Inapproximability of Combinatorial Optimization Problems

Luca Trevisan[*]

July 27, 2004

Abstract

We survey results on the hardness of approximating combinatorial optimization problems.

Contents

1 Introduction **2**
1.1 A Brief Historical Overview 2
1.2 Organization of This Survey 5
1.3 Further Reading 5

2 Some Technical Preliminaries **6**

3 Probabilistically Checkable Proofs **7**
3.1 PCP and the Approximability of Max SAT 8

4 Basic Reductions **10**
4.1 Max 3SAT with Bounded Occurrences 10
4.2 Vertex Cover and Independent Set 11
4.3 Steiner Tree 12
4.4 More About Independent Set 14

5 Optimized Reductions and PCP Constructions **15**
5.1 PCPs Optimized for Max SAT and Max CUT 15
5.2 PCPs Optimized for Independent Set 17

6 An Overview of Known Inapproximability Results **18**
6.1 Lattice Problems 18
6.2 Decoding Linear Error-Correcting Codes 19
6.3 The Traveling Salesman Problem 20
6.4 Coloring Problems 21
6.5 Covering Problems 22

[*] `luca@cs.berkeley.edu`. U.C. Berkeley, Computer Science Division. Work supported by NSF grant CCR-9984703, a Sloan Research Fellowship, and an Okawa Foundation Grant.

7 Other Topics **23**
7.1 Complexity Classes of Optimization Problems 23
7.2 Average-Case Complexity and Approximability 24
7.3 Witness Length in PCP Constructions 25
7.4 Typical and Unusual Approximation Factors 26
7.5 Inapproximability Results versus "Integrality Gaps" 27

8 Conclusions **28**

1 Introduction

Hundreds of interesting and important combinatorial optimization problems are NP-hard, and so it is unlikely that any of them can be solved by an efficient exact algorithm. Short of proving P = NP, when one deals with an NP-hard problem one can either hope to design an exact algorithm that runs in polynomial time on "many" instances but has exponential worst-case running time, or to design an efficient algorithm that finds sub-optimal solutions. In this paper we focus on *approximation algorithms*, that are algorithms of the second kind with a provably good worst-case ratio between the value of the solution found by the algorithm and the true optimum.

Formally, we say that an algorithm is r*-approximate* for a minimization problem (respectively, a maximization problem) if, on every input, the algorithm finds a solution whose cost is at most r times the optimum (respectively, at least $1/r$ times the optimum). The ratio r, which is always ≥ 1, is also called the *performance ratio* of the algorithm.

For some problems, it is possible to prove that even the design of an r-approximate algorithm with small r is impossible, unless P = NP. Results of this kind, called *inapproximability* results, are the subject of this survey.

1.1 A Brief Historical Overview

The seeming intractability of many combinatorial optimization problems was observed already in the 1960s, motivating the development of suboptimal heuristic algorithms and, in particular, the notion of approximation algorithm as defined above. An early example of analysis of an approximation algorithm is a paper of Graham on scheduling problems [Gra66].

The theory of NP-completeness [Coo71, Lev73, Kar72], and its success in classifying the complexity of optimization problems, provided added motivation to the study of efficient suboptimal algorithms and to the rigourus analysis of approximation algorithms. A 1973 seminal paper by Johnson [Joh74] gave the field its current foundations. Johnson considers the problems Max SAT, Set Cover, Independent Set, and Coloring.[1] He notes that there is a 2-approximate algorithm for Max SAT, and that this can be improved to 8/7 for Max E3SAT, the restriction of the problem to instances in which every clause contains exactly three literals. He also presents a $(1 + \ln k)$-approximate algorithm for Set Cover, where k is the size of the largest set in the collection, and he notes that natural heuristics for Coloring and Independent Set fail to give even a $n^{1-\varepsilon}$ approximation for $\varepsilon > 0$.

[1]The Max SAT problem is defined in Section 3.1, the Independent Set problem is defined in Section 4.2, the Coloring problem is defined in Section 6.4, and the Set Cover problem is defined in Section 6.5.

Thirty years after Johnson's paper, the design and analysis of approximation algorithms has become a large and successful research area. A book edited by Hochbaum [Hoc96] and two more recent textbooks [ACG+99, Vaz01] give an extensive overview of the field. The book by Ausiello et al. [ACG+99] also includes a list of known results for hundreds of problems.

As the field progressed and matured, it became apparent that different combinatorial optimization problems behaved different from the point of view of approximability. For some problems, researchers were able to devise polynomial time approximation schemes (abbreviated PTAS), that is, to devise an r-approximate algorithm for every $r > 1$. For other problems, such as Max SAT and Vertex Cover,[2] r-approximate algorithms for constant r were known, but no approximation scheme. Other problems like Set Cover had no known constant factor approximation algorithm, but were known to admit an approximation algorithm with a performance ratio that was slowly growing (logarithmic) in the input length. Finally, there were problems like Independent Set and Coloring for which not even a slowly growing performance ratio was known to be achievable in polynomial time. On the other hand, until 1990, there was no negative result showing that the existence of approximation schemes for any of the above problems would imply P = NP or other unlikely consequences. Indeed, few inapproximability results were known at all. Furthermore, there was a general intuition that, in order to prove inapproximability for the above mentioned problems (and others), it was necessary to depart from the standard techniques used to prove the NP-hardness of problems.

To see the limitations of standard NP-hardness proofs, consider for example the proof that solving optimally the Independent Set problem is NP-hard. The proof starts from a 3SAT instance φ and constructs a graph G and an integer k such that if φ is satisfiable then G has an independent set of size k, and if φ is not satisfiable then all independent sets in G have size at most $k - 1$. It then follows that a polynomial time exact algorithm for solving Independent Set, together with the reduction, would imply a polynomial time algorithm for 3SAT, and, from Cook's theorem, we would have a polynomial time algorithm for every problem in NP. Consider now the question of the approximability of the Independent Set problem. Looking at the reduction from 3SAT (that we describe in Section 4.2) we can see that if φ has an assignment that satisfies all the clauses except c ones, then G has an independent set of size $k-c$ and, given such an assignment, the independent set is easy to construct. Furthermore, the instances of 3SAT produced by Cook's theorem are such that it is easy to find assignments that satisfy all the clauses but one. In conclusion, the instances that we get by reducing a generic NP problem to Independent Set are very easy to approximate.

To derive an inapproximability result, we need a much stronger reduction. Suppose we want to prove that no 2-approximate algorithm exists for the Independent Set problem assuming P $\neq$ NP. Then a natural proof strategy would be to start from an NP-complete problem, say, 3SAT, and then reduce an instance φ of 3SAT to an instance G_φ of Independent Set, with the property that, for some k, if φ is satisfiable then

$$OPT_{\text{Independent Set}}(G_\varphi) \geq k \tag{1}$$

and if φ is not satisfiable then

$$OPT_{\text{Independent Set}}(G_\varphi) < k/2 \ . \tag{2}$$

Suppose we have such a reduction, and suppose we also have a 2-approximate algorithm for Independent Set; then the algorithm, given G_φ, will find a solution of cost at least $k/2$ if and only if φ

[2] We define the Vertex Cover problem in Section 4.2.

is satisfiable. The approximation algorithm then gives a polynomial time algorithm for 3SAT, and so no such algorithm can exist if P $\neq$ NP.

All the NP-completeness proofs for graph problems before 1990, however, can be essentially described as follows: we start from the computation of a generic non-deterministic Turing machine, then we encode its computation as a 3SAT formula, using the construction of Cook's theorem, and then we reduce 3SAT to the problem of interest (the reduction may be presented as a sequence of reductions involving several intermediate problems, but it can always be thought of as a direct reduction from 3SAT) by encoding variables and clauses of the formula as sub-graphs connected in a proper way. The computation of a Turing machine is very sensitive to small changes, and it seems impossible to generate an inapproximability gap starting from a fragile model and applying "local" reductions. The only inapproximability results that can be proved with such reductions are for problems that remain NP-hard even restricted to instances where the optimum is a small constant. For example, in the Metric Min k-Center problem it is NP-hard to decide whether the optimum has cost 1 or 2, and so no algorithm can have a performance ratio smaller than 2 unless P = NP [HS85]. Similarly, in the Coloring problem it is NP-hard to decide wether the optimum has cost 3 or 4, and so no algorithm has performance ratio smaller than 4/3 unless P = NP, and Garey and Johnson [GJ76] show that the gap can be "amplified" to k versus $2k - 4$ for constant k, ruling out also algorithms with performance ratio smaller than 2. Most interesting problems, however, become trivial when restricted to inputs where the optimum is a constant.

To prove more general inapproximability results it seemed necessary to first find a machine model for NP in which accepting computations would be "very far" from rejecting computations. Before such a model was discovered, an important piece of work on inapproximability was due to Papadimitriou and Yannakakis, who showed that, assuming that Max 3SAT does not have a PTAS, then several other problems do not have a PTAS [PY91]. Berman and Schnitger [BS92] proved that if Max 2SAT does not have a PTAS then, for some $c > 0$, the Independent Set problem cannot be approximated within a factor n^c.

The modern study of inapproximability was made possible by the discovery, due to Feige et al. [FGL$^+$96] in 1990, that probabilistic proof systems could give a robust model for NP that could be used to prove an inapproximability result for the Independent Set problem.[3] A year later, Arora et al. [AS98, ALM$^+$98] proved the PCP Theorem, a very strong characterization of NP in terms of proof systems, and showed how to use the PCP Theorem to prove that Max 3SAT does not have a PTAS. Using the reductions of [PY91] (and others [PY93, BP89]), the PCP Theorem gave inapproximability results for several other problems.

An explosion of applications to other problems soon followed, such as the work of Lund and Yannakakis [LY94] on Set Cover and Coloring and the work of Arora et al. [ABSS97] on lattice problems, that appeared in 1993. Later work focused on stronger inapproximability results, obtained both by proving stronger versions of the PCP Theorem and by devising better reductions. Despite great technical difficulties, very rapid progress was made in the mid 1990s. It is interesting to read a survey paper by Bellare [Bel96], written in the middle of the most exciting period of that time. Some of the open questions raised in the paper were solved a few months later. We now know inapproximability results for Max 3SAT [Hås01], Set Cover [Fei98], Independent Set [Hås99] and Coloring [FK98] showing that the performance ratios of the algorithms presented in Johnson's paper [Joh74] are best possible (assuming P $\neq$ NP or similar complexity assumptions).

[3] Another, less influential, connection between probabilistic proof checking and inapproximability was discovered around the same time by Condon [Con93].

Despite all these achievements, the study of inapproximability results is still a vibrant field, with radically new insights and major technical achievements emerging at a steady pace.

1.2 Organization of This Survey

To give justice to the breadth and the diversity of this field, we chose to present it from different perspectives.

After a review of technical definitions (Section 2), we disucss the PCP Theorem in Section 3, inapproximability results that follow from the PCP Theorem in Section 4, and stronger results that follow from "optimized" versions of the PCP theorem and/or from "optimized" reductions in Section 5. This is a "textbook" presentation of inapproximability results, similar, for example, to the one in [Vaz01, Chapter 29]. We include some proofs and we discuss problems for which the reductions are relatively simple and yield inapproximability results that are tight or almost tight. The focus of Sections 3, 4, and 5 is on *techniques* used to prove inapproximability results.

In Section 6 we review what is known for various fundamental problems. This section has mostly a reference character, and the focus is on *problems*. We give particular space to problems for which there are important open questions and that the reader could use as research topics.

After approaching the field from the perspectives of techniques and of results for specific problems, we discuss in Section 7 a number of alternate questions that have been pursued. One question relates to the fact that the best possible performance ratios for natural problems that do not have a PTAS were observed to often be $O(1)$, $O(\log n)$ or $n^{O(1)}$. A survey paper by Arora and Lund [AL96] emphasizes this regularity. For a subclass of optimization problems it was actually proved [Cre95, KSTW00] that only a finite number of approximability behaviour were possible. However, we now know of a natural problem for which the best possible performance ratio is $O(\log^* n)$, of another for which it is somewhere between $O((\log n)^2)$ and $O((\log n)^3$, and another for which it is somewhere between $O(\log \log n)$ and $O(\log n / \log \log n)$. Contrary to previous intuition, it now appears that it is not possible to classify the approximability of natural problems into a small number of simple cases. Another topic that we discuss is the relation between instances that are "hard" for linear and semidefinite relaxation and instances that are generated by approximation preserving reductions. We also discuss the study of *complexity classes* of combinatorial optimization problems, of relations between average-case complexity and inapproximability, and of the issue of *witness length* in PCP constructions.

We make some concluding remarks and we discuss a small sample of open questions in section 8.

1.3 Further Reading

Several excellent surveys on inapproximability results have been written.

A paper by Arora and Lund [AL96] gives a very good overview of the techniques used to prove inapproximability results. Our discussion in Section 7.4 is motivated by the classification scheme defined in [AL96]. Arora has written two more survey papers [Aro98a, Aro98a] on PCP and inapproximability.

The subject of strong inapproximability results is discussed in a paper by Bellare [Bel96], that we have already mentioned, and in a more recent paper by Feige [Fei02a].

A book by Ausiello and others [ACG+99] has an extensive compendium of known inapproximability results for hundreds of problems[4] and chapters devoted to PCP, inapproximability results,

[4] At the time of writing, the compendium is available at http://www.nada.kth.se/∼viggo/problemlist/compendium.html.

and complexity classes of optimization problems. The books by Papadimitriou [Pap94] and by Vazirani [Vaz01] have chapters devoted to inapproximability results.

2 Some Technical Preliminaries

We assume the reader is familiar with the basic notions of algorithm, running time and big-Oh notation. Such background can be found, for example, in the introductory chapter of Papadimitriou's textbook [Pap94].

NP-Completeness

We assume that all combinatorial objects that we refer to (graphs, boolean formulas, families of sets) are represented as binary strings. For a binary string x, we denote its length as $|x|$. We represent a decision problem as a language, that is, as the set of all inputs for which the answer is YES. We define P as the class of languages that can be decided in polynomial time. We define NP as the class of languages L such that there is a polynomial time computable predicate V and a polynomial $q()$ such that $x \in L$ if and only if there is w, $|w| \leq q(|x|)$ such that $V(x,w)$ accepts. We think of w as a *proof*, or *witness* that x is in the language.

For two languages L_1 and L_2, we say that L_1 reduces to L_2, and we write $L_1 \leq_m L_2$ if there is polynomial time computable f such that $x \in L_1$ if and only if $f(x) \in L_2$. A language A is NP-hard if every language L in NP reduces to A. A language is NP-complete if it is NP-hard and it belongs to NP.

NPO Problems

A combinatorial *optimization problem* O is defined[5] by a cost function $\mathrm{cost}_O()$ that given an *instance* x of the problem and a *solution* s outputs $\mathrm{cost}_O(x,s)$ which is either the cost of the solution (a non-negative real number) or the special value $\perp$ if s is not a *feasible* solution for x. For every x, there is only a finite number of feasible solutions s such that $\mathrm{cost}_O(x,s) \neq \perp$. If O is a *maximization* problem (respectively, *maximization*), then our goal is, given x to find a feasible solution s such that $\mathrm{cost}(x,s)$ is smallest (respectively, largest). We denote by $\mathrm{opt}_O(x)$ the cost of an optimal solution for x.

For example, in the independent set problem, an instance is an undirected graph $G = (V,E)$, a feasible solution is a subset $S \subseteq V$ such that no two vertices of S are connected by an edge. The cost of a feasible S is the number of vertices in S.

An optimization problem O is an NP-optimization problem, and it belongs to the class NPO, if $\mathrm{cost}_O()$ is computable in polynomial time and if there is a polynomial q such that for every instance and every solution s that is feasible for x we have $|s| \leq q(|x|)$. The independent set problem is clearly an NPO problem.

If P = NP, then for every NPO problem there is a polynomial time optimal algorithm. If P $\neq$ NP, then none of the NPO problems considered in this paper has a polynomial time optimal algorithm.

[5] Other conventions are possible, which are typically equivalent to the one we describe here.

Approximation

If O is an NPO minimization problem, $r > 1$, and A is an algorithm, we say that A is an r-*approximate* algorithm for O if, for every instance x, $A(x)$ is a feasible solution for x and $\text{cost}_O(x, A(x)) \leq r \cdot \text{opt}_O(x)$. If O is a maximization problem and $r \geq 1$, then we say that A is an r-approximate algorithm if $\text{cost}_O(x, A(x)) \geq \text{opt}_O(x)/r$. Another common convention, that we do not use here, is to say that an algorithm is r-approximate for a maximization NPO problem O, with $r \leq 1$ if, for every x, $\text{cost}_O(x, A(x)) \geq r \cdot \text{opt}_O(x)$. Finally, sometimes we let $r = r(n)$ be a function of the length of the instance x, or of some other parameter related to the size of x.

For example, a 2-approximate algorithm for the independent set problem would be an algorithm that given a graph G outputs a feasible independent set S such that $|S| \geq |S^*|/2$, where S^* is an optimal independent set. As we will se later, such an approximation algorithm can exist if and only if P = NP.

Complexity Classes

We have already defined the classes NP and P. We also define, for later reference, classes of problems solvable by efficient probabilistic algorithms. We define BPP as the class of decision problems (languages) that can be solved in polynomial time by a probabilistic algorithm that, on every input, has a probability of error at most 1/3. We also define two subclasses of BPP. The class RP contains decision problems that admit a polynomial time probabilistic algorithm that is correct with probability one if the correct answer is "NO," and is correct with probability at least 1/2 if the correct answer is "YES." The class ZPP contains decision problems that admit a probabilistic algorithm that, on every input, runs in average polynomial time and always returns the correct answer. (The average is taken only over the random choices of the algorithm.)

We say that an algorithm runs in quasi-polynomial time if, for some constant c, it runs in time $O(n^{(\log n)^c})$ on inputs of length n. We denote by QP the class of decision problems solvable by deterministic quasi-polynomial time algorithms, and by QRP the relaxation of RP to quasi-polynomial running time.

It is considered extremely unlikely that NP be contained in any of the complexity classes defined above. Furthermore, if, say NP $\subseteq$ RP, then for every NPO problem there is a quasi-polynomial time probabilistic algorithm that on every input outputs, with high probability, an optimal solution.

3 Probabilistically Checkable Proofs

As discussed in the introduction, probabilistically checkable proofs (PCPs) provide a "robust" characterization of the class NP. When we reduce a generic NP problem L to 3SAT using Cook's theorem, we give a way to transform an instance x into a 3CNF formula φ_x so that if $x \in L$ then φ_x is satisfiable, and if $x \notin L$ then φ_x is not satisfiable. Following the proof of Cook's theorem, however, we see that it is always easy (even when $x \notin L$) to construct an assignment that satisfies all the clauses of φ_x except one.

Using the PCP Theorem one can prove a stronger version of Cook's theorem, that states that, in the above reduction, if $x \in L$ then φ_x is satisfiable, and if $x \notin L$ then there is no assignment that satisfies even a $1 - \varepsilon$ fraction of clauses of φ_x, where $\varepsilon > 0$ is a fixed constant that does not depend on x. This immediately implies that Max 3SAT does not have a PTAS (unless P = NP), and that several other problems do not have a PTAS either (unless P = NP), using the reductions of Papadimitrious and Yannakakis [PY91] and others.

We define PCPs by considering a probabilistic modification of the definition of NP. We consider probabilistic polynomial time verifiers V that are given an input x and "oracle access" to a witness string w. We model the fact that V is a probabilistic algorithm by assuming that V, besides the input x and the witness w, takes an additional input R, that is a sequence of random bits. Then V performs a deterministic computation based on x, w and R. For fixed w and x, when we say "$V^w(x)$ accepts" we mean "the event that V accepts when given oracle access to witness w, input x, and a uniformly distributed random input R." When we refer to the "probability that $V^w(x)$ accepts," we take the probability over the choices of R.

We say that a verifier is $(r(n), q(n))$-restricted if, for every input x of length n and for every w, $V^w(x)$ makes at most $q(n)$ queries into w and uses at most $r(n)$ random bits.

We define the class $\mathrm{PCP}[r(n), q(n)]$ as follows. A language L is in $\mathrm{PCP}[r(n), q(n)]$ if there is an $(r(n), q(n))$-restricted verifier V such that if $x \in L$, then there is w such that $V^w(x)$ accepts with probability 1 and if $x \notin L$ then for every w the probability that $V^w(x)$ accepts is $\leq 1/2$.

We also consider the following more refined notation. We say that a language L is in $\mathrm{PCP}_{c(n),s(n)}[r(n), q(n)]$ if there is an $(r(n), q(n))$-restricted verifier V such that if $x \in L$, then there is w such that $V^w(x)$ accepts with probability at least $c(n)$, and if $x \notin L$ then for every w the probability that $V^w(x)$ accepts is at most $s(n)$. Of course, the definition makes sense only if $0 \leq s(n) < c(n) \leq 1$ for every n. The parameter $c(n)$ is called the *completeness* of the verifier and the parameter $s(n)$ is called the *soundness error*, or simply the *soundness* of the verifier.

Note that if $r(n) = O(\log n)$ then the proof-checking can be *derandomized*, that is, V can be simulated by a polynomial time deterministic verifier that simulates the computation of V on each of the $2^{r(n)} = n^{O(1)}$ possible random inputs and then computes the probability that $V^w(x)$ accepts, and then accepts if and only if this probability is one. It then follows that, for example, $\mathrm{PCP}[O(\log n), O(\log n)] \subseteq \mathrm{NP}$. The PCP Theorem shows a surprising converse.

Theorem 1 (PCP Theorem) $\mathrm{NP} = \mathrm{PCP}[O(\log n), O(1)]$.

The theorem was proved in [AS98, ALM+98], motivated by a relation between PCP and approximation discovered in [FGL+96]. The actual proof relies on previous work, as well as on several new results. A survey paper by David Johnson [Joh92] and the introductions of the theses of Madhu Sudan [Sud92] and Sanjeev Arora [Aro94] give an overview of the line of work that led to the PCP Theorem.

3.1 PCP and the Approximability of Max SAT

In the Max 3SAT problem we are given a 3CNF boolean formula, that is, a boolean formula in conjunctive normal form (AND-of-OR of literals, where a literal is either a variable or the negation of a variable) such that each term in the conjunction is the OR of at most three literals. The goal is to find an assignment that satisfies the largest possible number of terms.

Theorem 2 *The* PCP *Theorem implies that there is an* $\varepsilon_1 > 0$ *such that there is no polynomial time* $(1 + \varepsilon_1)$*-approximate algorithm for MAX-3SAT, unless* P = NP.

Proof: Let $L \in \mathrm{PCP}[O(\log n), q]$ be an NP-complete problem, where q is a constant, and let V be the $(O(\log n), q)$-restricted verifier for L. We describe a reduction from L to Max 3SAT. Given an instance x of L, our plan is to construct a 3CNF formula φ_x with m clauses such that, for some $\varepsilon_1 > 0$ to be determined,

$$\begin{aligned} x \in L &\Rightarrow \varphi_x \text{ is satisfiable} \\ x \notin L &\Rightarrow \text{no assignment satisfies more than } (1-\varepsilon_1)m \text{ clauses of } \varphi_x. \end{aligned} \tag{3}$$

Once (3) is proved, the theorem follows.

We enumerate all random inputs R for V. The length of each string is $r(|x|) = O(\log |x|)$, so the number of such strings is polynomial in $|x|$. For each R, V chooses q positions $i_1^R, \ldots, i_q^R$ and a Boolean function $f_R : \{0,1\}^q \to \{0,1\}$ and accepts iff $f_R(w_{i_1^R}, \ldots, w_{i_q^R}) = 1$.

We want to simulate the possible computation of the verifier (for different random inputs R and different witnesses w) as a Boolean formula. We introduce Boolean variables $x_1, \ldots, x_l$, where l is the length of the witness w.

For every R we add clauses that represent the constraint $f_R(x_{i_1^R}, \ldots, x_{i_q^R}) = 1$. This can be done with a CNF of size $\leq 2^q$, that is, we would need to add at most 2^q clauses if we were allowed to write a qCNF expression. But we also need to "convert" clauses of length q to clauses length 3, which can be done by introducing additional variables, as in the standard reduction from kSAT to 3SAT (for example $x_2 \vee x_{10} \vee x_{11} \vee x_{12}$ becomes $(x_2 \vee x_{10} \vee y_R) \wedge (\bar{y}_R \vee x_{11} \vee x_{12})$). Overall, this transformation creates a formula φ_x with at most $q2^q$ 3CNF clauses.

Let us now see the relation between the optimum of φ_z as an instance of Max 3SAT and the question of whether $z \in L$.

If $z \in L$, then there is a witness w such that $V^w(z)$ accepts for every R. Set $x_i = w_i$, and set the auxiliary variables appropriately, then the assignment satisfies all clauses, and φ_z is satisfiable.

If $z \notin L$, then consider an arbitrary assignment to the variables x_i and to the auxiliary variables, and consider the string w where w_i is set equal to x_i. The witness w makes the verifier reject for half of the $R \in \{0,1\}^{r(|z|)}$, and for each such R, one of the clauses representing f_R fails. Overall, at least a fraction $\varepsilon_1 = \frac{1}{2}\frac{1}{q2^q}$ of clauses fails. □

Interestingly, the converse also holds: any gap-creating reduction from an NP-complete problem to Max 3SAT implies that the PCP Theorem must be true.

Theorem 3 *If there is a reduction as in (3) for some problem L in* NP*, then $L \in$ PCP$[O(\log n), O(1)]$. In particular, if L is* NP*-complete then the* PCP *Theorem holds.*

Proof: We describe how to construct a verifier for L. V on input z expects w to be a satisfying assignment for φ_z. V picks $O(\frac{1}{\varepsilon_1})$ clauses of φ_z at random, and checks that w satisfies all of them. The number of random bits used by the verifier is $O(\frac{1}{\varepsilon_1}\log m) = O(\log |z|)$. The number of bits of the witness that are read by the verifier is $O(\frac{1}{\varepsilon_1}) = O(1)$.

$$\begin{aligned} z \in L &\Rightarrow \varphi_z \text{ is satisfiable} \\ &\Rightarrow \exists w \text{ such that } V^w(z) \text{ always accept.} \end{aligned}$$

$$\begin{aligned} z \notin L &\Rightarrow \forall w \text{ a fraction } \varepsilon_1 \text{ of clauses of } \varphi \text{ are unsatisfied by } w \\ &\Rightarrow \forall w\ V^w(z) \text{ rejects with probability } \geq \tfrac{1}{2} \end{aligned}$$

□

4 Basic Reductions

We have seen that the PCP Theorem is equivalent to the inapproximability of Max 3SAT and other constraint satisfaction problems. In this section we will see several reductions that prove inapproximability results for other problems.

4.1 Max 3SAT with Bounded Occurrences

We begin with a reduction from the Max E3SAT problem on general instances to the restriction of Max E3SAT to instances in which every variable occurrs only in a bounded number of clauses. The latter problem will be a useful starting point for other reductions.

For the reduction we will need *expander graphs* of the following type.

Definition 4 (Expander Graph) *An undirected graph $G = (V, E)$ is a 1-expander if, for every subset $S \subseteq V$, $|S| \leq |V|/2$, the number of edges $e(S, V - S)$ having one endpoint in S and one in $V - S$ is at least $|S|$.*

For our purposes, it will be acceptable for the expander graph to have multiple edges. It is easy to prove the existence of constant-degree 1-expanders using the probabilistic method. Polynomial-time constructible 1-expanders of constant degree can be derived from [GG81], and, with a smaller degree, from [LPS88]. Let d be a constant for which degree-d 1-expanders can be constructed in polynomial time. ($d = 14$ suffices using the construction of [LPS88].)

Let now φ be an instance of 3SAT with n variables $x_1, \ldots, x_n$ and m clauses. For each variable x_i, let occ_i be the number of *occurrences* of x_i, that is, the number of clauses that involve the literal x_i or the literal $\bar{x}_i$. We write $x_i \in C_j$ if the variable x_i occurs in clause C_j. Notice that $\sum_{i=1}^{n} occ_i = 3m$. For each i, construct a 1-expander graph $G_i = (V_i, E_i)$ where V_i has occ_i vertices, one for each occurrence of x_i in φ. We denote the vertices of V_i as pairs $[i, j]$ such that x_i occurrs in C_j. Each of these graphs has constant degree d.

We define a new instance ψ of Max E3SAT with $N = 3m$ variables $Y = \{y_{i,j}\}_{i \in [n], x_i \in C_j}$, one for each occurrence of each variable in φ. For each clause of φ we put an equivalent clause in ψ. That is, if $C_j = (x_a \vee x_b \vee x_c)$ is a clause in φ, then $(y_{a,j} \vee y_{b,j} \vee y_{c,j})$ is a clause in ψ. We call these clauses the *primary clauses* of ψ. Note that each variable of ψ occurs only in one primary clause.

To complete the construction of ψ, for every variable x_i in φ, and for every edge $([i, j], [i, j'])$ in the graph G_i, we add the clauses $(y_{i,j} \vee \bar{y}_{i',j})$ and $(\bar{y}_{i,j} \vee y_{i',j})$ to ψ. We call these clauses the *consistency clauses* of ψ. Notice that if $y_{i,j} = y_{i',j}$ then both consistency clauses are satisfied, while if $y_{i,j} \neq y_{i',j}$ then one of the two consistency clauses is contradicted.

This completes the construction of ψ. By construction, every variable occurrs in at most $2d + 1$ clauses of ψ, and ψ has $M = m + 3dm$ clauses.

We now claim that the cost of an optimum solution in ψ is determined by the cost of an optimum solution in φ and, furthermore, that a good approximation algorithm applied to ψ returns a good approximation for φ. We prove the claim in two steps.

Claim 5 *If there is an assignment for φ that satisfies $m - k$ clauses, then there is an assignment for ψ that satisfies $\geq M - k$ clauses.*

PROOF: This part of the proof is simple: take the assignment for φ and then for every variable $y_{i,j}$ of ψ give to it the value that the assignment gives to x_i. This assignment satisfies all the consistency clauses and all but k of the remaining clauses. $\square$

Claim 6 *If there is an assignment for ψ that leaves k clauses not satisfied, then there is an assignment for φ that leaves $\leq k$ clauses not satisfied.*

PROOF: This is the interesting part of the proof. Let $a_{i,j}$ be the value assigned to $y_{i,j}$. We first "round" the assignment so that all the consistency clauses are satisfied. This is done by defining an assignment b_i, where, for every i, the value b_i is taken to be the majority value of $a_{i,j}$ over all j such that $x_i \in C_j$, and we assign the value b_i to all the variables $y_{i,j}$. The assignment b_i satisfies all the consistency clauses, but it is possible that it contradicts some primary clauses that were satisfied by $a_{i,j}$. We claim that, overall, the b_i assignment satisfies at least as many clauses as the $a_{i,j}$ assignment. Indeed, for each i, if b_i differs from the $a_{i,j}$ for, say, t values of j, then there can be at most t primary clauses that were satisfied by $a_{i,j}$ but are contradicted by b_i. On the other hand, because of the consistency clauses being laid out as the edges of a 1-expander graph, at least t consistency clauses are contradicted by the $a_{i,j}$ assignment for that value of i alone, and so, the b_i assignment can be no worse.

We conclude that b_i assignment contradicts no more clauses of ψ than are contradicted by $a_{i,j}$, that is, no more than k clauses. When we apply b_i as an assignment for φ, we see that b_i contradicts at most k clauses of φ. $\square$

In conclusion:

- If φ is satisfiable then ψ is satisfiable;
- If every assignment contradicts at least an ε fraction of the clauses of φ, then every assignment contradicts at least an $\varepsilon/(1 + 3d)$ fraction of the clauses of ψ.

Theorem 7 *There are constants d and ε_2 and a polynomial time computable reduction from 3SAT to Max 3SAT-d such that if φ is satisfiable then $f(\varphi)$ is satisfiable, and if φ is not satisfiable then the optimum of $f(\varphi)$ is less than $1 - \varepsilon_2$ times the number of clauses. In particular, if there is an approximation algorithm for Max 3SAT-d with performance ratio better than $(1-\varepsilon_2)$, then* P = NP.

4.2 Vertex Cover and Independent Set

In an undirected graph $G = (V, E)$ a vertex cover is a set $C \subseteq V$ such that for every edge $(u, v) \in E$ we have either $u \in C$ or $v \in C$, possibly both. An independent set is a set $S \subseteq V$ such that for every two vertices $u, v \in S$ we have $(u, v) \notin E$. It is easy to see that a set C is a vertex cover in G if and only if $V - C$ is an independent set. It then follows that the problem of finding a minimum size vertex cover is the same as the problem of finding a maximum size independent set. From the point of view of approximation, however, the two problems are not equivalent: the Vertex Cover problem has a 2-approximate algorithm (but, as we see below, it has no PTAS unless P = NP), while the Independent Set problem has no constant-factor approximation unless P = NP.

We give a reduction from Max E3SAT to Independent Set. The reduction will also prove intractability of Vertex Cover. If we start from an instance of Max E3SAT-d we will get a bounded degree graph, but the reduction works in any case. The reduction appeared in [PY91], and it is similar to the original proof of NP-completeness of Vertex Cover and Independent Set [Kar72].

Starting from an instance φ of E3SAT with n variables and m clauses, we constuct a graph with $3m$ vertices; the graph has a vertex $v_{i,j}$ for every occurrence of a variable x_i in a clause C_j. For each clause C_j, the three vertices corresponding to the three literals in the clause are joined by

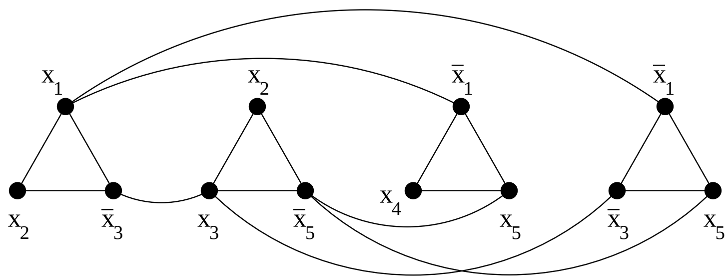

Figure 1: Graph construction corresponding to the 3CNF formula $\varphi = (x_1 \ \vee \ x_2 \ \vee \ \bar{x}_3) \wedge (x_2 \ \vee \ x_3 \ \vee \ \bar{x}_5) \wedge (\bar{x}_1 \ \vee \ x_4 \ \vee \ x_5) \wedge (\bar{x}_1 \ \vee \ \bar{x}_3 \ \vee \ x_5)$.

edges, and form a triangle (we call such edges *clause edges*). Furthermore, if a variable x_i occurrs positively in a clause C_j and negated in a clause $C_{j'}$, then there is an edge between the vertices $v_{i,j}$ and $v_{i,j'}$ (we call such edges *consistency edges*). Let us call this graph G_φ. See Figure 1 for an example of this construction.

Note that if every variable occurrs in at most d clauses then the graph has degree at most $d+2$.

Claim 8 *There is an independent set of size $\geq t$ in G_φ if and only if there is an assignment that satisfies $\geq t$ clauses in φ.*

PROOF: Suppose we have an assignment a_i that satisfies t clauses. For each clause C_j, let us pick a vertex $v_{i,j}$ that corresponds to a literal of C_j satisfied by a_i. We claim that the set of picked vertices is an independent set in G_φ. To prove the claim, we note that we picked at most one vertex from each triangle, so that we do not violate any clause edge, and we picked vertices consistent with the assignment, so that we could not violate any consistency edge.

For the other direction, suppose we have an independent set with t vertices. The vertices must come from t different triangles, corresponding to t different clauses. We claim that we can satisfy all such clauses. We do so by setting an assingment so that x_i takes a value consistent with the vertices $v_{i,j}$ in the independent set, if any. Since consistency edges cannot be violated, this is a well defined assignment, and it satisfies t clauses. $\square$

If we combine this reduction with Theorem 7, we get the following result.

Theorem 9 *There is a polynomial time computable function mapping instances φ of 3SAT into graphs G_φ of maximum degree $d+2$ such that if φ is satisfiable then G_φ has an independent set of size at least $N/3$ (and a vertex over of size at most $2N/3$, where N is the number of vertices, and if φ is not satisfiable then every independent set in G_φ has size at most $N \cdot (1-\varepsilon_2)/3$, and every vertex cover has size at least $N \cdot (2+\varepsilon_2)/3$. In particular, if there is an approximation algorithm for Independent Set in degree-$(d+2)$ graphs with performance ratio better than $1/(1-\varepsilon_2)$, or if there is an approximation algorithm for Vertex Cover in degree-$(d+2)$ graphs with performance ratio better than $1+\varepsilon_2/2$, then* $\mathrm{P} = \mathrm{NP}$.

4.3 Steiner Tree

In the Steiner tree problem we are given a graph $G = (V, E)$, with weights on the edges, and a subset $C \subseteq V$ of vertices. We want to find the tree of minimum cost that contains all the vertices of C. This problem is different from the minimum spanning tree problem because the tree is not required to contain the vertices in $V-C$, although it may contain some of them if this is convenient. An interesting special cases (called *Metric Steiner Tree*) arises when E is the complete graph and the weights are a metric. (That is, they satisfy the triangle inequality.) Without this restriction, it is not hard to show that the problem cannot be approximated within any constant.

We describe a reduction from the Vertex Cover problem in bounded degree graphs to the Steiner Tree problem. The reduction is due to Bern and Plassmann [BP89].

We start from a graph $G=(V,E)$, and we assume that G is connected.[6] We define a new graph G' that has $|V|+|E|$ vertices, that is, a vertex $[v]$ for each vertex v of G and a vertex $[u,v]$ for each edge (u,v) of G.

The distances in G' are defined as follows:

- For every edge $(u,v)\in E$, the vertices $[u]$ and $[u,v]$ are at distance one, and so are the vertices $[v]$ and $[u,v]$.
- Any two vertices $[u]$, $[v]$ are at distance one.
- All other pairs of vertices are at distance two.

We let C be the set of vertices $\{[u,v] : (u,v)\in E\}$. This completes the description of the instance of the Steiner Tree problem. Notice that, since all the distances are either one or two, they satisfy the triangle inequality, and so the reduction always produces an instance of Metric Steiner Tree.

Claim 10 *If there is a vertex cover in G with k vertices, then there is a Steiner tree in G' of cost $m+k-1$.*

PROOF: Let S be the vertex cover. Consider the vertices $\{[v] : v\in S\}$ and the vertices $\{[u,v] : (u,v)\in E\}$, and consider the weight-one edges between them. We have described a connected sub-graph of G', because every vertex in $\{[u] : u\in S\}$ is connected to every other vertex in the same set, and every vertex $[u,v]$ is connected to a vertex in $\{[u] : u\in S\}$. Let us take any spanning tree of this subgraph. It has $m+k-1$ edges of weight one, and so it is of cost $m+k-1$, and it is a feasible solution to the Steiner Tree problem. □

Claim 11 *If there is a Steiner tree in G' of cost $\leq m+k$, then there is a vertex cover in G with k vertices.*

PROOF: Let T be a feasible Steiner tree. We first modify the tree so that it has no edge of cost 2. We repeatedly apply the following steps.

- If there is an edge of cost 2 between a vertex $[w]$ and a vertex $[u,v]$, we remove it and add the two edges $([w],[u])$ and $([u],[u,v])$ of cost 1.
- If there is an edge of cost 2 between a vertex $[u,v]$ and a vertex $[v,w]$, we remove it and add the two edges $([u,v],[v])$ and $([v],[v,w])$ of cost 1.

[6] We proved inapproximability of Vertex Cover without guaranteeing a connected graph. Clearly, if we have an approximation algorithm that works only in connected graphs, we can make it work on general graphs with same factor. It follows that any inapproximability for general graphs implies inapproximability of connected graphs with the same factor.

- Finally, and this case is more interesting, if there is an edge of cost 2 between the vertices $[u, v]$ and $[w, z]$, we remove the edge, and then we look at the two connected components into which T has been broken. Some vertices $[u, v]$ are in one component, and some vertices are in the other. This corresponds to a partition of the edges of G into two subsets. Since G is connected, we see that there must be two edges on different sides of the partition that share an endpoint. Let these edges be (u, v) and (v, w) then we can reconnect T by adding the edges $([u, v], [v])$ and $([v], [v, w])$.

We repeat the above steps until no edges of cost two remain. This process will not increase the cost, and will return a connected graph. We can obtain a tree by removing edges, and improving the cost, if necessary.

The final tree has only edges of weight one, and it has a cost $\leq m + k - 1$, so it follows that it spans $\leq m + k$ vertices. The m vertices $\{[u, v] : (u, v) \in E\}$ must be in the tree, so the tree has $\leq k$ vertices $[v]$. Let S be the set of such vertices. We claim that this is a vertex cover for G. Indeed, for every edge (u, v) in G, the vertex $[u, v]$ is connected to v_0 in the tree using only edges of weight one, which means that either $[u]$ or $[v]$ is in the tree, and that either u or v is in S. □

If we combine the reduction with the results of Theorem 9, we prove the following theorem.

Theorem 12 *There is a constant ε_3 such that if there is a polynomial time $(1 + \varepsilon_3)$-approximate algorithm for Metric Steiner Tree then* $\mathrm{P} = \mathrm{NP}$.

4.4 More About Independent Set

In this section we describe a direct reduction from PCP to the Independent Set problem. This reduction is due to Feige et al. [FGL+96].

Let L be NP-complete, and V be a verifier showing that $L \in \mathrm{PCP}_{c,s}[q(n), r(n)]$. For an input x, let us consider all possible computations of $V^w(x)$ over all possible proofs w; a complete description of a computation of V is given by a specification of the randomness used by V, the list of queries made by V into the proof, and the list of answers. Indeed, for a fixed input x, each query is determined by x, the randomness, and the previous answers, so that it is enough to specify the randomness and the answers in order to completely specify a computation. We call such a description a *configuration*. Note that the total number of configuration is at most $2^{r(n)} \cdot 2^{q(n)}$, where n is the length of x.

Consider now the graph G_x that has a vertex for each *accepting* configuration of V on input x, and has an edge between two configurations c, c' if c and c' are *inconsistent*, that is, if c and c' specify a query to the same location and two different answers to that query. We make the following claims.

Claim 13 *If $x \in L$, then G_x has an independent set of size $\geq c \cdot 2^{r(n)}$.*

PROOF: If $x \in L$, then there is a proof w such that $V^w(x)$ accepts with probability at least c, that is, there is a proof w such that there are at least $c \cdot 2^{r(n)}$ random inputs that make $V^w(x)$ accept. This implies that there are at least $c \cdot 2^{r(n)}$ mutually consistent configurations in the graph G_x, and they form an independent set. □

Claim 14 *If $x \notin L$, then every independent set of G_x has size $\leq s \cdot 2^{r(n)}$.*

PROOF: We prove the contrapositive: we assume that there is an independent set in G_x of size $\geq s \cdot 2^{r(n)}$, and we show that this implies $x \in L$. Define a witness w as follows: for every configuration in the independent set, fix the bits in w queried in the configuration according to the answers in the configurations. Set the bits of w not queried in any configuration in the independent set arbitrarily, for example set them all to zero. The $s \cdot 2^{r(n)}$ configurations in the independent set correspond to as many different random strings. When $V^w(x)$ picks any such random string, it accepts, and so $V^w(x)$ accepts with probability at least s, implying $x \in L$. □

It follows that if there is a ρ-approximate algorithm for the independent set problem, then every problem in $\mathrm{PCP}_{c,s}[r(n), q(n)]$ can be solved in time $\mathrm{poly}(n, 2^{r(n)+q(n)})$, provided $c/s < \rho$.

From the PCP Theorem we immediately get that there cannot be a ρ-approximate algorithm for the independent set problem with $\rho < 2$ unless P = NP, but we can do better.

Suppose that V is a $(O(\log n), O(1))$-restricted verifier for an NP-complete problem, and that V has soundness $1/2$ and completeness 1. Define a new verifier V' that performs two independent repetitions of the computation of V, and that accepts if and only if both repetitions accept. Then V' has clearly soundness $1/4$ and completeness 1, and it is still $(O(\log n), O(1))$-restricted, thus showing that even an approximation better than 4 is infeasible. If we repeat V a constant number of times, rather than twice, we can rule out any constant factor approximation for the independent set problem.

In general, a verifier that makes $k(n)$ repetitions shows that $L \in \mathrm{PCP}_{1,1/2^{k(n)}}[O(k(n) \cdot \log n), O(k(n))]$, and the reduction to Independent Set produces graphs that have $2^{O(k(n)\cdot \log n)}$ vertices and for which $2^{k(n)}$-approximate algorithms are infeasible. If we let $k(n) = \log n$, then the graph has size $N = 2^{O((\log n)^2)}$ and the infeasible ratio is n, which is $2^{\Omega(\sqrt{\log N})}$. So, if we have an algorithm that on graphs with N vertices runs in polynomial time and has an approximation ratio $2^{o(\sqrt{\log N})}$, then we have an $O(n^{O(\log n)})$ algorithm to solve 3SAT, and NP $\subseteq$ QP. More generally, by setting $k(n) = (\log n)^{O(1)}$, we can show that if there is an $\varepsilon > 0$ such that Independent Set can be approximated within a factor $2^{O((\log n)^{1-\varepsilon})}$ then NP $\subseteq$ QP.

Finally, using random walks in expander graphs as in [IZ89], it is possible to use the PCP theorem to show that, for every $k(n)$, NP $= \mathrm{PCP}_{1,1/2^{k(n)}}[O(k(n) + \log n), O(k(n))]$. If we choose $k(n) = \log n$, then, in the reduction, we have a graph of size $2^{O(k(n)+\log n)} = n^{O(1)}$ for which an approximation ratio of n is infeasible. This shows the following result.

Theorem 15 *There is a constant $c > 1$ such that if there is a polynomial time n^c-approximate algorithm for Independent Set then* P = NP.

5 Optimized Reductions and PCP Constructions

In this section we give an overview of tighter inapproximability results proved with a combination of optimized PCP constructions and optimized reductions.

5.1 PCPs Optimized for Max SAT and Max CUT

In the reduction from a $(O(\log n), q)$-restricted verifier to Max SAT we lost a factor that was exponential in q. A more careful analysis shows that the loss depends on the size of the CNF that expresses a computation of the verifier. To prove stronger inapproximability results for Max SAT one needs a verifier that, at the same time, has a good separation between soundness and completeness and a simple acceptance predicate. Progress along these lines is reported in [BGLR93, FK94, BS94, BGS98].

The optimal inapproximability result for Max 3SAT was proved by Håstad [Hås01], based on the following PCP construction.

Theorem 16 (Håstad [Hås01]) *For every* $\varepsilon > 0$, $\mathrm{NP} = \mathrm{PCP}_{1-\varepsilon,1/2+\varepsilon}[O(\log n), 3]$. *Furthermore, the verifier behaves as follows: it uses its randomness to pick three entries* i, j, k *in the witness and a bit* b, *and it accepts if and only if* $w_i \oplus w_j \oplus w_k = b$.

The furthermore part immediately implies that for every problem in NP, say, SAT for concreteness, and for every $\varepsilon > 0$ there is a reduction that given a 3CNF formula φ constructs a system of linear equations over $GF(2)$ with three variables per equation. If φ is satisfiable then there is an assignment to the variables that satisfies a $1-\varepsilon$ fraction of the equations, and if φ is not satisfiable then there is no assignment that satisfies more than a $1/2+\varepsilon$ fraction of equations. In particular, it is not possible to approximate the Max E3LIN-2 problem to within a factor better than 2 unless $\mathrm{P} = \mathrm{NP}$.

To reduce an instance I of Max E3LIN-2 to an instance φ_I of Max 3SAT, we simply take every equation $x_i \oplus x_j \oplus x_k = b$ in I and express it as the conjunction of 4 clauses, then φ_I is the conjunction of all these clauses. Let m be the number of equations in I, then φ_I has $4m$ clauses. If $\geq m(1-\varepsilon)$ of the E3LIN equations could be satisfied, then $\geq 4m(1-\varepsilon)$ of the clauses of φ_I can be satsfied, using the same assignment. Conversely, if it is impossible to satisfy more than $m(1/2+\varepsilon)$ equations in the E3LIN instance I then it is impossible to satisfy more than $3.5m + \varepsilon m$ clauses in φ_I. An approximation algorithm for Max 3SAT with a performance ratio better than 8/7 allows to distinguish between the two cases, if ε is small enough, and so such an algorithm is impossible unless $\mathrm{P} = \mathrm{NP}$.

Theorem 17 *If there is an* r*-approximate algorithm for Max 3SAT, where* $r < 8/7$, *then* $\mathrm{P} = \mathrm{NP}$.

A similar "local" reduction is possible from Max E3LIN-2 to Max CUT. For the purpose of the reduction it is easier to think of Max CUT as the boolean constraint satisfaction problem where we have Boolean variables $x_1, \ldots, x_n$ (one for every vertex in the graph) and constraints of the form $x_i \neq x_j$ (one for every edge (i,j) in the graph); we call constaints of this form "cut constraints." To reduce a Max E3LIN-2 instance I to a Max CUT instance G_I, we take each constraint $x_i \oplus x_j \oplus x_k = b$ of I and we construct a collection of cut constraints over x_i, x_j, x_k, plus auxiliary variables. The construction (from [TSSW00]) is such that an assignment to x_i, x_j, x_k that satisfies the E3LIN-2 constraint with $b = 0$ can be extended to the auxiliary variable so that it satisfies 9 of the cut constraints, while an assignment that contradicts the E3LIN-2 constraint, no matter how it is extended to the auxiliary variables, satisfies at most 8 cut constraints (and 8 constraints can always be satisfied by setting the auxiliary variables appropriately). When $b = 1$, a similar construction exists and the number of satisfied constraints is 8 versus 7 (instead of 9 versus 8). In E3LIN-2 instances that are derived from Håstad's PCP there are as many equations with right-hand side 0 as equations with right-hand side 1. Given all these facts, simple computations show that

Theorem 18 *If there is an* r*-approximate algorithm for Max CUT, where* $r < 17/16$, *then* $\mathrm{P} = \mathrm{NP}$.

The best known approximation algorithm for Max CUT is due to Goemans and Williamson [GW95], and its performance ratio is $1/\alpha$ where $\alpha \approx .878$. Khot et al. [KKMO04] describe a PCP system that may prove that the algorithm of Goemans and Williamson is best possible. In particular, if the "unique game conjecture" is true[7] and if a conjecture about extremal properties of monotone functions is true, then there is no polynomial time approximation algorithm for Max CUT with performance ratio better than $1/\alpha$, unless $\mathrm{P} = \mathrm{NP}$.

5.2 PCPs Optimized for Independent Set

The graph in the reduction of Feige et al. [FGL+96] has a vertex for every *accepting* computation. This implies that the important parameter to optimize (in order to derive strong inapproximability results for Independent Set) is not the number of queries but rather the number of accepting computations. This intuition motivates the following definition.

Definition 19 (Free Bit Complexity) *We say that a $(r(n), q(n))$-restricted verifier V for a language L* has free bit complexity at most $f(n)$ *if, for every input x and every choice of the random input, there are at most $2^{f(n)}$ possible answers to the $q(n)$ queries made by the verifier that would make it accept.*

In the reduction from PCP to Independent Set, if we start from a $(r(n), q(n))$-restricted verifier of free bit complexity $f(n)$, then we construct a graph with $\leq 2^{r(n)+f(n)}$ vertices, so that, all other things being equal, a lower free bit complexity yields a smaller graph and so a stronger relation between inapproximability and graph size. A further inspection of the reduction of [FGL+96] shows that what really matters is the ratio $f(n)/\log_2(1/s(n))$, where $s(n)$ is the soundness of the verifier, assuming that the verifier has completeness 1. This measure is called the *amortized free bits complexity* of the verifier.

Theorem 20 (Optimized FGLSS Reduction [BGS98]) *Suppose that an* NP*-complete problem has an $(O(\log n), O(\log n))$-restricted verifier of amortized free bit complexity $\bar{f}$, and suppose that, for some $\varepsilon > 0$, there is an $n^{\varepsilon+1/(\bar{f}+1)}$-approximate algorithm for Independent Set. Then* NP = ZPP.

In order to prove strong inapproximability results it is then enough to provide a version of the PCP Theorem with a verifier having low amortized free bit complexity. Håstad proved the stronger possible result, stating that an arbitrarily small amortized free bit complexity is sufficient.

Theorem 21 (Håstad [Hås99]) *For every $\delta > 0$, 3SAT has an $(O(\log n), O(1))$-restricted verifier of amortized free bit complexity $\leq \delta$. Therefore, assuming* ZPP $\neq$ NP*, for every $\varepsilon > 0$ there is no $n^{1-\varepsilon}$-approximate algorithm for the Independent Set problem.*

The proof of this result was substantially simplified in [ST00, HW01].

If a graph $G = (V, E)$ has maximum degree d, then a maximal independent set contains at least $|V|/(d+1)$ vertices, and so is a $(d+1)$-approximate solution. This can be slightly improved, and an $O(d \log\log d/\log d)$-approximate algorithm is also known [KMS98, AK98, Vis96, Hal98]. Trevisan [Tre01] proves that no $(d/2^{O(\sqrt{\log d})})$-approximate algorithm exists unless $\mathrm{P} = \mathrm{NP}$.

[7] The unique game conjecture, posed by Khot [Kho02b], is about the existence of a PCP system for NP with certain properties. See also Section 8.

6 An Overview of Known Inapproximability Results

After the overview of techniques given in the previous sections, we devote this section to an overview of known inapproximability results for a few selected problems. Some more inapproximability results for specific problems are also discussed in Section 7.4 below.

6.1 Lattice Problems

Let $B = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a set of linearly independent vectors in $\mathbb{R}^n$. The *lattice* generated by B is the set of all points of the form $\sum_i a_i \mathbf{v}_i$ where a_i are integers. For example, suppose $n = 2$ and $B = \{(1/2, 0), (0, 1/2)\}$. Then the lattice generated by B contains all points in $\mathbb{R}^2$ whose coordinates are half-integral.

Shortest Vector Problem

The *Shortest Vector Problem*, abbreviated SVP, is the problem of finding the shortest non-zero vector of a lattice, given a base for the lattice. By "shortest vector," we mean the vector of smallest ℓ_2 norm, where the ℓ_2 norm of a vector $\mathbf{v} = (v_1, \ldots, v_n)$ is $||\mathbf{v}||_2 = \sqrt{\sum_i v_i^2}$.

SVP is a non-trivial problem because the shortest vector may not be any of the basis vectors, and may in fact be a complicated linear combination of the basis vectors. For example, suppose that the basis is $\{(3, 4), (4, 5)\}$. Then one can immediately see that a shorter vector $(1, 1)$ can be derived by subtracting one base vector from the other; it takes some more attention to see that a shortest vector can be derived as $(0, 1) = 4\cdot(3, 4)-3\cdot(4, 5)$. Of course the complexity of the problem escalates with the number of dimension and, perhaps surprisingly, the best known approximation algorithms for SVP achieve only an *exponential* approximation as a function of n, and even achieving such an approximation is highly non-trivial [LLL82, Sch87, AKS01]. The best known approximation algorithm has performance ratio $2^{O(n(\log\log n)/\log n)}$ [AKS01]. On the other hand, Aharonov and Regev [AR04] prove than an $O(\sqrt{n})$-approximation can be computed in NP $\cap$ *co*NP,[8] and so the problem of finding $O(\sqrt{n})$-approximate solutions cannot be NP-hard unless NP = *co*NP.

Interest in the SVP is motivated in part by a result of Ajtai [Ajt96], showing that if n^c approximation of SVP cannot be done in polynomial time, for a certain constant c, then there is a problem in NP that does not have any algorithm running in polynomial time on average. Ajtai's result was the first one to show that a problem in NP is hard on average provided that another problem in NP is hard on worst-case. Even more interestingly, Ajtai and Dwork [AD97] presented a public key cryptosystem that is secure provided that a version of SVP is hard to approximate. The results of Ajtai and Dwork [Ajt96, AD97] have been later improved by Regev [Reg03] and by Micciancio and Regev [MR04]. Currently, the cryptosystems and average-case complexity results of [Ajt96, AD97, Reg03, MR04] depend on the worst-case complexity of problems known to be in NP $\cap$ *co*NP, but it is conceivable that the analyses could be improved so that they rely on problems that are NP-hard to approximate. If so, there would be average-case hard problems, one-way functions and even private-key encryption be based on the assumption that P $\neq$ NP (or, more precisely, that NP $\not\subseteq$ BPP). Towards this goal, one needs, of course, strong inapproximability results for SVP.

[8] We have not quite defined what it means to have an approximation algorithm in NP $\cap$ *co*NP. See [AR04] for a discussion.

The first inapproximability result for SVP was proved by Micciancio [Mic01], who showed that there is no $(\sqrt{2} - \varepsilon)$-approximate algorithm for SVP for any $\varepsilon > 0$, unless RP = NP. This result has been substantially improved by Khot [Kho04a], who proves that no $O(1)$-approximate algorithm for SVP exists unless RP = NP, and that no $2^{(\log n)^{1/2-\varepsilon}}$-approximate algorithm exists unless NP $\subseteq$ QRP, for every $\varepsilon > 0$.

Closest Vector Problem

In the *Closest Vector Problem*, abbreviated CVP, we are given a lattice and a vector, and the problem is to find the lattice element that is closest (in ℓ_2 norm) to the given vector. The approximation algorithms that are known for the SVP can be adapted to approximate CVP within the same ratios [LLL82, Sch87, AKS01], and the techniques of Aharonov and Regev also show that achieving an $O(\sqrt{n})$ approximation is in NP $\cap$ *co*NP [AR04].

The best inapproximability result for CVP is due to Dinur et al., who show that there can be no $n^{1/O(\log \log n)}$-approximate algorithm unless P = NP [DKRS03].

Other Norms

Interesting variants of the SVP and CVP arise when we measure the length of vectors using other ℓ_p norms, where the ℓ_p norm of a vector $\mathbf{v} = (v_1, \ldots, v_n)$ is $||\mathbf{v}||_p = (\sum_i |v_i|^p)^{1/p}$; the problems are then denoted SVP_p and CVP_p, respectively. Of interest is also the case of the ℓ_∞ norm $||v||_\infty = \max_i |v_i|$, that defines the problems SVP_∞ and CVP_∞.

Any ℓ_p norm is a $O(\sqrt{n})$ approximation of the ℓ_2 norm, so the exponential approximation of [LLL82, Sch87, AKS01] also apply to other ℓ_p norm.

For SVP_p, Micciancio's result [Mic01] proves a hardness of approximation within $2^{1/p} - \varepsilon$ for any ℓ_p norm and $\varepsilon > 0$. For every $\delta > 0$ and for every sufficiently large p, Khot [Kho03] proves hardness $p^{1-\delta}$. The new result of Khot [Kho04a] rules out r-approximate algorithms for SVP_p for every constant r and every $1 < p < \infty$, assuming RP $\neq$ NP, and rules out $2^{(\log n)^{1/2-\varepsilon}}$-approximate algorithm for every $1 < p < \infty$ and $\varepsilon > 0$, assuming NP $\not\subseteq$ QRP. Micciancio's result remains the strongest known one for the ℓ_1 norm.

For CVP_p, the hardness result of Dinur and others [DKRS03], ruling out $n^{1/O(\log \log n)}$-approximate algorithms, assuming P $\neq$ NP applies to any ℓ_p norm $1 \leq p < \infty$.

Regarding the ℓ_∞ norm, Dinur [Din02] proves that there is no $n^{O(1/\log \log n)}$-approximate algorithm for SVP_∞ and CVP_∞, assuming P $\neq$ NP.

6.2 Decoding Linear Error-Correcting Codes

Let $\mathbb{F}$ be a field and A be a $n \times k$ matrix over $\mathbb{F}$. The mapping $C(x) = A \cdot x$ is a linear error-correcting code with minimum distance d if for every two disting inputs $x \neq x' \in \mathbb{F}^k$ the vectors $C(x)$ and $C(x')$ differ in at least d entries. It is easy to see that this happens if and only if for every $x \neq \mathbf{0}$ the vector $C(x)$ has at least d non-zero entries. In turn, this happens if and only if every d rows of A are linearly independent.

Error-correcting codes are used for the transmission (and storage) of information using unreliable media. Suppose that a sender has a message $x \in \mathbb{F}^k$ and that he wants to send it to a recipient using an unreliable channel that introduces transmission errors, and suppose also that the sender and the recipient know a linear error-correcting code $C : \mathbb{F}^k \to \mathbb{F}^n$ of minimum distance d. Then the sender can compute $C(x)$ and send it to the recipient through the channel. Due to transmission errors, the recipient receives a string $y \in \mathbb{F}^n$, that may be different from $C(x)$. Assuming that fewer that $d/2$ errors occurred, there is a unique string in the range of $C()$ that is closest to y, and this string is $C(x)$. If there is confidence that no more than $d/2$ errors occurred, then the recipient can reconstruct x this way.

Clearly, the larger is d the higher is the reliability of the system, because more transmission errors can be tolerated. Also, for fixed d and k, the smaller is n the more efficient is the transmission. Therefore, we would like to construct codes, that is, matrices, with large d and small n for given k. Asymptotically, we would like to find large ρ, δ such that for every n $k = \rho n$ and $d = \delta n$. It is beyond the scope of this paper to give an overview of known results about this problem. The reader is referred to a coding theory book like [vL99] for more information.

The natural optimization problems associated with linear codes are the following: Suppose we are given a linear code, specified by the encoding matrix A, can we compute its minimum distance? And given a good code C of large minimum distance and a string y, can we find the decoding x such that $C(x)$ is closest to y?

The first problem is similar to SVP, and it is formally defined as follows: given a matrix $A \in \mathbb{F}^{n\times k}$, where $\mathbb{F}$ is a field, find the non-zero vector $x \in \mathbb{F}^k$ such that $A \cdot x$ has the smallest number of non-zero entries. Dumer, Micciancio and Sudan [DMS03] prove that, assuming that NP $\not\subseteq$ QRP, the problem cannot be approximated within a factor $2^{(\log n)^{1-\varepsilon}}$. Their result holds even if the fielf $\mathbb{F}$ is restricted to be $\{0, 1\}$. The second problem, Nearest Codeword, is similar to CVP. Formally, our input is a matrix $A \in \mathbb{F}^{n\times k}$ and a vector $y \in \mathbb{F}^n$, and the goal is to find a vector $x \in \mathbb{F}^k$ such that the number of entries where $A \cdot x$ and y differ is minimized. Arora et al. [ABSS97] prove that the problem cannot be approximated within a factor $2^{(\log n)^{1-\varepsilon}}$, assuming NP $\not\subseteq$ QP.

6.3 The Traveling Salesman Problem

In the Traveling Salesman Problem (TSP) we are given a complete undirected graph $G = (V, E)$ with non-negative weights $w : E \to \mathbb{R}$ on the edges. We think of the vertices as being "cities" and the weight of the edge (u, v) as the "distance" between two cities. The goal is to find a route that touches each city exactly once and that has minimal total length.

If the distances are allowed to be arbitrary, then it is easy to see that even deciding whether the optimum is zero or not is an NP-hard problem, and so no approximation is possible [SG76]. One, however, is typically interested in instances where the distances satisfy the triangle inequality, and thus describe a metric on the set of cities. This special case is called the Metric TSP, or ΔTSP. The metric TSP is equivalent to the variation of TSP with arbitrary distances in which the route is required to pass through each city *at least* once, but it is allowed pass more than once through some city.

Almost 30 years ago, Christofides devised a 3/2-approximate algorithm for Metric TSP [Chr76], and the performance ratio of his algorithm has never been improved upon.

Papadimitriou and Yannakakis [PY93] present an approximation-preserving reduction from Max 3SAT-d to Metric TSP. The reduction, together with other reductions seen in the previous section and with the PCP Theorem, implies that there is no PTAS for Metric TSP unless P = NP. More specifically, there is a constant $\varepsilon_{TSP} > 0$ such that if there is a polynomial time $(1 + \varepsilon_{TSP})$-approximate algorithm for Metric TSP then P = NP. The value of ε_{TSP} implied by the PCP Theorem and the reductions is extremely small, and there has been work devoted to derive stronger inapproximability results. Papadimitriou and Vempala [PV00], using Theorem 16 (Håstad's three-query PCP construction [Hås01]) and a new reduction show that there is no polynomial time algorithm for Metric TSP with performance ratio better that 220/219, unless P = NP.

Grigni, Koutsopias and Papadimitriou show that Metric TSP has a PTAS in the special case in which the metric is the shortest path metric of a planar graph [GKP95]. Arora [Aro98b] and Mitchell [Mit99] show that there is a PTAS in the special case in which cities are points in $\mathbb{R}^2$ and distances are computed according to the ℓ_2 norm. Arora's PTAS also works for other ℓ_p metrics and in higher dimension, with running time doubly exponential in the number of dimensions. Trevisan [Tre00] shows that, if P $\neq$ NP, then Metric TSP in $\mathbb{R}^{d(n)}$ with an ℓ_p metric has no PTAS if $d(n) = \Omega(\log n)$. This implies that the double exponential dependency of the running time on the number of dimensions in Arora's PTAS cannot be improved unless every problem in NP has subexponential algorithms.

Another interesting version is Asymmetric TSP, in which the "distance" from a city u to a city v may be different from the distance from v to u. The Asymmetric ΔTSP is the restriction of Asymmetric TSP to asymmetric "distance" functions that satisfy the triangle inequality. A polynomial $O(\log n)$-approximate algorithm for Asymmetric ΔTSP is known [FGM82]. Papadimitriou and Vempala [PV00] prove hat there is no polynomial time algorithm for Metric TSP with performance ratio better that 117/116,[9] unless P = NP. It remains an interesting open question to rule out constant factor approximation for Asymmetric ΔTSP, or to devise a constant-factor approximation algorithm.

6.4 Coloring Problems

In the graph coloring problem we are given an undirected graph $G = (V, E)$ and we want to label each vertex with a "color" so that every edge has endpoints of different colors. The goal is to use as few colors as possible.

Feige and Kilian prove that, if ZPP $\neq$ NP, then, for every ε, there is no $n^{1-\varepsilon}$-approximate algorithm for this problem, where n is the number of vertices.

An interesting special case of the problem is to devise algorithms that color a 3-colorable graph with a minimum number of colors. (On other graphs, the algorithm may have an arbitrary behavior.) Blum and Karger [BK97] (improving previous work by Karger, Motwani and Sudan [KMS98] and by Wigderson [Wig83]), prove that there is a polynomial time algorithm that colors every 3-colorable graph with at most $n^{3/14} \cdot \text{poly}\log(n)$ colors. Khanna, Linial and Safra [KLS93] prove that there is no polynomial time algorithm that colors every 3-colorable graph using at most four colors, unless P = NP, and this is still the strongest known inapproximability result for this problem. This is one of the largest gaps between known approximation algorithms and known inapproximability results for a natural and well-studied problem.

It appears that methods based on probabilistically checkable proofs could be applied to the problem of coloring 3-colorable graphs, provided that one uses a different defnition of "soundness." Guruswami, Håstad and Sudan [GHS02] describe a type of PCP that could be used to prove hardness of 3-coloring. They are able to construct such PCPs with parameters that are sufficient to prove intractability for the *Hypergraph* Coloring problem. In the Hypergraph Coloring problem we are given a hypergraph $H = (V, E)$, where E is a collection of subsets of V, and we want to

[9] The conference version of [PV00] claimed that no polynomial $(129/128 - \varepsilon)$-approximate algorithm for Metric TSP and no $(42/41 - \varepsilon)$-approximate algorithm for Asymmetric ΔTSP exists for any $\varepsilon > 0$, assuming P $\neq$ NP. The stronger claim was due to an error in the proof. The corrected version of the paper, with the weaker inapproximability results, is available from the authors.

color the vertices with a minimum number of colors so that there is no edge that contains vertices all of the same color. A hypergraph is r-uniform if every hyperedge contains exactly r vertices. An undirected graph is a 2-uniform hypergraph. Guruswami et al. [GHS02] show that, if P $\neq$ NP, there is no polynomial time algorithm that, given a 2-colorable 4-uniform hypergraph, finds a feasible coloring that uses a constant number of colors. Dinur, Regev and Smyth [DRS02] (see also [Kho02a]) prove the same result for 3-uniform hypergraphs.

Regarding k-colorable graphs for large constant k, Khot [Kho01] proves that there is no polynomial time algorithm that, given a k-colorable graph, colors it with $k^{O(\log k)}$ colors.

6.5 Covering Problems

Using Håstad's PCP construction (Theorem 16) and the reduction of [FGL$^+$96], one can prove that there is no polynomial time algorithm for the Vertex Cover problem with performance ratio better than 7/6.

This result has been improved by Dinur and Safra [DS02], who prove that there is no approximation algorithm for Vertex Cover with performance ratio smaller than $10\sqrt{5}-21 = 1.36\ldots$- unless P = NP. A simple 2-approximate algorithm has been known to exist for thirty years [Gav74]. If the unique games conjecture is true, then 2 is the best possible performance ratio for polynomial time algorithms, unless P = NP [KR03]. (See also Section 8 below.)

In the Set Cover problem we are given a collection of subsets $S_1, \ldots, S_m$ of a set U such that $\bigcup_i S_i = U$. The goal is to find a smallest set $I \subseteq \{1, \ldots, m\}$ such that $\bigcup_{i \in I} S_i = U$. The problem can be approximated within a factor $\ln |U| + 1$, using either a greedy or a primal-dual algorithm [Joh74, Lov75, Chv79]. More precisely, if every set is of size at most k, then the approximation ratio of the algorithm is at most H_k, defined as $H_k = 1 + 1/2 + \ldots + 1/k \leq 1 + \ln k$. Feige [Fei98] shows that there cannot be a $(1-\varepsilon)\ln|U|$ approximate algorithm, for any $\varepsilon > 0$, unless NP $\subseteq$ QP. Trevisan [Tre01] notes that Feige's proof also implies that there is a constant c such that the Set Cover problem with sets of size k (where k is constant) has no $(\ln k - c \ln \ln k)$-approximate algorithm unless P = NP.

In the Hitting Set problem we are given collection of subsets $S_1, \ldots, S_m$ of a set U. The goal is to find a smallest set $S \subseteq U$ such that $S_i \cap S \neq \emptyset$ for every $i = 1, \ldots, m$. It is easy to see that the Set Cover problem and the Hitting Set problem are equivalent, and that there is an $r(k)$-approximate algorithm for the Set Cover problem, where k is the size of the largest set, if and only if there is an $r(t)$-approximate algorithm for the Hitting Set problem, where t is an upper bound to the number of sets to which an element of U may belong. Similarly, there is an $r(t)$-approximate algorithm for the Set Cover problem, where t is an upper bound to the number of sets to which an element of U may belong if and only if there is an $r(k)$-approximate algorithm for the Hitting Set problem, where where k is the size of the largest set. In particular, it follows from the results of [Joh74, Lov75, Chv79, Fei98] that there is a $(1 + \ln m)$-approximate algorithm for Hitting Set, and that there is no $(1-\varepsilon) \cdot \ln m$-approximate algorithm with $\varepsilon > 0$ unless NP $\subseteq$ QP.

Consider now the Hitting Set problem restricted to instances where every set is of size at most k. This is equivalent to the Set Cover problem restricted to instances where every element of U may appear in at most k sets. Another equivalent formulation of the problem is as the Vertex Cover problem in k-uniform hypergraphs. (In particular, it is just the vertex cover problem in graphs when $k = 2$.) A simple generalization of the 2-approximate algorithm for Vertex Cover yields a k-approximate algorithm for this problem [Hoc82]. Dinur et al. [DGKR03] prove there is no $(k-1-\varepsilon)$-approximate algorithm, $\varepsilon > 0$, unless P = NP. Assuming that the unique games conjecture holds, even $(k-\varepsilon)$-approximate algorithms are impossible [KR03].

7 Other Topics

7.1 Complexity Classes of Optimization Problems

Prior to the discovery of the connection between probabilistically checkable proofs and inapproximability, research on inapproximability focused on the study of approximation-preserving reductions among problems, and on classes of optimization problems and completeness results.

The first definitions of approximation-preserving reductions appear in work by Paz and Moran [PM81] and by Ausiello, D'Atri and Protasi [ADP80].

Orponen and Manilla [OM87] define the class NPO, and show that certain (artificial) problems are NPO-complete under approximation-preserving reductions. Kann [Kan93] finds problems that are complete for NPO-PB, the restriction of NPO to problems whose objective function is polynomially bounded in the length of the input. Crescenzi and Panconesi [CP91] study the class APX of problems that admit an $O(1)$-approximate algorithm. They show the existence of (artificial) complete problems in APX with respect to reductions that preserve the existence of approximation scheme. It is easy to see that, if P $\neq$ NP, there are problems in APX that do not have approximation schemes, and so, in particular, no APX-complete problem can have a PTAS unless P = NP. These results were meant as proofs-of-concept for the program of proving inapproximability results by proving completeness results under approximation-preserving reductions. None of these papers, however, proves inapproximability results for natural problems.

A very different, and more productive, direction was taken by Papadimitriou and Yannakakis [PY91]. They follow the intuition that Max 3SAT should be as central a problem to the study of constant factor approximation as 3SAT is to the study of decision problems in NP. Papadimitriou and Yannakakis identify a class of optimization problems, called Max SNP, that can be defined using a certain fragment of logic. They show that Max SNP includes Max 3SAT and other constraint satisfaction, and they prove that Max 3SAT is complete for Max SNP under approximation-preserving reductions. It follows that Max 3SAT has a PTAS if and only if every problem in Max SNP has a PTAS. More importantly, Papadimitriou and Yannakakis present several reductions from Max 3SAT using approximation-preserving reductions, including ones that we presented in earlier sections. The reductions proved in [PY91], and in other papers such as [BP89, PY93], implied that if one could rule out approximation schemes for Max 3SAT (or, equivalently, for any other problem in Max SNP), then inapproximability results would follow for many other problems. As we discussed earlier, the PCP Theorem finally proved the non-existence of approximation schemes for Max 3SAT (assuming P $\neq$ NP), and the reductions of [PY91, BP89, PY93] played the role that they were designed for. Around the same time, in another interesting result, Berman and Schnitger [BS92] proved that if Max 2SAT does not have a probabilistic PTAS (or equivalently, in light of [PY91], if Max 3SAT does not have a probabilistic PTAS), then there is a $c > 0$ such that there is no n^c-approximate algorithm for Independent Set. The reduction of Berman and Schnither and the PCP Theorem imply that there is a $c > 0$ such that there is no n^c-approximate algorithm for Indepedent Set, assuming RP $\neq$ NP. As we discussed in Section 5.2, Arora et al. [ALM$^+$98] achieve the same conclusion from the assumption that P $\neq$ NP, by improving the soundness of the PCP construction using random walks on an expander. As discussed in [AFWZ95], one can see the reduction of Arora et al. [ALM$^+$98] as a "derandomized" version of the reduction of [BS92].

The most interesting results in the paper of Papadimitriou and Yannakakis [PY91] were the approximation preserving reductions from Max 3SAT to other problems, but it was the idea of defining classes of optimization problems using fragments of logical theories that caught other people's immagination. Other papers followed with logic-based definitions of optimization classes having natural complete problems, such as the work of Kolaitis and Thakur [KT94, KT95] and of Panconesi and Ranjan [PR90].

The connection between proof checking and approximation and the PCP theorem allowed researchers to prove inapproximability results directly, without using notions of completeness, and gave central importance to reductions between specific problems.

Khanna, Motwani, Sudan and Vazirani [KMSV99] revisited the issue of completeness in approximation classes, and gave a definitive treatment. Khanna et al. [KMSV99] show how to use PCP to prove natural problems complete in their respective classes. For example they show that Max 3SAT is complete in APX, and that Independent Set is complete in poly-APX, the class of problems that are n^c-approximable for some $c > 0$, where n is the length of the input. Crescenzi et al. [CKST99] address some finer points about what definition of reduction should be used to prove general completeness results.

An interesting related line of work, developed by Chang and others [CGL97, Cha96, CKST99], is a characterization of the complexity of approximation problems in terms of "query complexity" to NP oracle. To see the point of this work, consider for example the problem of approximating Max 3SAT within a 1.1 versus a 1.01 factor. Both problems are NP-hard, and both problems can be solved in polynomial time if P = NP, so in particular if we had a polynomial time 1.1 approximate algorithm for Max 3SAT we could also design a polynomial time 1.01-approximate algorithm that uses the 1.1-approximate one as a subroutine. If we follow the proof of this result, we see that the hyptothetical 1.1-approximate algorithm is used more than once as a subroutine. Is this necessary? The answer turns out to be YES. Roughly speaking, computing a $1+\varepsilon$ approximation is equivalent to answering $\Theta(1/\varepsilon)$ non-adaptive queries to an NP oracle, and the power of what can be computed with k queries to an NP oracle is strictly less than what can be done with $k+1$ queries (unless the polynomial hierarchy collapses). In contrast, in Independent Set, for every two constants $1 < r_1 < r_2$, we can design an r_1-approximate algorithm for Independent Set that uses only once a an r_2-approximate algorithm. This property of the Independent Set problem (that different constant factor approximations have the same complexity) is called *self-improvability*. From the point of view of query complexity, computing a constant-factor approximation for Independent Set is equivalent to answering $\Theta(\log n)$ queries to an NP oracle. A consequence of these characterizations of approximation problems using query complexity is, for example, that one can prove that the Set Cover problem is not self-improvable unless the polynomial hierarchy collapses [CKST99], a result that it is not clear how to prove without using this machinery.

7.2 Average-Case Complexity and Approximability

Feige [Fei02b] proposes to use the conjectured average-case complexity of certain distributional problems[10] in order to prove inapproximability results. More specifically, Feige consider the 3SAT problem, together with the distribution where an instance with n variables and cn clauses, with c being a large constant, is picked uniformly at random from all instances with n variables and cn clauses. For sufficiently large constant c, such instances have an extremely high probability of

[10] A *distributional problem* is a decision problem equipped with a probability distribution over the inputs.

being unsatisfiable, but is seems difficult to *certify* that specific instances are unsatisfiable. Feige considers the assumption[11] that, for every choice of c, there is no polynomial time algorithm that on input a satisfiable formula never outputs "UNSAT," and that on input a formula picked from the above distribution outputs "UNSAT" with high probability. (The probability being computed over the distribution of formulas.)

Under this assumption, Feige shows that there is no $(4/3 - \varepsilon)$-approximate algorithm for the *Minimum Bisection* problem, for any $\varepsilon > 0$. The Minimum Bisection problem is defined as follows: we are given an undirected graph $G = (V, E)$, with V having even cardinality, and our goal is to find a partition of V into two subsets $S, \bar{S}$ of equal size such that a minimum number of edges crosses the partition. An approximation algorithm due to Feige and Krautghamer [FK02], and it achieves a factor of $O((\log n)^2)$. This has been recently improved to $O((\log n)^{1.5})$ by Arora, Rao and Vazirani [ARV04]. No inapproximability result was known for this problem prior to [Fei02b]. Khot [Kho04b] recently proved that the Minimum Bisection problem does not have an approximation scheme assuming that NP is not contained in sub-exponential time. More precisely, Khot shows that for every $\varepsilon > 0$ there is a constant $c_\varepsilon > 0$ such that there is no $(1 + c_\varepsilon)$-approximate algorithm for Minimum Bisection unless 3SAT can be solved in time $2^{O(n^\varepsilon)}$.

Alekhnovich [Ale03] defines other distributional problems (related to E3LIN) and shows that certain problems in coding theory are hard to approximate provided that the distributional problem is hard on average.

7.3 Witness Length in PCP Constructions

The proof of the PCP theorem shows that for every NP-complete problem, for example 3SAT, we can encode a witness in such a way that it can be tested with a constant number of queries. How long is that encoding? The original proof of the PCP theorem shows that the size of the encoding is polynomial in the number of variables and clauses of the formula. Improvements in [PS94, FS95] shows that one can achieve cubic encoding length in the PCP Theorem with a moderate query complexity, or encoding length $N^{1+\varepsilon}$, where N is the size of the formula, using query complexity that depends on ε. Recent work shows that the witness needs only be of nearly linear [HS00, BSSVW03, BSGH$^+$04] length in the size of the formula. It is conceivable that the witness could be of size linear in the number of clauses of the formula.

The question of witness length in PCP construction is studied mostly for its connections to coding theory, but it has an interesting connection to approximability as well.

If we let m denote the number of clauses and n the number of variables in a 3SAT instance, it is believed that the satisfiability problem cannot be solved in time $2^{o(n)}$, where n is the number of variables, even in instances where $m = O(n)$.[12] Suppose now that there were a proof of the PCP Theorem with a witness of linear size. Then this can be used to show that there is some $\varepsilon > 0$ such that a $(1 + \varepsilon)$-approximate algorithm for Max 3SAT running in $2^{o(n)}$ time implies an algorithm that solves 3SAT in $2^{o(n)}$ time. Since such a consequence is considered unlikely, we would have to conclude that no sub-exponential algorithm can even achieve a good approximation for Max 3SAT. Conversely, a $(1 + o(1))$-approximate algorithm for Max 3SAT running in $2^{o(n)}$ time would provide some evidence that the proof length in the PCP Theorem cannot be linear.

[11] He does not formulate it as a conjecture.

[12] In fact, Impagliazzo, Paturi and Zane [IPZ01] prove that, roughly speaking, if 3SAT has a $2^{o(n)}$ time algorithm that works only on instances where $m = O(n)$, then there is also a $2^{o(n)}$ time algorithm for 3SAT that works on all instances.

7.4 Typical and Unusual Approximation Factors

Looking at the inapproximability results proved in the 1990s using the PCP Theorem, it is easy to see a certain pattern. For several problems, such as Max 3SAT, Vertex Cover, Metric TSP, Metric Steiner Tree, Max CUT, we know that a polynomial time constant factor approximation exists, and there is an inapproximability result proving that for some $\varepsilon > 0$ there is no $(1+\varepsilon)$-approximate algorithm. For other problems, like Clique, Independent Set, and Chromatic Number, we can prove that there is an $c > 0$ such no n^c-approximate algorithm exists, where n is the size of the input. Then there are problems like Set Cover that are approximable within an $O(\log n)$ factor but not within a $o(\log n)$ factor, and finally there are some problems for which we can show an inapproximability result of the form $2^{(\log n)^c}$ for some constant $c > 0$, although we believe that the right lower bound is n^c for some $c > 0$.

For maximization problems, known results either prove that no PTAS exists (while a constant factor approximation is known), or that a n^c approximation is impossible, for some $c > 0$. For minimization problems, $\Omega(\log n)$ inapproximability results are also common, as well as inapproximability results of the form $2^{(\log n)^c}$.

Arora and Lund [AL96] notice this pattern in their survey paper and they classify optimization problems in four classes (called Class I to Class IV) according to the known inapproximability result.

Of course, a priori, there is no reason why the approximability of natural optimization problem should fall into a small number of cases, and, in fact, unless P = NP, for every (efficiently computable) function $f(n)$ there are (artificial) problems that can be approximated within $f(n)$ but no better. The interesting question is whether the pattern is an effect of our proof techniques, or whether the approximability of natural problems tends indeed to fall into one of a few typical cases. The latter possibilities has interesting precedents in complexity theory: for example almost all the natural decision problems in NP are either known to be solvable in polynomial or known to be NP-complete, even though, if P $\neq$ NP, there are (artificial) problems that have an infinite variety of intermediate complexities [Lad75]. Schaefer [Sch78] found in 1978 an interesting "explanation" for the latter phenomenon. He considers an infinite class of boolean satisfiability problems in NP, and he shows that each such problem is either in P or NP-complete. Creignou [Cre95], and Khanna, Sudan, Trevisan and Williamson [KSTW00] (see also the monograph [CKS01]) apply the same framework to optimization problems, and definite infinite classes of minimization of maximization problems that can be expressed in terms of boolean constraint satisfaction. They show that maximization problems in the class are either solvable in polynomial time, or can be approximated within a constant factor but not to arbitrary good constant factors (Class I in the classification of [AL96]), or are hard to approximate within a factor n^c for some $c > 0$ (Class IV), or it is intractable to even find a feasible solution. Minimization problems are either solvable in polynomial time, or fall into one of five equivalence classes (two corresponding to Class II, two corresponding to Class III and one corresponding to Class IV), or it is intractable to find a feasible solution. It is remarkable that the inifinite class of problems studied in [Cre95, KSTW00] would show only a constant number of distinct approximability behaviour, and that these cases would allign themselves with the classification of [AL96].

Recent results, however, have shown that some natural problems have approximation thesholds that do not fit the above scheme.

For example, Halperin and Krauthgamer [HK03] prove a $\Omega((\log n)^{2-\varepsilon})$ inapproximability result for the Group Steiner Tree problem for every $\varepsilon > 0$, a negative result that applies even to the restriction of the problem to trees. The problem can be approximated within a $O((\log n)^2)$ factors in trees [GKR00] and $O((\log n)^3)$ in general graphs [Bar96, FRT03]. This result shows that there are natural problems for which the best possible approximation is super-logarithmic but polylogarithmic.

Chuzhoy and Naor [CN04] prove an $\Omega(\log \log n)$ inapproximability result for the Min-congestion Unsplittable Flow problem. An $O(\log n / \log \log n)$-approximate algorithm by Raghavan and Thompson [RT87] is known for this problem. This result shows that there are natural problems for which the best possible approximation is sub-logarithmic but super-constant.

The most surprising result along these lines is the recent proof by Chuzhoy et al. [CGH$^+$04] of a $\log^* k - O(1)$ inapproximability result for the Asymmetric k-Center Problem. An $O(\log^* k)$-approximate algorithm is known [PV98, Arc01].

7.5 Inapproximability Results versus "Integrality Gaps"

Several approximation algorithms are based on linear programming or semidefinite programming relaxations of the problem of interest. Typically, in order to prove that an algorithm based on a relaxation is r-approximate, one proves the stronger statement that the algorithm finds a solution whose cost is within a factor r from the optimum of the relaxation. In particular, this shows that the optimum of the relaxation is within a factor r from the optimum of the combinatorial problem.

For example, the Goemans-Williamson approximation algorithm for Max CUT [GW95] finds a cut whose cost is at least $\alpha = .878\ldots$ times the cost of the optimum of the relaxation. Such an analysis also proves, as a consequence, that the optimum of the relaxation is always at most $1/\alpha$ times the true optimum.

In order to show that such an analysis is tight, one can try and construct an instance of the combinatorial problem for which the ratio between the true optimum and the optimum of the relaxation is as large as the bound implied by the analysis of the approximation algorithm. For example, for Max CUT, Feige and Schechtman [FS02] prove that there are graphs for which the ratio between the cost of the optimum of the relaxation and the cost of the maximum cut is arbitrarily close to α.

The worst-case ratio between the true optimum of a combinatorial problem and the optimum of a relaxation is called the *integrality gap* of the relaxation. Research on integrality gaps can be seen as the study of inapproximability results that apply to a restricted kind of algorithms (namely, algorithms that use the relaxation in their analysis). Arora et al. [ABL02] have recently shown that it is possible to find instances that give large integrality gaps for entire families of relaxations. It would be interesting to generalize their approach to semidefinite programming relaxations.

The study of integrality gap has, moreover, a more direct, if not well understood, relation to inapproximability results. For example, the inapproximability result for the Set Cover problem proved by Lund and Yannakakis [LY94] (and its refinements and improvements in [BGLR93, Fei98]) relies on a reduction from PCP that involves a family of sets similar to the one used to prove the integrality gap of the standard linear programming relaxation of Set Cover. Similarly, the inapproximability result for Group Steiner Tree [HK03] involves a reduction that constructs an instance similar to the one used in [HKK$^+$03] to prove an integrality gap. A similar progress from integrality gap to inapproximability result can be see in [CGH$^+$04] for the Asymmetric K-Center Problem. Recently, Khot and others [KKMO04] present a PCP construction that shows that Max CUT cannot be approximate to within a factor small than α, assuming that the unique games conjecture holds and that the majority function has certain extremal properties. The PCP construction of

[KKMO04] is stronly inspired by the graphs constructed by Feige and Schechtman [FS02] to bound the integrality gap of the Goemans-Williamson relaxation.

Of course it would not be surprising to see that the instances created by a reduction in an inaproximability result have a large integrality gap: if not, we would have P = NP. What is surprising is that instances defined specifically to prove integrality gaps turn out to be useful as components in reductions proving inapproximability results.

8 Conclusions

In this paper we surveyed inapproximability results, a field almost non-existent in 1990, and then defined by the connection with probabilistic proof-checking of [FGL$^+$96, AS98, ALM$^+$98]. Progress in the years after the PCP Theorem was extremely rapid, even outpacing the expectations of the researchers working in this area. For example, Bellare et al. and Arora [BGS98, Aro95] presented two independent arguments that "current techniques" could not prove an inapproximability result stronger than $\sqrt{n}$ for Independent Set just months before Håstad distributed a preliminary version of his work [Hås99] showing inapproximability within a factor $n^{1-\varepsilon}$. By 1997, optimal inapproximability results were known for Max 3SAT [Hås01], Independent Set [Hås99], Coloring [FK98] and Set Cover [Fei98], the problems discussed in Johnson's paper [Joh74] that defined the study of approximation algorithms.

After all these successes, the field is still very active and far from mature. Recent work [DS02, KKMO04] has emphasized the use of higher mathematics in the analysis of PCP verifier, and of constructions of verifiers that are so specialised to the application to the problem of interest that it is hard to draw the line between PCP construction and reduction from PCP to problem. The connection between average-case complexity and inapproximability is likely to become a major line of work in the future. It is notable that several important inapproximability results have been announced just in the last few months, while we were writing this paper, and are still unpublished at the time of writing [Kho04a, Kho04b, KKMO04].

Some important open questions seem to still be beyond the reach of current techniques, and one can look forward to the exciting new ideas to come in the future to address these questions. Here we mention some questions that are open at the time of writing and that the author feels are particularly interesting. Of course this is only a very small sample, and it is very biased by the author's interests.

Prove Optimal Results about 2-Query PCPs. Several PCP constructions and inapproximability results rely on versions of the PCP theorem in which the verifier reads only two entries in the proof, where each entry contains not just one bit but an element of a larger alphabet of size, say $t = t(n)$. It is believed[13] that there is a version of the PCP Theorem with a 2-query verifier that uses $O(\log n)$ randomness, accesses a proof written with over an alphabet of size $t(n)$, and has soundness $1/t^c(n)$ with $c > 0$. Raz [Raz98] proves such a result for constant t. For general $t(n)$, Raz's construction has a verifier that uses $O(t(n) \cdot \log n)$ randomness. Raz and Safra [RS97] and Arora and Sudan [AS97] prove a version of the result where the verifier makes a constant (larger than 2) number of queries, and $t(n)$ is restricted to be $n^{o(1)}$.

[13] This conjecture was first made in [BGLR93].

Settle the "Unique Games Conjecture." The unique games conjecture, formulated by Khot [Kho02b], is that a certain strong form of the PCP theorem holds. The conjecture implies the resolution of various important open questions [Kho02b, KR03], including the non-existence of $(2-\varepsilon)$-approximate algorithms for Vertex Cover and of $(k-\varepsilon)$-approximate algorithm for Vertex Cover in k-uniform hypergraphs. If the unique games conjecture holds, and a certain conjecture about extremal properties of monotone functions is true, then an optimal inapproximability result for Max CUT [KKMO04] would follow, which would be a remarkable breakthrough.

Prove a Strong Inapproximability Result for Metric TSP. The Metric TSP is one of the most well studied optimization problems, and there is still a very large gap between the known 3/2-approximate algorithm [Chr76] and the 220/219 inapproximability result [PV00]. The reduction of Papadimitriou and Vempala is an extremely optimized one, and it starts from the optimal version of the PCP Theorem of Håstad [Hås01], so it looks like a very different approach is needed to prove, say, a 1.1 inapproximability result. New results for Vertex Cover [DS02] blur the line between PCP construction and reduction, and use PCP techniques to directly construct an instance of the problem of instance. It is possible that a similar approach could give stronger inapproximability results for Metric TSP.

Make Progress on Graph Partitioning Problems. For many problems involving graph partitioning there is a big gap between algorithms and inapproximability results. Often, no inapproximability results are known at all. The *sparsest cut* problem is a typical case of an important graph partitioning problem for which no inapproximability result is known. In the *sparsest cut* problem we are given an undirected graph $G=(V,E)$, and the goal is to find a partition $(S,\bar{S})$ of V that minimizes

$$\frac{E(S,\bar{S})}{\min\{|S|,|\bar{S}|\}} .$$

The objective function is defined in such a way that the optimal solution tends to be a cut that is both balanced and sparse, a property that is useful in the design of divide-and-conquer algorithms.

The best known algorithm for Sparsest Cut is $O(\sqrt{\log n})$-approximate, where n is the number of vertices, and it is due to Arora, Rao and Vazirani [ARV04], improving an $O(\log n)$ approximation algorithm by Leighton and Rao [LR99]. The paper of Leighton and Rao [LR99] describes various application of approximate sparsest cut computations to design divide-and-conquer algorithms.

There is no known inapproximability result for this problem, and Khot's inapproximability result for the related problem Min Bisection [Kho04b] seem not to generalize to Sparsest Cut. Even using average-case assumptions as in [Fei02b] it is not known how to rule the existence of a PTAS for Sparsest Cut.

Acknowledgements

I thank Vangelis Paschos for giving me the opportunity to write this survey, and for being very patient with my inability to meet deadlines. I am grateful to James Lee and Robi Krautghamer for their very valuable suggestions that gave shape to Section 7. I wish to thank Uri Feige, Johan Håstad and Subhash Khot for their comments on an earlier version of this manuscript.

References

[ABL02] Sanjeev Arora, Béla Bollobás, and László Lovász. Proving integrality gaps without knowing the linear program. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 313–322, 2002. 27

[ABSS97] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997. 4, 20

[ACG+99] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, 1999. 3, 5

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 284–293, 1997. 18

[ADP80] G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21:136–153, 1980. 23

[AFWZ95] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized graph products. *Computational Complexity*, 5(1):60–75, 1995. 23

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 99–108, 1996. 18

[AK98] N. Alon and N. Kahale. Approximating the independence number via the θ function. *Mathematical Programming*, 80:253–264, 1998. 17

[AKS01] Miklos Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 601–610, 2001. 18, 19

[AL96] S. Arora and C. Lund. Hardness of approximations. In *Approximation Algorithms for NP-hard Problems*. PWS Publishing, 1996. 5, 26

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 298–307, 2003. 25

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in *Proc. of FOCS'92*. 4, 8, 23, 28

[AR04] Dorit Aharonov and Oded Regev. Lattice problems in NP∩coNP. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, 2004. 18, 19

[Arc01] Aaron Archer. Two $O(\log^* k)$-approximation algorithms for the asymmetric k-center problem. In *Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization*, pages 1–14, 2001. 27

[Aro94] Sanjeev Arora. *Probabilistic checking of proofs and the hardness of approximation problems.* PhD thesis, U.C. Berkeley, 1994. 8

[Aro95] S. Arora. Reductions, codes, PCP's and inapproximability. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 404–413, 1995. 28

[Aro98a] S. Arora. The approximability of NP-hard problems. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, pages 337–348, 1998. 5

[Aro98b] Sanjeev Arora. Polynomial time approximation schemes for Euclidean Traveling Salesman and other geometric problems. *Journal of the ACM*, 45(5), 1998. 21

[ARV04] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows and a $\sqrt{\log n}$-approximation to sparsest cut. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, 2004. 25, 29

[AS97] S. Arora and M. Sudan. Improved low degree testing and its applications. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 485–495, 1997. 28

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in *Proc. of FOCS'92*. 4, 8, 28

[Bar96] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996. 27

[Bel96] M. Bellare. Proof checking and approximation: Towards tight results. *Sigact News*, 27(1), 1996. 4, 5

[BGLR93] M Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 294–304, 1993. See also the errata sheet in *Proc of STOC'94*. 16, 27, 28

[BGS98] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP's and non-approximability – towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. Preliminary version in *Proc. of FOCS'95*. 16, 17, 28

[BK97] A. Blum and D. Karger. An $\tilde{O}(n^{3/14})$ coloring algorithm for 3-colorable graphs. *Information Processing Letters*, 61(1):49–53, 1997. 21

[BP89] M. Bern and P. Plassmann. The Steiner tree problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989. 4, 13, 23

[BS92] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation*, 96:77–94, 1992. Preliminary version in *Proc. of STACS'89*. 4, 23

[BS94] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 184–193, 1994. 16

[BSGH+04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, 2004. 25

[BSSVW03] Eli Ben-Sasson, Madhu Sudan, Salil P. Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via ε-biased sets. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 612–621, 2003. 25

[CGH+04] Julia Chuzhoy, Sudipto Guha, Eran Halperin, Sanjeev Khanna, Guy Kortsarz, and Joseph Naor. Asymmetric k-center is log^*n hard to approximate. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, 2004. 27

[CGL97] R. Chang, W. I. Gasarch, and C. Lund. On bounded queries and approximation. *SIAM Journal on Computing*, 26(1):188–209, 1997. Preliminary version in *Proc. of FOCS'93*. 24

[Cha96] R. Chang. On the query complexity of clique size and maximum satisfiability. *Journal of Computer and System Sciences*, 53(2):298–313, 1996. Preliminary version in *Proc. of Structures'94*. 24

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon University, 1976. 20, 29

[Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979. 22

[CKS01] Nadia Creignou, Sanjeev Khanna, , and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. SIAM Monographs on Discrete Mathematics and Applications 7, 2001. 26

[CKST99] P. Crescenzi, V. Kann, R. Silvestri, and L. Trevisan. Structure in approximation classes. *SIAM Journal on Computing*, 28(5):1759–1782, 1999. 24

[CN04] Julia Chuzhoy and Seffi Naor. New hardness results for congestion minimization and machine scheduling. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, 2004. 27

[Con93] A. Condon. The complexity of the max-word problem and the power of one-way interactive proof systems. *Computational Complexity*, 3:292–305, 1993. Preliminary version in *Proc. of STACS91*. 4

[Coo71] S.A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971. 2

[CP91] P. Crescenzi and A. Panconesi. Completeness in approximation classes. *Information and Computation*, 93:241–262, 1991. Preliminary version in *Proc. of FCT'89.* 23

[Cre95] Nadia Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, 51(3):511–522, 1995. 5, 26

[DGKR03] Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered pcp and the hardness of hypergraph vertex cover. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 595–601, 2003. 22

[Din02] Irit Dinur. Approximating SVP_∞ to within almost-polynomial factors is NP-hard. *Theoretical Computer Science*, 285(1):55–71, 2002. 19

[DKRS03] Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. An improved lower bound for approximating CVP. *Combinatorica*, 23(2):205–243, 2003. 19

[DMS03] Ilya Dumer, Daniele Micciancio, , and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003. 20

[DRS02] Irit Dinur, Oded Regev, and Clifford D. Smyth. The hardness of 3-Uniform hypergraph coloring. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 33–42, 2002. 22

[DS02] Irit Dinur and Shmuel Safra. The importance of being biased. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 33–42, 2002. 22, 28, 29

[Fei98] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998. 4, 22, 27, 28

[Fei02a] Uriel Feige. Approximation thresholds for combinatorial optimization problems. In *Proceedings of the International Congress of Mathematicians*, pages 649–658, 2002. Volume 3. 5

[Fei02b] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 534–543, 2002. 24, 25, 29

[FGL$^+$96] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. Preliminary version in *Proc. of FOCS91.* 4, 8, 14, 17, 22, 28

[FGM82] A. Frieze, G. Galbiati, and F Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12:23–39, 1982. 21

[FK94] U. Feige and J. Kilian. Two prover protocols - low error at affordable rates. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 172–183, 1994. 16

[FK98] Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998. 4, 28

[FK02] Uriel Feige and Robert Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal on Computing*, 31(4):1090–1118, 2002. 25

[FRT03] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 448–455, 2003. 27

[FS95] Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *Proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems*, pages 190–198, 1995. 25

[FS02] Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for MAX CUT. *Random Structures and Algorithms*, 20(3):403–440, 2002. 27, 28

[Gav74] F. Gavril. Manuscript cited in [GJ79], 1974. 22

[GG81] O. Gabber and Z. Galil. Explicit construction of linear sized superconcentrators. *Journal of Computer and System Sciences*, 22:407–425, 1981. 10

[GHS02] Venkatesan Guruswami, Johan Håstad, and Madhu Sudan. Hardness of approximate hypergraph coloring. *SIAM Journal on Computing*, 31(6):1663–1686, 2002. 21, 22

[GJ76] M.R. Garey and D.S. Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM*, 23:43–49, 1976. 4

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979. 34

[GKP95] M. Grigni, E. Koutsoupias, and C.H. Papadimitriou. An approximation scheme for planar graph TSP. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 640–645, 1995. 21

[GKR00] Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *J. of Algorithms*, 37(1):66–84, 2000. 27

[Gra66] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technology Journal*, 45:1563–1581, 1966. 2

[GW95] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995. Preliminary version in *Proc. of STOC'94*. 17, 27

[Hal98] M. Halldorsson. A survey on independent set approximations. In *APPROX'98*, pages 1–14, 1998. LNCS 1444, Springer-Verlag. 17, 39

[Hås99] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182:105–142, 1999. 4, 17, 28

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. 4, 16, 21, 28, 29

[HK03] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 585–594, 2003. 26, 27

[HKK+03] Eran Halperin, Guy Kortsarz, Robert Krauthgamer, Aravind Srinivasan, and Nan Wang. Integrality ratio for group Steiner trees and directed Steiner trees. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms*, pages 275–284, 2003. 27

[Hoc82] D. Hochbaum. Approximation algorithms for set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555–556, 1982. 22

[Hoc96] Dorit Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing, 1996. 3

[HS85] D.S. Hochbaum and D.B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. 4

[HS00] Prahladh Harsha and Madhu Sudan. Small PCPs with low query complexity. *Computational Complexity*, 9(3–4):157–201, 2000. 25

[HW01] Johan Håstad and Avi Wigderson. Simple analysis of graph tests for linearity and PCP. In *Proceedings of the 16th IEEE Conference on Computational Complexity*, pages 244–254, 2001. 17

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. 25

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 248–253, 1989. 15

[Joh74] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974. 2, 4, 22, 28

[Joh92] D.S. Johnson. The NP-completeness column: An ongoing guide: The tale of the second prover. *Journal of Algorithms*, 13:502–524, 1992. 8

[Kan93] V. Kann. Polynomially bounded minimization problems which are hard to approximate. In *Proceedings of 20th International Colloquium on Automata, Languages and Programming*, pages 52–63, 1993. 23

[Kar72] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972. 2, 11

[Kho01] Subhash Khot. Improved inaproximability results for maxclique, chromatic number and approximate graph coloring. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 600–609, 2001. 22

[Kho02a] Subhash Khot. Hardness results for coloring 3-colorable 3-uniform hypergraphs. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 23–32, 2002. 22

[Kho02b] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 767–775, 2002. 17, 29

[Kho03] Subhash Khot. Hardness of approximating the shortest vector problem in high L_p norms. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 290–297, 2003. 19

[Kho04a] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, 2004. 19, 28

[Kho04b] Subhash Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, 2004. 25, 28, 29

[KKMO04] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other two-variable CSPs? In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, 2004. 17, 27, 28, 29

[KLS93] S. Khanna, N. Linial, and S. Safra. On the hardness of approximating the chromatic number. In *Proceedings of the 2nd IEEE Israel Symposium on Theory of Computing and Systems*, pages 250–260, 1993. 21

[KMS98] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semi-definite programming. *Journal of the ACM*, 45(2):246–265, 1998. 17, 21

[KMSV99] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1999. Preliminary version in *Proc. of FOCS'94*. 24

[KR03] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2 - \varepsilon$. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, 2003. 22, 23, 29

[KSTW00] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2000. 5, 26

[KT94] P.G. Kolaitis and M.N. Thakur. Logical definability of NP optimization problems. *Information and Computation*, 115(2):321–353, 1994. 24

[KT95] P.G. Kolaitis and M.N. Thakur. Approximation properties of NP minimization classes. *Journal of Computer and System Sciences*, 50:391–411, 1995. Preliminary version in *Proc. of Structures91*. 24

[Lad75] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975. 26

[Lev73] L. A. Levin. Universal search problems. *Problemi Peredachi Informatsii*, 9:265–266, 1973. 2

[LLL82] A.K. Lenstra, H.W. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. 18, 19

[Lov75] L. Lovasz. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975. 22

[LPS88] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988. 10

[LR99] Frank T. Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46:787–832, 1999. 29

[LY94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994. Preliminary version in *Proc. of STOC'93*. 4, 27

[Mic01] Daniele Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 2001. 19

[Mit99] J.S.B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, K-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. 21

[MR04] Daniele Micciancio and Oded Regev. Stronger average case to worst case connections. Manuscript, 2004. 18

[OM87] P. Orponen and H. Mannila. On approximation preserving reductions: complete problems and robust measures. Technical Report C-1987-28, Department of Computer Science, University of Helsinki, 1987. 23

[Pap94] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. 6

[PM81] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximation. *Theoretical Computer Science*, 15:251–277, 1981. 23

[PR90] A. Panconesi and D. Ranjan. Quantifiers and approximation. In *Proceedings of the 22nd ACM Symposium on Theory of Computing*, pages 446–456, 1990. 24

[PS94] A. Polishchuk and D.A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 194–203, 1994. 25

[PV98] Rina Panigrahy and Sundar Vishwanathan. An $O(\log^* n)$ approximation algorithm for the asymmetric p-center problem. *J. of Algorithms*, 27(2):259–268, 1998. 27

[PV00] Christos H. Papadimitriou and Santosh Vempala. On the approximability of the traveling salesman problem. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 126–133, 2000. 20, 21, 29

[PY91] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88*. 4, 8, 11, 23, 24

[PY93] C.H. Papadimitriou and M. Yannakakis. The travelling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–11, 1993. 4, 20, 23

[Raz98] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. Preliminary version in *Proc. of STOC'95*. 28

[Reg03] Oded Regev. New lattice based cryptographic constructions. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 407–416, 2003. 18

[RS97] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 475–484, 1997. 28

[RT87] P. Raghavan and C.D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987. 27

[Sch78] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing*, pages 216–226, 1978. 26

[Sch87] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987. 18, 19

[SG76] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976. 20

[ST00] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000. 17

[Sud92] M. Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*. PhD thesis, University of California at Berkeley, 1992. 8

[Tre00] Luca Trevisan. When Hamming meets Euclid: The approximability of geometric TSP and MST. *SIAM Journal on Computing*, 30(2):475–485, 2000. 21

[Tre01] Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 453–461, 2001. 17, 22

[TSSW00] L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming. *SIAM Journal on Computing*, 29(6):2074–2097, 2000. 16

[Vaz01] Vijay Vazirani. *Approximation Algorithms*. Springer, 2001. 3, 5, 6

[Vis96] S. Vishwanathan. Personal communication to M. Halldorsson. Cited in [Hal98], 1996. 17

[vL99] Jacobus H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1999. 20

[Wig83] Avi Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM*, 30(4):729–735, 1983. 21