CHAPTER

10

HARDNESS OF APPROXIMATIONS

Sanjeev Arora

Carsten Lund

Abstract

This chapter is a self-contained survey of recent results about the hardness of approximating *NP*-hard optimization problems. [1]

This chapter surveys recent results on the hardness of approximating *NP*-hard optimization problems. According to these results, computing good approximate solutions to many problems is *NP*-hard — and hence no easier than computing exact solutions.

In general, proving the *NP*-hardness of an optimization problem involves a *reduction* from SAT (or any other *NP*-complete problem) to the problem. To prove the hardness of approximation, this reduction must produce a *gap* in the value of the optimum. For instance, proving the *NP*-hardness of approximating the *maximum clique problem* within a factor g requires coming up with a reduction from SAT to CLIQUE that maps satisfiable formulae to graphs with clique number at least K (for some K), and unsatisfiable formulae to graphs with clique number at most K/g.

For a long time it was unclear how to construct such *gap producing* reductions for Clique and many other important optimization problems. The Cook-Karp-Levin [Coo71, Kar72, Lev73] techniques for doing reductions seemed more suited

Sanjeev Arora is supported by NSF CAREER award CCR-9502747.

for proving the hardness of decision problems (for example, non-optimization problems such as SAT or TILING) than of approximation problems.

Recent work has yielded a fairly general technique for constructing gap-producing reductions. This new technique, which originated in the work of [FGL+91], relies upon new probabilistic characterizations of the NP class. The most well-known such characterization is the so-called PCP Theorem, written as $NP = PCP(\log n, 1)$ [AS92, ALM+92]. Section 10.7 provides an introduction to the PCP Theorem.

The proof of the PCP Theorem involves complicated algebraic techniques from complexity theory, and will not be given here. Luckily, understanding the proof is not a prerequisite for using the theorem in inapproximability results. In particular, this survey describes all major inapproximability results, while requiring from the reader only some familiarity with the basic notions of NP-completeness. The first few chapters of the well-known book by Garey and Johnson [GJ79] provide all necessary background.

As described in earlier chapters of this book, the chief parameter of interest when studying the approximability of a problem is the *approximation ratio* that can be achieved by a polynomial-time approximation algorithm. An algorithm *achieves an approximation ratio* α for a maximization problem if, for every instance, it produces a solution of value at least OPT/α, where OPT is the value of the optimal solution. (For a minimization problem, achieving a ratio α involves finding a solution of cost at most $\alpha\, OPT$.) Note that the approximation ratio is ≥ 1 by definition. (Other chapters of this book use a slightly different convention, according to which approximation ratios for maximization problems are ≤ 1.)

Recent inapproximability results divide problems into four broad classes, based on the approximation ratio that is provably hard to achieve. These approximation ratios are, respectively, $1+\epsilon$ for some fixed $\epsilon > 0$, $\Omega(\log n)$, $2^{\log^{1-\gamma} n}$ for every fixed $\gamma > 0$, and n^δ for some fixed $\delta > 0$ (throughout, n denotes the input size). The corresponding classes of problems are called Classes I, II, III, and IV respectively. Inapproximability results for problems within a class (sometimes also across classes) share common ideas. We devote a section (from 10.2 to 10.5) to each class. It should be noted that our classification reflects only (the limits of) our current understanding of the area; future work may reveal other natural classes, or collapse two classes, or move problems from one class to another. To give an example, all problems in Class III are believed to also lie in Class IV, although a proof of this eludes us.

This survey also proposes six "canonical" inapproximable problems, which can be used to derive all known inapproximability results in a fairly simple way. (Our framework of canonical problems is derived from the one in [Aro94].) Thus our six problems play a role in inapproximability results similar to that played by the six canonical problems of [GJ79] in proving the NP-completeness of exact optimization. Furthermore, just as 3SAT is the most basic canonical problem in [GJ79], its optimization version, MAX-3SAT, is the most basic canonical problem in this chapter. As we will show, (see Figure 10.1 on page 7) reductions from MAX-3SAT can prove the hardness of all the canonical problems.

We emphasize that the framework presented in this chapter is not a way to

Class	Factor of Approximation that is hard	Representative Problems
I	$1+\epsilon$	MAX-3SAT
II	$O(\log n)$	SETCOVER
III	$2^{\log^{1-\gamma} n}$	LABELCOVER
IV	n^{ϵ}	CLIQUE

Table 10.1: The four classes and their representative problems.

prove the best possible results for every problem. Rather, it is meant to be a way to prove results that are "in the same ballpark" as the best results known. To give an example, the best existing result for the clique problem shows the hardness of achieving an approximation ratio roughly $\sqrt[3]{N}$, where N is the number of vertices of the graph. Since the proof of that result is complicated and involves delving deep into the proof of the PCP Theorem, we instead present the weaker result — provable in our framework — that achieving a ratio N^{ϵ} is NP-hard, for some fixed $\epsilon > 0$.

The rest of the survey is organized as follows. Section 10.1 sets up the notational framework as well as the general philosophy of inapproximability results. It also defines the six canonical problems. Sections 10.2 to 10.5 prove inapproximability results for representative problems in Classes I to IV. Section 10.6 is a succinct (by no means exhaustive) overview of other inapproximabilty results and where they are proved. Section 10.7 is an introduction to the PCP Theorem and how it is used in proving inapproximability results. Section 10.8 lists important research problems. Finally, Section 10.9 briefly describes how the results of this chapter were discovered. That section also provides references to further reading.

REMARK 10.1 Notation We describe, using MAX-3SAT as an example, the notation we will use for describing optimization problems. Such problems involve optimizing some *objective function* on a set of *feasible solutions.* In the MAX-3SAT problem, the input is a 3CNF formula, and the goal is to find an assignment that maximizes the number of satisfied clauses. Thus the feasible solutions are truth assignments, and the value of the objective function on a truth assignment is the *fraction* of clauses that it satisfies. We let MAX-3SAT(I) denote the maximum value of this objective function for a 3CNF formula I. Note that MAX-3SAT$(I) \leq 1$. We will likewise express the objective function for many other problems as a ratio (i.e., a pure number).

REMARK 10.2 *Quasi-NP*-hardness Many inapproximability results in this chapter show that achieving a certain approximation ratio is *NP-hard.* This implies that if some polynomial-time algorithm achieves that approximation ratio, then $NP = P$. Other results in this chapter are slightly weaker; they show only that the approximation is *Quasi-NP-hard.* A problem is *Quasi-NP*-hard if any polynomial-time algorithm for it can be used to solve all NP-problems in quasi-polynomial (i.e., $2^{\mathrm{poly}(\log n)}$) time. Since NP-complete problems are conjectured to have no sub-exponential algorithms (deterministic or randomized) a proof of *Quasi-NP*-hardness is good evidence that the problem has no polynomial-time algorithm (deterministic or randomized).

10.1 HOW TO PROVE INAPPROXIMABILITY RESULTS

Throughout, we will use the same method to prove the inapproximability of a given problem: We start with a known inapproximable problem, and then perform a *gap-preserving reduction* from that problem to the given problem. Section 10.1.1 describes six canonical problems that often serve as the "known inapproximable problem" in such situations. Section 10.1.2 describes, without proof, the inapproximability results for the canonical problems; the proofs of these results appear later in the chapter. The concept of a gap-preserving reduction is defined in Section 10.1.3, and illustrated with some examples.

10.1.1 THE CANONICAL PROBLEMS

We describe six canonical problems. The most basic one is MAX-3SAT, an optimization version of the decision problem 3SAT. Reductions from MAX-3SAT will be used to prove all inapproximability results in this chapter. Thus MAX-3SAT plays a role in the theory of inapproximability analogous to the one played by 3SAT in the classical theory of *NP*-completeness. The next result, which proves that approximating MAX-3SAT is *NP*-hard, can therefore be viewed as the analogue of the Cook-Levin Theorem[2]. Section 10.9 gives a brief history of this and related results.

THEOREM 10.1 [ALM$^+$92] There is a fixed $\epsilon > 0$ and a polynomial-time reduction τ from SAT to MAX-3SAT such that for every boolean formula I:

$$I \in \text{SAT} \Longrightarrow \text{MAX-3SAT}(\tau(I)) = 1,$$
$$I \notin \text{SAT} \Longrightarrow \text{MAX-3SAT}(\tau(I)) < \frac{1}{1+\epsilon}.$$

In other words, achieving an approximation ratio $1+\epsilon$ for MAX-3SAT is *NP*-hard.

Theorem 10.1 is proved in Section 10.7.2

[2]The analogy between Cook's Theorem and Theorem 10.1 is not entirely correct. Cook's Theorem is important also because it motivated other researchers to study *NP*-completeness. The analogous result in the recent work on inapproximability is the paper of Feige et al. [FGL$^+$91] on CLIQUE.

Now we describe the other five canonical problems.

Other Canonical Problems:

MAX-3SAT(5): This is the subcase of MAX-3SAT in which every variable appears in at most 5 clauses. We will assume that each variable appears in exactly 5 clauses.

R-CLIQUE: The clique number of the graph, denoted ω, is the size of the largest clique (i.e., a set of vertices all adjacent to each other) in it. In the R-CLIQUE problem the input consists of a positive integer r, and an r-partite[3] graph G along with its r-partition. The goal is to find the largest clique in G. We define R-CLIQUE(G) to be k/r, where k is the size of the largest clique in G. Since a clique can have at most one vertex in common with an independent set, no clique in an r-partite graph has size more than r. Thus R-CLIQUE(G) ≤ 1.

LABELCOVER: This problem comes in a maximization and a minimization version, both of which are defined below (Definition 10.1).

SETCOVER: Given a ground set U and a collection of its subsets $S_1, S_2, \ldots, S_m$ satisfying $\bigcup_{i=1}^{m} S_i = U$, find the size of the minimum subcollection that covers U, i.e., the minimum sized $I \subset \{1, 2, \ldots, m\}$ such that $\bigcup_{i \in I} S_i = U$. The objective function is $|I|$, the number of sets in the subcollection.

COLORING: Given a graph G, assign a color (namely, an integer) to each vertex such that no two adjacent vertices have the same color. Minimize the total number of colors used. The objective function is the number of colors in a proper coloring. (The minimum of the objective function is called the *chromatic number* of the graph, denoted $\chi(G)$.)

The only problem unfamiliar from the traditional theory of NP-completeness is LABELCOVER, which we describe now.

DEFINITION 10.1 LABELCOVER
Input: (i) A regular[4] bipartite graph $G = (V_1, V_2, E)$ (ii) An integer N. This defines the set of *labels*, which are integers in $\{1, 2, \ldots, N\}$. (iii) For each edge $e \in E$ a partial function $\Pi_e : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, N\}$.

A *labelling* has to associate a non-empty set of labels with every vertex in $V_1 \cup V_2$. It is said to *cover* an edge $e = (u, v)$ (where $u \in V_1, v \in V_2$) if, for every label a_2 assigned to v, there is some label a_1 assigned to u such that $\Pi_e(a_1) = a_2$. The collection of all such partial functions id denoted as Π.

LABELCOVER$_{\mathrm{max}}$: The output is a labelling that assigns one label per vertex, and maximizes the fraction of covered edges. For a label cover instance $\mathcal{L} = ((V_1, V_2, E), N, \Pi)$, we let LABELCOVER$_{\mathrm{max}}(\mathcal{L})$ denote this fraction.
LABELCOVER$_{\mathrm{min}}$: The output is a labelling that covers *all* the edges, using more than one label per vertex if necessary. Furthermore, the labelling has min-

[3]An r-partite graph is one whose vertices can be partitioned into r disjoint independent sets

[4]For our purposes a bipartite graph is *regular* if for some integers d_1, d_2, every vertex on the left (resp., right) has degree d_1 (resp., d_2).

imum *cost,* which is

$$\sum_{v \in V_1} (\text{number of labels assigned to } v)$$

(that is, the total number of labels, counting multiplicities, assigned to vertices in V_1). The objective function is $\text{cost}/|V_1|$, that is, the average number of labels used per vertex.

Note that there is no reason thus far why $\text{LABELCOVER}_{\min}$ is a well-defined problem, since there may be no feasible solution (i.e., a labelling that covers all edges) at all. So we impose the following condition on the input: (iv) For each edge e, the label 1 has a pre-image under Π_e, that is to say, a $b \in \{1, 2, \ldots, N\}$ for which $\Pi_e(b) = 1$. Thus we have guaranteed the existence of a feasible solution of cost $\leq N$: the labelling that assigns assign 1 to each vertex in V_2 and the set $\{1, 2, \ldots, N\}$ to each vertex in V_1. This labelling clearly covers all the edges. (In particular, condition (iv) thus ensures that $\text{LABELCOVER}_{\min}(\mathcal{L}) \leq N$ for every instance $\mathcal{L}$.)

The reader may wish to read Example 10.1 below to properly understand the definition of Label Cover.

10.1.2 INAPPROXIMABILITY RESULTS FOR THE CANONICAL PROBLEMS

Recall that an inapproximability result involves a reduction from SAT (or any other NP-complete decision problem) to instances of the given problem, such that there is a gap in the optimum value of the objective function depending on whether or not the boolean formula is satisfiable. This section describes such results for our canonical problems. (Figure 10.1 gives an overview of the logical dependence between these results.)

To do further reductions from the canonical problem, it helps to know the exact nature of the gap produced. As an illustration of this point, consider two hypothetical polynomial-time reductions τ_1 and τ_2 from SAT to MAX-3SAT. For some fixed $\epsilon, c > 0$, reduction τ_1 satisfies

$$\begin{aligned} I \in \text{SAT} &\Longrightarrow \text{MAX-3SAT}(\tau_1(I)) = 1, \\ I \notin \text{SAT} &\Longrightarrow \text{MAX-3SAT}(\tau_1(I)) < \frac{1}{1+\epsilon} \end{aligned}$$

and reduction τ_2 satisfies

$$\begin{aligned} I \in \text{SAT} &\Longrightarrow \text{MAX-3SAT}(\tau_2(I)) = 1 - c, \\ I \notin \text{SAT} &\Longrightarrow \text{MAX-3SAT}(\tau_2(I)) < \frac{1-c}{1+\epsilon}. \end{aligned}$$

As inapproximability results for MAX-3SAT, the reductions are equally useful: both prove the NP-hardness of achieving an approximation ratio of $(1+\epsilon)$. But as an ingredient of further inapproximability results (which involve further reductions from MAX-3SAT) the first reduction appears to be far more useful.

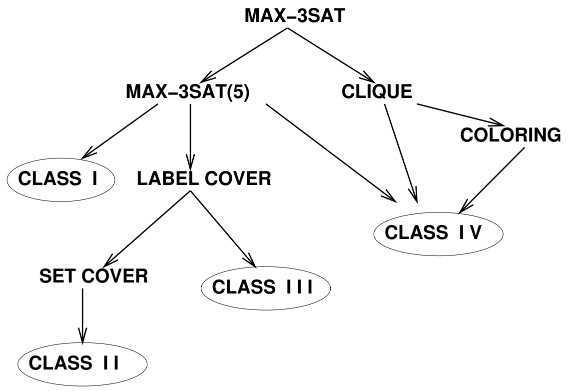

FIGURE 10.1

Diagram of the sequence of transformations used to prove the inapproximability of the six canonical problems. The six problems in turn lead to inapproximability results for problems in Classes I, II, III, and IV.

For example, we will later prove the hardness of Set-Cover and Nearest Lattice Vector problems using τ_1, whereas we do not know how to prove these results using τ_2. Luckily for us, Theorem 10.1 shows that reduction τ_1 exists.

Such peculiarities make it essential to know the precise statement of the inapproximability result for our canonical problems. The next theorem gives these statements.

THEOREM 10.2 The following reductions exist.

1. A polynomial-time reduction τ_1 from SAT to MAX-3SAT(5) that, for some fixed $\epsilon > 0$ and for all boolean formulae I, ensures:

$$I \in \text{SAT} \Longrightarrow \text{MAX-3SAT}(\tau_1(I)) = 1,$$
$$I \notin \text{SAT} \Longrightarrow \text{MAX-3SAT}(\tau_1(I)) < \frac{1}{1+\epsilon}.$$

2. A polynomial-time reduction τ_2 from SAT to R-CLIQUE that, for some fixed $\delta > 0$ and for all I, ensures:

$$I \in \text{SAT} \Longrightarrow \omega(\tau_2(I)) = 1,$$
$$I \notin \text{SAT} \Longrightarrow \omega(\tau_2(I)) \leq \frac{1}{n^\delta},$$

where n is the number of vertices in $\tau_2(I)$.

3. A quasi-polynomial-time reduction τ_3 from SAT to $\text{LABELCOVER}_{\max}$ that, for all I, ensures:

$$I \in \text{SAT} \Longrightarrow \text{LABELCOVER}_{\max}(\tau_3(I)) = 1,$$

$$I \notin \mathrm{SAT} \Longrightarrow \mathrm{LABELCOVER}_{\max}(\tau_3(I)) < \frac{1}{2^{\log^{1-\gamma} n}},$$

where γ is an arbitrarily small positive constant and n is the size of $\tau_3(I)$. (Note: Since the reduction runs in quasi-polynomial-time, n is at most $2^{\mathrm{poly}(\log |I|)}$.)

There is a similar reduction for $\mathrm{LABELCOVER}_{\min}$ that produces instances with optimum value either 1 or $> 2^{\log^{1-\gamma} n}$.

4. A quasi-polynomial-time reduction τ_4 from SAT to SETCOVER that, for all I, ensures:

$$I \in \mathrm{SAT} \Longrightarrow \mathrm{SETCOVER}(\tau_4(I)) = K(|I|),$$
$$I \notin \mathrm{SAT} \Longrightarrow \mathrm{SETCOVER}(\tau_4(I)) > K(|I|) \cdot \frac{\log n}{48},$$

where $K(|I|)$ is a polynomial-time (in $|I|$) computable function and n is the size of the ground set of the setcover instance $\tau_4(I)$. (The constant 48 can be improved somewhat.)

5. A polynomial-time reduction τ_5 from SAT to COLORING that, for some fixed $\delta > 0$ and for all I, ensures:

$$I \in \mathrm{SAT} \Longrightarrow \chi(\tau_5(I)) = K(|I|),$$
$$I \notin \mathrm{SAT} \Longrightarrow \chi(\tau_5(I)) > K(|I|)\, n^{\delta},$$

where $K(|I|)$ is a polynomial-time (in $|I|$) computable function and n is the number of vertices of $\tau_5(I)$.

Proof. The proofs appear in the following sections.

(1). 10.1.3.

(2). 10.5.1

(3). 10.4.1 and 10.4.2.

(4). 10.3.1.

(5). 10.5.2.

■

Notice that the following corollary follows immediately from the statement of Theorem 10.2.

COROLLARY 10.1 There exist fixed $\epsilon, \delta, \delta' > 0$ such that

1. Approximating MAX-3SAT and MAX-3SAT(5) within a factor $(1+\epsilon)$ is NP-hard.
2. Approximating R-CLIQUE and thus CLIQUE within a factor n^{δ} is NP-hard.
3. Approximating LABELCOVER within a factor $2^{\log^{1-\gamma} n}$ is *Quasi*-NP-hard, for any $\gamma > 0$.
4. Approximating SETCOVER within a factor $(\log n)/48$ is *Quasi*-NP-hard.

5. Approximating COLORING within a factor $n^{\delta'}$ is NP-hard.

10.1.3 GAP PRESERVING REDUCTIONS

Now we define gap-preserving reductions. We will prove inapproximability results by composing one of the reductions in Theorem 10.2 with a gap-preserving reduction. This technique is explained further in the note following the definition.

DEFINITION 10.2 Let Π and Π' be two maximization problems. A *gap-preserving reduction from* Π *to* Π' *with parameters* $(c, \rho), (c', \rho')$ is a polynomial-time algorithm f. For each instance I of Π algorithm f produces an instance $I' = f(I)$ of Π'. The optima of I and I', say $OPT(I)$ and $OPT(I')$ respectively, satisfy the following property.

$$\begin{aligned} OPT(I) \geq c &\Longrightarrow OPT(I') \geq c', \\ OPT(I) < \frac{c}{\rho} &\Longrightarrow OPT(I') < \frac{c'}{\rho'}. \end{aligned} \tag{10.1}$$

Here c and ρ are functions of $|I|$, the size of instance I, and c', ρ' are functions of $|I'|$. Also, $\rho(I), \rho'(I') \geq 1$.

Comments on Definition 10.2:

1. Suppose we wish to prove the inapproximability of problem Π'. Suppose further that we have a polynomial time reduction τ from SAT to Π that ensures, for every boolean formula φ:

$$\begin{aligned} \varphi \in \mathrm{SAT} &\Longrightarrow OPT(\tau(\varphi)) \geq c \\ \varphi \notin \mathrm{SAT} &\Longrightarrow OPT(\tau(\varphi)) < \frac{c}{\rho}. \end{aligned}$$

Then composing this reduction with the reduction of Definition 10.2 gives a reduction $f \circ \tau$ from SAT to Π' that ensures:

$$\begin{aligned} \varphi \in \mathrm{SAT} &\Longrightarrow OPT'(\,f(\tau(\varphi))\,) \geq c' \\ \varphi \notin \mathrm{SAT} &\Longrightarrow OPT'(\,f(\tau(\varphi))\,) < \frac{c'}{\rho'}. \end{aligned}$$

In other words, $f \circ \tau$ shows that achieving an approximation ratio ρ' for Π' is NP-hard. This idea of composing reductions underlies our inapproximability results.

2. Like most known reductions, ours will also map solutions to solutions in an obvious way. For instance, given a solution to I' of value at least c', a solution to I of value at least c can be produced in polynomial time. But we keep this aspect out of the definition for simplicity.

3. The above definition can be modified in an obvious way when one (or both) of the optimization problems involve minimization.

4. The gap-preserving reduction could behave arbitrarily on an instance I for which $c/\rho \leq OPT(I) < c$. Thus its "niceness" (namely, Equation 10.1) holds only on a partial domain. In this sense the gap-preserving reduction is a weaker notion than the *L-reduction* introduced in [PY91], whose "niceness" has to be maintained on *all* instances of Π. An L-reduction, coupled with an approximation algorithm for Π', yields an approximation algorithm for Π. This statement is false for a gap-preserving reduction. On the other hand, for exhibiting merely the hardness of approximation, it suffices (and is usually easier) to find gap-preserving reductions. For instance, we will prove the inapproximability of SETCOVER and the Nearest Lattice Vector problem using gap-preserving reductions, whereas no corresponding L-reductions are known.
5. The name "gap-preserving" is a bit inaccurate, since the new gap ρ' could be much bigger or much smaller than the old gap ρ.

To illustrate the concept of gap-preserving reductions, we give a reduction from MAX-3SAT to MAX-3SAT(5) that also proves Theorem 10.2, part 1.

Proof of Theorem 10.2, part 1. First we describe a reduction from MAX-3SAT to MAX-3SAT(29) (i.e., each variable appears in at most 29 clauses). Modifying instances of MAX-3SAT(29) to instances of MAX-3SAT(5) (preserving inapproximability) is easy, as we will show.

We describe a reduction τ from MAX-3SAT to MAX-3SAT(29) that is gap-preserving with parameters $(1, (1-\delta)^{-1}), (1, (1-\frac{\delta}{43})^{-1})$ for every $\delta > 0$. In other words, it ensures for every fixed $\delta > 0$ and for every 3CNF formula I, that

$$\text{MAX-3SAT}(I) = 1 \Longrightarrow \text{MAX-3SAT}(\tau(I)) = 1,$$
$$\text{MAX-3SAT}(I) < 1 - \delta \Longrightarrow \text{MAX-3SAT}(\tau(I)) < 1 - \frac{\delta}{43}.$$

Recall that Theorem 10.1 describes a reduction τ_1 from SAT to MAX-3SAT such that for some fixed $\epsilon > 0$ and for every SAT instance I:

$$I \in \text{SAT} \Longrightarrow \text{MAX-3SAT}(\tau_1(I)) = 1,$$
$$I \notin \text{SAT} \Longrightarrow \text{MAX-3SAT}(\tau_1(I)) < \frac{1}{1+\epsilon}.$$

Thus the composed reduction (that is, τ_1 followed by τ) is a reduction from SAT to MAX-3SAT(29) such that the fraction of satisfied clauses is either 1 or $1-\epsilon/(43(1+\epsilon))$ depending upon whether or not the SAT instance was satisfiable.

The description of reduction τ uses a special type of *expander* graphs. The relevant property of these graphs is that for every subset S of the vertices, the number of edges between S and its complement $\overline{S}$ is at least $\min\{|S|, |\overline{S}|\}$. As shown in [LPS88], such graphs are constructible. There is an algorithm A and a fixed integer n_0 such that given any integer $k > n_0$, algorithm A constructs in poly(k) time a 14-regular graph G_k on k vertices that is an expander.

Let I, the instance of MAX-3SAT, have n variables $y_1, y_2, \ldots, y_n$, and m clauses. Let m_i denote the number of clauses in which variable y_i appears. Let N denote the sum $\sum_i m_i$. Since a clause contains at most 3 variables, $N \leq 3m$.

Furthermore, since replicating each clause n_0 times does not change the fraction of satisfiable clauses, we can assume w.l.o.g. that each m_i is greater than n_0, the integer mentioned in the statement about expanders.

To create an instance of MAX-3SAT(29) do the following for each variable y_i. Replace y_i with m_i new variables $y_i^1, y_i^2, \ldots, y_i^{m_i}$. Use the jth new variable, y_i^j, in place of the jth occurrence of y_i. Next, to ensure that the optimum assignment assigns the same value to $y_i^1, y_i^2, \ldots, y_i^{m_i}$, add the following $14m_i$ new clauses. For each $j, l \le m_i$ such that (j, l) is an edge of the expander G_{m_i}, add a pair of new clauses $(y_i^l \vee \neg y_i^j)$ and $(\neg y_i^l \vee y_i^j)$. Together, this pair just says $(y_i^l \equiv y_i^j)$: an assignment satisfies the pair iff it assigns the same value to y_i^l and y_i^j.

Hence the new formula contains $14N$ new clauses and m old clauses. Each variable occurs in exactly 28 new clauses and 1 old clause. We claim that an optimum assignment, namely, one that satisfies the maximum number of clauses, satisfies all new clauses. For, suppose it does not satisfy a new clause corresponding to y_i. Then it does not assign the same value to all of $y_i^1, y_i^2, \ldots, y_i^{m_i}$. Divide these m_i variables into two sets S and $\overline{S}$ according to the value they were assigned. One of these sets has size at most $m_i/2$; say it is S. In the expander G_{m_i}, consider the set of $|S|$ vertices corresponding to vertices in S. Expansion implies there are at least $|S| + 1$ edges leaving this set. Each such edge yields an unsatisfied new clause. Hence by flipping the value of the variables in S, we can satisfy at least $1 + |S|$ clauses that weren't satisfied before, and possibly stop satisfying the (at most $|S|$) old clauses that contain these variables. The net gain is still at least 1. This contradicts the assumption that we started with an optimum assignment.

We have shown that the optimum assignment satisfies all new clauses, and thus assigns identical values to the different copies $y_i^1, y_i^2, \ldots, y_i^{m_i}$ of y_i for all $i \in \{1, 2, \ldots, n\}$. Thus the optimum assignment corresponds to an assignment to the old formula. Suppose the original instance of MAX-3SAT was satisfiable. Then so is the instance of MAX-3SAT(5) we created. Now suppose no assignment could satisfy more than $(1 - \delta)m$ clauses in the original formula. Then in the new formula no assignment can satisfy more than $14N + (1 - \delta)m$ clauses. Since $N \le 3m$, we see that the fraction of unsatisfied clauses is at least $\frac{\delta m}{42m+m} = \delta/43$. Hence the correctness of the reduction has been proved.

Finally, changing an instance of MAX-3SAT(29) into an instance of MAX-3SAT(5) is similar to the above transformation, but easier. Specifically, replace the expander in the above construction with a simpler graph: the cycle. Thus if a variable appears l times, replace it by l new variables, and add new clauses corresponding to edges in a cycle on l vertices. The reader can easily check that each variable in the new formula appears in exactly 5 clauses. Further, if only $1 - \gamma$ fraction of clauses could be satisfied in the old formula, then in the new formula the fraction of satisfiable clauses is at most $1 - \gamma/29$.

Note: The fact about expanders that we quoted above is only approximately correct. The construction in [LPS88] may be unable to construct expanders of size n for some integers n. However, it is able to construct an expander of size $\le n(1 + o(1))$ for every n. The reader can easily check that this does not affect the correctness of our construction. ■

Next, we give another example of a gap-preserving reduction, which will be useful later.

Example 10.1 We describe a reduction τ from MAX-3SAT(5) to LABELCOVER$_{\max}$ that is gap-preserving with $c = c' = 1$, and $\rho = (1-\epsilon)^{-1}$, $\rho' = (1-\epsilon/9)^{-1}$. In other words, for every input φ to MAX-3SAT(5) it ensures

$$\begin{aligned} \text{MAX-3SAT}(\varphi) = 1 &\Longrightarrow \text{LABELCOVER}_{\max}(\tau(\varphi)) = 1 \\ \text{MAX-3SAT}(\varphi) < 1 - \epsilon &\Longrightarrow \text{LABELCOVER}_{\max}(\tau(\varphi)) < 1 - \frac{\epsilon}{9}. \end{aligned} \tag{10.2}$$

Given an instance of MAX-3SAT(5), produce an instance of Label Cover as follows. Let V_1 have one vertex for each clause and V_2 have a vertex for every variable. Let adjacency correspond to the variable appearing in the clause, whether negated or unnegated. The number of labels, N, is 8. For a vertex in V_1, if the corresponding clause involves variables x_i, x_j, x_k, the reader should think of a label as a 3-bit binary vector (b_1, b_2, b_3), corresponding to the assignment $x_i = b_1, x_j = b_2, x_k = b_3$. For a vertex in V_2, say one corresponding to variable x_i, the labels will be either 0 or 1, corresponding respectively to a truth assignment to x_i (the remaining 6 labels will be dummy labels and not useful to cover any edges).

The edge function Π_e is described as follows. Suppose e is the edge (u, v) where $u \in V_1$ corresponds to clause C, and $v \in V_2$ corresponds to variable x_i. Thus we know that x_i appears in C and let j be the place where x_i appears. Let $\Pi_e(b_1, b_2, b_3)$ be defined and equal to b_j if and only if the assignment b_1, b_2, b_3 satisfies C.

Since each vertex is allowed only 1 label in the maximization version of LABELCOVER, the labels on the right hand size vertices constitute a boolean assignment to $x_1, x_2, \ldots, x_n$. The label on a left-hand vertex also constitute an assignment to the variables in the corresponding clause. The edge joining a clause-vertex and a variable-vertex is covered iff that variable is assigned the same value by both assignments and the assignment satisfies the clause.

Clearly, if all edges are covered then the assignment is a satisfying assignment. Conversely, if no assignment satisfies more than $1-\epsilon$ of the clauses, then every labelling must fail to cover an edge incident to at least ϵ fraction of the clause-vertices, in other words a fraction at least $\epsilon/3$ of all edges.

This completes the proof except for the fact that the graph (V_1, V_2, E) is not regular. It can be proven that we can extend the reduction by adding at most $2|E|$ dummy edges such that the new graph is regular and such that all the dummy edges are easily coverable. This will only decrease the gap by a factor of 3. The details are left to the reader. Thus this completes the proof of the property claimed in (10.2).

10.2 INAPPROXIMABILITY RESULTS FOR PROBLEMS IN CLASS I

Class I contains problems for which achieving an approximation ratio $1 + \epsilon$ is NP-hard, for some fixed $\epsilon > 0$. To prove that a problem is in Class I, we will give simple gap-preserving reductions from MAX-3SAT.

Papadimitriou and Yannakakis had identified Class I a few years before the discovery of the PCP Theorem. They called it the class of Max-SNP-hard problems. The next section describes their work.

10.2.1 MAX-SNP

NP-hard optimization problems exhibit a vast range of behaviors when it comes to approximation. Papadimitriou and Yannakakis [PY91] identified a large subclass of them that exhibit the same behavior. The authors defined a class of optimization problems, Max-SNP, as well as a notion of completeness for this class. Roughly speaking, a Max-SNP-complete problem is one that behaves just like MAX-3SAT in terms of approximability: MAX-3SAT is hard to approximate upto some constant factor iff so is every Max-SNP-complete problem. (This made MAX-3SAT a plausible candidate problem to prove hard to approximate, and in particular motivated the discovery of the PCP theorem.)

Max-SNP contains constraint-satisfaction problems, where the constraints are local. The goal is to satisfy as many constraints as possible. The concept of "local" constraints is formalized using logic: constraints are local iff they are definable using a quantifier-free propositional formula.

DEFINITION 10.3 A maximization problem is in Max-SNP if there is a sequence of relation symbols $G_1, \ldots, G_m$, a relation symbol S, and a quantifier-free formula $\phi(G_1, \ldots, G_m, S, x_1, \ldots, x_k)$ (where each x_i is a variable) such that the following are true (i) there is a polynomial-time algorithm that, given any instance I of the problem produces a set $\mathcal{U}$ and a sequence of relations $G_1^{\mathcal{U}}, \ldots, G_m^{\mathcal{U}}$ on $\mathcal{U}$, where each $G_i^{\mathcal{U}}$ has the same arity ("arity" refers to the number of arguments) as the relation symbol G_i. (ii) The value of the optimum solution on instance I, denoted $\mathrm{OPT}(I)$, satisfies

$$\mathrm{OPT}(I) = \max_{S^{\mathcal{U}}} \left| \left\{ (x_1, \ldots, x_k) \in \mathcal{U}^k : \phi(G_1^{\mathcal{U}}, \ldots, G_m^{\mathcal{U}}, S^{\mathcal{U}}, x_1, \ldots, x_k) = \text{ TRUE} \right\} \right|,$$

where $S^{\mathcal{U}}$ is a relation on $\mathcal{U}$ with the same arity as S, and $\mathcal{U}^k$ is the set of k-tuples of $\mathcal{U}$.

Note: The above definition is inspired by Fagin's model-theoretic characterization of NP [Fag74], and an explanation is in order for those unfamiliar with model theory. The sequence of relation symbols $G_1, \ldots, G_m, S$, as well as their arities, are fixed for the problem. Thus when the universe $\mathcal{U}$ has size n, the sequence of relations $G_1^{\mathcal{U}}, \ldots, G_m^{\mathcal{U}}$ implicitly defines an "input" of size $O(n^c)$ where c is the largest arity of a relation in G. Solving the optimization problem involves finding a relation $S^{\mathcal{U}} \subseteq \mathcal{U}^k$ that maximizes the number of k-tuples satisfying ϕ. Since $\mathcal{U}^k = n^k$, this relation $S^{\mathcal{U}}$ can be viewed as a "feasible solution" which can be specified using n^k bits.

Example 10.2 Let MAX-CUT be the problem of partitioning the vertex set of an undirected graph into two parts such that the number of edges crossing the partition is maximized. Here's how to see it is in Max-SNP. Let the universe $\mathcal{U}$ be the vertex set of the graph, and let G consist of E, a binary relation whose interpretation is "adjacency." Let S be a unary relation (interpreted as one side of the cut), and $\phi(E, S, (u, v)) = (u < v) \wedge E(u, v) \wedge (S(u) \neq S(v))$. Clearly, the optimum value of MAX-CUT on the graph is $\max_{S \subseteq \mathcal{U}} |\{(u, v) \in \mathcal{U}^2 : \phi(E, S, (u, v)) = \text{ TRUE}\}|$.

The reader should check that MAX-3SAT and MAX-3SAT(5) are also in Max-SNP. The paper [PY91] identified many other Max-SNP problems. It also showed that for every Max-SNP problem, there is some constant $c \geq 1$ such that some polynomial-time algorithm can achieve an approximation ratio c for the problem (see the exercises and [PY91]). The smallest value of c for which this is true remains an object of research. For example, an algorithm to approximate MAX-CUT within a factor 1.13 appeared recently in [GW94].

There is a notion of *completeness* in the class Max-SNP. According to the original definition, a Max-SNP problem is complete for the class if every Max-SNP problem can be reduced to it using a (so-called) L-reduction. For purposes of proving inapproximability results, it suffices to concentrate on a notion somewhat weaker than an L-reduction, as described in in the following definition.

DEFINITION 10.4 A maximization problem Π is *Max-SNP-hard* if for every every Max-SNP problem Γ and every two constants $c \leq 1, \rho > 1$, there are two constants $c' < 1, \rho' > 1$ such that there is a gap-preserving reduction from Γ to Π with parameters (c, ρ, c', ρ').

(Notes: (i) For notational ease, we are assuming, just as in the case of MAX-3SAT, that the value of the optimum is a fraction. Hence $c, c' \leq 1$. (ii) The definition implies that if there is a Max-SNP problem Γ which is hard to approximate within any fixed factor $\rho > 1$, then there is some $\rho' > 1$ such that approximating Π within a factor ρ' is hard.)

Examples of Max-SNP-hard problems include MAX-CUT, MAX-3SAT(5), and many others (see Section 10.6 for a partial list). Note that to prove the hardness of approximating *all* Max-SNP-hard problems, it suffices to exhibit just *one* problem $\Gamma \in$ Max-SNP and *some* $\epsilon > 0$, such that achieving an approximation ratio $(1 + \epsilon)$ for Γ is *NP*-hard. But we already exhibited such a problem in Max-SNP, namely MAX-3SAT! Thus the following corollary to Theorem 10.1

is immediate.

COROLLARY 10.2 For every Max-SNP-hard problem, there exists some $c > 1$ such that achieving an approximation ratio c for it is NP-hard.

We end this section with another example of a Max-SNP-hard problem: CLIQUE. This proof of Max-SNP-hardness will be used later to show a stronger inapproximability result for CLIQUE.

LEMMA 10.1 For every $\epsilon > 0$, there is a gap-preserving reduction from MAX-3SAT to CLIQUE that has parameters $(c, 1+\epsilon), (cN/3, 1+\epsilon)$, where N is the number of vertices in the new graph. In other words, CLIQUE is Max-SNP-hard.

Proof. A textbook reduction (a modification of the one in [GJ79]) from 3SAT to (the decision version of) CLIQUE works.

Let φ be a 3CNF formula in variables $x_1, x_2 \ldots, x_n$. By replicating literals within clauses we can ensure that each clause has 3 literals (e.g., if the clause is x_i then change it to $x_i \wedge x_i \wedge x_i$). Construct a new graph $\tau(\varphi)$ on $3m$ vertices as follows. Represent each clause with a triple of vertices, one per literal. Put no edges between vertices within the same triple. If u, v are vertices in two different triples, put an edge between them iff the literals they stand for are not the negations of each other.

Notice, a clique in this graph can contain only one vertex per triple. Furthermore, it cannot contain two vertices representing literals that are negations of each other. In other words, by looking at the literals represented in the clique we can write in a natural way a partial assignment that satisfies as many clauses as there are vertices in the clique. Thus,

$$\text{MAX-3SAT}(\varphi) = c \Longrightarrow \omega(\tau(\varphi)) = cm$$
$$\text{MAX-3SAT}(\varphi) < \frac{c}{1+\epsilon} \Longrightarrow \omega(\tau(\varphi)) < \frac{cm}{1+\epsilon}. \blacksquare$$

PROBLEMS

EXERCISE 10.1

Prove the following result from [PY91]: For every Max-SNP problem, there is some constant $c \geq 1$ such that some polynomial-time algorithm can achieve an approximation ratio c for the problem.

EXERCISE 10.2

Prove that the following optimization problems are in Max-SNP: MAX-3SAT and Minimum Vertex Cover restricted to graphs of degree 5.

INAPPROXIMABILITY RESULTS FOR PROBLEMS IN CLASS II

10.3

In this section we show how to prove the hardness of achieving an approximation ratio $O(\log n)$ for problems in Class II. The canonical problem in Class II is SETCOVER. In Section 10.3.1 we describe the hardness result for SETCOVER, and indicate how to use it to prove the inapproximability of other problems in the class.

10.3.1 SETCOVER

We prove Theorem 10.2, part 4 about the inapproximability of SETCOVER. Specifically, we give a polynomial-time reduction τ from LABELCOVER$_{\max}$ to SETCOVER. Composing it with the inapproximability result for LABELCOVER$_{\max}$ (namely, part 3 of Theorem 10.2, proved Section 10.4.1) gives the desired result.

For every Label-cover instance $\mathcal{L} = ((V_1, V_2, E), N, \Pi)$, the reduction in this section produces an instance $\mathcal{S} = \tau(\mathcal{L})$ of SETCOVER such that

$$\text{LABELCOVER}_{\max}(\mathcal{L}) = 1 \Longrightarrow \text{SETCOVER}(\mathcal{S}) = |V_1| + |V_2|,$$
$$\text{LABELCOVER}_{\max}(\mathcal{L}) \leq \frac{1}{\log^3(|\mathcal{L}|)} \Longrightarrow \text{SETCOVER}(\mathcal{S}) \geq \frac{\log|\mathcal{S}|}{48} \cdot (|V_1| + |V_2|),$$

where $|\mathcal{L}| = N \cdot |E|$ is the size of $\mathcal{L}$ and $|S|$ is the number of sets in the set-cover instance. (Note: in the instances of SETCOVER produced by the reduction, the number of sets and the size of the ground set are polynomially related. So $|S|$ in the above statement could also stand for the size of the ground set; then we need to change "48" to some other constant.)

The reduction uses the following set system as a basic building block.

DEFINITION 10.5 Let m and l be positive integers, B some finite set, and $C_1, C_2, \ldots, C_m$ a collection of m subsets of B. The subsets form an (m,l) *set-system* if for every set I of at most l indices from $\{1, 2, \ldots, m\}$,

$$\bigcup_{i \in I} D_i \neq B,$$

where each D_i is either C_i or complement of C_i.

Lemma 10.3 gives explicit constructions of (m,l) set-system for all m, l, where $|B| = O(2^{2l} m^2)$. Denote the set system as $\mathcal{B}_{m,l}$. We will use $l = O(\log m)$, in which case the size of $\mathcal{B}_{m,l}$ is poly(m).

Now we describe the reduction. First, we assume w.l.o.g. that $|V_1| = |V_2|$. For, if $|V_1| \neq |V_2|$, then just construct a new bipartite graph (V_1', V_2', E') with $|V_1||V_2|$ vertices on each side (i.e., $|V_2|$ copies of V_1 and $|V_1|$ copies of V_2), and the new set of edges E' consisting of copies of E between each new copy of V_1 and V_2. If the old instance had a labelling that covers all edges (i.e., has value 1), then so does the new one. Conversely, if the old instance had no labelling of value $\geq \rho$ then neither does the new one. Thus we can assume that $|V_1| = |V_2|$.

Let l be an even integer (to be determined later) and $m = N$. Let $\mathcal{B}_{m,l} = (B; C_1, C_2, \ldots, C_m)$ be an (m,l)-system. Since the labels in the LABELCOVER instance are integers from 1 to N, we can talk about the set C_a for any label a. The instance of SETCOVER is as follows. Its ground set U is $E \times B$. The given collection of subsets of $\mathcal{S}$ contains a set $S_{v,a}$ for every vertex v in $G = (V_1, V_2, E)$ and label a. For every $u \in V_1$ and $a_1 \in \{1, 2, \ldots, N\}$, let S_{u,a_1} be defined by

$$S_{u,a_1} = \{(e,b) | e = (u,v),\ \Pi_e(a_1) \text{ is defined and } b \notin C_{\Pi_e(a_1)}\}.$$

(That is, S_{u,a_1} is the union of all sets of the type $\{e\} \times (B - C_{a_2})$, where e is an edge containing u and a_2 is the image $\Pi_e(a_1)$ of a_1.)

For every $v \in V_2$ and $a_2 \in \{1, 2, \ldots, N\}$, let S_{v,a_2} be defined by

$$S_{v,a_2} = \{(e,b) | e = (u,v) \text{ and } b \in C_{a_2}\}.$$

(That is, $S_{v,a_2} = \cup_{e \ni v} \{e\} \times C_{a_2}$.)

Note that the ground set U can be viewed as $\cup_{e \in E} \{e\} \times B$, that is, as $|E|$ copies of B, one per edge. This observation underlies the next claim.

CLAIM 10.1 If LABELCOVER$_{\max}(\mathcal{L}) = 1$ then SETCOVER$(\mathcal{S}) = |V_1| + |V_2|$.

Proof. Consider an optimal label cover, which uses one label per vertex to cover all edges. For a vertex $u \in V_1 \cup V_2$, let a_u be the label it assigns to u. We show that the collection of $|V_1| + |V_2|$ sets

$$\{S_{w,a_w} : w \in V_1 \cup V_2\}$$

is a set cover.

Let $e = (u,v)$ be any edge. Since the pair of labels a_u, a_v covers it, a_v must be $\Pi_e(a_u)$. The definition of the set-cover instance ensures that $\{e\} \times C_{a_v}$ is contained in S_{v,a_v} and $\{e\} \times (B - C_{a_v})$ in S_{u,a_u}. Hence $\{e\} \times B \subseteq S_{u,a_1} \cup S_{v,a_2}$. Since the same is true for every edge e, it follows that

$$E \times B \subseteq \bigcup_{w \in V_1 \cup V_2} S_{w,a_w}.$$

Thus we have exhibited a set cover of size $|V_1| + |V_2|$. ∎

The next lemma is more nontrivial. It uses the following restatement of the property of an (m,l)-system $(B; C_1, C_2, \ldots, C_m)$: If the union of any collection of l sets out of $\{C_1, C_2, \ldots, C_m, \overline{C_1}, \ldots, \overline{C_m}\}$ is B, then the collection must contain both C_i and $\overline{C_i}$ for some i.

LEMMA 10.2 Suppose there is a set cover of size less than $\frac{l(|V_1|+V_2|)}{16}$. Then there is a labelling that uses 1 label per vertex and covers a $\frac{2}{l^2}$ fraction of the edges (in other words, $\text{LABELCOVER}_{\max}(\mathcal{L}) \geq \frac{2}{l^2}$).

Proof. Let I be the collection of sets in the set cover. We say that I *associates a label a with vertex u* if $S_{u,a} \in I$. Since $|V_1| = |V_2|$, and I associates at most $l/16$ labels per vertex on average, at least $3/4$ of the vertices in both V_1 and V_2 have fewer than $l/2$ labels associated with them.

Now define a labelling by picking, for each vertex, a random label from the set of labels associated with it. We show now that the expected fraction of edges covered by this labelling is at least $2/l^2$. Note that this also implies the *existence* of a labelling that covers at least this fraction.

First, we claim that for 1/2 the edges, the end-points have fewer than $l/2$ labels associated with them. To see this, imagine picking an edge $e = (u, v)$ uniformly at random. Since the bipartite graph is regular, the endpoints u, v of this edge are also distributed uniformly (but not necessarily independently) in V_1, V_2 respectively. With probability at least $1 - 2 \cdot 1/4 = 1/2$, both end points have at most $l/2$ labels. Thus our claim follows.

Now let $e = (u, v)$ be an edge such that both u, v have fewer than $l/2$ labels associated with them. We claim that with probability at least $(2/l)^2$ the randomly-picked labelling covers this edge. The reason is that $\mathcal{C}$ is a set cover, and so

$$\{e\} \times B \subseteq \left(\bigcup_{a_1} S_{u,a_1}\right) \cup \left(\bigcup_{a_2} S_{v,a_2}\right),$$

where the two unions range over the sets of labels associated with u and v respectively. But recall the property of an (m, l)-system: the only way to cover B with fewer than l sets is to use two sets that are complements of one another. In our construction, complementary sets correspond to pairs of labels that can cover the edge. We conclude that there are some labels a_1, a_2 associated with u, v respectively such that $\Pi_e(a_1) = a_2$. Now, let us pick labels randomly for u, v from among all labels associated with them. With probability at least $(2/l)^2$ we pick a_1 and a_2, in which case the labelling covers edge e.

To conclude, we have shown that for 1/2 the edges, the labelling has a probability at least $(2/l)^2$ of satisfying that edge. Hence the expected number of covered edges is at least $1/2 \cdot (4/l^2) = 2/l^2$. ∎

The following corollary finishes the proof of the correctness of the reduction. The proof also specifies the values of various parameters that were not stated above.

COROLLARY 10.3 Let $\mathcal{L}$ be any instance of $\text{LABELCOVER}_{\max}$ as described above, and $\mathcal{S}$ be the instance of SETCOVER produced using the above reduction. If $\text{LABELCOVER}_{\max}(\mathcal{L}) \leq 1/\log^3 |\mathcal{L}|$ then

$$\text{SETCOVER}(\mathcal{S}) > \frac{\log |\mathcal{S}|}{48}(|V_1| + |V_2|),$$

where $|\mathcal{S}| = N(|V_1| + |V_2|)$ is the number of sets in $\mathcal{S}$ and $|\mathcal{L}| = N \cdot |E|$ is the size of $\mathcal{L}$.

Proof. Let $l = 4\lceil \log |\mathcal{L}| \rceil$. Assume that $\mathcal{S}$ has a cover of size less than $\frac{l\,(|V_1|+V_2|)}{16}$. Then Lemma 10.2 implies that

$$\text{LABELCOVER}_{\max}(\mathcal{L}) \geq 2/l^2 = 1/8\lceil \log |\mathcal{L}| \rceil^2 > 1/\log^3 |\mathcal{L}|$$

for large $|\mathcal{L}|$. This is a contradiction. Hence SETCOVER$(\mathcal{S}) \geq \frac{l\,(|V_1|+V_2|)}{16}$.

Now we express l in terms of $\log |\mathcal{S}|$. Recall that $|B| = O(2^{2l}m^2)$ and $m = N$. Hence

$$\begin{aligned} \log |\mathcal{S}| &= \log(N(|V_1| + |V_2|)) &\leq& \log(m) + 2\log(|\mathcal{L}|) \\ && < & \ 3l \end{aligned}$$

Thus SETCOVER$(\mathcal{S}) > \frac{\log |\mathcal{S}|}{48}(|V_1| + |V_2|)$.

■

Note that the previous Corollary continues to hold if we define $|S|$ to be the sum of the number of sets in S and the size of their ground set.

To finish the description of the reduction, we show how to construct an (m, l)-system.

LEMMA 10.3 There is an explicit algorithm to construct, given integers m, l where $l < m$, an (l, m)-system $(B; C_1, C_2, \ldots, C_m)$ with $|B| = O(2^{2l}m^2)$. The algorithm runs in time poly$(|B|)$.

Proof. We note that to construct the (m, l) system it suffices to construct a collection X of m-bit vectors with the following property: Given any l distinct coordinates $i_1, i_2, \ldots, i_l$ and any l-bit sub-sequence $(y_1, y_2, \ldots, y_l)$ there exists a vector $x \in X$ whose components in the indicated coordinates exactly match the bits of y. That is, $x_{i_j} = y_j$ for $j = 1, 2, \ldots, l$.

To see that it suffices to construct this collection of vectors, assume we are given a collection X. Define the set-system with $B = X$ and C_i as $\{x \in X | x_i = 1\}$, the set of vectors with 1 in the ith coordinate. Suppose $i_1, i_2, \ldots, i_l$ is any set of indices in $\{1, 2, \ldots, m\}$ and $D_{i_1}, D_{i_2}, \ldots, D_{i_l}$ is a collection of subsets of X such that D_{i_j} is equal to either C_{i_j} or the complement of C_{i_j}. We claim that $\cup_{j \leq l} D_{i_j}$ cannot be B. For, define an l-bit subsequence y such that $y_j = 0$ if $D_{i_j} = C_{i_j}$ and 1 otherwise. The known property of X implies there exists a vector $x \in X$ such that $x_{i_j} = y_j$ for $j = 1, 2, \ldots, l$. By construction, $x \notin D_{i_j}$ for all $j = 1, 2, \ldots, l$. Thus we have shown $\cup_{j \leq l} D_{i_j} \neq B$. Since this is true for every sequence $i_1, \ldots, i_l$, it follows that $(B, C_1, \ldots, C_m)$ is an (m, l)-system.

An explicit construction[5] of the desired collection of vectors follows from a result of Naor and Naor on *ϵ-biased distributions* ([NN93]; for an improved construction, see [AGHP92]).

[5] A randomized construction is quite trivial; see the exercises.

Given m, l and a rational $\epsilon > 0$ they show how to construct a set X_ϵ of $O(m^2/\epsilon^2)$ vectors of m bits each, with the following property. Let $i_1, i_2, \ldots, i_l$ be any l distinct coordinates and $(y_1, y_2, \ldots, y_l)$ be any l-bit sub-sequence. Consider two experiments: *(i)* picking a sequence x uniformly at random from X_ϵ, and *(ii)* picking a sequence z uniformly at random from the set of all 2^m strings of length m. Then, regardless of i_j's and the y_j's, the collection satisfies the inequality $|p_z - p_x| < \epsilon$, where p_z and p_x are the probabilities that z and x respectively matches y in the indicated coordinates. But, $p_z = 2^{-l}$. Hence $p_x \in [\frac{1}{2^l} - \epsilon, \frac{1}{2^l} + \epsilon]$. By choosing $\epsilon = 2^{-(l+1)}$ we see that the number of vectors is $O(m^2 2^{2l})$. Furthermore, $p_x \geq \frac{1}{2^l} - \frac{1}{2^{l+1}} > 0$, implying there exists a $x \in X_\epsilon$ that matches y. Thus we have shown how to construct the desired set-system. ■

Briefly, we indicate the proof of Theorem 10.2, part 4.

Proof of Part 4 of Theorem 10.2. Part 3 of Theorem 10.2 describes a reduction from SAT to LABELCOVER$_{\max}$. By composing that reduction with the reduction described in this section, part 4 follows. ■

REMARK 10.3 The reduction to SETCOVER is easily modified to prove hardness results for HITTING SET, HYPERGRAPH TRANSVERSAL, DOMINATING SET and MINIMUM EXACT COVER (see Table 10.2). Also, the constant $c = 1/48$ in the inapproximability result is not obtimal. In fact, Feige [Fei95] proved for any $\epsilon > 0$ that approximating SETCOVER within a factor $(1+\epsilon)\ln|\mathcal{S}|$ is *Quasi-NP*-hard. It is an open question whether approximating SETCOVER within a factor $\Omega(\log n)$ is *NP*-hard. A result in [BGLR93] implies that approximating within any constant factor is *NP*-hard. An alternative proof of the [BGLR93] result follows from the reduction of this section, in conjunction with the *product* technique of Section 10.4.1, which implies that approximating LABELCOVER$_{\max}$ within any constant factor is *NP*-hard.

PROBLEMS

EXERCISE 10.3 (Randomized construction of (m, l) systems)

Let l, m be "large enough" positive integers such that $2l < \log m$. Let $X = \{0,1\}^m$ be the set of sequences of bits of length m. Suppose we pick a set B uniformly at random from among all subsets of X of size $2^{2l}m^2$, and set $C_i = \{x \in B : \text{ the } i\text{th bit of } x \text{ is } 1\}$. Prove that the probability that $(B, C_1, \ldots, C_m)$ is an (m, l) set system is $1 - o(1)$.

EXERCISE 10.4

Use the inapproximability of SETCOVER to prove the inapproximability of

HITTING SET, HYPERGRAPH TRANSVERSAL, DOMINATING SET and MINIMUM EXACT COVER. These problems are defined in [GJ79]. (*Hint:* Try to show that SETCOVER is a subcase of these problems.)

10.4 INAPPROXIMABILITY RESULTS FOR PROBLEMS IN CLASS III

In this section we show hardness results for problems in Class III. Achieving a ratio $2^{\log^{1-\gamma} n}$ for these problems is quasi-NP-hard, for every fixed $\gamma > 0$. LABELCOVER (in both its minimization and maximization versions) is the canonical problem in this class.

Sections 10.4.1 and 10.4.2 outline the inapproximability result for the max. and the min. versions of Label Cover. A hardness result for the nearest lattice vector problem appears in Section 10.4.3. This result, involving a gap-preserving reduction from Label Cover, is illustrative of the hardness results for Class III.

10.4.1 LABELCOVER (MAXIMIZATION VERSION)

We give a quasi-polynomial-time reduction from SAT to LABELCOVER (maximization version) that maps satisfiable formulae to LABELCOVER instances in which some labelling can cover all edges (i.e., the labelling has value 1), and unsatisfiable formulae to instances in which every labelling covers at most $2^{-\log^{1-\gamma} N}$ fraction of edges, where N is the size of the instance. The proof of the correctness of the reduction relies upon a difficult result from [Raz94], which we will not prove.

The starting point is a gap-preserving polynomial-time reduction τ from MAX-3SAT(5) to LABELCOVER$_{\max}$ from Example 10.1. The reduction ensures for all 3CNF formulae I and all $\epsilon > 0$ that:

$$\text{MAX-3SAT}(I) = 1 \Longrightarrow \text{LABELCOVER}_{\max}(\tau(I)) = 1,$$
$$\text{MAX-3SAT}(I) < (1-\epsilon) \Longrightarrow \text{LABELCOVER}_{\max}(\tau(I)) < (1 - \frac{\epsilon}{9}).$$

Note that by composing this reduction with the reduction from SAT to MAX-3SAT(5) in part 1 of Theorem 10.2, we get a polynomial-time reduction τ' from SAT to LABELCOVER$_{\max}$ which, for some $\delta > 0$, ensures

$$I \in \text{SAT} \Longrightarrow \text{LABELCOVER}_{\max}(\tau'(I)) = 1,$$
$$I \notin \text{SAT} \Longrightarrow \text{LABELCOVER}_{\max}(\tau'(I)) < (1 - \delta). \tag{10.3}$$

Thus the reduction produces a gap: the optimum value in the LABELCOVER$_{\max}$ instance is either 1 or less than $1-\delta$. We desire a bigger gap: the value should be either 1 or less than $2^{-\log^{1-\gamma} N}$ where N is the size of Label Cover instance in question, and γ is any fixed positive rational number. We use the following construction to "boost" the gap.

Given any instance $\mathcal{L} = (V_1, V_2, E, N, \Pi)$ of LABELCOVER$_{\max}$ we define its *kth power*, denoted $\mathcal{L}^k$, as follows. It is another instance of LABELCOVER$_{\max}$ whose underlying bipartite graph is (V_1^k, V_2^k, E^k), where V_i^k is the set of all k-tuples of V_i and E^k, the new set of edges, is the set of all k-tuples of old edges. The number of new labels is N^k, and the set of labels are to be viewed as k-tuples of labels of $\mathcal{L}$, i.e., as elements of $\{1, 2, \ldots, N\}^k$. (Thus a labelling must now assign k-tuples of labels from $\{1, 2, \ldots, N\}$ to vertices in $V_1^k \cup V_2^k$.) The new edge-functions, denoted Π^k, are defined as follows. Let e be a k-tuple of edges $(e_1, e_2, \ldots, e_k)$, where each e_i is an edge in the original graph. Then define

$$\Pi_e^k(a_1^1, a_1^2, \ldots, a_1^k) = (\Pi_{e_1}(a_1^1), \Pi_{e_2}(a_1^2), \ldots, \Pi_{e_k}(a_1^k)).$$

In other words, the pair of k-tuples of labels $((a_1^1, a_1^2, \ldots, a_1^k), (a_2^1, a_2^k, \ldots, a_2^k))$ can be used to cover the new edge $(e_1, \ldots, e_k)$ iff for $i = 1, \ldots, k$ the pair (a_1^i, a_2^i) can be used to cover the edge e_i.

The reduction consists in constructing $\mathcal{L}^k$ for some suitable k. Since $\mathcal{L}^k$ has size $N = n^{O(k)}$, where n is the size of $\mathcal{L}$, the running time of the reduction is poly(N). If LABELCOVER$_{\max}(\mathcal{L}) = 1$ to start with, then LABELCOVER$_{\max}(\mathcal{L}^k) = 1$, as is easily checked. But Lemma 10.4 below shows that if LABELCOVER$_{\max}(\mathcal{L}) < 1 - \delta$ to start with, then LABELCOVER$_{\max}(\mathcal{L}^k) < (1-\delta)^{ck}$, where $c > 0$ is a fixed constant. (Note that the number of labels, N, is $O(1)$ in instances of Label Cover obtained in Example 10.1.) Now for $k = (\log n)^{O(\frac{1-\gamma}{\gamma})}$ we get that the size of $\mathcal{L}^k$ m is $2^{\mathrm{poly}(\log n)}$ and the gap $1/(1-\delta)^{ck}$ is at least $2^{\log^{1-\gamma} m}$. This finishes the description of the reduction.

The next lemma, also called Raz's *Parallel Repetition Theorem* [6] shows that if LABELCOVER$_{\max}(\mathcal{L}) < 1$, then LABELCOVER$_{\max}(\mathcal{L}^k)$ decreases exponentially as we increase k.

LEMMA 10.4 Implicit in [Raz94] There is a fixed constant $c > 0$ such that for any LABELCOVER instance $\mathcal{L}$: LABELCOVER$_{\max}(\mathcal{L}^k) \leq$ LABELCOVER$_{\max}(\mathcal{L})^{\frac{ck}{\log N}}$, where N is the number of labels allowed in $\mathcal{L}$.

Raz's result implies that LABELCOVER is in the class of problems with a self-improvement property (see Section 10.5.1). His proof is quite complicated.

[6] This Lemma had been conjectured for many years in the literature on interactive proofs. It can be viewed as saying that parallel repetition of a 2 Prover 1 Round proof system drives down the error probability at an exponential rate.

10.4.2 LABELCOVER (MIN. VERSION)

The hardness result for the min. version of Label Cover follows from the hardness result for the max. version, since the two versions are linked by the following "weak duality."

LEMMA 10.5 For any instance I of LABELCOVER:

$$\text{LABELCOVER}_{\max}(I) \geq \frac{1}{\text{LABELCOVER}_{\min}(I)}.$$

To see why the inapproximability of the min. version follows, recall our quasi-polynomial-time reduction τ' from SAT to $\text{LABELCOVER}_{\max}$. It ensures for all I:

$$I \in \text{SAT} \Longrightarrow \text{LABELCOVER}_{\max}(\tau'(I)) = 1,$$
$$I \notin \text{SAT} \Longrightarrow \text{LABELCOVER}_{\max}(\tau'(I)) < \rho,$$

where $\rho = 2^{-\log^{1-\gamma} n}$ for some fixed $\gamma > 0$, and n is the size of the Label Cover instance.

But $\text{LABELCOVER}_{\max}(\tau'(I)) = 1$ means there is a way to assign one label per vertex such that all edges are covered, hence $\text{LABELCOVER}_{\min}(\tau'(I)) = 1$. Further, the weak duality implies that if $\text{LABELCOVER}_{\max}(\tau'(I)) < \rho$, then $\text{LABELCOVER}_{\min}(\tau'(I)) > 1/\rho$.

Thus τ' can be viewed as a reduction to $\text{LABELCOVER}_{\min}$ as well, with a gap of $2^{\log^{1-\gamma} N}$ in the cost of the minimum label cover depending on whether or not the boolean formula was satisfiable.

Thus to finish the proof of the inapproximability of $\text{LABELCOVER}_{\min}$ it suffices to prove the weak duality.

Proof of Lemma 10.5. Let $OPT_{\min}$ and $OPT_{\max}$ be shorthands for $\text{LABELCOVER}_{\min}(I)$ and $\text{LABELCOVER}_{\max}(I)$ respectively. Consider the solution of the minimization version, namely, a labelling using on an average $OPT_{\min}$ labels per vertex in V_1 and covering all the edges. For any vertex $u \in V_1$ let the labelling assign n_u labels to u. Then by definition of cost

$$\sum_{u \in V_1} n_u = OPT_{\min} \cdot |V_1|.$$

We randomly delete all labels for each vertex in $V_1 \cup V_2$ except one. This gives a labelling that assigns only 1 label per vertex, in other words, a candidate solution for the maximization version. The original labelling covered every edge. In other words, for each edge $e = (u, v)$ and for every label a_2 assigned to v it assigns a label a_1 to u such that $\Pi_e(a_1) = a_2$. The probability that this preimage a_1 survives is $1/n_u$. Therefore in the new (randomly constructed) labelling the expected number of edges still left covered is at least

$$\sum_{(u,v)\in E} \frac{1}{n_u}.$$

Since each vertex in V_1 has the same degree, say d, the number of edges, $|E|$, is $d\,|V_1|$, and the above expectation can be rewritten as

$$\begin{aligned}\sum_{u\in V_1} \frac{d}{n_u} &= d\sum_{u\in V_1}\frac{1}{n_u} &\geq d\frac{|V_1|^2}{\sum_{u\in V_1} n_u}\\ &= d\,\frac{|V_1|^2}{OPT_{\min}|V_1|} &= \frac{|E|}{OPT_{\min}}.\end{aligned}$$

The crucial fact used above is that $\sum_u 1/n_u$ is minimized when all n_u are equal.

Thus we have shown a randomized way to construct a candidate solution to the maximization version, such that the expected fraction of edges covered is at least $1/OPT_{\min}$. It follows that there must exist a candidate solution that covers this many edges. Hence we have proved

$$OPT_{\max} \geq \frac{1}{OPT_{\min}}.\blacksquare$$

Remarks: The Label Cover problem was implicit in [LY94], and explicitly defined for the first time in [ABSS93]. A weaker version of the "weak duality" described in Lemma 10.5 was implicit in a calculation in [LY94]; the version given here is from [Aro94].

Note that original proof of the hardness of approximating $\mathrm{LABELCOVER}_{\max}$ used [FL92] instead of [Raz94].

10.4.3 NEAREST LATTICE VECTOR PROBLEM

Let m, k be integers and $\{b_1, b_2, \ldots, b_m\}$ be a set of independent vectors in $\mathbf{Z}^k$. The *lattice generated by basis* $\{b_1, b_2, \ldots, b_m\}$, denoted $\mathcal{L}(b_1, b_2, \ldots, b_m)$, is the set of all their integer linear combinations, i.e., $\{\sum_{i=1}^m \alpha_i b_i : \alpha_i \in \mathbf{Z}\}$. Many important optimization problems can be phrased as problems involving lattices (see the survey by Kannan [Kan87]). The following lattice problem is quite basic.

DEFINITION 10.6 Nearest Lattice Vector Problem (NLVP) Given a lattice basis $\{b_1, b_2, \ldots, b_m\}$, and a point $b_0 \in \mathbf{Q}^k$, find the lattice vector nearest (in the Euclidean norm) to b_0.

In conjunction with the inapproximability result for $\mathrm{LABELCOVER}_{\min}$ (Theorem 10.2, part 3) the next result shows that NLVP is in Class III.

THEOREM 10.3 There is polynomial-time reduction τ from $\mathrm{LABELCOVER}_{\min}$ to NLVP that ensures for every number $\rho \geq 1$:

$$\mathrm{LABELCOVER}_{\min}(I) = 1 \Longrightarrow \mathrm{NLVP}(\tau(I)) = K(I),$$
$$\mathrm{LABELCOVER}_{\min}(I) > \rho \Longrightarrow \mathrm{NLVP}(\tau(I)) > \sqrt{\rho}\cdot K(I),$$

where $K(I)$ is a polynomial-time computable function.

This is an example (refer to the comments after Definition 10.2) of a gap-preserving reduction for which we do not know an equivalent L-reduction. However, this reduction is still good enough for proving inapproximability in conjunction with the inapproximability result for LABELCOVER stated in Theorem 10.2 part 3, since that result (quite conveniently, it seems) in one of the cases produces instances of LABELCOVER with optimum cost exactly 1.

Proof of Theorem 10.3. Let (V_1, V_2, E, Π, N) be an instance of LABELCOVER$_{\min}$. Remember, a labelling is a function $\mathcal{P}$ from vertices to sets of labels, where $\mathcal{P}(v)$ denotes the set of labels assigned to the vertex $v \in V_1 \cup V_2$. A labelling that covers every edge is called a *total cover*.

The reduction produces a lattice basis containing one vector for every pair (v, a), where v is a vertex and a is a label. For convenience, we index the basis vectors by such pairs, i.e., the basis is denoted by $\{b_{v\,a} | v \in V_1 \cup V_2, a \in \{1, 2, \ldots, N\}\}$.

Note, with every lattice vector $x = \sum_{v,a} c_{v\,a} b_{v\,a}$ we may associate a labelling $\mathcal{P}^x$, defined as $\mathcal{P}^x(v) = \{a : c_{v\,a} \neq 0\}$. That is, $\mathcal{P}^x$ assigns a label a to vertex v iff the basis vector $b_{v\,a}$ has a nonzero contribution to x. The reduction will construct a fixed point b_0 in such a way that the cost of the labelling $\mathcal{P}^x$ is related to the distance $||x - b_0||_2$; the distance is small iff $\mathcal{P}^x$ forms a total cover.

The basis vectors (and hence, also the lattice vectors) have $|E|(N+1)+|V_1|N$ coordinates: $N+1$ coordinates for each edge $e \in E$ and 1 for each pair (v_1, a_1) for $v_1 \in V_1$ and $a_1 \in \{1, 2, \ldots, N\}$. Call the coordinates corresponding to e as the *e-projection* of the vector, and the coordinate corresponding to pair (v_1, a_1) as the (v_1, a_1)-projection. The (v_1, a_1)-projection of a basis-vector $b_{v\,a}$ is 1 iff $(v, a) = (v_1, a_1)$. Thus only the basis vectors corresponding to vertices in V_1 can have a nonzero (v_1, a_1)-projection. As will be clear later, this corresponds to fact that in the Label Cover problem, only labels assigned to vertices in V_1 contribute to the cost of the cover.

Let K be the integer $|E|(N+1)$. For $j = 1, 2, \ldots, N$, let U_j be a vector with $N+1$ coordinates, in which the j'th entry is 1 and all the other entries are 0. Let $\vec{0}$ and $\vec{1}$ denote, respectively, the all-zero vector and all-one vector with $N+1$ coordinates.

The basis is defined as follows. Let e be any edge and a be a label. For vertex $v_1 \in V_1$,

$$\text{the } e\text{-projection of } b_{v_1 a} = \begin{cases} K \cdot (\vec{1} - U_{\Pi_e(a)}) & \text{if } e \text{ is incident to } v_1 \text{ and } \Pi_e(a) \text{ is defined} \\ \vec{0} & \text{otherwise} \end{cases}$$

For vertex $v_2 \in V_2$, the e-projection of the vector $b_{v_2 a}$ is $K \cdot U_a$ if e is incident to v_2, and $\vec{0}$ otherwise.

The fixed vector b_0 contains, for every edge e, the vector $K \cdot \vec{1}$ in its e-projection. It's (v_1, a_1)-projection is $\vec{0}$ for all pairs (v_1, a_1).

Thus for every edge e, the e-projections of the basis vectors are either $K \cdot U_a$ for some label a, or $K \cdot (\vec{1} - U_a)$, or $\vec{0}$. Suppose labels a_1, a_2 cover edge $e = (u, v)$.

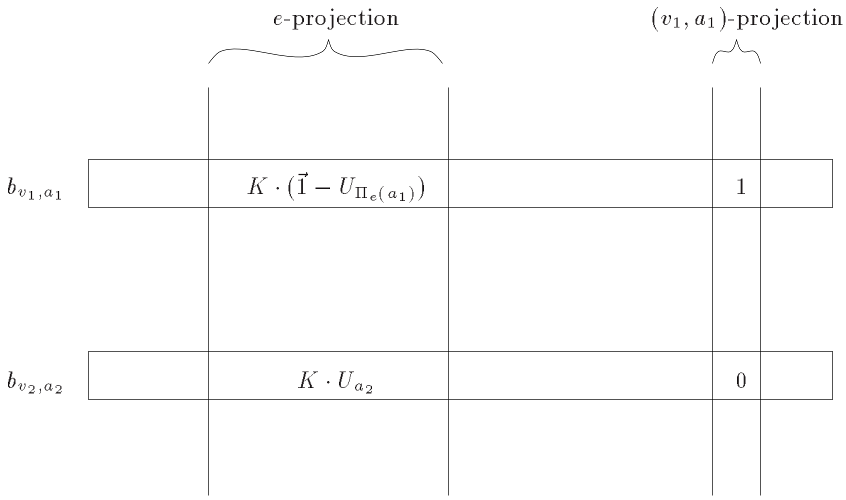

FIGURE 10.2

The e-projections of vectors b_{v_1,a_1} and b_{v_2,a_2} when $e = (v_1, v_2)$. In the figure we assume that $\Pi_e(a_1)$ is defined; otherwise the e-projection of b_{v_1,a_1} is $\vec{0}$.

Then $\Pi_e(a_1) = a_2$. Hence the e-projection of $V_{ua_1} + V_{va_2}$ is

$$K \cdot (\vec{1} - U_{\Pi_e(a_1)}) + K \cdot U_{a_2} \quad = K \cdot \vec{1}.$$

In particular, since the e-projection of b_0 is $K \cdot \vec{1}$, the e-projection of $V_{ua_1} + V_{va_2} - b_0$ is $\vec{0}$.

Now we prove the first property of the reduction. Assume there is a total cover with cost 1, i.e., which uses only one label per vertex. Let $\mathcal{P}(u)$ denote the label it assigns to vertex $u \in V_1 \cup V_2$. Let x be the lattice vector $x = \sum_{v_1 \in V_1} b_{v_1,\mathcal{P}(v_1)} + \sum_{v_2 \in V_2} b_{v_2,\mathcal{P}(v_2)}$. Then the e-projection of $x - b_0$ is $\vec{0}$ for every edge e. Further, exactly $|V_1|$ of the other coordinates in $x - b_0$ are 1. Hence the distance $||x - b_0||_2$ is exactly $\sqrt{|V_1|}$.

Next, we show the second property of the reduction. The following lemma shows that if the e-projection of any integer combination of basis vectors is $K \cdot \vec{1}$, then the combination contains some vectors b_{u,a_1} and b_{v,a_2} such that a_1, a_2 cover e.

LEMMA 10.6 For some integers $\{d_a : a \in \{1, \ldots, N\}\} \cup \{d'_a : a \in \{1, \ldots, N\}\}$, if

$$\sum_a (d_a \cdot U_a + d'_a \cdot (\vec{1} - U_a)) = \vec{1},$$

then $d_a = d'_a$ for every a and there exists some a such that d_a and d'_a are nonzero.

Proof. The vectors $\{U_1, U_2, \ldots, U_N\}$ are linearly independent and do not span $\vec{1}$. ■

Next, we show that if a lattice vector is "near" b_0 then its associated labelling is a total cover.

COROLLARY 10.4 If $||x - b_0||_2 < K$ for a lattice vector x, then $\mathcal{P}^x$ is a total cover.

Proof. Let $x = \sum_{v,a} c_{v\,a} b_{v\,a}$ and let e be any edge, say (u, v). Every coordinate in the e-projection of $x - b_0$ is an integer multiple of K. If $||x - V_0|| < K$, these coordinates must be all 0. Thus if $||x - V_0||_2 < K$, the e-projections of the basis vectors form a linear combination as described in the hypothesis of Lemma 10.6, where $\vec{1}$ in the lemma statement comes from the e-projection of b_0. We conclude that $c_{v\,a_2} = \sum_{a_1:\Pi_e(a_1)=a_2} c_{u\,a_1}$ for every a_2 and there is at least one a_2 such that $c_{v\,a_2} \neq 0$. Further, if $c_{v\,a_2} \neq 0$, then there exists some a_1 such that $c_{u\,a_1} \neq 0$ and $\Pi_e(a_1) = a_2$. Hence $\mathcal{P}^x$ covers e. ■

The next lemma shows that the distance $||x - b_0||_2$ is related to the cost of labelling $\mathcal{P}^x$. It also proves the second property of the reduction claimed in Theorem 10.3.

LEMMA 10.7 For every lattice vector x,

$$||x - b_0||_2 \geq \sqrt{|V_1| \,\mathrm{LABELCOVER}_{\min}(\mathcal{L})}.$$

Proof. The definition of $\mathrm{LABELCOVER}_{\min}$ implies that $\mathrm{LABELCOVER}_{\min}(\mathcal{L}) \leq N$. Since $K = |V_1|(N + 1)$, it follows that $|V_1| \,\mathrm{LABELCOVER}_{\min}(\mathcal{L}) < K$. Corollary 10.4 implies that if $\mathcal{P}^x$ is not a total cover then $||x - b_0||_2 \geq K$. Thus the lemma needs to be proven only when $\mathcal{P}^x$ is a total cover. The definitions of $\mathcal{P}^x$ and of the (v_1, a_1)-projections of basis vectors imply that

$$\begin{aligned} ||x - b_0||_2 &\geq \sqrt{\text{Number of nonzero coordinates in } x - b_0} \\ &= \sqrt{|\{(v_1, a_1) : \ (v_1, a_1)\text{-projection of } x \text{ is } \neq 0\}|} \\ &\geq \sqrt{\text{cost of } \mathcal{P}^x} \\ &\geq \sqrt{|V_1| \,\mathrm{LABELCOVER}_{\min}(\mathcal{L})} \end{aligned}$$

Thus the Lemma has been proved. ■

Remarks: The hardness result for the nearest vector problem is from [ABSS93], which also uses similar constructions to prove the hardness of a variety of problems involving lattices, codes, and linear systems.

PROBLEMS

The following two problems are from [ABSS93].

EXERCISE 10.5

In the NEAREST CODEWORD problem, we are given a matrix M with m rows and n columns ($m < n$) and a vector b_0 with n entries. All entries in M and b_0 are 0 or 1 (interpreted as elements of the field $GF(2)$). The goal is to find a 0/1 vector $x \in \{0,1\}^m$ such that $x^T \cdot M$ differs from b_0 in as few positions as possible. Let NEAREST-CODEWORD(M, b_0) denote the number of positions in which $x^T \cdot M$ and b_0 differ. (*Note:* In the language of coding theory, M is the generator matrix of a binary code and b_0 is a received message. We wish to find out the codeword it is nearest to.)

1. Show that the NEAREST CODEWORD problem is in Class I.

2. Show that the NEAREST CODEWORD problem is in Class III. (*Hint:* Modify our construction for the nearest lattice vector problem.)

EXERCISE 10.6

In the MIN-UNSATISFY problem we are given a system of n linear equations in m variables $x_1, \ldots, x_m$. The the coefficients of the system are integers and $n > m$ (i.e., there are more equations than variables). The goal is to remove as few equations from the system as possible, such that the remaining system is feasible (i.e., has a solution in which each x_i is a real number). For a system S, let MIN-UNSATISFY(S) denote the minimum number of equations that must be removed to make it feasible.

1. Show that the MIN-UNSATISFY problem is in Class I.

2. Show that the MIN-UNSATISFY problem is in Class III.

3. Consider the version of MIN-UNSATISFY problem in which linear system contains strict inequalities instead of equations. Prove the results in parts 1 and 2 for that problem.

10.5 INAPPROXIMABILITY RESULTS FOR PROBLEMS IN CLASS IV

In this section we demonstrate problems for which it is *NP*-hard to achieve an approximation ratio of n^ϵ for some $\epsilon > 0$. The canonical problems in this class are CLIQUE (or R-CLIQUE) and COLORING. We describe the inapproximability results for the canonical problems, and indicate how to prove the inapproximability of the other problems.

The inapproximability result for CLIQUE uses the fact that the clique number has a *self-improvement* property [GJ79, BS92]. Problems other than CLIQUE also display this property; see Section 10.6.1 for example. Also, a classical inapproximability result for COLORING[GJ76] uses a similar property.

10.5.1 CLIQUE

The hardness result for clique number relies upon its interesting *self-improvement* behavior when we take graph products. The next example describes this behavior.

DEFINITION 10.7 For a graph $G = (V, E)$ let $\hat{E}$ denote E with all self-loops added, that is, $\hat{E} = E \cup \{\{u, u\} : u \in V\}$. For graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their *product* $G_1 \times G_2$ is the graph whose vertex set is the set $V_1 \times V_2$, and edge set is

$$\left\{((u_1, v_1), (u_2, v_2)) : (u_1, u_2) \in \hat{E}_1 \text{ and } (v_1, v_2) \in \hat{E}_2\right\}.$$

Example 10.3 Let $\omega(G)$ be the size of the largest clique in a graph. It is easily checked that $\omega(G_1 \times G_2) = \omega(G_1)\omega(G_2)$.

Now suppose a reduction exists from SAT to clique, such that the graph G produced by the reduction has clique number either l, or $(1 - \epsilon)l$, depending on whether or not the SAT formula was satisfiable. In other words, achieving an approximation ratio $(1 - \epsilon)^{-1}$ for clique number is *NP*-hard. Then we claim that achieving any constant approximation ratio is *NP*-hard. Consider G^k, the product of G with itself k times. Then $\omega(G^k)$ is either l^k or $(1-\epsilon)^k l^k$. The gap in clique numbers, $(1 - \epsilon)^{-k}$, can be made arbitrarily large by increasing k enough. This is what we mean by self-improvement.

Note however that G^k has size n^k, so k must remain $O(1)$ if the above construction has to work in polynomial time.

The rapid increase in problem size when using self-improvement may seem hard to avoid. Surprisingly, the following combinatorial object (specifically, as constructed in Theorem 10.4) often allows us to do just that.

DEFINITION 10.8 Let n be an integer. A (k, n, α) *booster* is a collection $\mathcal{S}$ of subsets of $\{1, 2, \ldots, n\}$, each of size k. For every subset $A \subseteq \{1, 2, \ldots, n\}$, the sets in the collection that are subsets of A constitute a fraction between $(\rho-\alpha)^k$ and $(\rho+\alpha)^k$ of all sets in $\mathcal{S}$, where $\rho = \frac{|A|}{n}$. When $\rho < 1.1\alpha$, the quantity $(\rho-\alpha)^k$ should be considered to be 0.

Example 10.4 The set of all subsets of $\{1, 2, \ldots, n\}$ of size k is a booster with $\alpha \approx 0$. This is because for any $A \subseteq \{1, 2, \ldots, n\}$, $|A| = \rho n$, the fraction of sets contained in A is $\binom{\rho n}{k}/\binom{n}{k}$, which is $\approx \rho^k$. The problem with this booster is that its size is $\binom{n}{k} = O(n^k)$, hence k must be $O(1)$ if the booster has to be used in polynomial-time reductions.

The following theorem appears in [AFWZ93]. Its proof uses explicit constructions of expander graphs [GG81].

THEOREM 10.4 [AFWZ93] For any $k = O(\log n)$ and $\alpha > 0$, an (n, k, α) booster of size poly(n) can be constructed in poly(n) time.

Let G be a graph on n vertices. Using a booster, we can define the *booster product* of a graph, which is similar to the product in [BS92]. This is the graph whose vertices are the sets of the booster $\mathcal{S}$, and there is an edge between sets $S_i, S_j \in \mathcal{S}$ iff $S_i \cup S_j$ is a clique in G.

LEMMA 10.8 For any graph G, and any (k, n, α) booster, the clique number of the booster product of G lies between $(\omega(G)/n-\alpha)^k\,|\mathcal{S}|$ and $(\omega(G)/n+\alpha)^k\,|\mathcal{S}|$.

Proof. Let $A \subseteq \{1, 2, \ldots, n\}$ be a clique of size $\omega(G)$ in graph G. Then the number of sets from $\mathcal{S}$ that are subsets of A is between $(\omega(G)/n-\alpha)^k\,|\mathcal{S}|$ and $(\omega(G)/n+\alpha)^k\,|\mathcal{S}|$. Clearly, all such sets form a clique in the booster product.

Conversely, given the largest clique B in the booster product, let A be the union of all sets in the clique. Then A is a clique in G, and hence must have size at most $\omega(G)$. The booster property implies that the size of B is as claimed. ■

THEOREM 10.5 [ALM+92] There is an $\epsilon > 0$ such that approximating CLIQUE within a factor n^ϵ is *NP*-hard, where n is the number of vertices in the input graph.

Proof. Let G be the graph obtained from the reduction in Lemma 10.1, and suppose it has m vertices. The reduction ensures, for some fixed $\beta > 0$, that $\omega(G)$ is either at least $m/3$ or at most $m(1-\beta)/3$, and it is *NP*-hard to decide which case holds.

Now construct a $(m, \log m, \alpha)$ booster, $\mathcal{S}$, using Theorem 10.4, by choosing $\alpha = \beta/9$. Construct the booster product of G. The number of vertices in the booster product is $|\mathcal{S}|$, and Lemma 10.8 says the clique number is either at least $((3-\beta)/9)^{\log m}|\mathcal{S}|$ or at most $((3-2\beta)/9)^{\log m}|\mathcal{S}|$. Hence the gap is now m^γ for some $\gamma > 0$, and further, $|\mathcal{S}| = \text{poly}(m)$, so this gap is $|\mathcal{S}|^\epsilon$ for some $\epsilon > 0$. ■

The proof in Theorem 10.5 is the easiest known proof of the inapproximability of CLIQUE. Next, we describe the slightly more complicated reduction to R-CLIQUE, which also proves Theorem 10.2, part 2.

Proof of Theorem 10.2, part 2. For each $\epsilon > 0$ we give a gap-preserving reduction τ from MAX-3SAT to R-CLIQUE that has parameters $((1, 1+\epsilon), (1, n^{-\delta}))$, where n is the size of the new instance and $\delta > 0$ is some constant depending on ϵ. (Remember that R-CLIQUE(G) of an r-partite graph G is the ratio of $\omega(G)/r$.) In other words, τ ensures, for every 3CNF formula φ,

$$\text{MAX-3SAT}(\varphi) = 1 \Longrightarrow \text{R-CLIQUE}(\tau(\varphi)) = 1$$
$$\text{MAX-3SAT}(\varphi) \le \frac{1}{1+\epsilon} \Longrightarrow \text{R-CLIQUE}(\tau(\varphi)) \le n^{-\delta}.$$

Let $C_1 \vee C_2 \vee \cdots \vee C_m$ be a 3CNF formula in n variables $x_1, x_2, \ldots, x_n$. Construct a $(m, \log m, \alpha)$ booster, $\mathcal{S}$, using Theorem 10.4 and with $\alpha = \epsilon/100$. Let k denote the size of each set in the booster (i.e., $\log m$). Interpret each set $S_i \in \mathcal{S}$ as a set of k clauses. Construct a graph H on $n = |\mathcal{S}|3^k$ vertices as follows. Corresponding to each set $S_i \in \mathcal{S}$ it has 3^k vertices, each labelled by a $k+1$-tuple of the form $(i, t_1, t_2, \ldots, t_k)$, where each $t_i = 1, 2$, or 3. Notice, the first entry in the tuple indicates the set in $\mathcal{S}$ it corresponds to, and the remaining entries pick out a literal (numbered 1 to 3) from each clause appearing in this set. Put edges in the new graph as follows: vertices $(i, t_1, t_2, \ldots, t_k)$ and $(j, s_1, s_2, \ldots, s_k)$ are adjacent iff $i \ne j$ and if the literals picked out by $t_1, t_2, \ldots, t_k, s_1, \ldots, s_k$, do not contain both a variable and its negation.

Note that the graph is $|\mathcal{S}|$-partite; the vertices corresponding to any set in $\mathcal{S}$ are mutually nonadjacent. Also, in any clique of H, a variable and its negation cannot both appear. Thus a clique corresponds to a (partial) assignment that satisfies all the clauses appearing in the clique.

Suppose the 3CNF formula has a satisfying assignment. Then there is a corresponding clique in H of size $|\mathcal{S}|$: pick a true literal from each clause, and pick the corresponding k-tuple from each set in $\mathcal{S}$. Since all k-tuples involve true literals, they never involve both a variable and its negation. So all $|\mathcal{S}|$ of them are adjacent to each other, and we have thus identified a clique of size $|\mathcal{S}|$.

Conversely, assume every assignment satisfies less than $1/(1+\epsilon)$ fraction of the clauses. Let B be a clique in H and let A be the set of clauses appearing in it. Then $|A|/m \le m/(1+\epsilon)$. The property of the booster implies that $|B|$ is at most $(|A|/m+\alpha)^k|\mathcal{S}| \le (1/(1+\epsilon)+\alpha)^k|\mathcal{S}|$. Thus $\omega(H)$ is upperbounded by $(1/(1+\epsilon)+\alpha)^k|S| \le m^{-\gamma}|\mathcal{S}|$ for some small $\gamma > 0$. Hence $\omega(H) < \mathcal{S}/n^{-\delta}$ for some small $\delta > 0$. Thus the correctness of the reduction has been proved. ■

Comments: The first inapproximability result for the clique problem, due to [FGL+91], showed that achieving an approximation ratio of $2^{\log^{1-\delta} n}$ is Quasi-NP-hard. Then [AS92] showed that the same approximation is actually NP-hard. Soon after, [ALM+92] showed that achieving a ratio n^ϵ is NP-hard. The proof given in this section is due to [AFWZ93]; it is a simplification of the original proof in [ALM+92].

PROBLEMS

EXERCISE 10.7 Randomized Construction of Boosters

Let α be any positive fraction and c be any fixed positive integer. Let n be a "large enough" integer, and $k = c \log n$. Suppose we randomly pick $O(n/\alpha^k)$ subsets of size k from $\{1, \ldots, n\}$. Prove that the probability that these subsets form a (k, n, α) booster is at least $1 - o(1)$.

10.5.2 COLORING

Now we prove part 5 of Theorem 10.2, concerning the hardness of approximating the chromatic number of graphs. For ease of exposition, we prove the result for a trivial reformulation of chromatic number, the clique cover number. A *clique cover of size* k in a graph is a partition of the vertices into k sets, each of which is a clique. The *clique cover number* of G, denoted $\overline{\chi}(G)$, is the minimum size of a clique cover of graph G. Note that $\omega(G) \geq n/\overline{\chi}(G)$, where n is the number of vertices in G. The reason is that a partition of $V(G)$ into $\overline{\chi}(G)$ cliques must contain some clique of size $\geq n/\overline{\chi}(G)$

A k-coloring of a graph G is a partition of the vertices into k independent sets. Thus a k-coloring in G can be viewed as as a clique cover of size k in $\overline{G}$, and vice versa, where $\overline{G}$ is the graph containing exactly those edges that are absent in G. Thus $\chi(G) = \overline{\chi}(\overline{G})$, which shows that the clique cover number is a trivial reformulation of the chromatic number.

Now we give a gap-preserving reduction τ from R-CLIQUE to the clique cover problem. The following definition will be useful.

DEFINITION 10.9 For positive integers r and k, an $r \times k$ *graph* is a graph on rk vertices, which can be partitioned into r independent sets, each of size exactly k. We will assume that the vertices are labelled from the set $R \times K$, where $R = \{0, 1, \ldots, r-1\}$, $K = \{0, 1, \ldots, k-1\}$, and vertices labelled by

$\{j\} \times K$ constitute the jth independent set.

Note that given an r-partite graph — in other words, an instance of R-CLIQUE — we can easily convert it into an $r \times k$ graph (for some suitable k) by adding isolated vertices to equalize the sizes of the r-partitions. Thus we can trivially rephrase Theorem 10.2, part 2, so that it gives us a reduction τ' which reduces SAT to R-CLIQUE as follows, for some $\epsilon > 0$:

$$\begin{aligned} \forall I \quad & I \in \text{SAT} \Longrightarrow \text{R-CLIQUE}(\tau'(I)) = 1 \\ & I \notin \text{SAT} \Longrightarrow \text{R-CLIQUE}(\tau'(I)) < \frac{1}{n^{\epsilon}}, \end{aligned} \tag{10.4}$$

where $\tau'(I)$ is an $r \times k$ graph and $n = rk$ is the size of $\tau'(I)$.

In this section we describe a reduction τ that, given an $r \times k$ graph G, produces an $r \times k'$ graph G' such that, for every constant $\epsilon > 0$:

$$\text{R-CLIQUE}(G) = 1 \Longrightarrow \overline{\chi}(\tau(I)) = k' \tag{10.5}$$

$$\text{R-CLIQUE}(G) < 1/n^{\epsilon} \Longrightarrow \overline{\chi}(\tau(I)) > N^{\epsilon/7} \cdot k', \tag{10.6}$$

where $n = rk$ and $N = rk'$ are the numbers of vertices in G and $\tau(G)$ respectively.

Note that the NP-hardness of approximating $\overline{\chi}$ follows, once we compose this reduction with the one in (10.4).

Our reduction τ consists of applying two transformations —defined below— on the graph: first make it 2, 3-*uniquely shiftable*, and then produce the *extension* of the resulting graph.

First, we describe the *extension* of an $r \times k$ graph $G = (R \times K, E)$. For $i = 0, 1, \ldots, k-1$, define the *ith copy* of G as a graph $G_i = (R \times K, E_i)$ containing the same number of vertices and edges as G, but whose edges are *shifted* according to the following rule: $((j,q),(j',q'))$ is an edge in G_i if and only if $((j,(q+i) \bmod k),(j',(q'+i) \bmod k))$ is an edge in G. Since this shifting of edges may be viewed also as a renumbering of the vertices, it follows that graphs G and G_i are isomorphic.

DEFINITION 10.10 Let H be an $r \times k$ graph and $H_0, \ldots, H_{k-1}$ be its copies, as described above. The *extension of* H, denoted $\widetilde{H}$, is an $r \times k$ graph with the same number of vertices as H and whose edges are $\bigcup_{i=0}^{K-1} E(H_i)$.

The next proposition suggests how the reduction could ensure the condition in (10.5).

PROPOSITION 10.1 If H is an $r \times k$ graph, then

$$\text{R-CLIQUE}(H) = 1 \Longrightarrow \overline{\chi}(\widetilde{H}) = k.$$

Proof. Suppose C is a clique in H of size r. Define for $i = 0, 1, \ldots, k-1$, the set of vertices $C_i \stackrel{def}{=} \{(j,q) : (j,(q+i) \bmod k) \in C\}$. Each C_i is a clique in $\widetilde{H}$ of size r and furthermore, $C_0, \ldots, C_{K-1}$ are disjoint. It follows that $\overline{\chi}(\widetilde{H}) \le k$. In addition, $\overline{\chi}(\widetilde{H}) \ge k$, since $\widetilde{H}$ contains an independent set of size k and so a clique partition needs to have k cliques just to cover this independent set. Thus $\overline{\chi}(\widetilde{H}) = k$. ■

The next proposition suggests how the reduction could ensure the condition in (10.6).

PROPOSITION 10.2 Let H be an $r \times k$ graph such that $\omega(\widetilde{H}) = \omega(H)$. Then for every $\rho > 1$,

$$\text{R-CLIQUE}(H) < \frac{1}{\rho} \Longrightarrow \overline{\chi}(\widetilde{H}) > k\rho.$$

Proof. Recall that $kr/\omega(\widetilde{H})$ is a lower bound on $\overline{\chi}(\widetilde{H})$, since $\widetilde{H}$ has kr vertices. Hence if $\omega(H) = \omega(\widetilde{H})$ then $\omega(H) < r/\rho \Rightarrow \omega(\overline{\chi}(\widetilde{H}) > k \cdot \rho$. ■

Unfortunately Proposition 10.2 is not enough to ensure the condition in (10.6), since not every $r \times k$ graph G satisfies the property $\omega(G) = \omega(\widetilde{G})$. However, even if G does not satisfy this property, we show how to convert it in polynomial time into an $r \times k'$ graph G' that does. Here k' is roughly $(rK)^5$. Further, this new graph G' has the same clique number as G. Thus it follows from Propositions 10.1 and 10.2 that

$$\text{R-CLIQUE}(G) = 1 \quad \Longrightarrow \quad \text{R-CLIQUE}(G') = 1 \quad \Longrightarrow \quad \overline{\chi}(\widetilde{G'}) = k'$$

and

$$\text{R-CLIQUE}(G) < \frac{1}{\rho} \quad \Longrightarrow \quad \text{R-CLIQUE}(G') < \frac{1}{\rho} \quad \Longrightarrow \quad \overline{\chi}(\widetilde{G'}) > k'\rho.$$

As we already indicated, our reduction, when given a graph G, first constructs G' and then outputs its extension $\widetilde{G'}$. Thus it satisfies both properties desired for it.

To finish up, we describe the transformation from G to G'. For a positive integer l, an $r \times k$ graph G is l *uniquely-shiftable* if it satisfies the following: For every clique C in $\widetilde{G}$ of size l, there is a a unique *shift* $i \in \{0, 1, \ldots, k-1\}$ such that C is a clique in some copy G_i of G, and C is an independent set in every other copy $G_{i'}$ for $i' \neq i$. Graph G is said to be $2, 3$ *uniquely-shiftable* if it is both 2 uniquely-shiftable and 3 uniquely-shiftable.

Note that if G is $2, 3$ uniquely-shiftable then its extension $\widetilde{G}$ has certain special properties. For instance, 2 uniquely-shiftability implies that every edge e of $\widetilde{G}$ is obtained from exactly one edge of G, by using an appropriate shift i. (That is, for every two distinct edges e', e'' in G, the sth shift of e' does not coincide with the tth shift of e'' for every pair of shifts s, t.) Thus if G is 2 uniquely-shiftable, we may talk of this shift i as *the* shift of the edge e. If in addition G is 3 uniquely-shiftable, then in every triangle in $\widetilde{G}$, all three edges of the triangle have the same shift.

LEMMA 10.9 If an $r \times k$ graph G is $2, 3$ uniquely-shiftable, then $\omega(G) = \omega(\widetilde{G})$.

Proof. Clearly, $\omega(G) \leq \omega(\widetilde{G})$, since a clique in G is a clique in $\widetilde{G}$. Now let C be any clique in $\widetilde{G}$ of size $\omega(\widetilde{G})$. Since G is 2 uniquely-shiftable, we know that every edge in C has a unique shift associated with it. We show that this shift is the same for all edges in C, whence it follows that C is just the shifted version of some clique in in G and the proof is complete. So assume the shifts are not all the same. Then there exist three vertices v_1, v_2, v_3 in C such that the edges (v_1, v_2) and (v_2, v_3) have different shifts. But this contradicts the assumption that G is 3 uniquely-shiftable, since $\{v_1, v_2, v_3\}$ is a triangle in $\widetilde{G}$. Hence it follows that every edge in C has the same shift. ■

It only remains to show that every $r \times k$ graphs can be made 2, 3 uniquely-shiftable without much increase in size. We will need the mapping defined in the following lemma.

LEMMA 10.10 For every positive integer m, we can in poly(m) time construct an injective map $T : \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m'-1\}$, where $m' > 6m^5$ and for all distinct ordered triples (u_1, u_2, u_3) and (v_1, v_2, v_3) (i.e., $u_1 \leq u_2 \leq u_3$, $v_1 \leq v_2 \leq v_3$, and $(u_1, u_2, u_3) \neq (v_1, v_2, v_3)$),

$$T(v_1) + T(v_2) + T(v_3) \neq T(u_1) + T(u_2) + T(u_3) \bmod m'. \tag{10.7}$$

Proof. Construct T by greedily assigning values, in order, to $T(0), T(1), \ldots,$. Assuming all values up to $T(i)$ have been assigned, we only have to ensure that $T(i+1)$ does not satisfy any of $6i^5$ equality constraints imposed by the values of $T(0), \ldots, T(i)$. Since $6i^5 < m'$, we can choose a value for $T(i+1)$ easily, and continue on. ■

The next lemma shows how to make every $r \times k$ graph 2, 3 uniquely-shiftable.

LEMMA 10.11 Every $r \times k$ graph G can be changed in polynomial time to an $r \times k'$ graph G' that is 2, 3 uniquely-shiftable and satisfies $\omega(G) = \omega(G')$. Further, this is possible for $k' \leq 6(rk)^5$.

Proof. Transform G as follows. Choose a map T according to Lemma 10.10 by substituting $m = rk = |R \times K|$. Denote the integer m' obtained from the lemma as k', and let $W = \{0, 1, \ldots, k'-1\}$. We will think of T as a function from $R \times K$, the vertices of G, to W. The vertex set of the new graph G' is $R \times W$, and its edges are defined according to the following rule: For $j_1, j_2 \in R$ and $w_1, w_2 \in W$, the vertices (j_1, w_1) and (j_2, w_2) are adjacent iff there is an edge $((j_1, q_1), (j_2, q_2))$ in G such that $T(j_1, q_1) = w$ and $T(j_2, q_2) = w_2$. In other words, G' is obtained by mapping every vertex (j, q) in G to the vertex $(j, T(j, q))$ in G', and putting in edges accordingly. Since T is injective, it follows that cliques in G and G' correspond to each other and so $\omega(G) = \omega(G')$.

To show that G' is 2-uniquely shiftable, we need to show that for every edge in $\widetilde{G'}$, the extension of G', there is a unique shift $i \leq k'-1$ such that the edge belongs to the ith copy G'_i of G. Assume this is not the case. Then there are distinct edges e_1 and e_2 in G and distinct shifts $s, t \leq k'-1$ such that the sth shift of e_1 coincides with the tth shift of e_2. Hence if $e_1 = \{(j_1, q_1), (j_2, q_2)\}$ and $e_2 = \{(j_1, q'_1), (j_2, q'_2)\}$ we have

$$(T(j_1, q_1) + s) = (T(j_1, q'_1) + t)$$
$$(T(j_2, q_2) + s) = (T(j_2, q'_2) + t)$$

This implies that $T(j_1, q_1) - T(j_1, q'_1) = T(j_2, q_2) - T(j_2, q'_2)$ and hence that $T(j_1, q_1) + T(j_2, q'_2) + T(j_2, q'_2) = T(j_2, q_2) + T(j_1, q'_1) + T(j_2, q'_2)$. The property of T from Equation 10.7 implies that $(j_i, q_i) = (j_i, q'_i)$ for $i = 1, 2$, and thus e_1 and e_2 are the same edge. This is a contradiction. It follows that G' is 2-uniquely shiftable. Similarly, we can prove that G' is also 3 uniquely-shiftable. ■

Comments. The above reduction is due to Khanna et al [KLS93] and is a simplification of the original reduction in [LY94]. Note that it reduces R-CLIQUE to COLORING and strongly uses the structure of the R-CLIQUE problems. Can a similar reduction be done from CLIQUE to COLORING? This is still open, but partial progress was made in [LY94], which contains a reduction from CLIQUE to COLORING that proves the hardness of approximating COLORING within any constant factor.

, The inapproximability of the chromatic number implies inapproximability of many other problems, via gap-preserving reductions: CLIQUE PARTITION, CLIQUE COVER [Sim90], BICLIQUE COVER [Sim90] and FRACTIONAL CHROMATIC NUMBER [Lov75] (see Table ref10:table:results).

PROBLEMS

EXERCISE 10.8

The *fractional chromatic number* of a graph $G = (V, E)$, denoted $\chi_f(G)$, is the optimum value of the following linear program, which contains a variable x_S for every maximal independent set S in G.

$$\begin{aligned} \text{Minimize } \textstyle\sum_S x_S \ &\text{ subject to} \\ x_S \geq 0 \quad &\forall S \\ \textstyle\sum_{\{S: v \in S\}} x_S \geq 1 \quad &\forall v \in V. \end{aligned}$$

1. Prove that $\chi_f(G) \leq \chi(G) \leq \chi_f(G) \cdot 2(1+\ln|G|)$. (*Hint:* Use the randomized rounding technique of Chapter 11.)

2. Conclude that there is an $\epsilon > 0$ such that approximating $\chi_f(G)$ within a factor n^ϵ is *NP*-hard.

10.6 INAPPROXIMABILITY RESULTS AT A GLANCE

As already mentioned, optimization problems fall into four classes, based on the approximation ratio that is hard to achieve for them. In earlier sections we described inapproximability results for some of the problems in each class. Table 10.2 lists references where other inapproximability results can be found.

All results in the papers mentioned in Table 10.2 are provable using our canonical problems, except one: the result for the Shortest Lattice Vector problem in ℓ_∞ norm, a problem related to the nearest lattice vector problem described in Section 10.4.3. This problem lies in Class III [ABSS93], but its inapproximability result uses special geometric facts about the construction in [FL92].

We note that inapproximability results for most problems in Class III were originally proved (see Table 10.2) using reductions from 2-Prover 1-Round proof systems for *NP*. (Such proof systems first appear in [FL92], and predate the PCP Theorem.) In the (unified) framework we have presented here, all those results can be derived using the LABELCOVER$_{\max}$ problem. The reason is that the LABELCOVER$_{\max}$ problem exactly captures the combinatorial structure of 2-Prover 1-Round proof systems [ABSS93].

As mentioned in the introduction, we have not attempted to optimize the results of this chapter. Such optimization is the subject of many recent papers, however. For example, our hardness result for MAX-3SAT asserts the existence of an $\epsilon > 0$ such that achieving an approximation ratio $1+\epsilon$ is *NP*-hard. Optimizing the value of ϵ seems to involve delving into the proof of the PCP Theorem. The best result (in [BGS95], following earlier improvements in [BGLR93, FK94, BS94]) says that $\epsilon \geq 1/38$. Optimizing the inapproximability result for clique, on the other hand, has involved looking at the *free bit* parameter of the PCP verifier [FK94, BS94]. A recent result [BGS95] shows that the free bit parameter is intimately connected to inapproximability results for clique. In our terminology, the [BGS95] result can be restated thus: Every reduction to CLIQUE can be modified to produce instances of R-CLIQUE(a seemingly more restrictive problem). That paper also shows that approximating clique within a ratio $n^{1/3-o(1)}$ is hard. This has since been improved to $n^{0.5-\epsilon}$ [Has95]. Similarly, we know that achieving approximation ratio $n^{1/3-\epsilon}$ for Chromatic number is hard [FK95] (see also [Für95]).

Despite such encouraging progress, the provably hard approximation ratio for most problems is much worse than the best ratios that we can achieve for these problems in polynomial time. The best ratio achievable for CLIQUEis $O(n/\log^2 n)$ [BH92], for Chromatic number is $O(n(\log\log n)^2/\log^3 n)$ [Hal93], and for MAX-3SAT is ≈ 1.33 [Yan92, GW94]. Can this gap between inapproximability results and algorithmic results be bridged? A recent result in [Aro95] suggests that the gap may well be inherent: at least the current techniques cannot prove the *NP*-hardness of approximating clique within a factor better than $\sqrt{n}$.

Comments	Problem	Hard Ratio	Assumption
Class I	MAX-SAT ([PY91])	1.027	$NP \neq P$
	MAX-3SAT ([PY91])	1.027	$NP \neq P$
Referenced papers	MAX-CUT ([PY91])	1.012	$NP \neq P$
in second column	VERTEX COVER ([PY91])	1.04	$NP \neq P$
show the given	SHORTEST SUPERSTRING ([BJL+91])		$NP \neq P$
problem is	3D-MATCHING ([Kan92])	$1+\epsilon$	$NP \neq P$
Max-SNP-hard.	MAX-2SAT ([PY91])	1.010	$NP \neq P$
	METRIC TSP ([PY93])	$1+\epsilon$	$NP \neq P$
Hardness first proved	MULTIWAY CUTS ([DJP+92])	$1+\epsilon$	$NP \neq P$
in [ALM+92]; improve-	STEINER TREE ([BP89])	$1+\epsilon$	$NP \neq P$
-ments in factors	MAX-3-COLORABLE		
due to [BS94, BGS95]	-SUBGRAPH ([PY91])	$1+\epsilon$	$NP \neq P$
	INTEGER-MULTICOMMODITY		
	-FLOW ([GVY93])	$1+\epsilon$	$NP \neq P$
Class II	SETCOVER	$(1-\delta)\ln N$	$NP \not\subset \tilde{P}$
	HITTING SET	$(1-\delta)\ln N$	$NP \not\subset \tilde{P}$
All results are	DOMINATING SET	$(1-\delta)\ln N$	$NP \not\subset \tilde{P}$
from [LY94]; Improve-	HYPERGRAPH TRANSVERSAL	$(1-\delta)\ln N$	$NP \not\subset \tilde{P}$
-ments are in	MINIMUM EXACT COVER	$(1-\delta)\ln N$	$NP \not\subset \tilde{P}$
[BGLR93, Fei95]			
Class III	LABELCOVER ([ABSS93])	ρ	$NP \not\subset \tilde{P}$
	NEAREST LATTICE		
Referenced papers	VECTOR ([ABSS93])	ρ	$NP \not\subset \tilde{P}$
give reductions	NEAREST CODEWORD ([ABSS93])	ρ	$NP \not\subset \tilde{P}$
from 2-Prover	MIN-UNSATISFY ([ABSS93, AK93])	ρ	$NP \not\subset \tilde{P}$
proof systems.	LEARNING HALFSPACES		
These systems are	WITH ERROR ([ABSS93])	ρ	$NP \not\subset \tilde{P}$
equivalent to	QUADRATIC PROGRAMMING		
$\text{LABELCOVER}_{\max}$.	([FL92, BR93])	ρ	$NP \not\subset \tilde{P}$
	MAX-π-SUBGRAPH ([LY93])	ρ	$NP \not\subset \tilde{P}$
	SHORTEST LATTICE		
	VECTOR (ℓ_∞ norm) ([ABSS93])	ρ	$NP \not\subset \tilde{P}$
	LONGEST PATH ([KMR93]	ρ	$NP \not\subset \tilde{P}$
Class IV	CLIQUE ([FGL+91, AS92, ALM+92])	n^ϵ	$NP \neq P$
	RCLIQUE ([LY94])	n^ϵ	$NP \neq P$
Assuming stronger	INDEPENDENT SET		
complexity conjectures,	([FGL+91, AS92, ALM+92])	n^ϵ	$NP \neq P$
following factors	CHROMATIC NUMBER ([LY94])	n^ϵ	$NP \neq P$
are hard to achieve:	CLIQUE COVER ([Sim90],[LY94])	n^ϵ	$NP \neq P$
$n^{0.5-\epsilon}$ for CLIQUE	BICLIQUE COVER ([Sim90],[LY94])	n^ϵ	$NP \neq P$
[Has95],			
$n^{1/3-\epsilon}$ for	FRACTIONAL CHROMATIC		
CHROMATIC NUM-	NUMBER ([Lov75],[LY94])	n^ϵ	$NP \neq P$
BER [FK95]			
	MAX-PLANAR-SUBGRAPH ([LY93])	n^ϵ	$NP \neq P$
	MAX-SET-PACKING ([Zuc93]))	n^ϵ	$NP \neq P$
	MAX-SATISFY ([ABSS93, AK93]	n^ϵ	$NP \neq P$

Table 10.2: Known inapproximability results. Here ϵ denotes some fixed positive constant (that might depend upon the problem in question), δ is any positive constant, ρ denotes $2^{\log^{1-\delta} n}$ for any $\delta > 0$, where n is the input size. $NP \not\subset \tilde{P}$ means NP does not have quasipolynomial-time (i.e., time $n^{\text{poly}(\log n)}$ time) deterministic algorithms. For the results in Class I, we estimate ϵ to be of the order of 10^{-2}; for results in Class IV, ϵ is at least 0.1.

10.6.0.0.1 Older inapproximability results. Various inapproximability results were obtained in the 1970s, including those for the TRAVELING SALESPERSON PROBLEM without the triangle inequality [SG76] and for certain maximum-subgraph problems [Yan79]. Many other results of this type have also been discovered (see eg [CK94]). We did not describe these results, since the techniques involved are closer to those occurring in [GJ79], and seem not to generalize to the problems considered here.

10.6.1 HOW TO PROVE OTHER HARDNESS RESULTS: A CASE STUDY

The first step in proving an inapproximability result for a given problem is to check whether it is already known to be inapproximable. For instance, the problem might be listed in the compendium [CK94].

As a second step, one should try to prove that the problem is in Class I, specifically, to prove that it is MAX-SNP-hard. Note that Class I is the richest of the four classes in our classification. Usually one can find a problem in this class, such as MAX-3SAT or MAX-CUT, that reduces in a gap-preserving fashion to the problem at hand. (When stuck in this process, leafing through the many reductions in [PY91] is known to help.)

What if one is unable to show in a reasonable amount of time that the problem is in Class I? According to past experience, proving that the problem is in one of the other classes will then be nearly impossible. (Indeed, we know of only one problem — the Shortest Lattice Vector Problem under the ℓ_∞ norm — which is not known to be Max-SNP-hard, but is known to be in a higher class – namely, Class III.)

But if the problem does turn out to be in Class I, then it is worthwhile trying to place it in a higher class as well. For instance, one might try to think which of the other canonical problems it resembles. Is there a way to reduce SETCOVER to it? Or CLIQUE? Is the problem "self-improvable?"

There is no general recipe to the search we have outlined above. Therefore we illustrate it with a case-study: MAX-SATISFY. This is the problem of finding, in a system of m equations in n variables with rational coefficients, the largest feasible subsystem. (Note that the corresponding decision problem, in which we have to decide whether or not the entire system is feasible, is solvable in polynomial time using Gaussian elimination.) The objective function is the fraction of satisfied equations.

A simple greedy algorithm is known to approximate MAX-SATISFY within a factor $n+1$, and no substantially better algorithm is known. We will describe a proof from [ABSS93, AK93] that the problem is in Class IV, and hence NP-hard to approximate within a factor n^δ for some $\delta > 0$. But to illustrate our methodology, we will first show that the problem is in Classes I and III.

Let us first attempt to show Max-SNP-hardness. We use the fact that the problem MAX-2SAT is Max-SNP-hard, from which it follows that there exist $d, \epsilon > 0$ such that given an 2CNF φ it is NP-hard to distinguish the cases when MAX-2SAT$(\varphi) \geq d$ and when MAX-2SAT$(\varphi) < d(1-\epsilon)$. Given a 2CNF formula containing m clauses, construct a system of at most $3m$ equations as follows. First, assume w.l.o.g. that each clause contains exactly 2 literals (which could be the same literal). Introduce a rational variable x_i for each boolean variable z_i. For a clause $z_i \vee z_j$ write the following system of equations.

$$x_i + x_j = 1, \quad x_i = 1, \quad x_j = 1.$$

(If the clause involves $\neg z_i$ then use $1 - x_i$ instead of x_i). First, we claim that w.l.o.g. we may assume that the optimal assignment to the x_i's — that is, the one that satisfies the maximum fraction of equations — assigns 0 or 1 to all the variables. For, given an assignment that is not all 0/1, we can construct the following assignment: if $x_i \geq 1/2$, change x_i to 1 and otherwise change x_i to 0. This does not decrease the number of satisfied equations, as is clear by looking at each group of 3 equations representing a clause.

So assume that the optimal assignment is 0/1. Thus in each group of 3 equations representing a clause, either 2 equations are satisfied or 0 equations are. View this assignment as a boolean assignment in the obvious way. The boolean assignment satisfies a clause iff exactly 2 equations corresponding to it were satisfied. Thus the number of satisfied clauses is exactly 1/2 the number of satisfied equations.

Thus we have shown that given a MAX-SATISFY instance I, it is NP-hard to distinguish whether the fraction of satisfiable equations is either $\geq c$ or $< c(1-\epsilon)$, where $c = 2d/3$.

10.6.1.0.2 Self-improvement. Now we show a self-improvement property for MAX-SATISFY, which helps us show that it is in Class III. Suppose we have (as above) N equations, in which the answer to MAX-SATISFY is either c or $c \cdot (1-\epsilon)$ for some $c, \epsilon > 0$. Let the equations be written as $p_1 = 0, p_2 = 0, \ldots, p_N = 0$. Let k, T be integers (to be specified later). Write down a new set of equations containing, for every k-tuple $(p_{i_1}, p_{i_2}, \ldots, p_{i_k})$ of old equations, a set of T equations $\sum_{j=1}^{k} p_{i_j} y^j = 0$, where $y = 1, 2, \ldots, T$. Note that the number of equations in this new system is $\binom{N}{k} \cdot T$, whereas the number of variables is unchanged.

For each assignment to the variables and each k-tuple of the old equations, let us say the assignment *clears* the tuple if it satisfies all k equations in it. Note that if an assignment clears a tuple, then in the new system of equations it satisfies all T new equations corresponding to this tuple. Now we claim that if it does not clear the tuple, then it satisfies at most k of those T equations. The reason for the claim is that if $(p_{i_1}, p_{i_2}, \ldots, p_{i_k})$ is a k-tuple that that is not cleared by the assignment, then the following vector is not zero: $(v_{i_1}, v_{i_2}, \ldots, v_{i_k})$, where v_{i_j} denotes the value of the left-hand-side of equation p_{i_j} under this assignment. Thus the univariate polynomial $\sum_{j=1}^{k} v_{i_j} y^j$ is not identically zero, and therefore has at most k roots.

If some assignment to the variables satisfies a fraction c of the equations in the original system, then it clears $\binom{cN}{k}$ k-tuples, so the number of satisfied equations in the new system is at least $\binom{cN}{k} \cdot T$.

Conversely, suppose every assignment satisfies less than $cN \cdot (1-\epsilon)$ equations in the old system. Then each assignment clears at most $\binom{cN(1-\epsilon)}{k}$ tuples, each of which contributes T satisfied equations in the new system. The assignment does not clear the other $\binom{N}{k} - \binom{cN(1-\epsilon)}{k}$ tuples, so each of those contributes only k satisfied equations each. Thus no assignment can satisfy more than $\binom{N}{k} \cdot k + \binom{cN(1-\epsilon)}{k} \cdot (T-k)$ equations in the new system.

By choosing $T = N^{k+1}$, we see that the gap between the optima in the two cases is approximately $(1-\epsilon)^k$. Thus we have described a simple construction that allows us to "boost" the gap between the optimum value in the two cases. This shows that MAX-SATISFY is self-improvable.

Using $k = \text{poly}(\log N)$ in the above construction, we get a gap of $2^{\log^{1-\gamma} N'}$, where $N' = 2^{\text{poly}(\log n)}$ is the number of equations in the new system. Thus we have proved that achieving an approximation ratio $2^{\log^{1-\gamma} N'}$ for MAX-SATISFY is Quasi-NP-hard, in other words, that MAX-SATISFY is in Class III.

10.6.1.0.3 MAX-SATISFY is in Class IV. Finally, it should be clear that instead of using the trivial booster, namely, the set of all subsets of size k, we can use the booster of Theorem 10.4 in the above self-improvement construction. Write down T equations for every subset of k equations that form a set in the booster. Use $k = \log N$, and $\alpha < \epsilon c/100$. Thus the NP-hardness of N^{δ}-approximation follows for some $\delta > 0$.

10.7 PROBABILISTICALLY CHECKABLE PROOFS AND INAPPROXIMABILITY

The introduction mentioned the close connection between inapproximability and new probabilistic definitions of NP. This section fleshes out the connection.

10.7.1 THE PCP THEOREM

Let a *verifier* be a polynomial-time probabilistic Turing Machine containing an input tape, a work tape, a tape that contains a random string, and a tape called the *proof string* and denoted as Π. The proof string should be thought of as an array of bits; out of which the verifier will examine a few. The verifier works as follows. First, it reads the input and the random string, and writes down on its work tape some addresses of locations in the proof string Π. Next, it examines the bits in those locations in Π. The process of reading a bit from Π is called a *query*. Finally, the verifier decides to accept or reject, based on what the input, the random string, and the queried bits of Π were.

DEFINITION 10.11 A verifier is $(r(n), q(n))$-*restricted* if on each input of size n it uses at most $O(r(n))$ random bits for its computation, and queries at most $O(q(n))$ bits of the proof.

In other words, an $(r(n), q(n))$-restricted verifier has two associated integers c, k. The random string has length $c\,r(n)$. The verifier operates as follows on an input of size n. It reads the random string R, computes a sequence of $k\,q(n)$ locations $i_1(R), i_2(R), \ldots, i_{k\,q(n)}(R)$, and queries those locations in Π. Depending on what these bits were, it accepts or rejects.

Define $M^{\Pi}(x, R)$ to be 1 if M accepts input x, with access to the proof Π, using a string of random bits R, and 0 otherwise.

DEFINITION 10.12 A verifier M *probabilistically checks membership proofs* for language L if

- For every input x in L, there is a proof Π_x that causes M to accept for every random string (i.e., with probability 1).
- For any input x not in L, every proof Π is rejected with probability at least $1/2$. I.e.,

$$\Pr_R[M^{\Pi}(x, R) = 1] < \frac{1}{2}.$$

Note: The choice of probability $1/2$ in the second part is arbitrary. By repeating the verifier's program $O(1)$ times, (and rejecting if the verifier rejects even once) the probability of rejection $1/2$ in the second part can be reduced to any arbitrary positive constant. Thus we could have used any constant less than 1 instead of $1/2$.

DEFINITION 10.13 A language L is in $PCP(r(n), q(n))$ if there is an $(r(n), q(n))$-restricted verifier M that probabilistically checks membership proofs for L.

Note that $NP = PCP(0, \text{poly}(n))$, since $PCP(0, \text{poly}(n))$ is the set of languages for which membership proofs can be checked in deterministic polynomial-time. This set is exactly NP. The next theorem, called the PCP Theorem, gives an alternative characterization of NP in terms of PCP.

THEOREM 10.6 PCP Theorem [AS92, ALM+92] $NP = PCP(\log n, 1)$.

The proof of the PCP Theorem is complicated. Proving that $PCP(\log n, 1) \subseteq NP$ is easier, however; see Lemma 10.12 for a hint on how to proceed.

10.7.2 CONNECTION TO INAPPROXIMABILITY OF MAX-3SAT

The PCP Theorem, specifically, its nontrivial half $NP \subseteq PCP(\log n, 1)$, is both sufficient and necessary to prove Theorem 10.1 about the inapproximability of MAX-3SAT. That is to say, $NP \subseteq PCP(\log n, 1)$ iff there is a reduction τ from SAT to MAX-3SAT that ensures the following for some fixed $\epsilon > 0$:

$$\begin{aligned} I \in \text{SAT} &\Longrightarrow \text{MAX-3SAT}(\tau(I)) = 1, \\ I \notin \text{SAT} &\Longrightarrow \text{MAX-3SAT}(\tau(I)) < \frac{1}{1+\epsilon}. \end{aligned} \tag{10.8}$$

Let's first check that the "if" part holds: if the above reduction exists then $NP \subseteq PCP(\log n, 1)$. Given any NP language L and an input, the verifier reduces it to SAT, and then to MAX-3SAT using the above reduction. It expects the membership proof to contain a satisfying assignment to the MAX-3SAT instance. To check this membership proof probabilistically, it picks a clause uniformly at random, reads the values of the variables in it (notice, this requires reading only 3 bits), and accepts iff these values satisfy the clause. Clearly, if the original input is in L, there is a proof which the verifier accepts with probability 1. Otherwise every proof is rejected with probability at least $1 - 1/(1+\epsilon)$. Of course, repeating the verification $O(1/\epsilon)$ times makes the rejection probability $\geq 1/2$.

Now we turn to the "only if" part. Suppose $NP \subseteq PCP(\log n, 1)$. As a first step in proving the inapproximability of MAX-3SAT, we prove the inapproximability of the following problem.

DEFINITION 10.14 MAX-k-FUNCTION-SAT Let k be a fixed integer.
Input: Truth tables for m boolean functions, each a function of a set of k variables out of $y_1, y_2, \ldots, y_n$.

$$\{f_i(y_{i_1}, y_{i_2}, \ldots, y_{i_k}) : 1 \leq i \leq m\}.$$

Output: An assignment to $x_1, x_2, \ldots, x_n$ that maximizes the number of functions that evaluate to 1. The objective function is the fraction of functions that evaluate to 1.

The next lemma is based upon the observation that MAX-k-FUNCTION-SAT is exactly the following problem: given the program of a $(\log n, 1)$-restricted verifier that examines k bits in the proof, determine the maximum (over all possible proof strings) probability with which it accepts any proof string. Thus the statement $NP \subseteq PCP(\log n, 1)$ implies the inapproximability of MAX-k-FUNCTION-SAT for some constant k.

LEMMA 10.12 If $NP \subseteq PCP(\log n, 1)$, then there is an integer k such that there is a reduction τ' from SAT to MAX-k-FUNCTION-SAT that ensures

$$\begin{aligned} I \in \text{SAT} &\Longrightarrow \text{MAX-}k\text{-FUNCTION-SAT}\,(\tau'(I)) = 1, \\ I \notin \text{SAT} &\Longrightarrow \text{MAX-}k\text{-FUNCTION-SAT}\,(\tau'(I)) < \frac{1}{2}. \end{aligned} \tag{10.9}$$

Proof. Since SAT $\in NP$, it has a $(\log n, 1)$-restricted verifier. Suppose the verifier uses $c \log n$ random bits and examines k bits in the proof. Note that it has at most $2^{c \log n} = n^c$ different possible runs, and in each run it reads only k bits in the proof-string. Hence assume w.l.o.g. that the number of bits in any provided proof-string is at most kn^c. For concreteness, assume this number is N.

For boolean-valued variables $y_1, y_2, \ldots, y_N$, the set of possible assignments to $y_1, y_2, \ldots, y_N$ is in one-to-one correspondence with the set of possible proof-strings. Assume w.l.o.g. that the proof-string is an assignment to the variables $y_1, y_2, \ldots, y_N$.

Fixing the verifier's random string to $R \in \{0,1\}^{c \log n}$, fixes the sequence of locations that it will examine in the proof. Let $i_1(R), i_2(R), \ldots, i_k(R)$ be this sequence. The verifier's decision depends only on the assignments to $y_{i_1(R)}, y_{i_2(R)} \ldots, y_{i_k(R)}$. Define a boolean function on k bits, f_R, as $f_R(b_1, \ldots, b_k) =$ true iff the verifier accepts when the assignment to the sequence of variables $y_{i_1(R)}, \ldots, y_{i_k(R)}$ is $b_1, b_2, \ldots, b_k$. Since the verifier runs in polynomial time, we can compute the truth table of f_R in polynomial time by going through all possible 2^k values of $b_1, b_2, \ldots, b_k$, and computing the verifier's decision on each sequence.

The reduction consists in outputting the set of n^c functions $\left\{f_R : R \in \{0,1\}^{c \log n}\right\}$ defined above. By definition of $PCP(\log n, 1)$, when the input is in the language, there is an assignment to the $y_1, y_2, \ldots, y_N$ that makes all functions in this set evaluate to true. Otherwise no assignment makes more than $1/2$ of them evaluate to true. ■

Now we prove the "only if" part of our earlier assertion.

THEOREM 10.7 If $NP \subseteq PCP(\log n, 1)$ then the reduction described in (10.8) exists.

Proof. Given a SAT instance, reduce it to MAX-k-FUNCTION-SAT using Lemma 10.12. Let $y_1, y_2, \ldots, y_N$ be the set of boolean variables and $\{f_i : 1 \le i \le n^c\}$ the collection of functions in the instance of MAX-k-FUNCTION-SAT. We indicate how to rewrite them using a 3CNF formula.

Consider a function f_i from this collection. Let f_i be a function of variables $y_{i_1}, y_{i_2}, \ldots, y_{i_k}$. Then f_i can be expressed as a conjunction of at most 2^k clauses in these variables, each of size at most k. Let $C_{i,1}, C_{i,2}, \ldots, C_{i,2^k}$ denote these clauses. (From now on we use the terms k-clause and 3-clause to talk about clauses of size k and 3 respectively.)

Then the k-CNF formula

$$\bigwedge_{i=1}^{n^c} \bigwedge_{j=1}^{2^k} C_{i,j} \tag{10.10}$$

is satisfiable iff $x \in L$. Also, if $x \notin L$, then every assignment fails to satisfy half the f_i's, each of which yields an unsatisfied k-clause. So if $x \notin L$ the fraction of unsatisfied clauses is at least $\frac{1}{2} \cdot \frac{1}{2^k}$, which is some fixed constant.

To obtain a 3CNF formula rewrite every k-clause as a conjunction of clauses of size 3, as follows. For a k-clause $l_1 \vee l_2 \vee \cdots \vee l_k$ (the l_i's are literals), write the formula

$$(l_1 \vee l_2 \vee z_1) \wedge (l_{k-1} \vee l_k \vee \neg z_{k-2}) \wedge \bigwedge_{t=1}^{k-4} (\neg z_t \vee l_{t+2} \vee z_{t+1}), \tag{10.11}$$

where $z_1, z_2, \ldots, z_{k-2}$ are new variables which are not to be used again for any other k-clause. Clearly, a fixed assignment to $l_1, l_2, \ldots, l_k$ satisfies the original k-clause iff there is a further assignment to $z_1, z_2, \ldots, z_{k-2}$ that satisfies the formula in (10.11).

Thus the formula of (10.10) has been rewritten as a 3CNF formula that is satisfiable iff $x \in L$. Further, if $x \notin L$, every unsatisfied k-clause in (10.10) yields an unsatisfied 3-clause in the new formula, so the fraction of unsatisfied 3-clauses is at least $\frac{1}{k-2} \cdot \frac{1}{2^{k+1}}$.

Hence the lemma has been proved for the value of ϵ given by

$$\frac{1}{1+\epsilon} = 1 - \frac{1}{(k-2)2^{k+1}}. \blacksquare$$

PROBLEMS

EXERCISE 10.9

Note that the MAX-k-FUNCTION-SAT problem exactly represents the decision process of a $(\log n, 1)$-restricted verifier that examines k bits in membership proof. The optimum assignment for the MAX-k-FUNCTION-SAT instance can be viewed as a proof that maximizes the verifier's acceptance probability. The next exercise further clarifies the connection between the PCP Theorem and the inapproximability of MAX-SNP-hard problems.

1. Prove that MAX-k-FUNCTION-SAT is in Max-SNP for every fixed k.
2. Prove that for every Max-SNP problem, there is some integer k such that every instance of the problem can be represented as instances of MAX-k-FUNCTION-SAT.

Thus we have shown that MAX-k-FUNCTION-SAT is Max-SNP-hard.

10.7.3 WHERE THE GAP COMES FROM.

The gap of a factor of $1 + \epsilon$ in reduction (10.8) came from two sources: the gap 1 versus 1/2 in the fraction of satisfiable f_i's in Lemma 10.12, and the fact that each f_i involves $O(1)$ variables. But, remember that each f_i just represents a possible run of the $(\log n, 1)$-restricted verifier for SAT. Thus the description of each f_i — that is, a description of what function it is and which variables it depends upon — is derived from the program of the verifier. The current construction of this verifier is very involved. It involves defining a complicated algebraic object, which exists iff the input boolean formula is satisfiable. The verifier expects a membership proof to contain a representation of this object. In each of its runs the verifier examines a different part of this provided object. Thus the function f_i representing that run is a correctness condition for that part of the object.

For details on the algebraic object, we refer the reader to the relevant papers. A detail worth mentioning is that each part of the object — and thus, the definition of each f_i — depends on every input bit (i.e., on every clause of the input boolean formula). This imparts the reduction a global structure. In contrast, classical NP-completeness reductions usually perform local transformations of the input.

OPEN PROBLEMS 10.8

We list some open problems about the hardness of approximation.

OPEN PROBLEM 10.1

As shown in this chapter, all known inapproximability results can be derived from the PCP Theorem and Raz's Parallel Repetition Theorem [Raz94] (note that the latter was used to prove the inapproximability of Label Cover). The proofs of both these theorems are currently very difficult. Can they be simplified?

OPEN PROBLEM 10.2

For many problems such as CLIQUE, COLORING, and MAX-3SAT, there is a large gap between the approximation ratio that is provably hard to achieve, and the ratio that we can achieve in polynomial time (see Section 10.6). Can this gap be closed?

OPEN PROBLEM 10.3

Prove inapproximability results for edge deletion problems [Yan81], such as (DIRECTED) FEEDBACK ARC SET, GRAPH BISECTION, GRAPH EXPANSION, etc. Algorithms for many of these problems use similar ideas and achieve approximation ratios close to $O(\log n)$ [LR88].

OPEN PROBLEM 10.4

A look at Table 10.2 shows that many inapproximability results rely on an assumption stronger than $P \neq NP$. Can this assumption be weakened to $P \neq NP$? Currently, the reason for resorting to the stronger assumption is that the reduction to LABELCOVER in Theorem 10.2, part 3, uses a self-improvement property, and needs quasipolynomial time. A more efficient reduction could possibly merge Classes III and IV.

OPEN PROBLEM 10.5

Identify limitations of current techniques for proving inapproximability. For example, identify inapproximability results they inherently cannot prove. (A recent paper [Aro95] takes a first step in this direction.)

OPEN PROBLEM 10.6

Prove inapproximability results for the following problems: SHORTEST LATTICE VECTOR (in Euclidean norm), EUCLIDEAN TSP, and coloring a 3-colorable graph with the minimum possible number of colors. For the last problem, Khanna *et al.* [KLS93] have shown that it is NP-hard to color a 3 colorable graph using only 4 colors.

OPEN PROBLEM 10.7

The best approximation algorithm for GRAPH BISECTION [LR88] outputs an integer that approximates the *cost* of the minimum bisection, without outputting a bisection of this approximate cost (it outputs a $1/3 : 2/3$ cut of the graph and not a bisection). Approximation algorithms for some other problems also behave similarly; they output an approximation to the cost of the optimal solution without outputting a near-optimal solution. Prove (for any natural problem) that finding approximate solutions is harder than finding an approximation to the cost of the optimum solution.

OPEN PROBLEM 10.8

Explain why inapproximable problems form the four classes we have described. Are these classes "real," or does the classification reflect a limitation of our proof techniques. (A partial answer — namely, a nice explanation of class I — seems to lie in the Max-SNP class of [PY91].)

OPEN PROBLEM 10.9

Prove inapproximability results for some of the counting (or $\#P$) problems discussed in Chapter 12. For example, no good approximation algorithm is known for the problem of approximating the number of perfect matchings in a graph.

10.9 CHAPTER NOTES

10.9.0.0.4 Approximability. The question of approximability started receiving attention [Joh74, SG76] soon after *NP*-completeness was discovered. (See [GJ79] for a discussion.) Much of the work attempted to discover a classification framework for optimization (and approximation) problems analogous to the framework of *NP*-completeness for decision problems. (See [ADP77, ADP80, AMSP80] for some of these attempts.) The most successful attempt was due to Papadimitriou and Yannakakis, who based their classification around a complexity class they called Max-SNP (see Section 10.2.1). They proved that MAX-3SAT is complete for Max-SNP, in other words, any inapproximability result for MAX-3SAT transfers automatically to a host of other problems. The desire to prove such an inapproximability result motivated the discovery of the PCP theorem.

The idea (in [PY91]) of using logical characterizations of optimization problems to understand their approximability has also been extended in many directions [KT91, PR90].

10.9.0.0.5 Interactive Proof Systems. The PCP Theorem is a descendent of a long line of results in interactive proofs systems. These proofs systems were invented in [GMR89, Bab85], and quickly found applications in cryptography and complexity theory [GMW87, BGKW88]. The notion of probabilistically checkable membership proofs first appears in [FRS88]. For a survey of all probabilistic proof systems that have been defined and studied, see [Gol94].

Surprisingly, complexity classes defined in terms of interactive proofs are actually equivalent to conventional complexity classes. For example, IP = PSPACE [LFKN92, Sha92] and MIP = NEXPTIME [BFL91]. Lund's dissertation [Lun92] describes these characterizations.

10.9.0.0.6 Connection to inapproximability. Feige et al. [FGL$^+$91] showed how to extend some of the ideas in the MIP = NEXPTIME result to the class *NP*. They identified the notion of probabilistically checkable proofs (although under a different name) and proved a surprising result that in hindsight is a clear precursor of the PCP Theorem. Their work was paralleled by that in [BFLS91], who defined *transparent math proofs*, a notion related to probabilistically checkable proofs.

All the above results were very surprising. Even more surprising was the following result, which [FGL$^+$91] proved as a corollary of their result about proof-checking: Approximating the clique number within a factor $2^{\log^{1-\delta} n}$ is Quasi-NP-hard.

The [FGL$^+$91] result focused attention on the notion of probabilistically checkable proofs, and raised two important questions. First, could their techniques prove the inapproximability of any important problem other than clique? Second, could their techniques be sharpened to prove the *NP*-hardness of approximation, instead of *Quasi-NP*-hardness. The first question was partially answered in [Bel93, BR93, Zuc93], which showed the inapproximability of a variety of problems. Then the second question was answered — positively— in [AS92], which showed that approximating clique is *NP*-hard. Proving this result involved a new probabilistic characterization of *NP* in terms of PCP (a definition of the class PCP also makes an appearance in this paper). Then [ALM+92] gave further evidence of the usefulness of PCPs for proving inapproximability results. They proved the PCP Theorem, and showed that Max-SNP-hard problems do not have polynomial-time approximation schemes. The proof of the PCP Theorem uses many ideas from the then-unpublished [AS92] and papers on program checking [BK89, RS92]. Some observers attribute the PCP Theorem to [AS92, ALM+92] together.

An interesting development that went largely unnoticed is Condon's inapproximability result for the max-word problem [Con93]. She uses a different probabilistic characterization of *NP*.

The connection between interactive proof systems and inapproximability has provided the impetus for new constructions of various interactive proof systems [LS91, FL92, BGLR93, FK94, Raz94, BS94, BGS95].

10.9.0.0.7 Further reading. For a self-contained proof of the PCP Theorem and a detailed survey of related topics, see [Aro94]. The surveys by Babai [Bab94], Johnson [Joh92], and Goldreich [Gol94] provide three different perspectives (mostly without proofs) of the developments mentioned above. Sudan [Sud92] describes program checking for algebraic programs and how it figures in the proof of the PCP Theorem. For a listing of optimization problems according to their approximation properties, consult [CK94].

References

ABSS93. S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes and linear equations. In *Proc. 34th IEEE Symp. on Foundations of Computer Science*, pages 724–733, 1993.

ADP77. G. Ausiello, A. D'Atri, and M. Protasi. On the structure of combinatorial problems and structure preserving reductions. In *Proc. 4th Intl. Coll. on Automata, Languages and Programming*, 1977.

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.

AFWZ93. N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized graph products. Manuscript, 1993.

AGHP92. N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3, 1992. See also Addendum in *Random Structures and Algorithms* 4 pages ?? 1993.

AK93. M. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. Technical report, Department of Numerical Analysis and Computing Science, Royal Institue of Technilogy, Stockholm, 1993. Technical Report TRITA-NA-9313.

ALM+92. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 13–22, 1992.

AMSP80. G. Ausiello, A. Marchetti-Spaccamela, and M. Protasi. Toward a unified approach for the classification of NP-complete optimization problems. *Theoretical Computer Science*, 12:83–96, 1980.

Aro94. S. Arora. *Probabilistic checking of proofs and the hardness of approximation problems.* PhD thesis, U.C. Berkeley, 1994. Available via anonymous ftp as Princeton TR94-476.

Aro95. S. Arora. Reductions, codes, PCPs, and inapproximability. In *Proc. 36th IEEE Symp. on Foundations of Computer Science*, 1995.

AS92. S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 2–13, 1992.

Bab85. L. Babai. Trading group theory for randomness. In *Proc. 17th ACM Symp. on Theory of Computing*, pages 421–429, 1985.

Bab94. L. Babai. Transparent proofs and limits to approximations. In *Proceedings of the First European Congress of Mathematicians.* Birkhauser, 1994.

Bel93. M. Bellare. Interactive Proofs and approximation: Reductions from two provers in one round. In *Proceedings of the 2nd Israel Symposium on Theory and Computing Systems.* IEEE Computer Press, 1993. Preliminary version: IBM Research Report RC 17969 (May 1992).

BFL91. L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

BFLS91. L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 21–31, 1991.

BGKW88. M. Ben-or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi prover interactive proofs: How to remove intractability assumptions. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 113–121, 1988.

BGLR93. M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient multi-prover interactive proofs with applications to approximation problems. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 113–131, 1993.

BGS95. M. Bellare, O. Goldreich, and M. Sudan. Free bits and non-approximability– towards tight results. In *Proc. 36th IEEE Symp. on Foundations of Computer Science*, 1995. Full version available from the authors.

BH92. R. B. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32(2):180–196, June 1992.

BJL+91. A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 328–336, 1991.

BK89. M. Blum and S. Kannan. Designing programs that check their work. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 86–97, 1989.

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

BR93. M. Bellare and P. Rogaway. The complexity of approximating non-linear programs. In P.M. Pardalos, editor, *Complexity of Numerical Optimization*. World Scientific, 1993. Preliminary version: IBM Research Report RC 17831 (March 1992).

BS92. Piotr Berman and Georg Schnitger. On the complexity of approximating the independent set problem. *Information and Computation*, 96(1):77–94, 1992.

BS94. M. Bellare and M. Sudan. Improved non-approximability results. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 184–193, 1994.

CK94. P. Crescenzi and V. Kann. A compendium of NP optimization problems. manuscript, 1994.

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.

Coo71. S. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symp. on Theory of Computing*, pages 151–158, 1971.

DJP+92. E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 241–451, 1992.

Fag74. R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In Richard Karp, editor, *Complexity of Computer Computations*, pages 43–73. AMS, 1974.

Fei95. U. Feige. A threshold of $\ln n$ for approximating set cover. Manuscript, 1995.

FGL+91. U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 2–12, 1991.

FK94. U. Feige and J. Kilian. Two prover protocols–low error at affordable rates. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 172–183, 1994.

FK95. U. Feige and J. Kilian. Zero knowledge and the chromatic number. unpublished manuscript, 1995.

FL92. U. Feige and L. Lovász. Two-prover one-round proof systems: Their power and their problems. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 733–741, 1992.

FRS88. L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proceedings of the 3rd Conference on Structure in Complexity Theory*, pages 156–161, 1988.

Für95. M. Fürer. Improved hardness results for approximating the chromatic number. In *Proc. 36th IEEE Symp. on Foundations of Computer Science*, 1995.

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

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

GJ79. M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness.* W. H. Freeman, 1979.

GMR89. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proofs. *SIAM J. on Computing*, 18:186–208, 1989.

GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. 19th ACM Symp. on Theory of Computing*, pages 218–229, 1987.

Gol94. O. Goldreich. Probabilistic proof systems. Technical Report RS-94-28, Basic Research in Computer Science, Center of the Danish National Research Foundation, September 1994. (*To appear in the Proceedings of the International Congress of Mathematicians, 1994. Birkhauser Verlag.*).

GVY93. N. Garg, V.V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)-cut theorems and their applications. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 698 –707, 1993.

GW94. M. Goemans and D. Williamson. A 0.878 approximation algorithm for MAX-2SAT and MAX-CUT. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 422–431, 1994.

Hal93. Halldorsson. A still better performance guarantee for approximate graph coloring. *Information Processing Letters*, 45:19–23, 1993.

Has95. J. Hastad. Fast and efficient testing of the long code. unpublished manuscript, 1995.

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

Joh92. D. S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 13:502–524, 1992.

Kan87. R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3), 1987.

Kan92. V. Kann. *On the approximability of NP-complete optimization problems.* PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 1992.

Kar72. R. M. Karp. Reducibility among combinatorial problems. In Miller and Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

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

KMR93. D. Karger, R. Motwani, and G.D.S. Ramkumar. On approximating the longest path in a graph. In *Proceedings of Workshop on Algorithms and Data Structures*, pages 421–430. LNCS (Springer-Verlag), v. 709, 1993.

KT91. P. G. Kolaitis and M. N. Thakur. Approximation properties of NP minimization classes. In *Proc. of the 6th Conference on Structure in Complexity Theory*, pages 353–366, 1991.

Lev73. L. Levin. Universal'nyĭe perebornyĭe zadachi (universal search problems : in Russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973.

LFKN92. C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. of the ACM*, 39(4):859–868, October 1992.

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

LPS88. A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujam graphs. *Combinatorica*, 8:261–277, 1988.

LR88. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 422–431, 1988.

LS91. D. Lapidot and A. Shamir. Fully parallelized multi prover protocols for NEXPTIME. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 13–18, 1991.

Lun92. Carsten Lund. *The Power of Interaction*. MIT Press, Cambridge, Mass., 1992.

LY93. C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In *Proceedings of International Colloquium on Automata, Languages and Programming, ICALP*, pages 40–51, 1993.

LY94. Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

NN93. J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM J. on Computing*, 22:838–856, 1993. Prelim. version in ACM STOC'90.

PR90. A. Panconesi and D. Ranjan. Quantifiers and approximation. In *Proc. of the 22nd ACM Symp. on the Theory of Computing*, pages 446–456, 1990.

PY91. C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

PY93. C. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.

Raz94. R. Raz. A parallel repetition theorem. Manuscript, 1994.

RS92. R. Rubinfeld and M. Sudan. Testing polynomial functions efficiently and over rational domains. In *Proc. 3rd Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 23–32, 1992.

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

Sha92. A. Shamir. IP = PSPACE. *J. of the ACM*, 39(4):869–877, October 1992.

Sim90. H. U. Simon. On approximate solutions for combinatorial optimization problems. *SIAM J. Algebraic Discrete Methods*, 3:294–310, 1990.

Sud92. Madhu Sudan. *Efficient checking of polynomials and proofs and the hardness of approximation problems.* PhD thesis, U.C. Berkeley, 1992.

Yan79. M. Yannakakis. The effect of a connectivity requirement on the complexity of of maximum subgraph problems. *Journal of the ACM*, 26:618–630, 1979.

Yan81. M. Yannakakis. Edge deletion problems. *SIAM Journal of Computing*, 10:77–89, 1981.

Yan92. M. Yannakakis. On the approximation of maximum satisfiability. In *Proceedings of 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–9, 1992.

Zuc93. D. Zuckerman. NP-complete problems have a version that's hard to approximate. In *8th Structure in Complexity Theory Conf.*, pages 305–312, 1993.