times. The size of the final instance is polynomially bounded by that of the original instance (i.e., by m). To construct the reduction sequence takes polynomial time.

3.1 The Three Sub-Steps of Dinur's Basic Reduction Step

We obtain Φ' from Φ of the previous section in three steps: (1) structural Improvement, (2) gap amplification, (3) alphabet reduction. We say that a $[2, \Sigma]$CSP instance is d-regular, expanding, if the graph we obtain by replacing each constraint with the corresponding pair of variables is a d-regular expander. Let Φ be an arbitrary $[2, \Sigma]$CSP. Let $d = 11$ and t be a large enough constant to be determined later. The steps are as follows:

$$\begin{array}{lll}
\Phi \in [2,\Sigma]\text{CSP} & \rightarrow & \text{(Structural improvement)} \\
\Phi_{\text{reg}} \in d\text{-regular, expanding } [2,\Sigma]\text{CSP} & \rightarrow & \text{(Gap amplification)} \\
\Phi_{\text{big}} \in [2, \Sigma^{(d+1)^{\lceil \frac{t}{2} \rceil}}]\text{CSP} & \rightarrow & \text{(Alphabet reduction)} \\
\Phi' \in [2,\Sigma]\text{CSP} & &
\end{array}$$

Furthermore:

- If Φ is satisfiable, then so are Φ_{reg}, Φ_{big} and Φ'.
- There are constants C' and C'' such that

$$|\Phi_{\text{reg}}| \leq C'|\Phi|, \quad |\Phi_{\text{big}}| = (d+1)^{\lceil \frac{t}{2} \rceil - 1}|\Phi_{\text{reg}}|, \quad |\Phi'| \leq C''|\Phi_{\text{big}}|.$$

- There are $\epsilon > 0$, $D > 1$, and $\delta > 0$ with $D\delta \geq 20$ such that the satisfiability gaps change as follows:

$$\begin{aligned}
\overline{\text{sat}}(\Phi_{\text{reg}}) &\geq 0.1\, \overline{\text{sat}}(\Phi) \\
\overline{\text{sat}}(\Phi_{\text{big}}) &\geq \min\{D\, \overline{\text{sat}}(\Phi_{\text{reg}}), \epsilon/\delta\} \\
\overline{\text{sat}}(\Phi') &\geq \delta\, \overline{\text{sat}}(\Phi_{\text{big}}).
\end{aligned}$$

Notice that from Φ to Φ' the satisfiability gap increases by a factor of at least $0.1D\delta \geq 2$, unless it is already ϵ. Only the $\Phi_{\text{big}} \rightarrow \Phi'$ reduction requires classic PCP ideas, namely, the long code.

3.1.1 Structural Improvement

The $\Phi \rightarrow \Phi_{\text{reg}}$ reduction is a fairly standard transformation, which starts with creating $\deg\, v$ clones of every node v of the constraint graph of Φ. We distribute the outgoing edges among the clones, making every clone an end-point of exactly one outgoing edge. Then for every clone group (associated with the same original node), we place a degree 5 expander on the members of the clone group (each member is one node of the expander), and we put equality constraints on the new edges. This way the entire graph will be a 6-regular graph, and it may not be an expander itself, since we assume nothing about the structure of Φ. To ensure the expanding property, we now add new edges with empty constrains on them in such a way that the new edges form a degree 5 expander on all nodes. The final graph is $d = 11$-regular, and it is an expander (since expander + any graph = expander). Throughout the whole construction, we preserve multiple edges (possibly with different constraints). The parameter changes are easy to calculate.

3.1.2 Gap Amplification

The second reduction ($\Phi_{\text{reg}} \rightarrow \Phi_{\text{big}}$) is a wonderful new addition to PCP theory, and this is the one that gains us the gap. We define an operation on binary constraint systems called *powering*. Let G be a constraint graph and $t > 1$ be an integer. First we add a loop to each node (with an empty constraint). We denote the resulting graph with $G + I$. Then we construct $(G + I)^t$ in such a way that:

- The vertices of $(G + I)^t$ are the same as the vertices of G.
- Edges: u and v are connected by k edges in $(G + I)^t$ iff the number of paths of length t from u to v in $G + I$ (if the path includes a loop, it also counts towards the length) is exactly k.
- Alphabet: The alphabet of $(G + I)^t$ is $\Sigma^{(d+1)^{\lceil t/2 \rceil}}$, where every vertex (when the prover is honest) specifies values for all of its neighbors reachable in $\lceil t/2 \rceil$ steps.
- Constraints: The constraint associated with an edge (u, v) of $(G + I)^t$ is satisfied iff the assignments for u and v are consistent with an assignment that satisfies all of the constraints induced by the union of the $\lceil t/2 \rceil$ neighborhoods of u and v.

If G is satisfiable, then $(G + I)^t$ is satisfiable as well. To see what happens with the negative instances, Dinur shows that powering has the following gap-enlarging property:

Lemma 16 (Amplification Lemma [33]) *Let Σ be an arbitrary constant size alphabet. There exists a constant $\gamma = \gamma(d, |\Sigma|) > 0$ such that for any $t > 0$ and for any d-regular expanding constraint graph G,*

$$\overline{\text{sat}}((G + I)^t) \geq \gamma \sqrt{t} \min \left\{ \overline{\text{sat}}(G), \frac{1}{t} \right\} .$$

We define $\Phi_{\text{big}} = (\Phi_{\text{reg}} + I)^t$. Parameter t has to be chosen so that we get a large enough $\overline{\text{sat}}(\Phi_{\text{big}})/\ \overline{\text{sat}}(\Phi_{\text{reg}})$ ratio to compensate for the loss in the satisfiability gap in the first and third transformations and even gaining a factor of two over that. In other words, Lemma 16 ensures that we can choose D to be sufficiently large.

3.1.3 Alphabet Size Reduction

($\Phi_{\text{big}} \to \Phi'$): The gap amplification step has increased the alphabet size from Σ to Σ_{big}, and now we have to reduce the alphabet to Σ without loosing much of the gap we have gained.

Since $\Sigma = \{0,1\}^3$, we can identify $\Sigma_{\text{big}} \overset{\text{def}}{=} \Sigma^{(d+1)^{\lceil \frac{t}{2} \rceil}}$ with $\{0,1\}^{3(d+1)^{\lceil \frac{t}{2} \rceil}}$. If Φ_{big} is satisfiable, the true prover encodes each variable of the satisfying assignment (i.e., element of Σ_{big}) with some encoding function $E : \{0,1\}^{3(d+1)^{\lceil \frac{t}{2} \rceil}} \to \{0,1\}^{10\times 3(d+1)^{\lceil \frac{t}{2} \rceil}}$ that corrects constant fraction of errors (from coding theory we know that such encoding exists). To represent the bits of the code word, we shall use Σ-valued variables, somewhat wastefully $10 \times 3(d+1)^{\lceil \frac{t}{2} \rceil}$ of them. The necessity of the error-correcting encoding will be explained later.

These are, however, not all the variables the true prover provides. If v_i and v_j are two variables on which Φ_{big} has a constraint, we install a Dinur assignment tester for the (v_i, v_j) pair. This means an additional constant number of Σ-valued variables per every constraint of Φ_{big}. For a constraint on (v_i, v_j) Dinur's tester checks, using the additional information given by the prover, if the desired relation holds between the (encoded) labels of v_i and v_j. The details are as follows:

Lemma 17 (Assignment Tester of Dinur) *Let h be an arbitrary positive integer, $\mathcal{T} \subseteq \{0,1\}^h$. Then there are positive integers l and f, and a 2-query verifier V, that uses a random string $r \in \{0,1\}^f$ accesses a pair of oracles* $(\mathsf{theorem}, P) \in \{0,1\}^h \times \Sigma^l$ *and satisfies:*

1. *If* $\mathsf{theorem} \in \mathcal{T}$, *then there is $P \in \Sigma^l$ such that $Prob_r(V(\mathsf{theorem}, P, r) = 1) = 1$.*
2. *If $z \in \{0,1\}^h$ is η-far from all words in $\mathcal{T}$, then for every $P \in \Sigma^l$, we have $Prob_r(V(z, P, r) = 0) \geq \eta/100$.*

It is crucial, that the verifier does not read theorem *that it verifies.*

We do not prove Lemma 17 here, only mention, that to construct P of the true prover, Dinur employs the folded long code together with her test described in Sect. 2.6 and standard techniques.

For every constraint (v_i, v_j) of Φ_{big}, Dinur installs an assignment tester to verify the property:

$$\mathcal{T}_{i,j} = \{(E(\sigma_i), E(\sigma_j)) | (\sigma_i, \sigma_j) \in \Sigma_{\text{big}}^2 \text{ satisfy all constraints of } \Phi_{\text{big}} \text{ on } v_i, v_j\}.$$

Why is error-correcting encoding E necessary at all, when Dinur's assignment tester in Lemma 17 does not require encoded input? Notice that the success of the test depends on the distance of the assignment from $\mathcal{T}$. The tester should not accept its input with high probability, unless there is an underlying assignment for Φ_{big} that satisfies the constraint that the tester tests. If $\mathcal{T}_{i,j}$ was simply defined as the collection of all (σ_i, σ_j) laid out in binary (without any encoding) that satisfy all constraints of Φ_{big} on v_i, v_j, then if Σ_{big} is large, changing a single bit in the pair could make the constraint of Φ_{big} fail on (σ_i, σ_j), yet the assignment tester would still accept it with high probability.

In contrast, if we change one bit of $E(\sigma_i) \cup E(\sigma_j)$, σ_i is still decodable from $E(\sigma_i)$ and σ_j from $E(\sigma_j)$, using the closest code-word decoding. (It is crucial that the decoding procedure depends on the members of the pair individually and not on the entire pair globally.) In fact, one has to change a constant fraction of the bits of $E(\sigma_i) \cup E(\sigma_j)$ to make either σ_i or σ_j un-decodable. Thus, when the constraints of Φ_{big} are replaced with assignment testers and the average acceptance probability of all assignment testers is $1 - \epsilon'$, the closest distance decoding decodes to an assignment for Φ_{big} that has satisfiability gap at most ϵ'/δ, where δ is some fixed constant.

Summarizing the above, the true prover transforms the proof $\sigma_1 \ldots \sigma_n$ of Φ_{big} as follows:

$$\begin{array}{lll} \sigma_1 \ldots \sigma_n & \rightarrow & \text{(Encodes the alphabet)} \\ E(\sigma_1) \ldots E(\sigma_n) & \rightarrow & \text{(Adds assignment testers)} \\ E(\sigma_1)' \ldots E(\sigma_n)' P_{i_1,j_1} \ldots P_{i_{m'},j_{m'}}. & & \end{array}$$

Here $m' = |\Psi_{\text{big}}|$, and $P_{i,j}$ is the proof of the assignment tester for constraint (i, j). The apostrophes in $E()'$ refer to the embedding of every bit of $E()$ into an element of Σ: $0 \rightarrow 000$ and $1 \rightarrow 100$. The new tests are those of the assignment testers, combined, appropriately weighted.

It is easy to see that when Φ_{big} is satisfied and the prover is faithful to the protocol, all tests are accepting. Assume now that Φ_{big} is a negative instance. As was sketched a little earlier, in this case the satisfiability gap of Φ' is at least $\delta\overline{\text{sat}}(\Phi)_{\text{big}}$ for some fixed δ, *independently of* $|\Sigma_{\text{big}}|$ (independence of $|\Sigma_{\text{big}}|$ is crucial, since we need the freedom to set t in the $\Phi_{\text{reg}} \rightarrow \Phi_{\text{big}}$ step to achieve a gap enlargement that more than compensates us for the cumulative loss in the $\Phi_{\text{big}} \rightarrow \Phi'$ and the $\Phi \rightarrow \Phi_{\text{reg}}$ steps). The transformation blows up the instance size by only a constant factor (the constant depends on $|\Sigma_{\text{big}}|$, but it is all right). Since the tester looks at binary constraints over the alphabet $\Sigma = \{0, 1\}^3$, by Fact 1 it corresponds to a $[2, \{0, 1\}^3]CSP$.

4 The Unique Games Conjecture

Amplification of the PCP theorem via parallel repetition results in the non-approximability of the label cover problem. When the label cover is composed with the long code inner verifier, Hastad's Fourier analytic technique yields optimal non-approximability bounds for problems like MAX3SAT [55], MAX3LIN [55], and Max Clique [54]. It is not known, however, how to prove such results for 2CSPs like vertex cover, MAX2LIN, MIN2SAT Deletion. The primary barrier is that it is not possible to efficiently perform both the code-word and consistency tests for projection constraints with just two queries. The barrier gets larger, when the image of the projection becomes much smaller than the domain. This state of affairs compelled Subhash Khot [61] to investigate what happens if we substitute the

$\mathrm{P} \neq \mathrm{NP}$ assumption with the assumption that label cover with bijective constraints are hard to approximate for large label size. Optimally, this could be NP-hard too, but as of now no one can prove this.

An instance of the *unique games ($\mathcal{UG}$)* problem (*bipartite* unique games problem) is a tuple $\mathcal{U} = (G, \Sigma, \Pi)$, where:

1. $G = (V, E)$ is a graph ($G = (V, W, E)$ is a bipartite graph with bipartition V, W) with edge set E.
2. Nodes in V (in both V and W) are assigned labels from Σ.
3. $\Pi = \{\pi_{vw} \mid (v, w) \in E\}$, is a set of bijections (permutations), $\pi_{vw} : \Sigma \to \Sigma$.

An assignment $l : V \to \Sigma$ ($l : V \cup W \to \Sigma$) is a labeling of the nodes with the elements of Σ. An edge (v, w) is satisfied iff $\pi_{vw}(l(v)) = l(w)$. Notice that unique games are special label cover instances, where projections are bijections.

Conjecture 1 (Unique Games Conjecture (UGC) [61]) *For every $\epsilon, \delta > 0$, there exists a fixed alphabet Σ, such that $Gap([2, \Sigma]\mathcal{UG}, \delta, 1 - \epsilon)$ is NP-hard.*

A weaker version of the conjecture is that $Gap([2, \Sigma]\mathcal{UG}, \delta, 1 - \epsilon)$ is not in polynomial time.

Remark 5 Khot, Kindler, Mossel, and O'Donnell [63] proved that the UGC is equivalent to the special case where each projection constraint is a linear constraint modulo q, i.e., $\Sigma = [q]$ and π_{uv} is defined via $l(u) = l(v) + c_{uv} \pmod q$. Khot and Regev [65] proved that it is enough to consider constraint graphs that are *left regular*.

Conditioned on the conjectured hardness of unique games, [61] was able to prove optimal bounds for problems including MIN2SAT Deletion and MAX2LIN.

4.1 Algorithms for Unique Games

Before discussing further remarkable consequences of the UGC, let us investigate in what sense the unique games problem is different from (easier than) the general label cover problem. Several algorithms solve the unique games in polynomial time for special cases, but none is strong enough to disprove the conjecture. In general, these algorithms impose relations between the parameters ϵ, δ, and $|\Sigma|$ or constraints on the underlying graph, and their existence tells us that one has to be careful with the parameters and the structure of the instance, when trying to prove the conjecture.

It is in P to tell if a unique games instance is perfectly satisfiable. It is also easy to see that a random assignment of labels to the nodes satisfies at least $\frac{1}{|\Sigma|}$ fraction of edges on expectation, implying $\delta > \frac{1}{|\Sigma|}$.

From the result of Charikar, Makarychev, and Makarychev [22], it follows, that if $\epsilon \in O\left(\frac{1}{\log |\Sigma|}\right)$, the UGC is in polynomial time (regardless of δ).

In order to disprove the UGC, it is enough to have a polynomial-time algorithm that finds an assignment that satisfies $\delta(\epsilon)$ fraction of the constraints, given a $1 - \epsilon$

satisfiable unique games instance and $\delta \to c$ as $\epsilon \to 0$, where $c > 0$ is some universal constant. Importantly, δ should not depend on $|\Sigma|$.

Motivated by sparsest cut, researchers have considered the special case, where the unique games graph, G, is a good *spectral expander*. If $\lambda = \lambda_2(G)$ is the second smallest eigenvalue of the Laplacian of G, Arora, Khot, Kolla, Steurer, Tulsiani, and Vishnoi [8] (improved by Makarychev and Makarychev [75]) give a polynomial-time algorithm with $\delta(\epsilon) = 1 - \frac{\epsilon}{\lambda} \log \frac{\lambda}{\epsilon}$.

In a recent result Arora, Barak, and Steurer [7] give an algorithm with $\delta(\epsilon) = 1 - \epsilon^{\alpha}$ that runs in $exp(n^{\epsilon^{\alpha}})$ time for a fixed constant $\alpha > 0$. Underlying their analysis is a graph decomposition theorem, stating that the vertex set of a graph can be partitioned in such a way that each resulting subgraph has few large eigenvalues ($\leq n^{O(\epsilon)}$) and there are at most a constant fraction (dependent on ϵ) of edges that go across the parts. The result involves applying this decomposition to the unique games graph and then solving the unique games problem for the induced subgraphs in the partition, using similar methods to the one used by Kolla [68].

We refer the reader to Khot's survey [62], which has an excellent discussion of many of the details.

4.2 Variants of the Unique Games Conjecture

There are several non-approximability results that have been proved starting from variants of Unique Games. Here we mention some of the variants.

A $d \to 1$ game is a special case of the label cover problem, and it is a generalization of unique games, where $|\Sigma_1| = d|\Sigma_2|$ and each projection constraint π is a $d \to 1$ map, i.e., $|\pi^{-1}(x)| \leq d$ for every $x \in \Sigma_2$. When $\Sigma_1 = [dq]$ and $\Sigma_2 = [q]$, we denote such a game by $[2,q]\mathcal{PG}_{d\to 1}$. For every positive integer $d \geq 2$, Khot [61] makes the following conjecture:

Conjecture 2 (*d* $\to$ **1-conjecture [61])** *For every* $\epsilon > 0$, *there exists a* $q = q(\epsilon)$ *such that* $Gap([2,q]\mathcal{PG}_{d\to 1}, \epsilon, 1)$ *is NP-hard.*

The $d \to 1$-conjecture has the perfect completeness property which is useful in some situations. It is shown by O'Donnell and Wu [78] that given a satisfiable instance of a Boolean 3CSP, it is $2 \to 1$-hard to find an assignment that satisfies more than $\frac{5}{8}$ fraction of the constraints.

Although the unique games is easy if the underlying graph is a good expander, it may be still hard if the graph is mildly expanding. A variant of the UGC states exactly this:

Conjecture 3 (UGC with expansion, [8]) *There exists a universal constant* $1/2 < t < 1$ *with the following properties: For every* $\epsilon, \delta > 0$, *there exists a* $\Sigma(\epsilon, \delta)$ *such that given a unique games instance* $\mathcal{U} = (G, \Sigma, \Pi)$, *it is NP-hard to distinguish between the following cases:*

- *$\mathcal{U}$ is at least $1 - \epsilon$-satisfiable.*
- *$\mathcal{U}$ is at most δ-satisfiable and for every partition $A \,\dot\cup\, B \,\dot\cup\, C$ of the vertex set of $\mathcal{U}$, where $|A| \leq 0.001|V|$ and $|B|$ and $|C| \geq 0.1|V|$, the (B, C) cut has at least ϵ^t fraction of the edges across.*

Among the consequences of the above conjecture is that the balanced separator problem, and therefore the sparsest cut problem, is hard to approximate within any constant factor.

4.3 Optimal Bounds Under the Unique Games Conjecture

Assuming the unique games conjecture, exact non-approximability bounds can be proved for several classes of problems. These include max-CSPs [83], strict CSPs [69], ordering problems [50], and clustering problems [64]. There are also problems for which the UGC does not give optimal bounds but still better ones than the $P \neq NP$ assumption alone. These include graph partitioning [2, 23, 66] and graph coloring [35, 60]. Some of the results, like those for graph coloring, are based on variants of the UGC. There are several excellent surveys that have details regarding these problems and the UGC [62, 96]. In this section, we discuss some outstanding consequences of the UGC, including Raghavendra's general hardness result for CSPs.

4.3.1 Optimal Non-approximability for Max Cut

Given a graph G, the Max Cut problem is to find a bipartition of $V(G)$ such that the fraction of edges crossing between the parts is maximized. In [55] an $\frac{17}{16} - \epsilon$ non-approximability bound was shown under $P \neq NP$. The best polynomial-time algorithm, due to Goemans and Williamson [48] approximates Max Cut within a factor of $\approx$ 1.1389 and employs breakthrough *semidefinite programming (SDP)* techniques.

The celebrated Goemans–Williamson (GW) approximation bound, which is precisely $\frac{\pi}{2} \max_{0<\theta\leq\pi} \frac{1-\cos\theta}{\theta}$, at first was not conjectured to be optimal. In a surprise move, however, Khot, Kindler, Mossel, and O'Donnell [63] proved that assuming UGC, the GW ratio is essentially the best possible. Their result was later beautifully extended by Raghavendra [83], who showed that a natural class of semidefinite programs solves all CSP optimally under UGC. We shall sketch the ideas in KKMO in this section and its generalization in Sect. 4.3.3.

For proving the non-approximability of Max Cut, we build our PCP from a bipartite unique games instance $\mathcal{U} = (G, \Sigma, \Pi)$ with $G = (V, W, E)$ as the outer verifier. By Remark 5 we will assume that the instance is left regular, i.e., that all $v \in V$ have the same degree. Depending on how closely we want to get to the GW bound, we choose the alphabet size, $|\Sigma| = R$, of $\mathcal{U}$ to be an appropriately large constant. Next, we need to set a parameter $-1 < \rho < 0$ optimally. For any ρ in this range, we get that to tell instances apart that are $\frac{1-\rho}{2}$ satisfiable (positive instances) from instances that are slightly more than $\frac{\arccos\rho}{\pi}$ satisfiable (negative instances) is unique games hard. The choice for ρ that gives the largest relative gap leads to the GW bound. The inner verifier is designed a little differently from the usual reductions, where the outer verifier is the label cover problem:

- The (true) prover encodes the labels, $l(w)$, of every $w \in W$ with the long code $long_R$. (But it does not use the labels of $v \in V$ anyhow!)
- The verifier picks a vertex $v \in V$ at random and two of its neighbors $w, w' \in W$ at random. Let $\pi = \pi_{vw}$ and $\pi' = \pi_{vw'}$ be the respective bijections for edges (v, w) and (v, w').

At this point we apply the following inner verifier:

1. Let y^w and $y^{w'}$ be the supposed long codes of the labels of w and w', respectively.
2. Pick $f : R \to \{0, 1\}$ uniformly and randomly.
3. Pick $\mu : R \to \{0, 1\}$ by choosing each function value independently to be 0 with probability $\frac{1}{2} + \frac{1}{2}\rho$ and 1 with probability $\frac{1}{2} - \frac{1}{2}\rho$.
4. Accept iff $y^w_g \neq y^{w'}_{g'}$, where $g = f \circ \pi$ and $g' = (f \circ \pi') \oplus \mu$. Here operation $\oplus$ means the exclusive or of two Boolean-valued function.

It is easy to see that the completeness of the above test is almost (where the "almost" comes from the imperfect completeness of the outer verifier) at least $\frac{1}{2} - \frac{1}{2}\rho$. Analyzing the soundness requires Fourier analytic techniques and the brilliant majority is stablest theorem of Mossel, O' Donnell, and Oleszkiewicz [77]. If the soundness of the above PCP is slightly bigger than $(\arccos \rho)/\pi$, then a solution for $\mathcal{U}$ exists satisfying a small, but constant fraction of the constraints (independently of M), which is a contradiction, if M was chosen large enough (which makes the soundness of the outer verifier small enough). Without giving the details of the calculation, we just hint that reason for this is that for those $v \in V$, whose average "neighbor-pair" inner verifiers have average success rate at least $(\arccos \rho)/\pi$ + small constant, a "probabilistic" (supposed) long code word exists, namely, $\frac{1}{\text{degv}} \sum_{v \sim w} \pi_{v,w} y^w$ (notice, we needed to shuffle the coordinates of y^w), from which a label l for v can be decoded, which correlates with the (supposed) long codes for v's neighbors as follows: The individual (supposed) long codes, y^w, of a constant fraction of the neighbors w of v have the property that from them we can decode a constant length list of labels such that at least one from the list is $\pi_{v,w}(l)$. A standard random assignment technique (which picks a random label from each list) now gives a labeling for $\mathcal{U}$, which satisfies a constant fraction of the constraints. Important was that the list for each w was recovered only from the supposed long code of w, without looking at v.

4.3.2 Semidefinite Programming and Integrality Gap

Semidefinite programming has resulted in the best known approximation ratios for various classes of optimization problems. For the performance analysis of semidefinite programs, it is crucial to understand the notion of *integrality gap*.

Fix instance Φ of a maximization problem Λ (think of Λ as Max Cut). The first step in designing an SDP-based algorithm for Φ is to translate it to a quadratic program QP_Φ, such that the optimal solutions for Φ and QP_Φ are the same. Then we *relax* the variables, allowing them to take any real vector values. Multiplications between the variables of QP_Φ become scalar products ($xy \to (\vec{x}, \vec{y})$) in the corresponding semidefinite programming instance, SDP_Φ.

An optimal solution, $SDOPT_\Phi$, to SDP_Φ is then computed, which is *rounded* to a solution of QP_Φ. Let OPT_Φ be the optimal value for Φ (and QP_Φ). Suppose the value of the rounded solution for SDP_Φ is $ROUND_\Phi$. To prove an approximation ratio of r_Φ, it is sufficient to have $r_\Phi ROUND_\Phi \geq SDOPT_\Phi$. The approximation ratio follows because

$$SDOPT_\Phi \geq OPT_\Phi \geq ROUND_\Phi \geq \frac{1}{r_\Phi} SDOPT_\Phi. \tag{18}$$

A barrier to this rounding technique is the *integrality gap* of the relaxation, $IGap_\Lambda = \max_\Phi \frac{SDOPT_\Phi}{OPT_\Phi}$. A lower bound on the integrality gap is usually proved by exhibiting a suitable sequence of instances Φ with integrality gap arbitrarily close to $IGap_\Lambda$.

4.3.3 Exact Non-approximability for All CSPs

In a remarkable recent result, Raghavendra [83] proved that assuming the UGC, SDP rounding algorithms yield essentially the best possible approximation ratios achievable in polynomial time. This result has confirmed former beliefs about the extraordinary strength of semidefinite programming. For every CSP Λ, Raghavendra explicitly constructs semidefinite programs whose maximal integrality gap is the optimal non-approximability bound for Λ.

We will describe the SDP for any 2-variable CSP maximization problem. Let Λ be a 2-variable CSP over the alphabet $\Sigma = [q]$. We can think of each instance Φ of Λ as a directed graph $G = (V, E)$ with nodes denoting variables and edges denoting constraints. Each edge (u, v) has a corresponding *pay-off* function $P_{uv} : \Sigma^2 \to [0, 1]$ (generalized from $\{0, 1\}$). Without loss of generality, we can assume that the constraints have nonnegative weights w_{uv} summing up to one, which is a probability distribution over E. Our goal is to find an assignment $l : V \to \Sigma$ such that

$$\sum_{(u,v)\in E} w_{uv} P_{uv}(l(u), l(v))$$

is maximized. When we turn this into a quadratic program, for every node u of G and for every element $i \in \Sigma$, we introduce a variable u_i that takes value one, if u gets label i in the optimal solution, and zero otherwise. The quadratic program is the same in appearance as the analogous semidefinite relaxation, where the u_is are now vectors and the "dot" means scalar product:

SDP_Λ

Maximize $\sum_{(u,v)\in E} w_{uv} \left(\sum_{i,j\in[q]} P_{uv}(i, j) u_i \cdot v_j \right)$
Subject to,
$\forall u, v \in V, i, j \in [q] : u_i \cdot u_j = 0, u_i \cdot v_j \geq 0$
$\forall u \in V : \sum_{i\in[q]} |u_i|^2 = 1$
$\forall u, v \in V : \sum_{i,j\in[q]} u_i \cdot v_j = 1$

Raghavendra [83] creates a long code test $Test_\Lambda(\Phi)$ from instance Φ, whose completeness and soundness ratio is the integrality gap of Φ. The long code test is a generalization of the folded long code, where $\{0, 1\}$ valued functions are replaced with $[q]$ valued functions. The key idea is that to define the query distribution of $Test_\Lambda(\Phi)$, Raghavendra uses an optimal solution of the above SDP. For each edge $(u, v) \in E$ and for each node $u \in V$, we define distributions $D_{uv} : Prob[i, j] = u_i \cdot v_j$ and $D_u : Prob[i] = |u_i|^2$, respectively. The following is the long code test for $z \in [q]^{q^R}$:

```
$Test_\Lambda(\Phi, \epsilon)$ for $z \in [q]^{q^R}$
Choose an edge $(u, v) \in E$ according to $w_{uv}$
Pick $R$ independent copies from $D_{uv}$ to obtain $x, y \in [q]^R$
Form $\hat{x}$ from $x$ by replacing each $x_i$ independently, with probability $\epsilon$, from the distribution $D_u$
Form $\hat{y}$ from $y$ by replacing each $y_i$ independently, with probability $\epsilon$, from the distribution $D_v$
Return "yes" with probability $P_{uv}(z_{\hat{x}}, z_{\hat{y}})$
```

From the definition of the test it is fairly straightforward that the completeness of the test $Test_\Lambda(\Phi)$ is $SDOPT_\Phi - o_\epsilon(1)$. Recall that completeness means the probability with which a word of the long code is accepted, minimized over all long code words.

To talk about soundness, Raghavendra first defines what it means that a word is far away from all words of the long code. In his definition, this happens when the word to be checked is pseudorandom, which he expresses in terms of the Fourier coefficients of the word. The soundness analysis is nontrivial and crucially depends on Mossel's invariance principle of [76]. It turns out that the soundness cannot be much larger than OPT_Φ. More precisely Raghavendra shows that any word w that is sufficiently pseudo-random gives a randomized rounding procedure round$_w$ to round the optimal solution of the semidefinite program $SDP_\Lambda(\Phi)$, yielding an integral solution (i.e. a solution to the original Φ), whose value is close to the success probability of $Test_\Lambda(\Phi)$ on w. Here "close" means in terms of ϵ and the pseudorandomness parameters. To understand this beautiful duality better we refer the reader to [83].

Using the unique games outer verifier of [63] in the manner as described in Sect. 4.3.1 for the special case of Max Cut, we obtain the following result:

Theorem 18 *Let Φ be an instance of the CSP Λ. Then for every $\gamma > 0$, there exist $\epsilon, \delta > 0$ such that there is a polynomial-time reduction from $Gap(\mathcal{UG}, \delta, 1 - \epsilon)$ to $Gap(\Lambda, SDOPT_\Phi - \gamma, OPT_\Phi + \gamma)$.*

4.3.4 Small Set Expansion and Unique Games

The unique games conjecture has contributed to the great progress in understanding optimal non-approximability bounds. Since unique games conjecture is so central, it is natural to look for equivalent ways of stating it. Finding natural approximation-equivalent problems to unique games, however, have eluded us for a long time. To remedy this situation, Raghavendra and Steurer [84] consider the *small set expansion (SSE)* problem, a natural and widely applicable graph partitioning problem. They observe that the current techniques do not seem to be able to solve the problem and hence propose the *small set expansion conjecture (SSEC)*; that SSE is NP-hard. They proved that the small set expansion conjecture implies the unique games conjecture. In the rest of the section, we will state the small set expansion conjecture and briefly discuss the recent results related to this conjecture.

Let $G = (V, E)$ be a d-regular undirected graph, $E(A, B)$ for $A, B \subseteq V$, $A \cap B = \emptyset$ be the set of edges between A and B, and μ be the uniform measure on V. The *edge expansion* of a subset of vertices S is defined as $\Phi_G(S) = \frac{|E(S, V \setminus S)|}{d|S|}$ and $\Phi_G(\delta) = \min_{\mu(S)\approx\delta} \Phi_G(S)$, where $0 \leq \delta \leq 1$, and $a \approx b$ means $1/2 \leq a/b \leq 2$. (We have opted to use Φ, which is a standard notation for conductance, and has nothing to do with the Φ of earlier sections, where it denotes two-query PCP instances.) Then $Gap(SSE, \eta, \delta)$ problem is distinguish between the cases $\Phi_G(\delta) \geq 1 - \eta$ and $\Phi_G(\delta) \leq \eta$. Raghavendra and Steurer make the following conjecture:

Conjecture 4 (SSE Conjecture) *For every $\eta > 0$, there exists a $\delta > 0$ such that $Gap(SSE, \eta, \delta)$ is NP-hard.*

Small set expansion comes up in several contexts in relation to unique games. If a unique games instance $\mathcal{U}$ with alphabet Σ satisfies at least $1 - \epsilon$ fraction of the edges, then the *label extended* graph has a less-than-ϵ-expanding set of measure $\frac{1}{|\Sigma|}$. In the first step of their sub-exponential algorithm for unique games, [7] use a graph decomposition scheme that partitions the unique games graph into almost-non-expanding small sets. Also, all known integrality gap instances for unique games and various other CSPs have very good small set expansion properties.

The small set expansion conjecture is interesting for several reasons. As is the case with the unique games conjecture, it is in a way a "win–win" situation; a positive resolution would of course prove the unique games conjecture and a negative resolution would solve a very important natural problem, likely with powerful new techniques. Since it is a stronger conjecture, it could potentially lead to optimal non-approximability results for problems like sparsest cut which have so far been out of the reach of the unique games conjecture. In fact, in a recent result, Raghavendra, Steurer, and Tulsiani [85] prove better non-approximability bounds for problems including balanced separator and min linear rearrangement, than those that are known via UGC. Another interesting line of research is to have a unifying conjecture by reducing SSEC to the variants of UGC [62].

5 Structural Consequences of the PCP Theory

What makes some NPO problems hard to approximate, while others easy? This natural question had already been raised by Johnson in [56] long before the PCP theory was developed: "Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain [approximation] results, or are they due to some structural similarity between the problems as we define them?" In this section we present two completeness results that are consequences of the basic PCP theorem. These results give an evidence that different problems in the same non-approximability classes have the same reason for their hardness of approximation.

5.1 Polynomial-Time Approximation Schemes

Not all natural NP-hard optimization problems are hard to approximate. Easy classes include PTAS and FPTAS. For the definition of these classes and for that of the class APX, see the table below.

	FPTAS	PTAS	APX
Quantifiers	$\exists c\ \forall \epsilon > 0\ \exists A_\epsilon$	$\exists c(\epsilon)\ \forall \epsilon\ \exists A_\epsilon$	$\exists c\ \exists C > 1\ \exists A$
Algorithm	A_ϵ	A_ϵ	A
Running Time	$(\lvert x\rvert + 1/\epsilon)^c$	$\lvert x\rvert^{c(\epsilon)}$	$\lvert x\rvert^c$
Approximability Factor	$1+\epsilon$	$1+\epsilon$	C

Clearly, the following inclusions hold as follows:

$$FPTAS \subseteq PTAS \subseteq APX \subseteq NPO.$$

These inclusions are strict if $P \neq NP$. Cesati and Trevisan [21] also introduce the class $EPTAS$ which differs from $FPTAS$ in that the bound on the running time is $f(\epsilon)n^c$, where $f(\epsilon)$ is an arbitrary function of ϵ and $c > 0$ is an arbitrary constant.

An optimization problem is practically tractable if it has a fully polynomial-time approximation scheme or somewhat weaker, an EPTAS. If a problem has a PTAS, but not EPTAS, then there is ϵ such that the $|x|^{c(\epsilon)}$ running time practically prohibits approximating it within a factor better than $1+\epsilon$.

5.2 Approximation Preserving Reductions

Approximation preserving reductions in between NPO problems are used to show that if a problem P_1 is easy to approximate, then any problem P_2 is also easy to approximate which reduces to P_1. Since the easiness of an approximation problem is associated with its membership in $PTAS$, almost all approximation preserving reductions preserve membership in PTAS (Fig. 5).

The first paper which defines an *approximation preserving reduction* was that of Orponen and Mannila [79]. Up to the present time, there are at least eight notions

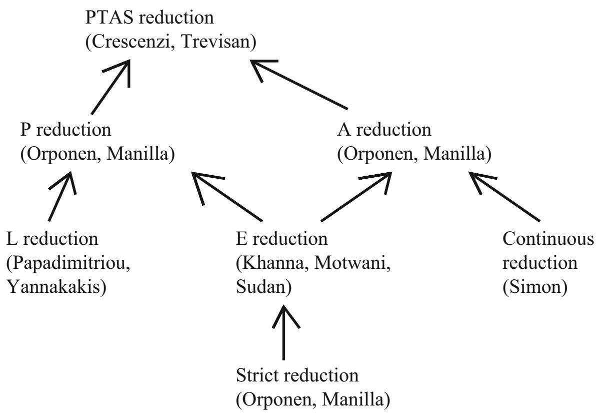

Fig. 5 The taxonomy of approximation preserving reducibilities (Courtesy of Crescenzi et. al. [30])

of approximation preserving reductions in use with a similar overall scheme. This scheme is the following:

Let $P_1(x,y)$ and $P_2(x',y')$ be two polynomial-time functions that are to be optimized for y and y' (maximized or minimized in an arbitrary combination). Let $OPT_1(x)$ and $OPT_2(x')$ be the optimum of these problems. A *reduction* assumes two maps:

1. A map f to transform instances x of P_1 into instances $x' = f(x)$ of P_2 [instance transformation]
2. A map g to transform (input, witness) pairs (x', y') of P_2 into witnesses y of P_1 [witness transformation]

Let $OPT_1 = OPT_1(x)$, $OPT_2 = OPT_2(h(x))$, $APPR_1 = P_1(x, g(h(x), y'))$, and $APPR_2 = P_2(h(x), y')$. The centerpiece of any approximation scheme is a relation which is required to hold between these four quantities. This relation must roughly say: "If $APPR_2$ well approximates OPT_2, then $APPR_1$ well approximates OPT_1." To see that indeed this is what we need, assume that we have a PTAS for OPT_2 and that P_1 reduces to P_2. In order to get a good approximate solution for $OPT_1(x)$, where x is an arbitrary input instance of P_1, first we construct $h(x)$ and find a witness y' such that $P_2(h(x), y')$ approximates $OPT_2(h(x))$ well. By the central assumption of the reduction, $P_1(x, y)$ well approximates $OPT_1(x)$ for $y = g(h(x), y')$. For the above argument to hold, f and g must be computable in polynomial time.

Different reductions differ in the relation required in between the four quantities. The L-reduction of Papadimitriou and Yannakakis [81] requires that OPT_2 is upper bounded by $c_1 OPT_1$ and that $|APPR_1 - OPT_1|$ is upper bounded by $c_2|APPR_2 - OPT_2|$ for some constants c_1 and c_2. It follows from the next lemma that L-reduction preserves PTAS.

Lemma 18 *A reduction scheme preserves PTAS iff it enforces that* $|APPR_1 - OPT_1|/OPT_1 \to 0$ *whenever* $|APPR_2 - OPT_2|/OPT_2 \to 0$.

5.3 APX Complete Problems

APX is the class of constant factor approximable NPO problems, and APX-PB is its subclass with problems that have a polynomial bound on their objective function. Our goal in this section is to give complete problems for APX and $APX - PB$.

Completeness proofs assume an underlying reduction. Even though in the theory of APX, the L-reduction is the most widely used reduction, in [31] it has been shown that it does not allow to reduce some problems which are known to be easy to approximate to problems which are known to be hard to approximate. Another flaw of the L-reduction is that it is too weak in another sense, namely, it is not always approximation preserving unless $P = NP \cap co - NP$ [30]. Therefore, we cannot hope for completeness results for APX or $APX - PB$ with respect to the L-reduction. Instead, S. Khanna, R. Motwani, M. Sudan, and U. Vazirani [59] define the E reduction which, using the notation of the previous section, requires that

$$\max\left\{\frac{APPR_1}{OPT_1}, \frac{OPT_1}{APPR_1}\right\} \leq 1 + c\left(\max\left\{\frac{APPR_2}{OPT_2}, \frac{OPT_2}{APPR_2}\right\} - 1\right)$$

for some $c > 0$. It can be easily shown that the E reduction preserves PTAS. Using the basic PCP theorem, Khanna et al. show the following:

Theorem 19 **[59]** *The $MAX\ \ SAT$ problem is complete for $APX - PB$ with respect to the E reduction. Also,*

$$APX - PB = \overline{MAX\ SNP} = \overline{MAX\ NP},$$

where the closure means closure under the E reduction.

The E reducibility is still somewhat too strict. In [31] it has been shown that natural PTAS problem exists, such as $MAX\ \ KNAPSACK$, which are not E reducible to polynomially bounded APX problems such as $MAX3SAT$. This drawback is mainly due to the fact that an E reduction preserves optimum values (see [31]). Crescenzi, Kann, Silvestri, and Trevisan [30] develop a reduction where functions f and g (see previous section) are allowed to depend on the *performance ratio*, where the performance ratio of an NPO problem A is defined as the function

$$R_A(x, y) = \max\left\{\frac{A(x, y)}{OPT_A(x)}, \frac{OPT_A(x)}{A(x, y)}\right\}.$$

Definition 10 (AP Reduction [30]) Let A and B be two NPO problems. A is said to be AP *reducible* to B, if two functions f and g and a positive constant α exist such that:

1. For any x and for any $r > 1$, $f(x, r)$ is computable in time $t_f(|x|, r)$.
2. For any x and for any $r > 1$ and for any y, $g(x, y, r)$ is computable in time $t_g(|x|, |y|, r)$.

3. For any fixed r, both $t_f(.,r)$ and $t_g(.,.,r)$ are bounded by a polynomial.
4. For any fixed n, both $t_f(n,.)$ and $t_g(n,n,.)$ are nonincreasing functions.
5. For any x and any $r > 1$ and for any y, $R_B(f(x,r),y) \leq r$ implies

$$R_A(x, g(x,y,r)) \leq 1 + \alpha(r-1).$$

The triple (f, g, α) is said to be an LP reduction from A to B.

In [30] the following is claimed:

Theorem 20 *$MAX\ SAT$ is APX complete with respect to the AP reduction.*

6 Conclusion

The major challenge of PCP theory is to build PCP reductions. The intuition to these reductions came from Arthur–Merlin games and zero knowledge proofs. From a combinatorial point of view, PCP reductions correspond to isoperimetric inequalities. From a complexity theoretical standpoint, they are pseudorandom constructs. We cannot, however, identify a single tool for building them. PCP theory takes its tools from many different branches of mathematics and computer science: complexity theory, polynomials, coding theory, combinatorics, pseudorandomness, probability theory, and geometry (for semidefinite programming). PCP reductions serve as the atoms of the theory, from which its proofs are composed.

The theory of PCP consists of the PCP theorem and its consequences. The PCP theorem states that every proof has a "transparent" version that admits a quick Monte Carlo verification algorithm with a constant number of check bits. Its current shortest proof, due to Irit Dinur, differs structurally from the previous proofs. The transformation of an ordinary proof into a transparent one is a PCP reduction itself.

The greatest consequence of PCP theory is the conditional hardness of approximating NPO. The theory of NP had cast serious doubt that for many important problems in NP we will ever find a polynomial-time solution, but hope still existed that algorithms will be available that output increasingly good approximate solutions. This hope has been shattered by the PCP theory: we know now that our long-term inability to design approximation algorithms for classical optimization problems such as *MAX CLIQUE*, *SET COVER*, *COLORING*, and *METRIC* TSP is due to the NP-hardness of the associated gap problems. PCP theory had a very major effect on the theory of CSP. Not only that the PCP theorem itself is an inapproximability statement of a CSP problem but due to the search for sharper inapproximability bounds, our understanding of CSPs has gone through an exponential acceleration. Much of this investigation has been centered around the so-called long code.

The most important spin-off of PCP theory is hardness studies based on the unique game conjecture (UGC). The UGC allows us to fully characterize the approximation bounds of all CSPs.

PCP theory has also provided a wealth of new algebraic and combinatorial ideas, like the notion of checkable codes and Raz's parallel repetition theorem for two-prover games.

Although PCP reductions have mostly replaced the approximation preserving reductions of the pre-PCP era, the latter still may be interesting when we study syntactically defined classes of optimization problems.

In spite of the tremendous advances, many questions remain open, like the NP-hardness of the unique games problem itself, the behavior of chromatic number under approximation for small chromatic graphs, and whether asymmetric TSP is constant factor approximable. These problems will give exciting projects for future generations of complexity theorists.

Cross-References

►Algorithms for the Satisfiability Problem
►Binary Unconstrained Quadratic Optimization Problem
►Complexity Issues on PTAS
►Fractional Combinatorial Optimization
►Max-Coloring

Recommended Reading

1. M. Ajtai, J. Komlós, E. Szemeredi, Deterministic simulation in logspace, *Proceedings of 19th Annual Symp. on the Theory of Computing, ACM*, 1987, pp. 132–140
2. C. Ambühl, M. Mastrolilli, O. Svensson, Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling, in *FOCS*, 2007, pp. 329–337
3. N. Alon, U. Feige, A. Wigderson, D. Zuckerman, Derandomized graph products. Comput. Complex. 60–75 (1995)
4. S. Arora, C, Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and the intractability of approximation problems. J. ACM **45**(3), 501–555 (1998). Preliminary version in *Proc. of FOCS'92*
5. S. Arora, S. Safra, Probabilistic checking of proofs: a new characterization of NP. J. ACM **45**(1), 70–122 (1998). Preliminary version in *Proc. of FOCS'92*
6. S. Arora, M. Sudan, Improved low degree testing and its applications, in *Proceedings of the 29th ACM Symposium on Theory of Computing*, 1997, pp. 485–495
7. S. Arora, B. Barak, D. Steurer, Subexponential algorithms for unique games and related problems, in *FOCS*, 2010, pp. 563–572
8. S. Arora, S. Khot, A. Kolla, D. Steurer, M. Tulsiani, N.K. Vishnoi, Unique games on expanding constraint graphs are easy: extended abstract, in *STOC*, 2008, pp. 21–28
9. L. Babai, Trading group theory for randomness, in *Proceedings of the Seventeenth Annual Symposium on the Theory of Computing*, ACM, 1985
10. L. Babai, E-mail and the unexpected power of interaction, in *Proc. 5th IEEE Structure in Complexity Theory conf.*, Barcelona, 1990, pp. 30–44
11. L. Babai, L. Fortnow, C. Lund, Non-deterministic exponential time has two-prover interactive protocols. Comput. Complex. **1**, 3–40 (1991)

12. L. Babai, L. Fortnow, L. Levin, M. Szegedy, Checking computations in polylogarithmic time, in *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing*, ACM, 1991
13. L. Babai, S. Moran, Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. J. Comput. Syst. Sci. **36**, 254–276 (1988)
14. M. Bellare, O. Goldreich, M. Sudan, Free bits, PCPs and non-approximability—towards tight results. To appear SIAM J. Comput. **27**(3), 804–915 (1998). Preliminary version in *Proc. of FOCS'95*. Full version available as TR95-024 of ECCC, the *Electronic Colloquium on Computational Complexity*, http://www.eccc.uni-trier.de/eccc/
15. M. Bellare, S. Goldwasser, C. Lund, A. Russell, Efficient probabilistically checkable proofs, in *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing*, ACM, 1993. (See also Errata sheet in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994)
16. M. Bellare, M. Sudan, Improved non-approximability results, in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994, pp. 184–193
17. M. Ben-Or, S. Goldwasser, J. Kilian, A. Wigderson, Multi-prover interactive proofs: how to remove intractability assumptions, in *Proceedings of the Twentieth Annual Symposium on the Theory of Computing*, ACM, 1988
18. P. Berman, G. Schnitger, On the complexity of approximating the independent set problem. Inform. Comput. **96**, 77–94 (1992)
19. A. Blum, Algorithms for Approximate Graph Coloring, Ph.D. thesis, Massachusetts Institute of Technology, 1991 (MIT/LCS/TR-506, June 1991)
20. R. Boppana, M.M. Halldórsson, Approximating maximum independent sets by excluding subgraphs. Bit Numer. Math. **32**, 180–196 (1992)
21. M. Cesati, L. Trevisan, On the efficiency of polynomial time approximation schemes. Inform. Process. Lett. **64**(4), 165–171 (1997)
22. M. Charikar, K. Makarychev, Y. Makarychev, Near-optimal algorithms for unique games, in *STOC*, 2006, pp. 205–214
23. S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, D. Sivakumar, On the hardness of approximating multicut and sparsest-cut. Comput. Complex. **15**(2), 94–114 (2006)
24. V. Chvatal, A greedy heuristic for the set-covering problem. Math. Oper. Res. **4**, 233–235 (1979)
25. A. Cohen, A. Wigderson, Dispersers, deterministic amplification, and weak random sources, in *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science*, IEEE, 1989
26. A. Condon, J. Feigenbaum, C. Lund, P. Shor, Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions, in *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing*, ACM, 1993
27. A. Condon, J. Feigenbaum, C. Lund, P. Shor, Random debaters and the hardness of approximating stochastic functions. SIAM J. Comput. **26**(2), 369–400 (1997)
28. S. Cook, The complexity of theorem-proving procedures, in *Proceedings of the Third Annual Symposium on the Theory of Computing*, ACM, 1971
29. P. Crescenzi, V. Kann, A compendium of NP optimization problems. Technical Report, Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", SI/RR-95/02, 1995. The list is updated continuously. The latest version is available by anonymous ftp from `nada.kth.se` as `Theory/Viggo-Kann/compendium.ps.Z`. Web address: http://www.nada.kth.se/~viggo/problemlist/compendium.html
30. P. Crescenzi, V. Kann, R. Silvestri, L. Trevisan, *Structure in Approximation Classes,* Electronic Colloquium on Computational Complexity (ECCC)(066): (1996); SIAM J. Comput. **28**(5), 1759–1782 (1999)
31. P. Crescenzi, L. Trevisan, On approximation scheme preserving reducibility and its applications. Theor. Comput. Syst. **33**(1), 1–16 (2000)
32. A. Deshpande, M. Tulsiani, N.K. Vishnoi, Algorithms and hardness for subspace approximation, in *SODA*, 2011

33. I. Dinur, The PCP theorem by gap amplification, in *Proceedings of the 31st ACM Symposium on Theory of Computing*, 2006, pp. 29–40
34. I. Dinur, O. Reingold, Assignment testers: towards a combinatorial proof of the PCP theorem. SIAM J. Comput. **36**(4), 975–1024 (2006)
35. I. Dinur, E. Mossel, O. Regev, Conditional hardness for approximate coloring. SIAM J. Comput. **39**(3), 843–873 (2009)
36. L. Engebretsen, J. Holmerin, Clique is hard to approximate within $n^{1-o(1)}$. Manuscript
37. U. Feige, A threshold of ln n for set cover, in *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing*, ACM, 1996. Journal Version: J. ACM **45**(4), 634–652 (1998)
38. U. Feige, S. Goldwasser, L. Lovász , S. Safra, M. Szegedy, Interactive proofs and the hardness of approximating cliques. J. ACM **43**(2), 268–292 (1996)
39. U. Feige, J. Kilian, Two prover protocols—low error at affordable rates, in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994
40. U. Feige, J. Kilian, Zero knowledge and chromatic number, in *Proceedings of the Eleventh Annual Conference on Complexity Theory*, IEEE, 1996, pp. 172–183
41. U. Feige, L. Lovász , Two-prover one-round proof systems: their power and their problems, in *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing*, ACM, 1992
42. V. Feldman, V. Guruswami, P. Raghavendra, Yi. Wu, Agnostic learning of monomials by halfspaces is hard. CoRR, abs/1012.0729 (2010)
43. L. Fortnow, J. Rompel, M. Sipser, On the power of multi-prover interactive protocols. Theor. Comput. Sci. **134**(2), 545–557 (1994)
44. M. Fürer, Improved hardness results for approximating the chromatic number, in *Proceedings of the Thirty Sixth Annual Symposium on the Foundations of Computer Science*, IEEE, 1995
45. M.R. Garey, R.L. Graham, J.D. Ullman, Worst case analysis of memory allocation algorithms, in *Proceedings in the 4th Annual ACM Symposium on the Theory of Computing*, 1972, pp. 143–150
46. M. Garey, D. Johnson, The complexity of near-optimal graph coloring. J. ACM **23**, 43–49 (1976)
47. M. Garey, D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman, 1979)
48. M. Goemans, D. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
49. S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof-systems. SIAM J. Comput. **18**(1), 186–208 (1989)
50. V. Guruswami, R. Manokaran, P. Raghavendra, Beating the random ordering is hard: inapproximability of maximum acyclic subgraph, in *FOCS*, 2008, pp. 573–582
51. V. Guruswami, P. Raghavendra, R. Saket, Yi. Wu, Bypassing ugc from some optimal geometric inapproximability results. *Electronic Colloquium on Computational Complexity*, 2010
52. M. Halldórsson, A still better performance guarantee for approximate graph coloring. Inform. Process. Lett. **46**, 169–172
53. J. Håstad, Testing of the long code and hardness for clique, in *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing*, ACM, 1996
54. J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, in *Proceedings of the Thirty Seventh Annual Symposium on the Foundations of Computer Science*, IEEE, 1996, pp. 627–636
55. J. Håstad, Some optimal inapproximability results. J. ACM **48**(4), 798–859 (2001)
56. D. Johnson, Approximation algorithms for combinatorial problems. J. Comput. Syst. Sci. **9**, 256–278 (1974)
57. H. Karloff, U. Zwick A 7/8-approximation for MAX 3SAT?, in *Proc. 38th Ann. IEEE Symp. on Foundations of Comput. Sci.* (IEEE Computer Society, 1997), pp. 406–415
58. S. Khanna, N. Linial, S. Safra, On the hardness of approximating the chromatic number, in *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, 1993

59. S. Khanna, R. Motwani, M. Sudan, U. Vazirani, On syntactic versus computational views of approximability. SIAM J. Comput. **28**, 164–191 (1998). Preliminary Version: Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, 1994, pp. 819–830. (with S. Khanna, M. Sudan, U. Vazirani) Also available as: ECCC Report No. TR95-023, Electronic Colloquium on Computational Complexity, http://www.eccc.uni-trier.de/eccc/, 1995
60. S. Khot, Improved inapproximability results for maxclique, chromatic number and approximate graph coloring, in *FOCS*, 2001, pp. 600–609
61. S. Khot, On the power of unique 2-prover 1-round games, in *STOC*, 2002, pp. 767–775
62. S. Khot, On the unique games conjecture (invited survey), in *IEEE Conference on Computational Complexity*, 2010, pp. 99–121
63. S. Khot, G. Kindler, E. Mossel, R. O'Donnell, Optimal inapproximability results for max-cut and other 2-variable csps? SIAM J. Comput. **37**(1), 319–357 (2007)
64. S. Khot, A. Naor, Sharp kernel clustering algorithms and their associated grothendieck inequalities, in *SODA*, 2010, pp. 664–683
65. S. Khot, O. Regev, Vertex cover might be hard to approximate to within 2-epsilon. J. Comput. Syst. Sci. **74**(3), 335–349 (2008)
66. S. Khot, N.K. Vishnoi, The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into l_1, in *FOCS*, 2005, pp. 53–62
67. G. Kindler, A. Naor, G. Schechtman, The UGC hardness threshold of the p grothendieck problem. Math. Oper. Res. **35**(2), 267–283 (2010)
68. A. Kolla, Spectral algorithms for unique games, in *IEEE Conference on Computational Complexity*, 2010, pp. 122–130
69. A. Kumar, R. Manokaran, M. Tulsiani, N.K. Vishnoi, On lp-based approximability for strict csps, in *SODA*, 2011
70. L. Levin, Universal'nyĭe perebornyĭe zadachi (Universal search problems: in Russian). Problemy Peredachi Informatsii **9**(3), 265–266 (1973). A corrected English translation appears in an appendix to Trakhtenbrot [93]
71. N. Linial, U. Vazirani, Graph products and chromatic numbers, in *Proc. 30th IEEE Symp. on Foundations of Computer Science*, 1989, pp. 124–128
72. L. Lovász, On the ratio of optimal integral and fractional covers. Discrete Math. **13**, 383–390 (1975)
73. C. Lund, L. Fortnow, H. Karloff, N. Nisan, Algebraic methods for interactive proof systems. J. ACM **39**, 859–868 (1992)
74. C. Lund, M. Yannakakis, On the hardness of approximating minimization problems. J. ACM **41**(5), 960–981 (1994)
75. K. Makarychev, Y. Makarychev, How to play unique games on expanders, in *WAOA*, 2010, pp. 190–200
76. E. Mossel, Gaussian bounds for noise correlation of functions and tight analysis of long codes, in *FOCS*, 2008, pp. 156–165
77. E. Mossel, R. O'Donnell, K. Oleszkiewicz, Noise stability of functions with low influences invariance and optimality, in *FOCS*, 2005, pp. 21–30
78. R. O'Donnell, Yi. Wu, Conditional hardness for satisfiable 3-csps, in *STOC*, 2009, pp. 493–502
79. P. Orponen, H. Manilla, On approximation preserving reductions: complete problems and robust measures. Technical Report C-1987-28, Department of Computer Science, University of Helsinki, 1987
80. C. Papadimitriou, M. Yannakakis, Optimization, approximation and complexity classes. J. Comput. Syst. Sci. **43**, 425–440 (1991)
81. C. Papadimitriou, M. Yannakakis, The traveling salesman problem with distances one and two. Math. Oper. Res. (1992)
82. E. Petrank, The hardness of approximation: gap location. Comput. Complex. **4**, 133–157 (1994)
83. P. Raghavendra, Optimal algorithms and inapproximability results for every csp? in *STOC*, 2008, pp. 245–254

84. P. Raghavendra, D. Steurer, Graph expansion and the unique games conjecture, in *STOC*, 2010, pp. 755–764
85. P. Raghavendra, D. Steurer, M. Tulsiani, Reductions between expansion problems (2010)
86. R. Raz, A parallel repetition theorem, in *Proceeding of the 27th STOC*, 1995, pp. 447–456. Journal version in: SIAM J. Comput. **27**(3), 763–803 (1998)
87. R. Raz, S. Safra, A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP, in *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing,* ACM, 1997, pp. 475–484
88. S. Sahni, T. Gonzales, P-complete approximation problems. J. ACM **23**, 555–565 (1976)
89. A. Samorodnitsky, L. Trevisan, A pcp characterization of np with optimal amortized query complexity, in *STOC*, 2000, pp. 191–199
90. A. Shamir, IP = PSPACE. J. ACM **39**(4), 869–877 (1992)
91. M. Sudan, L. Trevisan, Probabilistically checkable proofs with low amortized query complexity, in *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, 1998
92. M. Szegedy, Many valued logics and holographic proofs, in *ICALP*, 1999
93. B. Trakhtenbrot, A survey of Russian approaches to *Perebor* (brute-force search) algorithms. Ann. Hist. Comput. **6**, 384–400 (1984)
94. L. Trevisan, Recycling queries in PCPs and in linearity tests, in *Proceedings of the 30th ACM Symposium on Theory of Computing*, 1998
95. L. Trevisan, G.B. Sorkin, M. Sudan, D.P. Williamson, Gadgets, approximation, and linear programming, in *Proceedings of the 37th Symposium on Foundations of Computer Science*, (IEEE, 1996), pp. 617–626
96. L. Trevisan, Inapproximability of combinatorial optimization problems. Electron. Colloq. Comput. Complex. (065) (2004)
97. L. Trevisan, G.B. Sorkin, M. Sudan, D.P. Williamson, Gadgets, approximation, and linear programming. SIAM J. Comput. **29**(6), 2074–2097 (2000)
98. D. Zuckerman, On unapproximable versions of NP-complete problems. SIAM J. Comput. **25**(6), 1293–1304 (1996)
99. D. Zuckerman, Linear degree extractors and the inapproximability of max clique and chromatic number. Theor Comput. **3**(1), 103–128 (2007)