Free Bits, PCPs and Non-Approximability—Towards Tight Results

(3rd Version)

Mihir Bellare[1] Oded Goldreich[2] Madhu Sudan[3]

December 31, 1995

In honor of Shimon Even's 60^{th} *birthday.*

[1] Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093, USA. e-mail: `mihir@cs.ucsd.edu`.

[2] Department of Applied Mathematics, Weizmann Institute of Sciences, Rehovot, Israel. e-mail: `oded@wisdom.weizmann.ac.il`. Partially supported by grant No. 92-00226 from the US–Israel Binational Science Foundation (BSF), Jerusalem, Israel.

[3] IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA. e-mail: `madhu@watson.ibm.com`.

Abstract

This paper continues the investigation of the connection between proof systems and approximation. The emphasis is on proving *tight* non-approximability results via consideration of measures like the "free bit complexity" and the "amortized free bit complexity" of proof systems.

The first part of the paper presents a collection of new proof systems based on a new error-correcting code called the long code, and means to test it. We provide a proof system which has amortized free bit complexity of $2+\epsilon$, implying that approximating Max Clique within $N^{\frac{1}{3}-\epsilon}$, and approximating the Chromatic Number within $N^{\frac{1}{5}-\epsilon}$, are hard assuming NP $\neq$ coRP, for any $\epsilon > 0$. We also derive the first explicit and reasonable constant hardness factors for Min Vertex Cover, Max-2-SAT, and Max Cut, and improve the hardness factor for Max-3-SAT. We note that our non-approximability factors for Max-SNP problems are appreciably close to the values known to be achievable by polynomial time algorithms. Finally we note a general approach to the derivation of strong non-approximability results under which the problem reduces to the construction of certain "gadgets."

The increasing strength of non-approximability results via the proof checking connection motivates us to ask how far this can go, and whether proofs are inherent in any way. This is addressed in the second part of the paper. Recall that [FGLSS] showed how to translate proof systems for NP into NP-hardness of approximation results for Max Clique. We begin with a result of a novel nature which essentially reverses this connection, showing how any NP-hardness of approximation result yields a proof system for NP. Roughly our result says that for any constant f if Max Clique is NP-hard to approximate within $N^{1/(1+f)}$ then NP is in the class $\overline{\mathrm{FPCP}}[\log, f]$ of languages possessing proofs of logarithmic randomness and amortized free bit complexity f. This indicates that proofs are inherent to obtaining non-approximability results. But it does more: it provides a tight relation indicating that to get large hardness factors we must minimize the amortized free bit complexity.

The third part of our paper initiates a systematic investigation of the properties of PCP and FPCP as a function of the various parameters: randomness, query complexity, free bit complexity, amortized free bit complexity, proof size, etc. We are particularly interested in "triviality" results which indicate which combinations of parameters are *not* powerful enough to capture NP. We also distill the role of randomized reductions in this area, and provide a variety of useful transformations between proof checking complexity classes.

Table of Contents

1 Introduction **5**

1.1 Overview of main results . . . 6

1.1.1 New proof systems and non-approximability results . . . 6

1.1.2 Proofs and approximation: Potential and limitations . . . 6

1.1.3 PCP and FPCP: Properties and transforms . . . 7

1.1.4 Conceptual contributions . . . 7

1.1.5 Previous version, current version and future versions . . . 7

1.2 Some background and definitions . . . 8

1.3 New proof systems and non-approximability results . . . 10

1.3.1 New proof systems . . . 10

1.3.2 New non-approximability results . . . 11

1.3.3 Techniques . . . 12

1.4 Proofs and approximation: Potential and limits . . . 12

1.4.1 Reversing the connection: Making proofs from gaps . . . 12

1.4.2 A lower bounds on amortized free-bits . . . 13

1.5 Properties and transforms of PCP and FPCP . . . 14

1.5.1 Triviality results . . . 14

1.5.2 Other results . . . 15

1.5.3 Transformations between proof systems . . . 16

1.6 History . . . 16

1.7 Related work . . . 18

1.8 Directions for further research . . . 18

1.9 Acknowledgments . . . 19

2 Notation and Definitions **20**

2.1 General notation and definitions . . . 20

2.2 Proof systems . . . 20

2.3 Randomized reductions 22
2.4 History 23

3 New proof systems and non-approximability results 25
3.1 Overview and guidemap 25
3.2 Preliminaries 26
3.3 Evaluation operators and the long code 28
3.4 Recursive verification of proofs 30
3.4.1 Outer verifiers 30
3.4.2 Inner verifiers 31
3.4.3 Composition of verifiers 32
3.4.4 Constant-prover proofs in PCP — perspective 34
3.5 The atomic tests 35
3.5.1 Atomic linearity test 37
3.5.2 Atomic respect of monomial basis test 38
3.5.3 Atomic projection test 42
3.5.4 Atomic circuit test 43
3.6 The MAX SNP verifier 43
3.6.1 The inner verifier 43
3.6.2 Main application: the MaxSNP verifier 46
3.6.3 Another application: minimizing soundness error in 3-query pcp 48
3.7 Satisfiability problems: Max-3-SAT and Max-2-SAT 48
3.7.1 Definitions 48
3.7.2 Previous work 49
3.7.3 New Results 50
3.7.4 Gadgets and the Hardness of MaxSAT 50
3.7.5 Maximum Satisfiable Linear Constraints (Parity Clauses) 55
3.8 Max-CUT 56
3.8.1 Definitions 56
3.8.2 Previous work 56
3.8.3 New Result 56
3.8.4 Gadgets and the hardness of Max-CUT 57
3.9 Free bits and vertex cover 60
3.9.1 Minimizing the error achievable with two free bits 61
3.9.2 Hardness of vertex cover 65
3.9.3 On using the MaxSNP verifier to establish Min VC hardness 66
3.10 Minimizing the number of queries 67
3.10.1 The PCP inner verifier 68
3.10.2 The new proof system 72
3.11 The iterated tests 72
3.11.1 Linearity and randomness 72
3.11.2 Iterated projection test 73

3.11.3 Technical claim 74
3.11.4 Iterated linearity test 74
3.11.5 Iterated RMB test 75
3.11.6 Putting some things together 76
3.12 Amortized free bits, Max Clique, and Coloring 76
3.12.1 Definitions 76
3.12.2 Sources of our improvements 77
3.12.3 Construction and results 77
3.12.4 Previous work 79
3.13 The coding theory bound 81
3.14 On the optimality of some choices in our analysis 82

4 Proofs and approximation: Potential and limitations 86
4.1 The reverse connection and its consequences 86
4.1.1 The Clique-Gap Verifier 86
4.1.2 Main Consequences 89
4.1.3 More Consequences 94
4.2 On the Limitations of Some Common Approaches 97
4.2.1 The tasks 97
4.2.2 Lower Bound for the Codeword Test 99
4.2.3 Lower Bound for the Projection Test 101
4.2.4 Lower Bound for the Combined Test 103

5 PCP: Properties and Transformations 106
5.1 The Complexity of PCP and FPCP 106
5.1.1 Query complexity and amortized query complexity 106
5.1.2 Free-bit complexity 112
5.1.3 Query complexity versus free-bit complexity 114
5.2 Transformations of FPCP Systems 115
5.2.1 Gap amplification maintaining amortized free-bit complexity 115
5.2.2 Trading-off gap location and free-bit complexity 119

List of Figures

1.1 *New PCP Systems for NP, all with logarithmic randomness.* 10

1.2 *Approximation factors attainable by polynomial-time algorithms (Approx) versus factors we show are hard to achieve (Non-Approx).* 11

3.1 *Constant prover PCPs achieving error which is a fixed, but arbitrarily small, constant ϵ. We indicate the number of provers, the randomness and answer sizes, and whether or not the system is canonical. The notation ?? means "don't know and don't care because stronger things have become available." In all cases the randomness and answer sizes hide factors which depend on ϵ.* 35

3.2 *The atomic tests and their passing probabilities.* 36

3.3 *The Max-SNP inner verifier V_{SNPinner}* 44

3.4 *Non-approximability results for Max-3-SAT indicating the factor shown hard and the assumption under which this was done.* 49

3.5 *The Max-E3-SAT Gadgets* 52

3.6 *The Max-2-SAT Gadgets* 53

3.7 *The Enhanced RMB test and its passing probability.* 61

3.8 *The two free-bit inner verifier V_{2inner}* 62

3.9 *Worst case (q) and average (q_{av}) number of queries needed to get $1/2$ soundness with logarithmic randomness; that is, results of the form of Eq. (3.14).* 68

3.10 *The PCP inner verifier V_{PCPinner}* 69

3.11 *The iterated tests and their passing probabilities.* 75

3.12 *The free inner verifier $V_{\mathrm{mrfree\text{-}in}}$* 77

3.13 *Some Milestones in the project of proving non-approximability of the Clique number: Approximation Factor (in terms of the graph size N) which is infeasible to achieve under an indicated Assumption. In stating results from [BGLR] on, we ignore N^{ϵ} terms in which $\epsilon > 0$ can be arbitrary small.* 80

C H A P T E R 1

Introduction

In the Max Clique problem we are given a graph G and must find the value of $\mathsf{MaxClique}(G) = \max\{\,|S| : S \text{ is a clique in } G\,\}$. It is an example of an NP-optimization problem, of which others are to find the chromatic number of a graph; to find the size of the smallest vertex cover; etc. These problems arise in a large and varied number of settings, and efficient solutions are much desired. Unfortunately, many important NP-optimization problems (those mentioned above in particular) are NP-hard to solve. So algorithm designers seek efficient (polynomial time) approximation algorithms.

An approximation algorithm A delivers a number that is supposed to be close to optimal. The quality of the algorithm is measured in terms of what factor of optimal is the delivered number. For example, a factor $\alpha(\cdot) \geq 1$ approximation for Max Clique is one which given G outputs a value v satisfying $\mathsf{MaxClique}(G)/\alpha(N) \leq v \leq \mathsf{MaxClique}(G)$ where N is the number of nodes in G.

The search for efficient approximation algorithms achieving good factors has met with varied success. In some cases, good approximation algorithms were found. But many important problems, including Max Clique, Chromatic Number and Min Vertex Cover, escaped efforts to be approximated at all (in the case of the first two problems) or reasonably well (in the case of the last). Algorithm designers want to know whether this is due to some inherent intractability, or only to the lack of cleverness in algorithm design.

Some early non-approximability results were able to indicate, in some cases, that very good approximation (ie. achieving factors very close to optimal) can be NP-hard. But the real breakthrough came more recently, when a connection was established between proof checking and approximation, yielding a strong non-approximability result for Max Clique. This connection has, over the last few years, been broadened and deepened: more and more problems have fallen to this approach, and meanwhile the factors that one can indicate hard to approximate increase. Indeed, in some cases, even tight results seem in sight.

We will now provide a high level overview of our main contributions. Then we will provide some definitions and state precise theorems.

The above, and most of the following discussion has omitted, for explanatory simplicity, the historical story that accompanies the technical advances. In Section 1.6 we provide a history of the main flow of works and ideas in the area. More detailed credits and historical discussions on specific topics can be found in the text relating to the topic in question, and pointers to these discussions are also given in Section 1.6.

1.1 Overview of main results

This paper continues and expands the research in non-approximability via proof systems, with a focus on the obtaining of tight results. Here we briefly summarize our contributions. Later we will state the results more precisely.

1.1.1 New proof systems and non-approximability results

Our first set of results continues previous work by building new and more efficient proof systems and thus improving (increasing) factors shown non-approximable for a wide variety of optimization problems.

We obtain improved non-approximability results for Max Clique, Chromatic Number, and Max-3-SAT. We also obtain the first reasonable and explicit constant factor non-approximability result for the Min Vertex Cover problem, Max Cut, and Max-2-SAT. Several of these results are strong enough to indicate that the gap between factors that are attainable by polynomial time algorithms, and those we can indicate are not, is now quite narrow. See Figure 1.2.

The technical foundation of these results is a new code, call the long code, and a collection of associated tests. The tests are used to construct proof systems for NP. Key to the improvements in non-approximability factors (for some of the above problems) is the focus on certain measures of proof checking complexity such as "free-bits" and "amortized free-bits." In the latter domain the main result is a proof system for NP using two amortized free-bits, and directly yielding a $N^{1/3}$ non-approximability factor for Max Clique.

We emphasize a general framework for the derivation of strong non-approximability results for Max-SNP problems which results from our tests and proof systems: obtaining a non-approximability result for a particular problem is reduced to the construction of appropriate "gadgets" to "represent" two simple functions: boolean XOR and boolean AND.

1.1.2 Proofs and approximation: Potential and limitations

As the above indicates, non-approximability results are getting steadily stronger, especially for Max Clique. How far can they go? And, in minimizing amortized free-bits, are we on the right track? Are there other ways? The next set of results provides answers to these kinds of questions.

A reverse connection

We focus on the Max Clique problem. We present a result which indicates that proof checking is *necessary* to getting non-approximability results. Furthermore, it indicates that not just proof checking, but the minimization of the amortized free-bit complexity is necessary.

Roughly, we show that if, for some $f > 0$, Max Clique is NP-hard to approximate within $N^{1/(1+f)}$ then NP has proof systems of (logarithmic randomness and) amortized free-bit complexity f. This result can be viewed as "inverting," in a strong way, the FGLSS-connection.

So our current efforts (recall that we have the amortized free-bit complexity down to two, yielding a $N^{1/3}$ hardness for Max Clique) are in the right direction. To prove that, say Max Clique is hard to approximate within $\sqrt{N}$, our reverse connection says we must construct proof systems with amortized free-bit complexity one.

A lower bound on amortized free-bits

Now that we know we must minimize amortized free-bits, we ask ourselves how low we can take them. Our approach here is to look at current techniques and assess their limitations. We stress that this approach makes various assumptions about methods, and is intended to show that significantly novel techniques are required to go further. But it does not suggest an *inherent* limitation. Indeed, if we believe Max Clique is hard to approximate within $N^{1-o(1)}$ then our reverse connection says proof systems with arbitrarily small constant amortized free-bit complexity exist; we are just saying they may be hard to find.

1.1.3 PCP and FPCP: Properties and transforms

Probabilistic proofs involve a vast arena of complexity parameters: query complexity, free-bit complexity, amortized free-bit complexity, randomness, and proof sizes to name a few. Some might, at first glance, seem less "natural" than others; yet all are important in applications. A better understanding of the basic properties and relations between these parameters would help move us forward.

We initiate, accordingly, a systematic investigation of the properties of pcp complexity classes as a function of the parameter values. Besides providing new results we take the opportunity to state and prove a few folklore ones.

We focus in particular on "triviality" results. These are results which say that certain parameter combinations yield classes probably not capable of capturing NP. For example, the class of languages recognizable with error 1/2 and logarithmic randomness using one (non-amortized!) free-bit is in P— so don't expect to prove NP using just one free-bit. (But nothing rules this out when amortization is considered).

We also investigate transformations: to reduce the randomness, error or other complexities at various costs.

1.1.4 Conceptual contributions

The reverse connection does more than guide our choice of parameters. It provides a new conceptual tool because it enables us to reflect, in the language of proof systems, theorems, properties and transformations of graphs, and vice versa. This turns out to be very useful and revealing. It also leads, in some cases to new results derived by turning graphs into proof systems via our connection, and then back to graphs via the FGLSS connection, in the process gaining some property. As an example we show how all known hardness results for chromatic number can be viewed (with almost no loss in efficiency) as reductions from Max Clique — even though these were essentially hardness results based on proof checking. Other examples demonstrating the usefulness of the equivalence may be found in Section 4.1.3. We believe that exploring and exploiting further this duality is a fruitful avenue to pursue.

A second (and related) conceptual contribution of this work is to distill and formalize the role of randomized reductions. These transforms provide an elegant and concise way of stating connections between proofs and approximability, or just between different kinds of proof systems, and make it easier to manipulate the many connections that exist to derive new results.

1.1.5 Previous version, current version and future versions

This is a *third version* of our work. Both previous versions, dated May and August 1995, respectively, are available from ECCC, the *Electronic Colloquium on Computational Complexity*, `http://www.eccc.uni-trier.de/eccc/`.

The second version improves over the first one in the analysis of the MAX-SNP verifier and consequently in the hardness factors achieved via this verifier (i.e., for Max-3-SAT, Max-2-SAT, and Max-CUT). In addition, a new transformation of pcp systems is presented (Proposition 5.2.9) resolving an open problem mentioned in the first version (i.e., showing a pcp system for NP with perfect completeness, logarithmic randomness, soundness error $s < 0.943 < 1$ and free-bit complexity $\log_2 3 < 2$). Finally, some minor flaws in the original expositions were removed (e.g., see Definition 3.3.1 and "double folding").

The current version presents improvements for many of the results of Chapter 3. In particular, we get hardness factors of 27/26, 74/73, 66/65 and 16/15 for Max3SAT, Max2SAT, MaxCUT and MinVC, respectively.[1] In addition, we obtain $\mathrm{NP} = \mathrm{PCP}_{1,0.5}[\log, 11] = \mathrm{FPCP}_{1,0.5}[\log, 6]$ and $\mathrm{NP} = \mathrm{PCP}_{1,0.851}[\log, 3] = \mathrm{FPCP}_{1,0.794}[\log, 2]$.[2] A key ingredient in obtaining all these improvements is a new (adaptive!) RMB test, replacing the previous RMB test.

1.2 Some background and definitions

In the next sections we will state more precisely the results and theorems corresponding to the above discussion. In order to do this we have to recall some minimal number of definitions and background. Here we will be informal and as brief as possible; formal definitions can be found in Chapter 2.

PROOF SYSTEMS AND PARAMETERS. A probabilistic proof system is described by a probabilistic, polynomial time *verifier* V. It takes an input x of length n and tosses coins R. It has oracle access to a poly(n) length string σ describing the proof: to access a bit it writes a $O(\log n)$ bit address and is returned the corresponding bit of the proof. Following its computation it will either accept or reject its input x. The accepting probability, denoted $\mathtt{ACC}[\,V(x)\,]$, is the maximum, over all σ, of the probability (over R) that V accepts x on coins R and proof string σ. While the task is typically language recognition, we will, more generally, consider promise problems (A, B) consisting of a set A of "positive" instances and a set B of "negative" instances [ESY]. (Languages are a special case of promise problems; a language L is represented by the promise problem $(L, \overline{L})$.)

Of interest in the applications are various parameters of the system. The completeness probability $c = c(n)$ and the soundness probability $s = s(n)$ are defined in the usual ways. In case $c = 1$ we say that the system has **perfect completeness**. The gap is $g = c/s$. The query complexity is the maximum (over all coin tosses and proof strings) of the number of bits of the proof that are examined by the verifier. The free-bit complexity, roughly speaking, is the logarithm of number of possible accepting configurations of V on coins R and input x. (For example a verifier which makes 3 queries and accepts iff the parity of the answers is odd has 4 accepting configuration and thus free-bit complexity 2.)

Either the query or the free-bit complexity may be considered in amortized form: e.g. the amortized free-bit complexity is the free-bit complexity (of a proof system with perfect completeness) divided by the logarithm of the gap. (That is, the number of free-bits needed per factor of 2 increase in the gap.) Also, either the query or free-bit complexity may be considered on the average, the average being over the random string of the verifier.

We use the notation $\mathrm{PCP}_{c,s}[r, q]$ to denote the class of promise problems recognized by verifiers tossing r coins, having query complexity q, and achieving completeness probability c and soundness

[1] The corresponding factors in Version 2 were 38/37, 94/93, 82/81 and 27/26.

[2] Improving over $\mathrm{NP} = \mathrm{PCP}_{1,0.5}[\log, 19] = \mathrm{FPCP}_{1,0.5}[\log, 11]$ and $\mathrm{NP} = \mathrm{PCP}_{1,0.8999}[\log, 3] = \mathrm{FPCP}_{1,0.884464}[\log, 2]$, respectively, obtained in Version 2.

probability s. $\mathrm{FPCP}_{c,s}[r, f]$ is defined analogously with f being the free-bit complexity. $\overline{\mathrm{PCP}}[r, q]$ is defined analogously with q being the amortized query complexity, and $\overline{\mathrm{FPCP}}[r, f]$ is defined analogously with f the amortized free-bit complexity.

Max Clique approximation. Although we look at many optimization problems there is a particular focus on Max Clique. Recall the best known polynomial time approximation algorithm for Max Clique achieves a factor of only $N^{1-o(1)}$ [BoHa], scarcely better than the trivial factor of N. (Throughout the paper, when discussing the Max Clique problem, N denotes the number of vertices in the graph.) There is not even a heuristic algorithm that is conjectured to do better. (The Lovász Theta function had been conjectured to approximate the Max Clique size within $\sqrt{N}$ but this conjecture was disproved by Feige [Fei].)

The question of whether one can do even slightly better is of interest. Namely, can one present an $N^{1-\epsilon}$ factor approximation algorithm for Max Clique for some $\epsilon < 1$? An additional motivation for searching for such "weak" approximation algorithms was suggested by Blum. He showed that a polynomial-time $N^{1-\epsilon}$-factor approximation algorithm for Max Clique implies a polynomial time algorithm to color a three colorable graph with $O(\log N)$ colors [Bl], which is much better than currently known [KMS].

But perhaps $N^{1-o(1)}$ is the best possible. Resolving the approximation complexity of this basic problem seems, in any case, to be worth some effort.

Gaps in clique size. Hardness of approximation (say of Max Clique) is typically shown via the construction of promise problems with gaps in max clique size. Specifically, let $\mathsf{Gap}\text{-Clique}_{c,s}$ be the promise problem (A, B) defined as follows: A is the set of all graphs G with $\mathsf{MaxClique}(G)/N \geq c(N)$, and B is the set of all graphs G with $\mathsf{MaxClique}(G)/N \leq s(N)$. The gap is defined as c/s. Now, a hardness result will typically specify a value of the gap $g(N) = c(N)/s(N)$ for which $\mathsf{Gap}\text{-Clique}_{c,s}$ is NP-hard under a (randomized) Karp reduction. This means that there is no polynomial time algorithm to approximate the Max Clique size of an N node graph within $g(N)$ unless NP has randomized polynomial time algorithms.

Gap problems can be similarly defined for all the other optimization problems we consider. From now on, we discuss approximation in terms of these gap problems.

The connection: Making gaps from proofs. We need to recall something about the manner in which proof systems are translated into (NP-hard) gap problems. We will refer to the FGLSS-reduction, which we recall is a reduction of a promise problem (A, B), or rather a pcp system for (A, B), which maps an input $x \in A \cup B$ to a graph G_x so that $\mathsf{MaxClique}(G_x)$ reflects $\mathtt{ACC}[\,V(x)\,]$. For the best results one typically uses a randomized form of this reduction due to [BeSc, Zu] and it is this that we will assume henceforth.

A NP-hard gap problem is obtained roughly as follows. First, one exhibits an appropriate proof system for NP. Then one applies the FGLSS reduction. The factor indicated hard depends on the proof system parameters. A key factor in getting better results has been the distilling of appropriate pcp-parameters. The sequence of works [FGLSS, ArSa, ALMSS, BGLR, FeKi, BeSu] lead us through a sequence of parameters: query complexity, free-bit complexity and, finally, for the best known results, amortized free-bit complexity. The connection in terms of amortized free-bits can be stated as follows: if NP reduces to $\overline{\mathrm{FPCP}}[\log, f]$ then NP also reduces to $\mathsf{Gap}\text{-Clique}_{c,s}$, with gap $c(N)/s(N) = N^{1/(1+f)}$. (In both cases the reduction is via randomized Karp reductions, and terms of $\epsilon > 0$ which can be arbitrarily small are ignored.) In particular if $\mathrm{NP} \subseteq \overline{\mathrm{FPCP}}[\log, f]$ then approximating the max clique size of an N vertex graph within $N^{1/(1+f)}$ in polynomial time is not possible unless NP has efficient randomized polynomial time algorithms.

1.3 New proof systems and non-approximability results

This section describes the proof systems that we construct and the non-approximability results that we derive from them. All proof systems are based on the long code and its checking machinery. For some of the non-approximability results we introduce new reductions or improve currently known reductions.

1.3.1 New proof systems

The following theorem summarizes the new proof systems that we obtain. Some are motivated by applications, others purely as interesting items in proof theory. Following the theorem is the discussion and motivation.

Theorem 1.3.1 We provide the following new proof systems for NP—

(1) For every $\epsilon > 0$ it is the case that $\mathrm{NP} \subseteq \overline{\mathrm{FPCP}}[\log, 2+\epsilon]$.

(2) $\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}[\log, 11]$.

(3) $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,s}[\log, 2]$ for $s = 0.794$.

(4) $\mathrm{NP} \subseteq \mathrm{PCP}_{1,s}[\log, 3]$ for any $s > 0.85$.

The search for proof systems of low amortized free-bit complexity is motivated of course by the FGLSS reduction. Bellare and Sudan [BeSu] have shown that $\mathrm{NP} \subseteq \overline{\mathrm{FPCP}}[\log, 3+\epsilon]$ for every $\epsilon > 0$. The first result above improves upon this, presenting a new proof system with amortized free-bit complexity $2+\epsilon$.

The question of how low one can get the (worst-case and average) query complexity required to attain soundness error 1/2 was investigated a lot in earlier works because they were applying the result to obtain Max Clique hardness results. We now know we can do better with amortized free-bit complexity. Nevertheless, the original question is still one to which we are curious to know the answer.

Minimizing the soundness error obtainable using only two (non-amortized!) free-bits is important for a more pragmatic reason. It enables us to get the first explicit and reasonably strong constant non-approximability result for the Min Vertex Cover problem. This application is discussed below.

Finally, what soundness one can achieve using only three query bits is a natural question given the Max 3SAT gap results. Indeed, if there is an NP-hard Max 3SAT gap problem with certain gap then one can easily get a three query proof system with the same gap. But in fact one can do better as indicated above.

focus	error	queries	free-bits	previous related result
3 queries	0.85	3	2	error $\frac{72}{73}$ via MaxSAT [BeSu]
2 free-bits	0.794	$O(1)$	2	
error 1/2	$\frac{1}{2}$	11	7	32 queries (24 on average) [FeKi]
amortized free-bits	$O(2^{-m})$	2^{3m}	$2m$	$3m$ free-bits [BeSu]

Figure 1.1: *New PCP Systems for NP, all with logarithmic randomness.*

Problem	**Approx**		**Non-Approx**		
	Factor	Due to	New Factor	Previous Factor	Assumption
Max-3-SAT	1.258	[Ya, GoWi2, SSTW]	1.038	$1+\frac{1}{72}$ [BeSu]	P $\neq$ NP
Max-E3-SAT	$1+\frac{1}{7}$	folklore	$1+\frac{1}{26}$	unspecified [ALMSS]	P $\neq$ NP
Max-2-SAT	1.075	[GoWi2, FeGo]	1.013	$1+\frac{1}{504}$ (implied [BeSu])	P $\neq$ NP
Max-CUT	1.139	[GoWi2]	1.015	unspecified [ALMSS]	P $\neq$ NP
Min-VC	$2-o(1)$	[BaEv, MoSp]	$1+\frac{1}{15}$	unspecified [ALMSS]	P $\neq$ NP
Max-Clique	$N^{1-o(1)}$	[BoHa]		$N^{\frac{1}{4}}$ [BeSu]	NP $\not\subseteq$ co$\tilde{\text{RP}}$
			$N^{\frac{1}{3}}$	$N^{\frac{1}{5}}$	coRP $\neq$ NP
			$N^{\frac{1}{4}}$	$N^{\frac{1}{6}}$ [BeSu]	P $\neq$ NP
Chromatic Number	$N^{1-o(1)}$	[BoHa]		$N^{\frac{1}{10}}$ [BeSu]	NP $\not\subseteq$ co$\tilde{\text{RP}}$
			$N^{\frac{1}{5}}$	$N^{\frac{1}{13}}$	coRP $\neq$ NP
			$N^{\frac{1}{7}}$	$N^{\frac{1}{14}}$ [BeSu]	P $\neq$ NP

Figure 1.2: *Approximation factors attainable by polynomial-time algorithms (Approx) versus factors we show are hard to achieve (Non-Approx).*

In Figure 1.1 we present a table which depicts the parameters of our new proof systems and compares them to previous related result. The last row in the table corresponds to the proof system used to establish Part (1) of Theorem 1.3.1.

1.3.2 New non-approximability results

Again we first state the theorem and then discuss it. But the best thing to do is look at Figure 1.2.

Theorem 1.3.2 The following indicate factors not achievable in polynomial time for the indicated problems, and the assumption under which the result is shown. Here $\epsilon > 0$ is an arbitrary constant and N is, for the first two results, the number of vertices in the graph–

(1) A factor of $N^{\frac{1}{3}-\epsilon}$ for Max Clique assuming NP $\not\subseteq$ coRP.

(2) A factor of $N^{\frac{1}{5}-\epsilon}$ for Chromatic Number assuming NP $\not\subseteq$ coRP.

(3) A factor of 16/15 for Min Vertex Cover assuming P $\neq$ NP.

(4) A factor of 27/26 for Max-3-SAT and Max Exact 3SAT assuming P $\neq$ NP.

(5) A factor of 66/65 for Max-CUT assuming P $\neq$ NP.

(6) A factor of 74/73 for Max-2-SAT assuming P $\neq$ NP.

The conclusion for Max Clique follows, of course, from the FGLSS-reduction and Part (1) of Theorem 1.3.1. The conclusion for the Chromatic Number follows from a recent reduction of Furer [Fu], which in turn builds on reductions in [LuYa, KLS, BeSu].

The improvements for the Max-SNP problems are perhaps more significant than the Max Clique one: for the first time, we see hardness results for Max-SNP problems which are comparable to the factors achieved by known polynomial time approximation algorithms.

We are obtaining the first explicit and reasonable non-approximability factor for Max-2-SAT, Max-CUTand minimum Vertex Cover. Recall that the latter is approximable within 2-o(1) [BaEv, MoSp]. Our results for Max-CUTand Max-2-SAT show that it is not possible to find a solution with value which is only 1% away from being optimal. This may be contrasted with the recent results of [GoWi2, FeGo] which shows that solutions which are within 14% and 7.5%, respectively, of the optimum are obtainable in polynomial time. Thus even though, we do not know if the "pcp approach" allows to get the best possible non-approximability results for these problems, we feel that the current results are not ridiculously far from the known upper bounds. Consider, for example, the ratio $\frac{u-1}{l-1}$, where u and l are the currently known upper and lower bounds, respectively. Then, the ratios for the above mentioned Max-SNP problems are 3.7 for Max Exact 3SAT, 5.5 for Max-2-SAT, 6.8 for Max-3-SAT, 9 for Max-CUT, and 15 for MinVC (Minimum Vertex Cover).

In Figure 1.2 we present a table which depicts, for each problem we have considered, the best known factor achievable by a polynomial time algorithm, our lower bound, and the best previous lower bound. We ignore, as usual, terms of N^ϵ where $\epsilon > 0$ is an arbitrary positive constant.

1.3.3 Techniques

As in all recent constructions of efficient pcp's our construction also relies on the use of recursive construction of verifiers, introduced by Arora and Safra [ArSa]. We have the advantage of being able to use, at the outer level, the verifier of Raz [Raz] which appeared only recently and was not available to previous works. The inner level verifier relies on the use of a "good" encoding scheme. Since [ALMSS], constructions of this verifier have used the Hadamard Code for this purpose. In this paper we change this aspect of the protocol and use instead a much more redundant code which we call the long code. This code encodes an n-bit string as a 2^{2^n} bit string which consists of the value of every boolean function on the n-bit string. It is easy to see such codes have large Hamming distance. What is important is that this code is also easily "testable" and "correctable". This is shown in Section 3, where we show how this code translates into the theorem described above.

A second aspect of the improved hardness result is the fact that we use direct reductions from verifiers to the problems of interest. This follows and extends [BGLR], prior to which results had used "generic" reductions, which did not take advantage of the nature of the tests performed by the verifier. In particular, in our case it turns out that the verifier only performs two kinds of tests — (1) verify that $a + b + c = 0 \pmod 2$; and (2) verify that $a = b_c$, where a, b, b_0, b_1, c are all elements of $\mathrm{GF}(2) = \{0, 1\}$. By constructing local gadgets (i.e., one gadget per random coin toss sequence) to verify each of the verifier's tests, we achieve better non-approximability results than using more general reductions. In particular our work seems to suggest that optimizing for gadgets which "check" the two conditions listed above will lead to reasonably good lower bounds for many Max-SNP problems.

1.4 Proofs and approximation: Potential and limits

Next we describe the results concerned with exploring the limitations of proof theoretic techniques in approximation.

1.4.1 Reversing the connection: Making proofs from gaps

The FGLSS Reduction Lemma indicates that one route to good non-approximability results for Max Clique is to show $\mathrm{NP} \subseteq \overline{\mathrm{FPCP}}[\log, f]$ for values of f which are as small as possible. Our reverse connection says that, in fact, this is the *only* way to proceed. Namely, we "invert" the above FGLSS-reduction. The following states an equivalence: (2)$\Rightarrow$(1) is just the FGLSS-reduction; (1)$\Rightarrow$(2) is our reversed connection. The following statement ignores terms of $\epsilon > 0$ which can be arbitrarily small. The proof and a more precise statement are in Section 4.1.

Theorem 1.4.1 Let f be a constant. Then the following statements are equivalent:
(1) NP reduces to $\mathsf{Gap}\text{-Clique}_{c,s}$ with gap $c(N)/s(N) = N^{1/(1+f)}$.
(2) NP reduces to $\overline{\mathrm{FPCP}}[\log, f]$.
In both cases the reduction is randomized. Furthermore the statement holds both for Karp and for Cook reductions. Also, if (1) holds with a deterministic Karp reduction then NP $\subseteq \overline{\mathrm{FPCP}}'[\log, f]$, where FPCP$'$ is defined as being the amortized free-bit complexity of proof systems with almost-perfect completeness (i.e., $c = 1 - o(1)$).

In other words ANY method of proving NP-hardness of Max Clique approximation to a factor of $N^{1/(1+f)}$ implies that NP has proof systems of amortized free-bit complexity f.

We stress both the "qualitative" and the "quantitative" aspects of this result. Qualitatively, it provides an answer to the following kind of a question: "What do proofs have to do with approximating clique size, and can we not prove non-approximability results without using proof checking?" The result indicates that proofs are inherent, and explains, perhaps, why hardness results avoiding the proof connection have not appeared.

However, at this stage it is the quantitative aspect that interests us more. It says that to get tighter results on Max Clique hardness, we must construct proof systems to minimize the amortized free-bit complexity. Thus our work with the long code was in the right direction. A question is whether the amortized free-bit bound of 2 can be improved.

1.4.2 A lower bounds on amortized free-bits

The following text has appeared in the previous version of our work [BGS]:

> We show that, under the framework used within this and previous papers on this subject, amortized free-bit complexity of 2 seems to be a natural barrier: any proof system in this framework must use $2 - \epsilon$ amortized free-bits, where $\epsilon > 0$ as usual can be arbitrarily small. The result, including a definition of what we mean by the "framework," is in Section 4.2. Loosely speaking, it considers proof systems which, among other things, probe two oracles in order to check that one oracle is "close" to a codeword (i.e., a codeword test) and the second oracle encodes a projection of the information encoded in the first oracle (i.e., a projection test). We also prove a lower bound of $1 - \epsilon$ on the amortized free-bit complexity of performing only the codeword test (resp., only the projection test). Our lower bound refers to a codeword test that is required to reject oracles which are at distance at least $d/2$ from the code, where d is the distance of the code. All three lower bounds are tight (by proof systems presented in this paper). A more relaxed definition of a codeword test only requires the test to reject oracles at distance more that $(1 - \epsilon) \cdot d$ from the code.[3] We do not know whether our lower bound (on the amortized free-bit complexity) holds also for the relaxed codeword test.
>
> All known constructions (of reasonably efficient pcp systems) fall into the framework discussed above (i.e., perform both a codeword test and a projection test). Furthermore,

[3] In contrast to the original definition, passing a relaxed codeword test does not guarantee unique decoding. However, as we see in Section 3.4, this does not matter.

a pcp system of amortized free-bit complexity $2+\epsilon$ (cf. Theorem 1.3.1 Part 1) can be constructed both by using the relaxed and non-relaxed forms of the codeword test. Thus improving on the amortized free-bit count of $2+\epsilon$ requires either departure from the abovementioned framework or the construction of a relaxed codeword test with amortized free-bit complexity significantly lower than 1. In the latter case (i.e., when remaining in the above framework), such a construction is necessary but not sufficient in order to obtain a pcp system for NP with free-bit complexity lower than 2 (since one needs to perform the projection test also in case the oracles are close but not equal to codewords). Furthermore, in such a case the lower bound of 1 on the free-bit complexity of the projection test still holds.

Meanwhile, Hastad [Has] has constructed a pcp system of amortized free-bit complexity $1+\epsilon$, $\forall\epsilon > 0$. Hastad's construction builds on the framework presented in the current work but introduces a (different type of a) relaxed codeword test which is conducted within amortized free-bit complexity ϵ. Thus, we currently consider our lower bound (of $1-\epsilon$) on the amortized free-bit complexity of the projection test to be the most important result of Section 4.2. The projection test seems inherent to the way pcp systems are currently constructed. We conclude that improving over Hastad's result (i.e., amortized free-bit complexity of $1+\epsilon$) would require some significant changes in the design of pcp verifiers. It follows from our reverse connection that proving a larger than $N^{1/2}$ non-approximability factor for Max Clique would also require significant new techniques.

We stress that these lower bounds point out limitations of techniques, not limitations of results. We are *not* saying there is any reason to disbelieve-believe the existence of, say, of a pcp verifier with amortized free-bit complexity of $\epsilon > 0$ for all NP languages, where $\epsilon > 0$ is an arbitrary constant. Indeed, if we believe Max Clique is hard to approximate within $N^{1-o(1)}$ then such verifiers exist! We are just saying they may be hard to find.

1.5 Properties and transforms of PCP and FPCP

The results mentioned in the first two subsections can be found in Section 5.1; whereas the results in the last subsection are from Section 5.2.

1.5.1 Triviality results

We begin our investigation of the roles of various parameters with triviality results. These results are directed at seeing what kinds of parameter combinations we can expect are too weak to recognize NP.

Perhaps the first thing to ask is whether, instead of amortized free-bit complexity, we could work with any of the simpler measures. After all $\overline{\mathrm{FPCP}}[\log, f]$ contains each of the following classes: (1) $\mathrm{PCP}_{1,1/2}[\log, f]$; (2) $\overline{\mathrm{PCP}}[\log, f]$; (3) $\mathrm{FPCP}_{1,1/2}[\log, f]$. Thus it would suffice to minimize the query complexity to get error 1/2; or the amortized query complexity; or the free-bit complexity to get error 1/2. However it turns out these complexities will not enable us to reach our target (of reducing the complexity to almost zero and thus proving that clique is hard to approximate to within a $N^{1-\epsilon}$ factor, for every $\epsilon > 0$). This is because of the following (where the first result is folklore and included here only for completeness).

Theorem 1.5.1 The following classes are all contained in P–

(1) $\mathrm{PCP}_{1,1/2}[\log, 2]$

(2) $\overline{\mathrm{PCP}}[\log, 1]$

(3) $\mathrm{FPCP}_{1,1/2}[\log, 1]$.

Thus we cannot expect to construct pcp systems for NP with query complexity 2; amortized query complexity 1; or free-bit complexity 1. However it is a feature of amortized free-bit complexity that so far it seems entirely possible that NP reduces to $\overline{\mathrm{FPCP}}[\log, f]$ with f an arbitrarily small constant. Indeed, if we believe (conjecture) that Max Clique is hard to approximate with $N^{1-\epsilon}$ for any $\epsilon > 0$ then such proof systems must exist, by virtue of Theorem 1.4.1 above. In fact, even if we do not believe that Max Clique is hard to approximate with $N^{1-\epsilon}$ for any $\epsilon > 0$, it turns out that the amortized free bit parameter will be too weak to capture the hardness of the clique function. In fact if Max Clique is hard to approximate to within N^{α}, then the best hardness result obtainable from the amortized query bit parameter would be of the form $N^{\frac{\alpha}{2-\alpha}}$. This is shown by invoking Corollary 5.1.9 which shows that the amortized query complexity parameter is always one larger than the amortized average free bit parameter (and we know that the amortized free bit parameter captures the hardness of Max Clique tightly).

1.5.2 Other results

We have already mentioned above (cf., Theorem 1.5.1) that strict limitations on various query parameters make PCP very weak. Actually, for every $s < 1$, $\mathrm{PCP}_{1,s}[\log, 2]$ and $\mathrm{FPCP}_{1,s}[\log, 1]$ collapse to P. This means that pcp systems with perfect completeness are very weak when restricted to either *two queries* or to *free-bit complexity one*. However, pcp systems with completeness error and the very same query (resp., free-bit) bounds are not so weak. In particular, it is well known that $\mathrm{NP} = \mathrm{PCP}_{c,s}[\log, 2]$ for some $0 < s < c < 1$ (e.g., by using the NP-hardness of approximating Max2SAT). We show that $\mathrm{NP} = \mathrm{FPCP}_{c,s}[\log, 1]$ for some $0 < s < c < 1$ (specifically, $c = \frac{1}{2}$ and $s = 0.885 \cdot c$). Furthermore, for some smaller $0 < s < c < 1$, the following holds

$$\mathrm{NP} = \mathrm{FPCP}_{c,s}[\log, 0] \tag{1.1}$$

(specifically, with $c = \frac{1}{4}$ and $s = 0.885 \cdot c$). We find the last assertion quite intriguing. It seems to indicate that one needs to be very careful when making conjectures regarding free-bit complexity. Furthermore, one has to be very careful also when making conjectures regarding amortized free-bit complexity; for example, the result $\mathrm{P} = \overline{\mathrm{PCP}}[\log, 1]$ holds also when one allows non-perfect completeness (in the definition of $\overline{\mathrm{PCP}}[\cdot, \cdot]$) as long as the gap is greater than 2^q per q queries, but an analogous result cannot hold for *two-sided error* amortized free-bit complexity (i.e., $\overline{\mathrm{FPCP}}[\cdot, \cdot]$).

Trying to understand the power of pcp systems with low free-bit complexity, we have waived the bound on the randomness complexity. Recall that in this case pcp systems are able to recognize non-deterministic exponential time (i.e., $\mathrm{NEXPT} = \mathrm{PCP}_{1,1/2}[\mathrm{poly}, \mathrm{poly}]$) [BFL]. Thus, it may be of interest to indicate that for every $s < 1$,

$$\mathrm{FPCP}_{1,s}[\mathrm{poly}, 0] \subseteq \mathrm{coNP} \tag{1.2}$$

$$\mathrm{FPCP}_{1,s}[\mathrm{poly}, 1] \subseteq \mathrm{PSPACE} \tag{1.3}$$

It seems that $\mathrm{FPCP}_{1,1/2}[\log, 0]$ is not contained in BPP, since Quadratic Non-Residuosity and Graph Non-Isomorphism belong to the former class. (Specifically, the interactive proofs of [GMR] and [GMW] can be viewed as a pcp system with polynomial randomness, query complexity 1 and free-bit complexity 0.) Thus, it seems that also the obvious observation $\mathrm{PCP}_{1,s}[\mathrm{poly}, 1] \subseteq \mathrm{AM}$ (for every $s < 1$, where AM stands for one round Arthur-Merlin games), would be hard to improve upon.

1.5.3 Transformations between proof systems

We provide various useful transformation of pcp systems. These transformations are analogous to transformations which can be applied to graphs with respect to the max-clique problem. In view of the relation (mentioned above), between FPCP and the clique promise problem, this analogy is hardly surprising.

One type of transformations amplify the gap (i.e., the ratio between completeness and soundness bounds) of the proof system while preserving its amortized free-bit complexity and incurring a relatively small additional cost in the randomness complexity. Specifically, using a randomized reduction we can transform $\mathrm{FPCP}_{1,\frac{1}{2}}[\log, f]$ into $\mathrm{FPCP}_{1,2^{-k}}[\log + k, k \cdot f]$. (This transformation is analogous to the well-known transformation of Berman and Schnitger [BeSc].) Alternatively, using a known deterministic amplification method based on [AKS, LPS] one can transform $\mathrm{FPCP}_{1,\frac{1}{2}}[\log, f]$ into $\mathrm{FPCP}_{1,2^{-k}}[\log + 2k, k \cdot f]$ (ignoring multiplicative factors of $1+\epsilon$ for arbitrarily small $\epsilon > 0$). (To the best of our knowledge this transformation has never appeared with a full proof.) Both alternatives are important ingredients in transforming pcp results into clique in-approximability results via the FGLSS method.

A second type of transformations are ones which move the location of the gap (or, equivalently, the completeness parameter). The gap itself is preserved by the transformation but moving it is related to changing the free-bit complexity (and thus the amortized free-bit complexity is not preserved). Moving the gap 'up' requires increasing the free-bit complexity, whereas moving the gap 'down' allows to decrease the free-bit complexity. For example, we randomly reduce $\mathrm{FPCP}_{c,s}[\log, f]$ to $\mathrm{FPCP}_{1,s\cdot\log}[\log, f + \log(1/c) + \log\log]$. On the other hand, for every $k \leq f$, we (deterministically) reduce $\mathrm{FPCP}_{c,s}[\log, f]$ into $\mathrm{FPCP}_{\frac{c}{2^k},\frac{s}{2^k}}[\log, f-k]$, provided that the original system has at least 2^k accepting configurations per each possible sequence of coin-tosses. (This condition is satisfied in many natural pcp systems, even for $k = f$.)

1.6 History

Early work in non-approximability includes that of Garey and Johnson [GJ1] showing that it is NP-hard to approximate the chromatic factor within a factor less than two. The indication of higher factors, and results for other problems, had to wait for the interactive proof approach.

Interactive proofs were introduced by Goldwasser, Micali and Rackoff [GMR] and Babai [Bab]. Ben-Or, Goldwasser, Kilian and Wigderson [BGKW] extended these ideas to define a notion of multi-prover interactive proofs. Fortnow, Rompel and Sipser [FRS] showed that the class, MIP, of languages possessing multi-prover interactive proofs equals the class of languages which have (using todays terms) probabilistically checkable proofs (of unrestricted, and thus polynomial, randomness and query complexity).

First indication to the power of interactive proof systems was given in [GMW], where it was shown that interactive proofs exist for Graph Non-Isomorphism (whereas this language is not known to be in NP). However, the real breakthrough came with the result of Lund, Fortnow, Karloff and Nisan [LFKN] who used algebraic methods for showing that all coNP languages (and actually, all languages in $\mathrm{P}^{\#P}$) have interactive proof systems. These techniques were used by Shamir [Sh] to show that IP = PSPACE.

A central result which enabled the approximation connection is that of Babai, Fortnow and Lund [BFL] who showed that the class MIP equals the class NEXP (i.e., languages recognizable in non-deterministic exponential time). The latter result has been "scaled-down" to the NP-level by two independent groups of researchers. Babai, Fortnow, Lund and Szegedy [BFLS] showed that if the input is encoded using a special error-correcting code (for which encoding and decoding can be performed in polynomial-time) then NP has transparent proof systems (i.e., it is possible to verify the correctness of the proof in poly-logarithmic time). Feige, Goldwasser, Lovász, Safra and Szegedy [FGLSS] showed that NP has probabilistically checkable proofs of poly-logarithmic randomness and query complexity; namely, $\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}[r,q]$, where $r(n) = q(n) = O(\log n \cdot \log\log n)$.

The breakthrough connection to approximation was made by Feige, Goldwasser, Lovász, Safra and Szegedy [FGLSS]. They have shown that $\mathrm{NP} \subseteq \mathrm{PCP}_{1,s}[r,q]$ implies that approximating the maximum clique in a $2^{r(n)+q(n)}$-vertices graph to within a $1/s(n)$ factor is infeasible (i.e., not doable in polynomial-time), provided that NP is not in $\mathrm{Dtime}(2^{O(r+q)})$. (Here n is the length of the input x to the pcp verifier.) Combined with the above-mentioned results, they have obtained the first in a sequence of strong non-approximability results for Max Clique: a non-approximability factor of $2^{\log^{1-\epsilon} N}$, $\forall\epsilon > 0$, assuming NP did not have quasi-polynomial time algorithms.

After the work of [FGLSS] the field took off in two major directions. One was to extend the interactive proof approach to apply also to other optimization problems. Direct reductions from proofs were used to show hardness of quadratic programming [BeRo, FeLo], Max-3-SAT [ALMSS], set cover [LuYa], and other problems [Be]. Also, reductions from Max Clique lead to hardness results for the chromatic number [LuYa] and other problems [Zu], while previous reductions from Max-3-SAT lead to hardness results for all of Max-SNP [PaYa].

The other direction was to increase factors and reduce assumptions for problems already shown hard to some factor under some assumption, by improving the efficiency of the underlying proof systems and/or the efficiency of the reductions.

The first stage of this enterprise started with the work of Arora and Safra [ArSa] which, showing that $\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}[\log, o(\log)]$, provided the first strong NP-hardness result for Max Clique (specifically, a hardness factor of $2^{\sqrt{\log N}}$). This work has introduced the idea of recursive proof checking which turned out to play a fundamental role in all subsequent developments. Interestingly, the idea of encoding inputs in an error-correcting form (as suggested in [BFLS]) is essential to make "recursion" work. Arora, Lund, Motwani, Sudan and Szegedy [ALMSS], have reduced the query complexity of pcp systems for NP to a constant, while preserving the logarithmic randomness complexity; namely, they have shown that $\mathrm{NP} = \mathrm{PCP}_{1,1/2}[\log, O(1)]$. This immediately implied the NP-hardness of approximating Max Clique within N^{ϵ}, for some $\epsilon > 0$. Furthermore, it also implied that Max-3-Sat is NP-hard to approximate to within some constant factor [ALMSS] and so is the entire class Max-SNP [PaYa].

Attempts to improve the constant in the exponent of the Max Clique hardness factor, and also improve the constant values of the hardness factors in the Max-SNP hardness results, begin with Bellare, Goldwasser, Lund and Russell [BGLR]. They presented new proof systems minimizing query complexity and exploited a slightly improved version of the FGLSS-reduction due to [BeSc, Zu] to get a $N^{1/30}$ hardness of approximation factor for Max Clique. Feige and Kilian [FeKi], however, observed that one should work with free-bits, and noted that the free-bit complexity of the system of [BGLR] was 14, yielding a $N^{1/15}$ hardness factor. Bellare and Sudan then suggested the notion of amortized free-bits and built new proof systems achieving amortized free-bit complexity three, and in particular a $N^{1/4}$ hardness for Max Clique assuming $\mathrm{NP} \not\subseteq \mathrm{co}\widetilde{\mathrm{RP}}$.

Detailed histories for specific topics are given in the sections addressing this topic. In particular see Section 2.4 for history of PCP and its growing list of parameters; Section 3.4 for a perspective of the role of constant prover proofs; Section 3.10 for previous work in query complexity minimization; Section 3.7 for previous work, both on approximation algorithms and hardness results, for Max-3-SAT and Max-2-SAT; Section 3.12 for previous work on Max Clique and history of various chromatic number reductions.

1.7 Related work

Following the presentation of our results, Arora has also investigated the limitations of proof checking techniques in proving non-approximability results [Ar]. Like in our free-bit lower bound result, he tries to assess the limitations of current techniques by making some assumptions about these techniques and then showing a lower bound. His focus is on the reductions, which he assumes are "code like." In this setting he can show that one should not expect to prove non-approximability of Max Clique within $N^{1/2}$. (The assumptions made by us and by Arora do not seem to be comparable: neither implies the other.)

1.8 Directions for further research

Following the publication of an early version of this work and building on its techniques, Hastad [Has] has constructed pcp systems for NP with amortized free-bit complexity $1+\epsilon$, $\forall \epsilon > 0$. This has resolved our previous challenge of reducing the amortized free-bit complexity to below 2, but left open the more difficult challenge of reducing the amortized free-bit complexity to below 1. Section 4.2 demonstrates that this cannot be done by using the current paradigms for constructing pcp systems, and specifically while using a Projection Test. We conjecture that, for every $\epsilon > 0$, $\mathrm{NP} \subseteq \overline{\mathrm{FPCP}}[\log, \epsilon]$ and challenge the reader to prove or refute this conjecture.

Two questions of a *de-randomization* flavor follow. As stated above, we know that $\overline{\mathrm{FPCP}}[\log, f]$ is *randomly* reducible to $\mathrm{FPCP}_{1,2^{-k}}[\log + k, k \cdot f]$. On the other hand, the former class is contained in (i.e., is *deterministically* reduced to) the class $\mathrm{FPCP}_{1,2^{-k}}[\log + (2+\epsilon)k, (1+\epsilon)k \cdot f]$, for arbitrarily small $\epsilon > 0$. Can one obtain the best of both worlds; namely, a deterministic reduction of $\overline{\mathrm{FPCP}}[\log, f]$ to, say, $\mathrm{FPCP}_{1,2^{-k}}[\log + (1+\epsilon)k, (1+\epsilon)k \cdot f]$, for arbitrarily small $\epsilon > 0$. An affirmative answer will allow to infer from $\mathrm{NP} \subseteq \overline{\mathrm{FPCP}}[\log, f]$ that approximating Max Clique to within an $N^{\frac{1}{1+f+\epsilon}}$ factor is NP-hard (rather than infeasible under the assumption that NP is not contained in BPP).

One ingredient of our method for reversing the FGLSS-reduction is the randomized reduction of the class $\mathrm{FPCP}_{c,s}[\log, f]$ to the class $\mathrm{FPCP}_{1,\frac{\log}{c}\cdot s}[\log, f + \log(1/c) + \log\log]$. (This statement follows the exposition in Section 5.2. An alternative exposition, making use of a randomized graph-layering process, is given in Section 4.1.) Anyhow, randomness plays an essential role in obtaining a pcp system with perfect completeness.[4] The question is whether the class $\mathrm{FPCP}_{c,s}[\log, f]$ is contained in the class $\mathrm{FPCP}_{1,\frac{\log}{c}\cdot s}[\log, f + \log(1/c) + \log\log]$ (rather than being randomly reducible to it).

Our NP-hardness (of approximation) results for MaxSNP make use of problem-dependent gadgets which implement two simple tests (i.e., testing that $x + y = z$ and testing that $x = y_z$, for variables/oracle-answers x, y, y_0, y_1 and z). For example, when proving Max3SAT we construct 3CNF formulae, over these and auxiliary variables, so that the formula is satisfied if and only if the basic variables satisfy the test. Specifically, the formula for each test has 4 clauses (and no auxiliary variables). In general, what matters is the relation between the number of clauses satisfied by the best assignment extending values which satisfy the test and the number of clauses satisfied by the best assignment extending values which do not satisfy the test. Let α_i (resp., $\alpha_i - \beta_i < \alpha_i$) denote the first (resp., second) number, for the i^{th} test, and let $\rho_i = \frac{\alpha_i}{\beta_i}$. Then, the non-approximability factor has the form $\frac{1}{c_1\rho_1 + c_2\rho_2}$, where c_1 and c_2 depend on the proof system. Thus, constructing 3CNF (resp., 2CNF) formulae for which the ratios ρ_i are small is a key ingredient in getting better non-approximability results. Currently, for 3CNF we have $\rho_1 = \rho_2 = 4$, whereas for 2CNF we have

[4] This makes our results more elegant, but actually – as indicated in Section 4.1, we could have settled for "almost perfect" completeness which suffices for presenting an inverse of the "FGLSS-reduction".

$\rho_1 = \rho_2 = 11$. Defining analogous quantities for Max Cut, we currently have $\rho_1 = 9$ and $\rho_2 = 11$. (For MinVC we could obtain $\rho_1 = 6$ and $\rho_2 = 7$, but used an alternative method instead – see Section 3.9). We suggest the construction of better gadgets as an open problem.

Regarding (non-amortized) free-bits, we know that $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,0.7936}[\log, 2]$ and on the other hand that $\mathrm{FPCP}_{1,s}[\log, 1] \subseteq \mathrm{P}$, for every $s < 1$. As motivation to the following questions we note that the first result was used to establish the NP-hardness of approximating Min Vertex Cover upto a $\frac{16}{15}$ factor. In general, $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,s}[\log, f]$ implies that approximating Min Vertex Cover up to a $\frac{2^f - s}{2^f - 1}$ factor is NP-hard. We ask whether

(1) $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,s}[\log, 2]$ for every $s > 0$ (this would imply a hardness factor of $\frac{4}{3} - \epsilon$, $\forall \epsilon > 0$).

(2) $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,s}[\log, \log_2 3]$ for every $s > 0$ (this would imply a hardness factor of $\frac{3}{2} - \epsilon$, $\forall \epsilon > 0$).

Note that obtaining a result for $s < 2^{-f}$, where f is the free bit complexity, would imply amortized free-bit complexity lower than 1. Thus, it may be easier to try to obtain soundness bounds of $s \approx \frac{1}{4}$ and $s \approx \frac{1}{3}$, respectively (yielding non-approximation factors of $\frac{5}{4}$ and $\frac{4}{3}$, resp.).

1.9 Acknowledgments

We thank Viggo Kann, Marcos Kiwi and Luca Trevisan for carefully reading the previous version of our work and pointing out several flaws and improvements. We also wish to thank Uri Feige for helpful discussions.

CHAPTER 2

Notation and Definitions

2.1 General notation and definitions

For integer n let $[n] = \{1, \ldots, n\}$. A graph always means an undirected graph with no self-loops, unless otherwise indicated. We let $\|G\|$ denote the number of vertices in graph $G = (V, E)$.

A probabilistic machine K has one or more inputs $x_1, x_2, \ldots$ and tosses some random coins R, usually of some length $r(\cdot)$ which is a function of the (lengths of the) inputs. We let $K(x_1, x_2, \ldots; R)$ denote the output of K when it uses the particular sequence of coin tosses given by R. Typically we are interested in the probability space associated to a random choice of R.

A function is *admissible* if it is polynomially bounded and polynomial time computable. We will ask that all functions measuring complexity (e.g. the query complexity $q = q(n)$) be admissible.

In defining complexity classes we will consider promise problems rather than languages.[1] Following Even et. al. [ESY], a promise problem is a pair of disjoint sets (A, B), the first being the set of "positive" instances and the second the set of "negative" instances. A language L is identified with $(L, \overline{L})$. (We refer the reader to [ESY] for issues in promise problems.)

2.2 Proof systems

A verifier is a probabilistic machine V taking one or more inputs and also allowed access to one or more oracles. Let x denote the sequence of all inputs to V and let n denote its length. During the course of its computation on coins R and input x it makes queries of its oracles. Its final decision to accept or reject is a function $\mathsf{DEC}_V(x, a; R)$ of x, R and the sequence a of all the bits obtained from the oracle in the computation. Contrary to standard terminology, acceptance in this paper will correspond to outputting 0 and rejection to outputting 1.

Oracles are formally functions, with the context specifying for each the domain and range; sometimes, however, we may write strings, to be interpreted as functions in the natural way. Let π denote the sequence (tuple) of all proof oracles supplied to the verifier V. Now for verifier V examining the proofs π and having input x, we let

$$\mathtt{ACC}[\, V^\pi(x) \,] \;=\; \Pr_R\,[V^\pi(x; R) = 0]$$

[1] This convention is adopted since approximation problems are easily casted as promise problems.

denote the probability that V accepts in this particular case. We then let

$$\mathtt{ACC}[\,V(x)\,] \;=\; \max_{\pi}\ \mathtt{ACC}[\,V^{\pi}(x)\,]$$

denote the maximum accepting probability, over all possible choices of proof sequences π; the domain from which the proofs are chosen depending, as mentioned above, on the context.

Let $\mathsf{pattern}_V(x;R)$ be the set of all sequences a such that $\mathsf{DEC}_V(x,a;R) = 0$. (That is, all sequences of oracle answers leading to acceptance). A *generator* for V is a poly(n)-time computable function G such that $\mathsf{pattern}_V(x;R) = G(x,R)$ for all x, R. (That is, it can efficiently generate the set of accepted patterns.)

We are interested in a host of parameters which capture various complexity measures of the proof checking process. They are all functions of the length n of the input x given to the verifier V. In the following σ denotes the concatenation of all the proof strings given to the verifier. Also recall we are interested in proof systems for promise problems (A,B) rather than just for languages.

coins = Number of coins tossed by verifier. Typically denoted r

pflen = Length of the proof provided to the verifier. Typically denoted l.

c = Completeness probability. Namely $\min\{\,\mathtt{ACC}[\,V(x)\,] : x \in A \text{ and } |x| = n\,\}$.

s = Soundness probability. Namely $\max\{\,\mathtt{ACC}[\,V(x)\,] : x \in B \text{ and } |x| = n\,\}$.

g = Gap. Namely c/s.

Now we move to various measures of the "information" conveyed by the oracle to the verifier. For simplicity we consider here only oracles which return a single bit; that is, they correspond to "written" proofs.

query = The query complexity on input x is the maximum, over all possible coin tosses R of V, of the number of bits of σ accessed by V on input x. The query complexity of the system $q = q(n)$ is the maximum of this over all inputs $x \in A \cup B$ of length n.

$\mathsf{query}_{\rm av}$ = The average query bit complexity on input x is the average, over R, of the number of bits of the proof σ accessed by V on input x and coins R. The average query complexity of the system is the maximum of this over all $x \in A \cup B$ of length n. Typically denoted $q_{\rm av}$.

$\overline{\mathsf{query}}$ = V is said to have amortized query bit complexity $\bar{q}$ if $q/\lg(g) \le \bar{q}$ where q is the query bit complexity and g is the gap, and, furthermore, q is at most logarithmic in n.

free = The free bit complexity of V is f if there is a generator G such that $|G(x,R)| \le 2^f$ for all R and all $x \in A \cup B$ of length n.

$\mathsf{free}_{\rm av}$ = The average free bit complexity of V is $f_{\rm av}$ if there is a generator G such that $\mathbf{E}_R\left[|G(x,R)|\right] \le 2^{f_{\rm av}}$ for all $x \in A \cup B$ of length n.

$\overline{\mathsf{free}}$ = V is said to have amortized free bit complexity $\bar{f}$ if $f/\lg(g) \le \bar{f}$ where f is the free bit complexity and g is the gap.

Notice that amortized query complexity is restricted to be at most logarithmic. We don't need to explicitly make this restriction for the amortized free bit complexity: it is a consequence of the efficient generation condition.

In case the completeness parameter equals 1 (i.e., $c = 1$), we say that the system is of **perfect completeness.** In case the completeness parameter, c, satisfies $c(n) = 1 - o(1)$, we say that the system is of **almost-perfect completeness.**

The consideration of all these parameters give rise to a potentially vast number of different complexity classes. We will use a generic notation in which the parameter values are specified by name, except that, optionally, the completeness and soundness can, if they appear, do so as subscripts. Thus for example we have things like:

$$\mathrm{PCP}_{c,s}[\,\mathsf{coins} = r\ ;\ \mathsf{query} = q\ ;\ \mathsf{pflen} = 2^r\ ;\ \mathsf{free} = f\ldots]\ .$$

However most often we'll work with the following abbreviations:

$$\begin{aligned}
\mathrm{PCP}_{c,s}[r,q] &\stackrel{\mathrm{def}}{=} \mathrm{PCP}_{c,s}[\,\mathsf{coins} = r\ ;\ \mathsf{query} = q\,]\\
\overline{\mathrm{PCP}}_c[\,r,\,q\,] &\stackrel{\mathrm{def}}{=} \mathrm{PCP}_{c,\cdot}[\,\mathsf{coins} = r\ ;\ \overline{\mathsf{query}} = q\,]\\
\mathrm{FPCP}_{c,s}[r,f] &\stackrel{\mathrm{def}}{=} \mathrm{PCP}_{c,s}[\,\mathsf{coins} = r\ ;\ \mathsf{free} = f\,]\\
\mathrm{FPCP}_{c,s}[r,f,l] &\stackrel{\mathrm{def}}{=} \mathrm{PCP}_{c,s}[\,\mathsf{coins} = r\ ;\ \mathsf{free} = f\ ;\ \mathsf{pflen} = l\,]\\
\overline{\mathrm{FPCP}}_c[\,r,\,f\,] &\stackrel{\mathrm{def}}{=} \mathrm{PCP}_{c,\cdot}[\,\mathsf{coins} = r\ ;\ \overline{\mathsf{free}} = f\,]\ .
\end{aligned}$$

We stress that in the definitions of the amortized classes, $\overline{\mathrm{PCP}}_c[\,r,\,q\,]$ and $\overline{\mathrm{FPCP}}_c[\,r,\,f\,]$, we refer to the completeness parameter c (but not to the soundness parameter). In case $c = 1$, we may omit this parameter and shorthand the amortized classes of perfect completeness by $\overline{\mathrm{PCP}}[\,r,\,q\,]$ and $\overline{\mathrm{FPCP}}[\,r,\,f\,]$, respectively. Namely,

$$\begin{aligned}
\overline{\mathrm{PCP}}[\,r,\,q\,] &\stackrel{\mathrm{def}}{=} \overline{\mathrm{PCP}}_1[\,r,\,q\,]\\
\overline{\mathrm{FPCP}}[\,r,\,f\,] &\stackrel{\mathrm{def}}{=} \overline{\mathrm{FPCP}}_1[\,r,\,f\,]
\end{aligned}$$

2.3 Randomized reductions

We will consider reductions between promise problems. A (randomized) Karp reduction from (A_1, B_1) to (A_2, B_2) is a probabilistic, polynomial time function T which takes two arguments: an input x and a security parameter k, the latter written in unary. The transformation is required to have the property that

$$\begin{aligned}
x \in A_1 &\implies \Pr\left[\,T(x, 1^k) \in A_2\,\right] \stackrel{\mathrm{def}}{=} p_1(x,k) \geq 1 - 2^{-k}\\
x \in B_1 &\implies \Pr\left[\,T(x, 1^k) \in B_2\,\right] \stackrel{\mathrm{def}}{=} p_2(x,k) \geq 1 - 2^{-k}\ .
\end{aligned}$$

The probability is over the coin tosses of T. We say the reduction has perfect completeness if $p_1 = 1$ and perfect soundness if $p_2 = 1$. (In the special case of deterministic transformations it must be that $p_1 = p_2 = 1$.) We write $(A_1, B_1) \leq_R^K (A_2, B_2)$ if there is a randomized Karp reduction from (A_1, B_1) to (A_2, B_2). If the reduction is deterministic we omit the subscript of "R," or, sometimes, for emphasis, replace it by a subscript of "D."

An example is the randomized FGLSS transformation [FGLSS, BeSc, Zu]. Here (A_1, B_1) is typically an NP-complete language L, and (A_2, B_2) is $\mathsf{Gap\text{-}Clique}_{c,s}$ for some c, s which are determined by the transformation. This transformation has perfect soundness, while, on the other hand, it is possible to get $p_1 = 1 - 2^{-\mathrm{poly}(n)}$.

Similarly one can define (randomized) Cook reductions. The notation for reductions is $\leq_R^C$.

Let C be a complexity class (e.g. NP). We say that C reduces to (A_2, B_2) if for every (A_1, B_1) in C it is the case that (A_1, B_1) reduces to (A_2, B_2). An example is to say that NP reduces to Gap-Clique$_{c,s}$. We say that C_1 reduces to C_2, where C_1 and C_2 are complexity classes, if for every (A_1, B_1) in C_1 there is an (A_2, B_2) in C_2 such that (A_1, B_1) reduces to (A_2, B_2). An example is to say that NP reduces to $\overline{\mathrm{FPCP}}[\log, f]$. The notation of $\leq_R^K$ or $\leq_R^C$ extends to these cases as well.

Notice that our definition of reducibility ensures that this relation is transitive.

For simplicity we sometimes view a reduction T as a function only of x, and write $T(x)$. In such a case it is to be understood that the security parameters has been set to some convenient value, such as $k = 2$.

2.4 History

The model underlying what are now known as "probabilistically checkable proofs" is the "oracle" model of Fortnow, Rompel and Sipser [FRS], introduced as an equivalent (with respect to language recognition power) version of the multi-prover model of Ben-Or, Goldwasser, Kilian and Wigderson [BGKW]. Interestingly, as shown by [BFLS, FGLSS], this framework can be applied in a meaningful manner also to languages in NP. These works provide the verifier V with a "written" proof, modeled as an oracle to which V provides the "address" of a bit position in the proof string and is returned the corresponding bit of the proof. Babai et. al. [BFLS] suggested a model in which the instances are encoded in a special (polynomial-time computable and decodable) error-correcting code and the verifier works in polylogarithmic time. Here we follow the model of Feige et. al. [FGLSS] where the verifier is probabilistic polynomial-time (as usual) and one considers finer complexity measures such as the query and randomness complexity. The reduction of [FGLSS] identified the parameters of query complexity (number of binary queries), randomness complexity and error. The class $\mathrm{PCP}_{1,1/2}[r, q]$ was made explicit by [ArSa].

The parameterization was expanded by [BGLR] to explicitly consider the answer size (the oracle is now allowed to return more than one bit at a time) and query size– their notation included five parameters: randomness, number of queries, size of each query, size of each answer, and error probability. They also similarly parameterized (single round) multi-prover proofs, drawing attention to the analogue with pcp. This served to focus attention on the roles of various parameters, both in reductions and in constructions. Also they introduced the consideration of average query complexity, the first in a sequence of parameter changes towards doing better for clique.

Free bits are implicit in [FeKi] and formalized in [BeSu]. Amortized free bits are introduced in [BeSu] but formalized a little better here.

Proof sizes were considered in [BFLS, PoSp]. We consider them here for a different reason– they play an important role in that the randomized FGLSS reduction [BeSc, Zu] depends actually on this rather than the randomness.

To deal with the now huge array of parameters we have generalized the notation of [BGLR] to allow specification of parameters by name.

We've followed the common tradition regarding the names of polynomial-time reductions: many-to-one reductions are called Karp-reductions whereas (polynomial-time) Turing reductions are called Cook-reductions. This terminology is somewhat unfair towards Levin whose work on NP-completeness [Lev] was independent of those of Cook [Co] and Karp [Ka]. Actually, the reductions considered by Levin are more restricted as they also efficiently transform the corresponding NP-witnesses (this is an artifact of Levin's desire to treat search problems rather than decision problem). In fact, such reductions (not surprisingly termed Levin-reductions) are essential for results such as

Corollary 4.1.12. (Yet, this is the only example in the current paper.)

CHAPTER 3

New proof systems and non-approximability results

This chapter presents some new proof systems minimizing complexity under various measures. These proof systems are then used to derive the best known in-approximability results for Max-3-SAT, Max-E3-SAT (Max Exact 3SAT), Max-2-SAT, Max Cut, Min Vertex Cover (Min VC), Max Clique, and Chromatic number. This is a long chapter and it will help to begin with some indication of what we will be doing.

3.1 Overview and guidemap

The starting point for all our proof systems is a two-prover proof system achieving arbitrarily small but fixed constant error with logarithmic randomness and constant answer size, as provided by Raz [Raz]. This proof system has the property that the answer of the second prover is supposed to be a predetermined function of the answer of the first prover. Thus, verification in it amounts to checking that the first answer satisfies some predicate and that the second answer equals the value obtained from the first answer. Following the "proof composition" paradigm of Arora and Safra [ArSa], we will "encode" the answers of the two provers under a suitable code and then, "recursively", check these encodings. As usual, we will check both that these encodings are valid and that they correspond to answer which would have been accepted by the original verifier.

Our main technical contribution is a new code, called the *long code*, and means to check it. The long code of an n-bit information word a is the sequence of 2^{2^n} bits consisting of the values of all possible boolean functions at a. The long code is certainly a disaster in terms of coding theory, but it has big advantages in the context of proof verification, arising from the fact that it carries enormous amounts of data about a. The difficulty will be to check that a prover claiming to write the long code of some string a is really doing so.

The long code is described in Section 3.3. In Section 3.5 we provide what we call the "atomic" tests for this code. These tests and their analysis are instrumental to all that follows. Section 3.4 is also instrumental to all that follows. This section sets up the framework for recursive proof checking which is used in all the later proof systems.

The atomic tests are exploited in Section 3.6, introducing a verifier which queries the proof at 3 locations and performs one of two simple checks on the answers obtained. These simple checks are implemented by gadgets of the MaxSNP problem at hand, yielding the non-approximability results. Section 3.7 presents gadgets which are CNF formulae of the corresponding type and Section 3.8 presents Max-CUT gadgets. The non-approximability results for Max-3-SAT, Max-E3-SAT, Max-2-SAT and Max-CUT follow. The verifier of Section 3.6 benefits from another novel idea which is referred to as *folding* (see Section 3.2). We stress that folding contributes to the improved results for Max-3-SAT, Max-E3-SAT, Max-2-SAT and Max-CUT, but not to the results regarding Max Clique (and Chromatic Number).

A reasonable non-approximability result for Min-VC (minimum Vertex Cover) can be obtained by the above procedure, but a better result is obtained by constructing a different verifier which uses exactly two-free bits. The computation of this verifier is then reduced to the vertex cover problem (by means of the FGLSS reduction). The latter approach is presented in Section 3.9 where we try to minimizing the soundness error attainable using exactly two free-bits.

In Section 3.10 we minimize the number of bits queried in a PCP to attain soundness error 1/2 — the result is not of direct applicability, but it is intruiging to know how low this number can go.

We then turn to Max Clique (and Chromatic Number). In Section 3.11 we provide the "iterated" tests (in which the atomic tests are sequentially invoked many times). These iterations will be related to one another (pairwise independent to be more specific) leading to a proof system in which the number of *amortized free-bits* used is two. We then draw the implications for Max Clique (and Chromatic Number). A reader interested only in the (amortized) free-bit and Max Clique results can proceed directly from Section 3.5 to Section 3.11 and Section 3.12.

The improvement in the complexities of the proof systems is the main source of our improved non-approximability results. In addition we also use (for the Max-SAT and Max-CUT problems) a recent improvement in the analysis of linearity testing [BCHKS] and introduce special (problem specific) gadgets which represent the various tests.

Credits and histories pertaining to each topic are discussed alongside the topic. Thus each subsection contains the historical material relevant to it.

3.2 Preliminaries

In this chapter, $\Sigma = \{0,1\}$ will be identified with the finite field of two elements, the field operations being addition and multiplication modulo two. If X and Y are sets then $\mathsf{Map}(X,Y)$ denotes the set of all maps of X to Y. For any m we regard Σ^m as a vector space over Σ, so that strings and vectors are identified.

Linearity. Let G, H be groups. A map $f\colon G \to H$ is *linear* if $f(x+y) = f(x) + f(y)$ for all $x, y \in G$. Let $\textsc{Lin}(G,H)$ denote the set of all linear maps of G to H.

Distance. The **distance** between functions f_1, f_2 defined over a common finite domain D is

$$\mathrm{Dist}(f_1,f_2) \;=\; \Pr_{x \overset{R}{\leftarrow} D}\left[f_1(x) \neq f_2(x)\right].$$

Functions f_1, f_2 are **ϵ-close** if $\mathrm{Dist}(f_1,f_2) < \epsilon$. If f maps a group G to a group H we denote by $\mathrm{Dist}(f,\textsc{Lin})$ the minimum, over all $g \in \textsc{Lin}(G,H)$, of $\mathrm{Dist}(f,g)$. (Note the notation does not specify G, H which will be evident from the context). We are mostly concerned with the case of G being a vector space V over Σ and H being Σ. Notice that in this case we have $\mathrm{Dist}(f,\textsc{Lin}) \le 1/2$ for all $f\colon V \to \Sigma$.

Boolean Functions. Let l be an integer. We let $\mathcal{F}_l \overset{\mathrm{def}}{=} \mathsf{Map}(\Sigma^l,\Sigma)$ be the set of all maps of Σ^l to Σ. We regard $\mathcal{F}_l$ as a vector space (of dimension 2^l) over Σ. Addition and multiplication of functions are defined in the natural way.

We let $\mathcal{L}_m \subseteq \mathcal{F}_m$ be the set $\text{Lin}(\Sigma^m, \Sigma)$ of linear functions of Σ^m to Σ, and let $\mathcal{L}_m^* = \mathcal{L}_m - \{0\}$ be the non-zero linear functions.

Let $g \in \mathcal{F}_m$ and $\vec{f} = (f_1, \ldots, f_m) \in \mathcal{F}_l^m$. Then $g \circ \vec{f}$ denotes the function in $\mathcal{F}_l$ which assigns the value $g(f_1(x), \ldots, f_m(x))$ to $x \in \Sigma^l$.

If $a \in \Sigma^m$ then $a^{(i)}$ denotes its i-th bit. Similarly, if f is any function with range Σ^m then $f^{(i)}$ denotes the i-th bit of its output.

The Monomial Basis. For each $S \subseteq [l]$ we let $\chi_S \in \mathcal{F}_l$ be the monomial corresponding to S, defined for $x \in \Sigma^l$ by

$$\chi_S(x) \;=\; \prod_{i \in S} x^{(i)} \,.$$

The empty monomial, namely $\chi_\emptyset$, is defined to be the constant-one function (i.e., $\chi_\emptyset(x) \;=\; \bar{1}$, for all $x \in \Sigma^l$). The functions $\{\chi_S\}_{S \subseteq [l]}$ form a basis for the vector space $\mathcal{F}_l$ which we call the **monomial basis**. This means that for each $f \in \mathcal{F}_l$, there exists a unique vector $\mathcal{C}(f) = (\, C_f(S)\,)_{S \subseteq [l]} \in \Sigma^{2^l}$ such that

$$f \;=\; \sum_{S \subseteq [l]} C_f(S) \cdot \chi_S \,. \tag{3.1}$$

The expression of Equation (3.1) is called the *monomial series* for f, and the members of $\mathcal{C}(f)$ are called the *coefficients of* f with respect to the monomial basis. We note that $\mathcal{C}\colon \mathcal{F}_l \to \Sigma^{2^l}$ is a bijection.

Folding. Fix $\prec$ to be some canonical, polynomial time computable total order (reflexive, anti-symmetric, transitive) on the set $\mathcal{F}_l$. Given functions $A\colon \mathcal{F}_l \to \Sigma$ and $h \in \mathcal{F}_l \setminus \{0\}$ (i.e., h is not the constant function 0) and bit $b \in \Sigma$, the (h,b)-**folding** of A is the function $A_{(h,b)}\colon \mathcal{F}_l \to \Sigma$ given by

$$A_{(h,b)}(f) = \begin{cases} A(f) & \text{if } f \prec h + f \\ A(f+h) - b & \text{otherwise.} \end{cases}$$

(Notice that the above is well-defined for any $h \neq 0$.) For sake of technical simplicity (see Definition 3.4.3), we define the $(0,0)$-folding of A to be A itself; namely, $A_{(0,0)}(f) = A(f)$, for every $f \in \mathcal{F}_l$. As shown below, the (h,b)-folding of a function A is forced to satisfy $A_{(h,b)}(f+h) = A_{(h,b)}(f) + b$, for every $f \in \mathcal{F}_l$ (whereas A itself may not necessarily satisfy these equalities). Before proving this, let us generalize the notion of folding to folding over several, specifically two, functions $h_1, h_2 \in \mathcal{F}_l$ (and bits $b_1, b_2 \in \Sigma$).

Definition 3.2.1 (folding): Let $f, h_1, h_2 \in \mathcal{F}_l$. The (h_1,h_2)-**span of** f, denoted $\text{span}_{h_1,h_2}(f)$, is defined as the set $\{f + \sigma_1 h_1 + \sigma_2 h_2 : \sigma_1, \sigma_2 \in \Sigma\}$. Let $A\colon \mathcal{F}_l \to \Sigma$, $h_1 \neq h_2 \in \mathcal{F}_l \setminus \{0\}$ and $b_1, b_2 \in \Sigma$. The **folding** of A **over** (h_1,b_1) **and** (h_2,b_2), denoted $A_{(h_1,b_1),(h_2,b_2)}$, is defined for every $f \in \mathcal{F}_l$ by

$$A_{(h_1,b_1),(h_2,b_2)}(f) = A(f + \sigma_1 h_1 + \sigma_2 h_2) - \sigma_1 b_1 - \sigma_2 b_2$$

where $\sigma_1, \sigma_2 \in \Sigma$ so that the function $f + \sigma_1 h_1 + \sigma_2 h_2$ is the smallest function (according to $\prec$) in $\text{span}_{h_1,h_2}(f)$.

The definition extends naturally to the the following two case. In case $(h_1,b_1) = (h_2,b_2)$, folding over the two (identical) pairs is defined as folding over one pair. In case $h_1 \equiv 0$ and $b_1 = 0$, folding over both (h_1,b_1) and (h_2,b_2) is defined as folding over (h_2,b_2). Note that folding over two pairs is invariant under the order between the pairs; namely, $A_{(h_1,b_1),(h_2,b_2)} \equiv A_{(h_2,b_2),(h_1,b_1)}$. Finally, observe that a function $A\colon \mathcal{F}_l \to \Sigma$ that is folded over two functions (i.e., over both (h_1,b_1) and (h_2,b_2)) is folded over each of them (i.e., over each (h_i,b_i)).

Proposition 3.2.2 (folding forces equalities): Let $A: \mathcal{F}_l \to \Sigma$, $h_1, h_2 \in \mathcal{F}_l$ and $b_1, b_2 \in \Sigma$ (with $b_i = 0$ in case $h_i \equiv 0$). Then, for every $f \in \mathcal{F}_l$,

$$A_{(h_1,b_1),(h_2,b_2)}(f + h_1) = A_{(h_1,b_1),(h_2,b_2)}(f) + b_1$$

Proof: By definition, $A_{(h_1,b_1),(h_2,b_2)}(f) = A(f + \sigma_1 h_1 + \sigma_2 h_2) - \sigma_1 b_1 - \sigma_2 b_2$, where the function $f + \sigma_1 h_1 + \sigma_2 h_2$ is the smallest function in $\text{SPAN}_{h_1,h_2}(f)$. Since $\text{SPAN}_{h_1,h_2}(f + h_1) = \text{SPAN}_{h_1,h_2}(f)$, we have $A_{(h_1,b_1),(h_2,b_2)}(f + h_1) = A(f + \sigma_1 h_1 + \sigma_2 h_2) - (\sigma_1 - 1)b_1 - \sigma_2 b_2$. The claim follows. ▮

It may be instructive to hint that the verifiers constructed below make virtual access to folded functions rather to the function themselves. Virtual access to a folding of A is implemented by actual accessing A itself according to the definition of folding (e.g., say one wants to access $A_{(h,0)}$ at f then one determines whether $f \prec h+f$ or not and accesses either $A(f)$ or $A(f+h)$, accordingly). One benefit of folding in our context is illustrated by Proposition 3.3.3; in case a (h, b)-folded function is close to a codeword (in the long code), we infer that the codeword encodes a string a satisfying $h(a) = b$. We will see that folding (the long code) over $(h, 0)$ allows us to get rid of a standard ingredient in proof verification; the so-called "circuit test".

In the sequel, we will use folding over the pairs $(h, 0)$ and $(\bar{1}, 1)$, where $h \in \mathcal{F}_l$ is an arbitrary function (typically not identically zero) and $\bar{1}$ is the constant-one function. Folding over $(\bar{1}, 1)$ allows us to simplify the "codeword" test (w.r.t. the long-code).

3.3 Evaluation operators and the long code

Let $a \in \Sigma^l$. We define the map $E_a: \mathcal{F}_l \to \Sigma$ by $E_a(f) = f(a)$ for all $f \in \mathcal{F}_l$. We say that a map $A: \mathcal{F}_l \to \Sigma$ is an *evaluation operator* if there exists some $a \in \Sigma^l$ such that $A = E_a$. We now provide a useful characterization of evaluation operators. First we need a definition.

Definition 3.3.1 (respecting the monomial basis): A map $A: \mathcal{F}_l \to \Sigma$ is said to respect the monomial basis if $A(\chi_\emptyset) = 1$ and

$$\forall\ S, T \subseteq [l] \ : \ A(\chi_S) \cdot A(\chi_T) = A(\chi_{S \cup T}) .$$

Proposition 3.3.2 (characterization of the evaluation operator): A map $\tilde{A}: \mathcal{F}_l \to \Sigma$ is an evaluation operator if and only if it is linear and respects the monomial basis.

Proof: Let $a \in \Sigma^l$. It is easy to see that E_a is linear: $E_a(f+g) = (f+g)(a) = f(a)+g(a) = E_a(f)+E_a(g)$. It is also easy to see E_a respects the monomial basis. Firstly we have $E_a(\chi_\emptyset) = \chi_\emptyset(a) = 1$. Next, for every $S, T \subseteq [l]$,

$$E_a(\chi_S) \cdot E_a(\chi_T) \ = \ \chi_S(a) \cdot \chi_T(a) \ = \ \textstyle\prod_{i \in S} a^{(i)} \cdot \prod_{i \in T} a^{(i)} .$$

However $x^2 = x$ for any $x \in \Sigma$ so

$$\textstyle\prod_{i \in S} a^{(i)} \cdot \prod_{i \in T} a^{(i)} \ = \ \displaystyle\prod_{i \in S \cup T} a^{(i)} \ = \ \chi_{S \cup T}(a) \ = \ E_a(\chi_{S \cup T})$$

Now we turn to the converse. Let $\tilde{A}: \mathcal{F}_l \to \Sigma$ be linear and respecting the monomial basis. For $i = 1, \ldots, l$, let $a_i \stackrel{\text{def}}{=} \tilde{A}(\chi_{\{i\}})$, and let $a \stackrel{\text{def}}{=} a_1 \ldots a_l$. We claim that $\tilde{A} = E_a$. The proof is as follows. We first claim that

$$\forall\ S \subseteq [l] \ : \ \tilde{A}(\chi_S) = \chi_S(a) . \tag{3.2}$$

Since $\tilde{A}$ respects the monomial basis we have $\tilde{A}(\chi_\emptyset) = 1$ which in turn equals $\chi_\emptyset(a)$, proving Eq. (3.2) for $S = \emptyset$. To establish Eq. (3.2) for $S = \{i_1, \ldots, i_t\} \neq \emptyset$, we write

$$\tilde{A}(\chi_S) \;=\; \tilde{A}\left(\chi_{\{i_1\}\cup\cdots\cup\{i_t\}}\right) \;=\; \prod_{j=1}^{t} \tilde{A}(\chi_{\{i_j\}}) \;=\; \prod_{j=1}^{t} a_{i_j} \;=\; \chi_S(a)\,.$$

where the second equality is due to the fact that $\tilde{A}$ respects the monomial basis. This establishes Eq. (3.2). Now for any $f \in \mathcal{F}_l$ we can use the linearity of $\tilde{A}$ to see that

$$\tilde{A}(f) \;=\; \tilde{A}(\textstyle\sum_S C_f(S)\cdot\chi_S)) \;=\; \sum_S C_f(S)\cdot\tilde{A}(\chi_S) \;=\; \sum_S C_f(S)\cdot\chi_S(a) \;=\; f(a) \;=\; E_a(f)\,.$$

Thus $\tilde{A} = E_a$. ▮

The *long code* $E\colon \Sigma^l \to \mathsf{Map}(\mathcal{F}_l, \Sigma)$ is defined for any $a \in \Sigma^l$ by $E(a) = E_a$. Thus, formally, a codeword is a map of $\mathcal{F}_l$ to Σ. Intuitively, think of the codeword $E(a)$ as the 2^{2^l} bit string which in position $f \in \mathcal{F}_l$ stores the bit $f(a)$. It is thus an extremely "redundant" code, encoding an l-bit string by the values, at a, of *all* functions in $\mathcal{F}_l$. In some sense E is the longest possible code: E is the longest code which is not repetitive (i.e., does not have two positions which are identical in all codewords).

We let $\mathrm{Dist}(A, \textsc{Eval}) = \min_{a\in\Sigma^l} \mathrm{Dist}(A, E_a)$ be the distance from A to a closest codeword of E. It is convenient to define $E^{-1}(A) \in \Sigma^l$ as the lexicographically least $a \in \Sigma^l$ such that $\mathrm{Dist}(A, E_a) = \mathrm{Dist}(A, \textsc{Eval})$. Notice that if $\mathrm{Dist}(A, \textsc{Eval}) < 1/4$ then there is exactly one $a \in \Sigma^l$ such that $\mathrm{Dist}(A, E_a) = \mathrm{Dist}(A, \textsc{Eval})$, and so $E^{-1}(A)$ is this a. The following is useful in relating folding to the long code.

Proposition 3.3.3 (folding and the evaluation operator): Let $A\colon \mathcal{F}_l \to \Sigma$, $h \in \mathcal{F}_l$, $b \in \Sigma$ and $a \in \Sigma^l$. Suppose that for any $f \in \mathcal{F}_l$ it is the case that $A(f+h) = A(f)+b$. Then $\mathrm{Dist}(A, E_a) < 1/2$ implies $h(a) = b$. Consequently, if $\mathrm{Dist}(A_{(h,b),(h',b')}, E_a) < 1/2$ then $h(a) = b$, provided $b = 0$ if $h \equiv 0$.

Proof: By the hypothesis, we have $A(h + f) = A(f) + b$, for every $f \in \mathcal{F}_l$. Suppose that $\mathrm{Dist}(A, E_a) < 1/2$. Then, noting that E_a is linear and applying Corollary 3.5.2 (below), we get $E_a(h) = b$. Using the definition of the Evaluator operator (i.e., $E_a(h) = h(a)$) we have $h(a) = b$. The consequence for $A_{(h,b),(h',b')}$ follows since by Proposition 3.2.2 we have $A_{(h,b),(h',b')}(f + h) = A_{(h,b),(h',b')}(f) + b$ for any $f \in \mathcal{F}_l$. ▮

The long code is certainly a disaster in terms of coding theory, but it has a big advantage in the context of proof verification. Consider, for example, the so-called "circuit test" (i.e., testing that the answer of the first prover satisfies some predetermined predicate/circuit). In this context one needs to check that the codeword corresponds to a string which satisfies a predetermined predicate (i.e., the codeword encodes some $w \in \{0,1\}^n$ which satisfies $h(w) = 0$, for some predetermined predicate h). The point is that the value of this predicate appears explicitly in the codeword itself, and furthermore it can be easily "self-corrected" by probing the codeword for the values of the functions f and $f + h$, for a uniformly selected function $f : \{0,1\}^n \to \{0,1\}$ (as all these values appear explicitly in the codeword). Actually, the process of verifying, via self-correction, that the value under h is zero can be incorporated into the task of checking the validity of the codeword; this is done by the notion of "$(h, 0)$-folding" (see above). The fact that we can avoid testing whether the codeword encodes a string which satisfies a given function (or that this testing does not cost us anything) is the key to the complexity improvements in our proof systems (over previous proof systems in which a "circuit test" was taking place).

3.4 Recursive verification of proofs

This section specifies the basic structure of proof construction, and in particular provides the definitions of the notions of inner and outer verifiers which will be used throughout. It is useful to understand these things before proceeding to the tests.

OVERVIEW. The constructions of efficient proofs that follow will exploit the notion of recursive verifier construction due to Arora and Safra [ArSa]. We will use just one level of recursion. We first define a notion of a *canonical outer verifier* whose intent is to capture two-prover one-round proof systems [BGKW] having certain special properties; these verifiers will be our starting point. We then define a canonical inner verifier. Recursion is captured by an appropriate definition of a composed verifier whose attributes we relate to those of the original verifiers in Theorem 3.4.5.

The specific outer verifier we will use is one obtained by a recent work of Raz [Raz]. We will construct various inner verifiers based on the long code and the tests in Section 3.5 and Section 3.11. Theorem 3.4.5 will be used ubiquitously to combine the two.

For a better understanding of the role of constant-prover proof systems in this context, and an explanation of what the use of [Raz] buys as opposed to the use of other systems, we have provided at the end of this subsection an explanatory history.

3.4.1 Outer verifiers

As mentioned above, outer verifiers will model certain special kinds of two-prover, one-round proof systems. We think of the verifier as provided with a pair of proof oracles π, π_1, and allowed one query to each. The desired properties concern the complexity of the system and a certain behavior in the checking of the proof, as we now describe.

Let $r_1, s, s_1 \colon \mathcal{Z}^+ \to \mathcal{Z}^+$ and let l and l_1 be positive integers. An (l, l_1)-**canonical outer verifier** V_{outer} takes as input $x \in \Sigma^n$, and has oracle access to a pair of proofs $\bar{\pi} \colon [s(n)] \to \Sigma^l$ and $\bar{\pi}_1 \colon [s_1(n)] \to \Sigma^{l_1}$. He does the following.

- Picks a random string R_1 of length $r_1(n)$.
- Computes, as a function of x and R_1, queries $q \in [s(n)]$ and $q_1 \in [s_1(n)]$, and a (circuit computing a) function $\sigma \colon \Sigma^l \to \Sigma^{l_1}$ (which is determined by x and R_1). Determines, based on x and q, a function $h \colon \Sigma^l \to \Sigma$ (and computes an appropriate representation of it).
  (We stress that h does not depend on R_1, only on q and x).
- Lets $a = \bar{\pi}(q)$ and $a_1 = \bar{\pi}_1(q_1)$.
- If $h(a) \neq 0$ then **rejects.**
- If $\sigma(a) \neq a_1$ then **rejects.**
- Otherwise **accepts.**

We call s, s_1 the *proof sizes* for V_{outer} and r_1 the *randomness* of V_{outer}.

Recall that by the conventions in Section 2, $\texttt{ACC}[\, V_{\text{outer}}^{\bar{\pi},\bar{\pi}_1}(x)\,]$ denotes the probability, over the choice of R_1, that V_{outer} accepts, and $\texttt{ACC}[\, V_{\text{outer}}(x)\,]$ denotes the maximum of $\texttt{ACC}[\, V_{\text{outer}}^{\bar{\pi},\bar{\pi}_1}(x)\,]$ over all possible proofs $\bar{\pi}, \bar{\pi}_1$.

Definition 3.4.1 (soundness of outer verifier): An outer verifier V_{outer} is ϵ-**good** for a language L if

(1) If $x \in L$ then $\texttt{ACC}[\, V_{\text{outer}}(x)\,] = 1$.

(2) If $x \notin L$ then $\texttt{ACC}[\, V_{\text{outer}}(x)\,] \leq \epsilon$.

Employing the FRS-method [FRS] to any PCP(log,O(1))-system for NP (e.g., [ALMSS]) one gets a canonical verifier which is δ-good for some $\delta < 1$. Using the Parallel Repetition Theorem of Raz, we obtain our starting point –

Lemma 3.4.2 (construction of outer verifiers [Raz]): Let $L \in$ NP. Then for every $\epsilon > 0$ there exist positive integers l, l_1 and c such that there exists an (l, l_1)-canonical outer verifier which is ϵ-good for L and uses randomness $r(n) = c \log_2 n$.

Actually, Raz's Theorem [Raz] enables one to assert that l, l_1 and c are $O(\log \epsilon^{-1})$; but we will not need this fact. Also, the function σ determined by this verifier is always a projection, but we don't use this fact either.

3.4.2 Inner verifiers

We now describe the form of a typical inner verifier. It may be illustrative to remember that the inner verifier will perform a combination of the *atomic linear test*, the *atomic respect of monomial basis test* and the *atomic projection test*. It turns out that the inner verifiers never need to perform a "circuit test" (i.e., test that $h(a) = 0$). This is achieved by use of the folding mechanism introduced in Section 3.2, and we refer the reader there for the notation "$A_{(h,b)}$" that is used below.

Let $r_2, l, l_1 \in \mathcal{Z}^+$. A (l, l_1)-**canonical inner verifier** V_{inner} takes as inputs functions $\sigma \colon \Sigma^l \to \Sigma^{l_1}$ and $h \in \mathcal{F}_l$. (It may also take additional inputs, depending on the context). It has oracle access to a pair of functions $A \colon \mathcal{F}_l \to \Sigma$ and $A_1 \colon \mathcal{F}_{l_1} \to \Sigma$, and uses r_2 random bits. The parameters $\delta_1, \delta_2 > 0$ in the following should be thought as extremely small: in our constructions, they are essentially 0 (see comment below).

Definition 3.4.3 (soundness of inner verifier): An inner verifier V_{inner} is $(\rho, \delta_1, \delta_2)$-**good** if for all σ, h as above–

(1) Suppose $a \in \Sigma^l$ is such that $h(a) = 0$. Let $a_1 = \sigma(a) \in \Sigma^{l_1}$. Then $\texttt{ACC}[\, V_{\text{inner}}^{E_a, E_{a_1}}(\sigma, h)\,] = 1$.

(2) Suppose A, A_1 are such that $\texttt{ACC}[\, V_{\text{inner}}^{A, A_1}(\sigma, h)\,] \geq \rho$. Then there exists $a \in \Sigma^l$ such that:
  (2.1) $\text{Dist}(A_{(h,0),(\bar{1},1)}, E_a) < 1/2 - \delta_1$.
  (2.2) $\text{Dist}(A_1, E_{\sigma(a)}) < 1/2 - \delta_2$.

We stress that although the inner verifier has access to the oracle A (and the hypothesis in condition (2) of Definition 3.4.3 refers to its computations with oracle A), the conclusion in condition (2.1) refers to A folded over both $(h, 0)$ and $(\bar{1}, 1)$, where $\bar{1}$ is the constant-one function. (Typically, but not necessarily, the verifier satisfying Definition 3.4.3 accesses the virtual oracle $A_{(h,0),(\bar{1},1)}$ by actual access to A according to the definition of folding.) Furthermore, by Proposition 3.3.3, condition (2.1) implies that $h(a) = 0$. (Thus, there is no need to explicitly require $h(a) = 0$ in order to make Theorem 3.4.5 work.) We comment that the upper bounds in conditions (2.1) and (2.2) are chosen to be the largest ones which still allow us to prove Theorem 3.4.5 (below). Clearly, the complexity of the inner verifier decreases as these bounds increase. This is the reason for setting δ_1 and δ_2 to be extremely small. We stress that this optimization is important for the MaxSNP results but not for the Max Clique result. In the latter case, we can use δ_i's greater than $\frac{1}{4}$ which simplifies a little the analysis of the composition of verifiers (below).
A tedious remark: The above definition allows h to be identically zero (although this case never occurs in our constructions nor in any other reasonable application). This is the reason that we had to define folding over (0,0) as well. An alternative approach would have been to require $h \not\equiv 0$ and assert that this is the case with respect to the outer verifier of Lemma 3.4.2.

3.4.3 Composition of verifiers

We now describe the canonical composition of a canonical outer verifier with a canonical inner verifier.

Let $V_{\rm outer}$ be a (l, l_1)-canonical outer verifier with randomness r_1 and proof sizes s, s_1. Let $V_{\rm inner}$ be a (l, l_1)-canonical inner verifier with randomness r_2. Their composed verifier $\langle V_{\rm outer}, V_{\rm inner}\rangle$ takes as input $x \in \Sigma^n$ and has oracle access to proofs $\pi\colon [s(n)] \times \mathcal{F}_l \to \Sigma$ and $\pi_1\colon [s_1(n)] \times \mathcal{F}_{l_1} \to \Sigma$. We ask that it does the following –

- Picks random strings for both $V_{\rm outer}$ and $V_{\rm inner}$; namely, picks a random string R_1 of length $r_1(n)$ and a random string R_2 of length $r_2(n)$.
- Computes queries q and q_1 and functions σ and h as $V_{\rm outer}$ would compute them given x, R_1
- Outputs $V_{\rm inner}^{A,A_1}(\sigma, h; R_2)$ where $A(\cdot) = \pi(q, \cdot)$ and $A_1(\cdot) = \pi_1(q_1, \cdot)$.

The randomness complexity of the composed verifier is $r_1 + r_2$ whereas its query and free-bit complexities equal those of $V_{\rm inner}$.

We show how the composite verifier $\langle V_{\rm outer}, V_{\rm inner}\rangle$ inherits the goodness of the $V_{\rm outer}$ and $V_{\rm inner}$. To do so we need the following Lemma. It is the counterpart of a claim in [BGLR, Lemma 3.5] and will be used in the same way. The lemma can be derived from a coding theory bound which is slight extension of bounds in [MaSl, Ch. 7] and is provided in Section 3.13.

Lemma 3.4.4 Suppose $0 \le \delta \le 1/2$ and $A\colon \mathcal{F}_l \to \Sigma$. Then

$$\left|\{\, a \in \Sigma^l \;:\; {\rm Dist}(A, E_a) \le 1/2 - \delta \,\}\right| \;\le\; \frac{1}{4\delta^2} \,.$$

Furthermore, for $\delta > 1/4$ the above set contains at most one string.

Proof: We know that E_a is linear for any a (cf. Proposition 3.3.2). So it suffices to upper bound the size of the set

$$\mathcal{A} \;=\; \{\, X \in {\rm L{\scriptsize IN}}(\mathcal{F}_l, \Sigma) \;:\; {\rm Dist}(A, X) \le 1/2 - \delta \,\} \,.$$

This set has the same size as

$$\mathcal{B} \;=\; \{\, X - A \;:\; X \in {\rm L{\scriptsize IN}}(\mathcal{F}_l, \Sigma) \text{ and } {\rm Dist}(A, X) \le 1/2 - \delta \,\} \,.$$

Let $n = 2^{2^l}$ and identify $\mathsf{Map}(\mathcal{F}_l, \Sigma)$ with Σ^n in the natural way. Let $w(\cdot)$ denote the Hamming weight. Now note that $Z = X - A \in \mathcal{B}$ implies $w(Z)/n = {\rm Dist}(X, A) \le 1/2 - \delta$. Furthermore if $Z_1 = X_1 - A$ and $Z_2 = X_2 - A$ are in $\mathcal{B}$ then ${\rm Dist}(Z_1, Z_2) = {\rm Dist}(X_1, X_2)$ and the latter is $1/2$ if $X_1 \ne X_2$, since X_1, X_2 are linear. So we can apply Lemma 3.13.1 (with $\alpha = \delta$ and $\beta = 0$) to upper bound the size of $\mathcal{B}$ as desired. Finally, when $\delta > 1/4$ the triangle inequality implies that we cannot have $a_1 \ne a_2$ so that ${\rm Dist}(A, E_{a_i}) \le 1/2 - \delta < 1/4$ for both $i = 1, 2$. ▌

In some applications of the following theorem, $\delta_1, \delta_2 > 0$ will first be chosen to be so small that they may effectively be thought of as 0. (This is done in order to lower the complexities of the inner verifiers.) Once the δ_i's are fixed, ϵ will be chosen to be so much smaller (than the δ_i's) that $\epsilon/(16\delta_1^2\delta_2^2)$ may be thought of as effectively 0. The latter explains why we are interested in outer verifiers which achieve a constant, but arbitrarily small, error ϵ. For completeness we provide a proof, following the ideas of [ArSa, ALMSS, BGLR].

Theorem 3.4.5 (the composition theorem): Let $V_{\rm outer}$ be a (l, l_1)-canonical outer verifier. Suppose it is ϵ-good for L. Let $V_{\rm inner}$ be an (l, l_1)-canonical inner verifier that is $(\rho, \delta_1, \delta_2)$-good. Let $V = \langle V_{\rm outer}, V_{\rm inner}\rangle$ be the composed verifier, and let $x \in \Sigma^*$. Then —

(1) If $x \in L$ then $\mathtt{ACC}[\,V(x)\,] = 1$
(2) If $x \notin L$ then $\mathtt{ACC}[\,V(x)\,] \le \rho + \frac{\epsilon}{16\delta_1^2\delta_2^2}$.
For $\delta_1, \delta_2 > 1/4$ the upper bound in (2) can be improved to $\rho + \epsilon$.

(The latter case (i.e., $\delta_1, \delta_2 > 1/4$) suffices for the Max Clique results.)

Proof: Let $n = |x|$, and let s, s_1 denote the proof sizes of V_{outer}.

Suppose $x \in L$. By Definition 3.4.1 there exist proofs $\bar{\pi}\colon [s(n)] \to \Sigma^l$ and $\bar{\pi}_1\colon [s_1(n)] \to \Sigma^{l_1}$ such that $\mathtt{ACC}[\,V_{\text{outer}}^{\bar{\pi},\bar{\pi}_1}(x)\,] = 1$. Let $\pi\colon [s(n)] \times \mathcal{F}_l \to \Sigma$ be defined by $\pi(q, f) = E_{\bar{\pi}(q)}(f)$. (In other words, replace the l bit string $\bar{\pi}(q)$ with its 2^{2^l} bit encoding under the long code, and let the new proof provide access to the bits in this encoding). Similarly let $\pi_1\colon [s_1(n)] \times \mathcal{F}_{l_1} \to \Sigma$ be defined by $\pi_1(q_1, f_1) = E_{\bar{\pi}_1(q_1)}(f_1)$. Now one can check that the item (1) properties in Definitions 3.4.1 and 3.4.3 (of the outer and inner verifier, respectively) imply that $\mathtt{ACC}[\,V^{\pi,\pi_1}(x)\,] = 1$.

Now suppose $x \notin L$. Let $\pi\colon [s(n)] \times \mathcal{F}_l \to \Sigma$ and $\pi_1\colon [s_1(n)] \times \mathcal{F}_{l_1} \to \Sigma$ be proof strings for V. We will show that $\mathtt{ACC}[\,V^{\pi,\pi_1}(x)\,] \le \rho + \epsilon/(16\delta_1^2\delta_2^2)$. Since π, π_1 were arbitrary, this will complete the proof.

We set $N_1 = \lfloor 1/(4\delta_1^2) \rfloor$ and $N_2 = \lfloor 1/(4\delta_2^2) \rfloor$ (with $N_1 = 1$ if $\delta_1 > 1/4$ and $N_2 = 1$ if $\delta_2 > 1/4$). The idea to show $\mathtt{ACC}[\,V^{\pi,\pi_1}(x)\,] \le \rho + N_1 N_2 \cdot \epsilon$ is as follows. We will first define a collection of N_1 proofs $\bar{\pi}^1, \ldots, \bar{\pi}^{N_1}$ and a collection of N_2 proofs $\bar{\pi}_1^1, \ldots, \bar{\pi}_1^{N_2}$ so that each pair $(\bar{\pi}^i, \bar{\pi}_1^j)$ is a pair of oracles for the *outer* verifier. Next we will partition the random strings R_1 of the *outer* verifier into two categories, depending on the performance of the *inner* verifier on the inputs (i.e., the functions σ, h and the oracles A, A_1) induced by R_1. On the "bad" random strings of the *outer* verifier, the *inner* verifier will accept with probability at most ρ; on the "good" ones, we will use the soundness of the *inner* verifier to infer that that the *outer* verifier accepts under some oracle pair $(\bar{\pi}^i, \bar{\pi}_1^j)$, for $i \in [N_1]$ and $j \in [N_2]$. The soundness of the *outer* verifier will be used to bound the probability of such acceptances. Let us now proceed to the actual proof.

We now turn to the actual analysis. We define N_1 proofs $\bar{\pi}^1, \ldots, \bar{\pi}^{N_1}\colon [s(n)] \to \Sigma^l$ as follows. Fix $q \in [s(n)]$ and let $A = \pi(q, \cdot)$. Let $B_q = \{\, a \in \Sigma^l : \mathrm{Dist}(A_{(h,0),(\bar{1},1)}, E_a) < 1/2 - \delta_1 \,\}$. (Notice that for this set to be well-defined we use the fact that h is well-defined given q.) Note that $|B_q| \le N_1$ by Lemma 3.4.4. Order the elements of B_q in some canonical way, adding dummy elements to bring the number to exactly N_1, so that they can be written as $a^1(q), \ldots, a^{N_1}(q)$. Now set $\bar{\pi}^i(q) = a^i(q)$ for $i = 1, \ldots, N_1$. In a similar fashion we define $\bar{\pi}_1^j(q_1) = a_1^j(q_1)$ for $j = 1, \ldots, N_2$, where each $a_1^j = a_1^j(q_1)$ satisfies $\mathrm{Dist}(\pi_1(q_1, \cdot), E_{a_1^j}) \le 1/2 - \delta_2$.

Let R_1 be a random string of V_{outer}. We say that R_1 is *good* if

$$\mathtt{ACC}[\,V_{\text{inner}}^{\pi(q,\cdot),\pi_1(q_1,\cdot)}(\sigma, h)\,] \;\ge\; \rho\;,$$

where q, q_1, σ, h are the queries and functions specified by R_1. If R_1 is not good we say it is *bad*. The claim that follows says that if R_1 is good then there is some choice of the above defined proofs which leads the outer verifier to accept on coins R_1.

Claim. Suppose R_1 is good. Then there is an $i \in [N_1]$ and a $j \in [N_2]$ such that $V_{\text{outer}}^{\bar{\pi}^i,\bar{\pi}_1^j}(x; R_1) = 0$.

Proof. Let q, q_1, σ, h be the queries and functions specified by R_1. Let $A = \pi(q, \cdot)$ and $A_1 = \pi_1(q_1, \cdot)$ (be the oracles accessed by the inner verifier). Since R_1 is good we have $\mathtt{ACC}[\,V_{\text{inner}}^{A,A_1}(\sigma, h)\,] \ge \rho$. So by Item (2) of Definition 3.4.3 there exists $a \in \Sigma^l$ such that $\mathrm{Dist}(A_{(h,0),(\bar{1},1)}, E_a) < 1/2 - \delta_1$ and $\mathrm{Dist}(A_1, E_{\sigma(a)}) < 1/2 - \delta_2$. Let $a_1 = \sigma(a)$. Since $\mathrm{Dist}(A_{(h,0),(\bar{1},1)}, E_a) \le 1/2 - \delta_1$ it must be the case that $a \in B_q$, and hence there exists $i \in [N_1]$ such that $a = \bar{\pi}^i(q)$. Similarly $\mathrm{Dist}(A_1, E_{\sigma(a)}) < 1/2 - \delta$

implies that there is some $j \in [N]$ such that $a_1 = \bar{\pi}_1^j(q_1)$. By Proposition 3.3.3 we have $h(a) = 0$, and we have $\sigma(a) = a_1$ by (the above) definition. Now, by definition of the (execution of the) canonical outer verifier, $V_{\text{outer}}^{\bar{\pi}^i, \bar{\pi}_1^j}(x; R_1) = 0$ holds. □

By conditioning we have $\texttt{ACC}\,[\, V^{\pi,\pi_1}(x)\,] \leq \alpha + \beta$ where

$$\begin{aligned} \alpha &= \Pr_{R_1}[R_1 \text{ is good}] \\ \beta &= \Pr_{R_1,R_2}[\, V^{\pi,\pi_1}(x; R_1R_2) = 0 \mid R_1 \text{ is bad}\,] \ . \end{aligned}$$

The definition of badness implies $\beta \leq \rho$. On the other hand we can use the Claim to see that

$$\begin{aligned} \alpha &\leq \Pr_{R_1}\left[\exists i \in [N_1]\ ,\ j \in [N_2]\ :\ V_{\text{outer}}^{\bar{\pi}^i, \bar{\pi}_1^j}(x; R_1) = 0\right] \\ &\leq \textstyle\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \Pr_{R_1}\left[V_{\text{outer}}^{\bar{\pi}^i, \bar{\pi}_1^j}(x; R_1) = 0\right] \\ &\leq N_1 N_2 \cdot \epsilon\ , \end{aligned}$$

the last by the soundness of V_{outer} (i.e., Item (2) of Definition 3.4.1). Using the bound on N_1 and N_2, the proof is concluded. ∎

3.4.4 Constant-prover proofs in PCP — perspective

Constant-prover proofs have been instrumental in the derivation of non-approximability results in several ways. One of these is that they are a good starting point for reductions— examples of such are reductions of two-prover proofs to quadratic programming [BeRo, FeLo] and set cover [LuYa]. However, it is a different aspect of constant prover proofs that is of more direct concern to us. This aspect is the use of constant-prover proof systems as the penultimate step of the recursion, and begins with [ALMSS]. It is instrumental in getting PCP systems with only a constant number of queries. Their construction requires that these proof systems have low complexity: error which is any constant, and randomness and answer sizes that are preferably logarithmic. The number of provers and the randomness and query complexity determine the quality of many non-approximability results (e.g., poly-logarithmic rather than logarithmic complexities translate into non-approximability results using assumptions about quasi-polynomial time classes rather than polynomial time ones). The available constant-prover proof systems appear in Figure 3.1 and are discussed below.

The two-prover proofs of Lapidot-Shamir and Feige-Lovász [LaSh, FeLo] had poly-logarithmic randomness and answer sizes, so [ALMSS] used a modification of these, in the process increasing the number of provers to a constant much larger than two. The later constructions of few-prover proofs of [BGLR, Ta, FeKi] lead to better non-approximability results.

Bellare and Sudan [BeSu] identified some extra features of constant prover proofs whose presence they showed could be exploited to further increase the non-approximability factors. These features are captured in their definition of canonical verifiers. But the proof systems of [FeKi] that had worked above no longer sufficed— they are not canonical. So instead [BeSu] used (a slight modification of) the proofs of [LaSh, FeLo], thereby incurring poly-logarithmic randomness and answer sizes, so that the assumptions in their non-approximability results pertain to quasi-polynomial time classes. (Alternatively they modify the [FeKi] system to a canonical three-prover one, but then incur a decrease in the non-approximability factors due to having more provers).

Due to	**Provers**	**Coins**	**Answer size**	**Canonical?**	**Can be made canonical?**
[LaSh, FeLo]	2	polylog	polylog	No	Yes [BeSu]
[ALMSS]	poly(ϵ^{-1})	log	polylog	No	??
[BGLR]	4	log	polyloglog	No	??
[Ta]	3	log	$O(1)$	No	??
[FeKi]	2	log	$O(1)$	No	At cost of one more prover [BeSu]
[Raz]	2	log	$O(1)$	Yes	(NA)

Figure 3.1: *Constant prover PCPs achieving error which is a fixed, but arbitrarily small, constant ϵ. We indicate the number of provers, the randomness and answer sizes, and whether or not the system is canonical. The notation ?? means "don't know and don't care because stronger things have become available." In all cases the randomness and answer sizes hide factors which depend on ϵ.*

Our outer verifiers ask for almost the same canonicity properties. (The only difference is that they have required σ to be a projection function, whereas we can deal with an arbitrary function. But we don't take advantage of this fact.) In addition we need answer sizes of $O(\log\log n)$ as opposed to the $O(\log n)$ of previous methods, for reasons explained below. This means that even the (modified) [LaSh, FeLo] type proofs won't suffice for us. We could use the three-prover modification of [FeKi] but the cost would wipe out our gain. Luckily this discussion is moot since we can use the recent result of Raz [Raz] to provide us with a canonical two-prover proof having logarithmic randomness, constant answer size, and any constant error. This makes an ideal starting point. To simplify the definitions above we insisted on constant answer size and two provers from the start.

The inner verifiers used in all previous works are based on the use of the Hadamard code constructions of [ALMSS]. (The improvements mentioned above are obtained by checking this same code in more efficient ways). We instead use a new code, namely the long code, as the basis of our inner verifiers. Note the codewords (in the long code) have length double exponential in the message, explaining our need for $O(\log\log n)$ answer sizes in the outer verifier. We also incorporate into the definitions the new idea of folding which we will see means we don't need a circuit test (a hint towards this fact is already present in the definition of a good inner verifier).

3.5 The atomic tests

Motivation. Our constructions of proofs systems will use the outer verifier of Lemma 3.4.2, composed via Theorem 3.4.5 with inner verifiers to be constructed. The brunt of our constructions is the construction of appropriate inner verifiers. The inner verifier will have oracle access to a function $A\colon \mathcal{F}_l \to \Sigma$ and a function $A_1\colon \mathcal{F}_{l_1} \to \Sigma$. In all our applications, A is supposed to be a folding of an encoding of the answer a of the first prover (in a two-prover proof system) and A_1 is supposed to be the encoding of the answer a_1 of the second prover. The verifier will perform various tests to determine whether these claims are true. The subject of this subsection is the design of these tests.

The atomic tests we provide here will be used directly in the proof systems for showing non-

The Atomic Tests. Here $A\colon \mathcal{F}_l \to \Sigma$ and $A_1\colon \mathcal{F}_{l_1} \to \Sigma$ are the objects being tested. The tests also take additional inputs or parameters: below $f, f_1, f_2, f_3 \in \mathcal{F}_l$; $g \in \mathcal{F}_{l_1}^m$; and $\sigma\colon \Sigma^l \to \Sigma^{l_1}$.

LinTest$(A; f_1, f_2)$ (Linearity Test)
If $A(f_1) + A(f_2) = A(f_1 + f_2)$ then output 0 else output 1.

MBTest$(A; f_1, f_2, f_3)$ (Respecting-Monomial-Basis Test)
If $A(f_1) = 0$ then check if $A(f_1 \cdot f_2 + f_3) = A(f_3)$
Otherwise (i.e. $A(f_1) = 1$) then check if $A(f_1 \cdot f_2 + f_2 + f_3) = A(f_3)$
Output 0 if the relevant check succeeded, else output 1.

ProjTest$_\sigma(A, A_1; f, g)$ (Projection Test)
If $A_1(g) = A(g \circ \sigma + f) - A(f)$ then output 0, else output 1.

The Passing Probabilities. These are the probabilities we are interested in:

$$\begin{aligned}
\textsc{LinPass}(A) &= \Pr_{f_1, f_2 \xleftarrow{R} \mathcal{F}_l}\left[\, \mathbf{LinTest}(A; f_1, f_2) = 0 \,\right] \\
\textsc{MBPass}(A) &= \Pr_{f_1, f_2, f_3 \xleftarrow{R} \mathcal{F}_l}\left[\, \mathbf{MBTest}(A; f_1, f_2, f_3) = 0 \,\right] \\
\textsc{ProjPass}_\sigma(A, A_1) &= \Pr_{f \xleftarrow{R} \mathcal{F}_l\, ;\, g \xleftarrow{R} \mathcal{F}_{l_1}}\left[\, \mathbf{ProjTest}_\sigma(A, A_1; f, g) = 0 \,\right]
\end{aligned}$$

Figure 3.2: *The atomic tests and their passing probabilities.*

approximability of Max-3-SAT, Max-2-SAT and Max-CUT. Furthermore, they are also the basis of iterated tests which will lead to proof systems of amortized free-bit complexity ≈ 2, which in turn are used for the Max Clique and Chromatic Number results. We remark that for the applications to the above-mentioned Max-SNP problems it is important to have the best possible analysis of our atomic tests, and what follows strives to this end. We stress that the exposition and analysis of these tests, in this subsection, is independent of the usage of the codes in our proof systems.

Testing for a codeword. The first task that concerns us is to design a test which, with high probability, passes if and only if A is close to an evaluation operator (i.e., a valid codeword). The idea is to exploit the characterization of Proposition 3.3.2. Thus we will perform (on A) a linearity test, and then a "respect of monomial basis" test. Linearity testing is well understood, and we will use the test of [BLR], with the analyses of [BLR, BGLR, BCHKS]. The main novelty is the respect of monomial basis test.

Circuit and projection. Having established that A is close to some evaluation operator E_a, we now want to test two things. The first is that $h(a) = 0$ for some predetermined function h. This test which would normally be implemented by "self-correction" (i.e., evaluating $h(a)$ by uniformly selecting $f \in \mathcal{F}_l$ and computing $A(f + h) - A(f)$) is not needed here, since in our applications we will use an $(h,0)$-folding of A instead of A. Thus, it is left to test that the two oracles are consistent in the sense that A_1 is not too far from an evaluation operator which corresponds to $\sigma(a)$ for some predetermined function σ.

SELF-CORRECTION. The following self-correction lemma is due to [BLR] and will be used throughout.

Lemma 3.5.1 (Self Correction Lemma [BLR]): Let $A, \tilde{A}\colon \mathcal{F}_l \to \Sigma$ with $\tilde{A}$ linear, and let $x = \mathrm{Dist}(A, \tilde{A})$. Then for every $f \in \mathcal{F}_l$:

$$\Pr_{h \xleftarrow{R} \mathcal{F}_l}\left[A(f+h) - A(h) = \tilde{A}(f)\right] \;\geq\; 1 - 2x \;.$$

Proof:

$$\begin{aligned}
\Pr_{h \xleftarrow{R} \mathcal{F}_l}\left[A(f+h) - A(h) = \tilde{A}(f)\right] &\geq \Pr_{h \xleftarrow{R} \mathcal{F}_l}\left[A(f+h) = \tilde{A}(f+h) \;\text{ and }\; A(h) = \tilde{A}(f)\right] \\
&\geq 1 - \left(\Pr_{h \xleftarrow{R} \mathcal{F}_l}\left[A(f+h) \neq \tilde{A}(f+h)\right] + \Pr_{h \xleftarrow{R} \mathcal{F}_l}\left[A(h) \neq \tilde{A}(h)\right]\right) \\
&= 1 - 2x
\end{aligned}$$

▮

Corollary 3.5.2 Let $A, \tilde{A}\colon \mathcal{F}_l \to \Sigma$ with $\tilde{A}$ linear, and suppose $x \stackrel{\text{def}}{=} \mathrm{Dist}(A, \tilde{A}) < 1/2$. Suppose also that $A(f+h) = A(h) + \sigma$, for some $f \in \mathcal{F}_l$, $\sigma \in \Sigma$ and every $h \in \mathcal{F}_l$. Then $\tilde{A}(f) = \sigma$.

Proof: By the hypothesis, we have $A(f+h) - A(f) = \sigma$ for all h's. Thus, we can write

$$\Pr_{h \xleftarrow{R} \mathcal{F}_l}\left[A(f+h) - A(f) = \tilde{A}(f)\right] \;=\; \Pr_{h \xleftarrow{R} \mathcal{F}_l}\left[\sigma = \tilde{A}(f)\right] .$$

But the right hand side (and hence the left) is either 0 or 1. However, by Lemma 3.5.1 the left hand side is bounded below by $1 - 2x > 0$ and so the corollary follows. ▮

CONVENTION. All our tests output a bit, with 0 standing for accept and 1 for reject.

3.5.1 Atomic linearity test

The *atomic linearity test* shown in Figure 3.2 is the one of Blum, Luby and Rubinfeld [BLR]. We want to lower bound the probability $1 - \mathrm{LinPass}(A)$ that the test rejects when its inputs f_1, f_2 are chosen at random, as a function of $x = \mathrm{Dist}(A, \mathrm{Lin})$. The following lemma, due to Bellare et. al. [BCHKS], gives the best known lower bound today. Detailed description of the history of developments in this area follows.

Lemma 3.5.3 [BCHKS] Let $A\colon \mathcal{F}_l \to \Sigma$ and let $x = \mathrm{Dist}(A, \mathrm{Lin})$. Then $1 - \mathrm{LinPass}(A) \geq \Gamma_{\mathrm{lin}}(x)$, where the function $\Gamma_{\mathrm{lin}}\colon [0,1/2] \to [0,1]$ is defined as follows:

$$\Gamma_{\mathrm{lin}}(x) \stackrel{\text{def}}{=} \begin{cases} 3x - 6x^2 & 0 \leq x \leq 5/16 \\ 45/128 & 5/16 \leq x \leq 45/128 \\ x & 45/128 \leq x \leq 1/2. \end{cases}$$

The above lower bound is composed of three different bounds with "phase transitions" at $x = \frac{5}{16}$ and $x = \frac{45}{128}$. It was shown in [BCHKS] (see below) that this combined lower bound is close to the best one possible.

History. The general problem of linearity testing as introduced and studied by Blum et. al. [BLR] is stated as follows: given a function $A: G \to H$, where G, H are groups, obtain a lower bound on δ_A as a function of x_A, where

$$\begin{aligned} \delta_A &= \Pr_{a,b \overset{R}{\leftarrow} G}[A(a) + A(b) \neq A(a+b)] \\ x_A &= \mathrm{Dist}(A, \textsc{Lin}) \ . \end{aligned}$$

Blum et. al. showed that $\delta_A \geq \frac{2}{9}x_A$, for every A. Their analysis was used in the proof system and Max-3-SAT non-approximability result of [ALMSS]. Interest in the tightness of the analysis from the point of view of improving the Max-3-SAT non-approximability began with [BGLR]. They showed that $\delta_A \geq 3x_A - 6x_A^2$, for every A. This establishes the first segment of the lower bound quoted above (i.e., of the function $\Gamma_{\rm lin}$). Also, it is possible to use [BLR] to show that $\delta_A \geq 2/9$ when $x_A \geq 1/4$. Putting these together implies a two segment lower bound with phase transition at the largest root of the equation $3x - 6x^2 = \frac{2}{9}$ (i.e., at $\frac{1}{4} + \frac{\sqrt{33}}{36}$). This lower bound was used in the Max-3-SAT analyses of [BGLR] and [BeSu].

However, for our applications (i.e., linearity testing over $\mathcal{F}_l$ as in Lemma 3.5.3), the case of interest is when the underlying groups are $G = \mathrm{GF}(2)^n$ and $H = \mathrm{GF}(2)$ (since $\mathcal{F}_l$ may be identified with $\mathrm{GF}(2)^n$ for $n = 2^l$). The work of [BCHKS] focused on this case and improved the bound on δ_A for the case $x_A \geq \frac{1}{4}$ where $A: \mathrm{GF}(2)^n \to \mathrm{GF}(2)$. Specifically, they showed that $\delta_A \geq 45/128$ for $x_A \geq \frac{1}{4}$ which establishes the second segment of $\Gamma_{\rm lin}$. They also showed that $\delta_A \geq x_A$, for every $A: \mathrm{GF}(2)^n \to \mathrm{GF}(2)$. Combining the three lower bounds, they have derived the three-segment lower bound stated in Lemma 3.5.3.

The optimality of the above analysis has been demonstrated as well in [BCHKS]. Essentially[1], for every $x \leq 5/16$ there are functions $A: \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ witnessing $\delta_A = \Gamma_{\rm lin}(x_A)$ with $x_A = x$. In particular, for $x = 5/16$ (and $n \geq 4$), there is a function A with $1/4 < x_A < 1/2$ and $\delta_A = 45/128 = \Gamma_{\rm lin}(x_A)$. For the interval $(\frac{5}{16}, \frac{1}{2}]$, no tight results are known. Instead, [BCHKS] reports of computer constructed examples of functions $A: \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ with x_A in every interval $[\frac{k}{100}, \frac{k+1}{100}]$, for $k = 32, 33, ..., 49$, and $\delta_A < \Gamma_{\rm lin}(x_A) + \frac{1}{20}$. Furthermore, they showed that there exist such functions with both x_A and δ_A arbitrarily close to $\frac{1}{2}$.

3.5.2 Atomic respect of monomial basis test

Having determined that A is close to linear, the *atomic respect of monomial basis* test makes sure that the linear function close to A respects the monomial basis. Let us denote the latter function (i.e., the linear function closest to A) by $\tilde{A}$. Recalling Definition 3.3.1 we need to establish two things: namely, that $\tilde{A}(\chi_\emptyset) = 1$ and that $\tilde{A}(\chi_S) \cdot \tilde{A}(\chi_T) = \tilde{A}(\chi_{S \cup T})$, for every $S, T \subseteq [l]$. Recall that we do not have access to $\tilde{A}$ but rather to A; still, the Self-Correction Lemma provides an obvious avenue to bypass the difficulty provided $\mathrm{Dist}(A, \tilde{A}) < 1/4$. This would have yielded a solution but quite a wasteful one (alas sufficient for the Max Clique and Chromatic Number results). Instead, we adopt the following more efficient procedure.

Firstly, by considering only oracles folded over $(\bar{1}, 1)$, we need not check that $\tilde{A}(\chi_\emptyset) = 1$. (This follows by combining Corollary 3.5.2 and the fact that the $(\bar{1}, 1)$-folded oracle A satisfies $A(f + \bar{1}) = A(f) + 1$, for all $f \in \mathcal{F}_l$.) Secondly, we test that $\tilde{A}(\chi_S) \cdot \tilde{A}(\chi_T) = \tilde{A}(\chi_{S \cup T})$, for

[1]Actually, the statement holds only for x's which are integral multiple of 2^{-n}

every $S, T \subseteq [l]$, by taking random linear combinations of the S's and T's to be tested. Such linear combinations are nothing but uniformly selected functions in $\mathcal{F}_l$. Namely, we wish to test $\tilde{A}(f) \cdot \tilde{A}(g) = \tilde{A}(f \cdot g)$, where f and g are uniformly selected in $\mathcal{F}_l$. Since A is close to $\tilde{A}$, we can inspect $A(f)$ (resp., $A(g)$) rather than $\tilde{A}(f)$ (resp., $\tilde{A}(g)$) with little harm. However, $f \cdot g$ is not uniformly distributed (when f and g are uniformly selected in $\mathcal{F}_l$) and thus Self-Correction will be applied here. The resulting test is

$$A(f_1) \cdot A(f_2) = A(f_1 \cdot f_2 + f_3) - A(f_3) \tag{3.3}$$

This test was analyzed in the previous version of this work [BGS]; specifically, this test was shown to reject a folded oracle A with $\tilde{A}$ (the linear function closest to A) which does not respect the monomial basis with probability at least $(1-2x) \cdot (\frac{3}{8} - x + \frac{x^2}{2}) = \frac{3}{8} - \frac{7}{4}x + \frac{5}{2}x^2 - x^3$, where $x = \mathrm{Dist}(A, \tilde{A})$. Here we present an adaptive version of the above test, which performs even better. We observe that if $A(f_1) = 0$ then there is no need to fetch $A(f_2)$ (since the l.h.s. of Eq. (3.3) is zero regardless of $A(f_2)$). Thus, we merely test whether $A(f_1 \cdot f_2 + f_3) - A(f_3) = 0$. But what should be done if $A(f_1) = 1$? In this case we may replace f_1 by $f_1 + \bar{1}$ (yielding $A(f_1 + \bar{1}) = A(f_1) + 1 = 0$) and test whether $A((f_1 + \bar{1}) \cdot f_2 + f_3) - A(f_3) = 0$. The resulting test is depicted in Figure 3.2. To analyze the performance of this test, we need some technical lemmas. The reader may skip their proofs, in first reading, and proceed below to their usage (in Lemma 3.5.7).

Technical lemmas. First we recall the following lemma of [BGLR] which provides an improved analysis of Freivalds's matrix multiplication test in the special case when the matrices are symmetric with common diagonal.

Lemma 3.5.4 (symmetric matrix multiplication test [BGLR]): Let M_1, M_2 be N-by-N symmetric matrices over Σ which agree on their diagonals. Suppose that $M_1 \neq M_2$. Then

$$\Pr_{x,y \stackrel{R}{\leftarrow} \Sigma^N}[xM_1y \neq xM_2y] \geq \frac{3}{8}.$$

Furthermore, $\Pr_{x \stackrel{R}{\leftarrow} \Sigma^N}[xM_1 \neq xM_2] \geq 3/4$.

Proof: Let $M \stackrel{\text{def}}{=} M_1 - M_2$. The probability that a uniformly selected combination of the rows of M yields an all-zero vector is 2^{-r}, where r is the rank of M. Since M is symmetric, not identically zero and has a zero diagonal, it must have rank at least 2. Thus, $\Pr_{x \stackrel{R}{\leftarrow} \Sigma^N}[xM \neq 0^N] \geq 3/4$ and the lemma follows. ▌

Suppose that A is actually linear. In that case, the following lemma provides a condition under which A respects the monomial basis. We start with a definition.

Definition 3.5.5 (RMB detector): Let $A: \mathcal{F}_l \to \Sigma$ and $f \in \mathcal{F}_l$. We say that f is a detector for A if

$$\Pr_{g \stackrel{R}{\leftarrow} \mathcal{F}_l}[A(f' \cdot g) \neq 0] \geq 1/2.$$

where $f' = f$ if $A(f) = 0$ and $f' = f + \bar{1}$ otherwise.

The number of detectors is clearly related to the rejection probability of the RMB test. Suppose that A (or rather $\tilde{A}$) is linear. Clearly, if A respects the monomial basis then it has no detectors. On the other hand, the following lemma asserts that if A does not respect the monomial basis then it has many detectors.

Lemma 3.5.6 (RMB test for linear functions): Suppose $\tilde{A}\colon \mathcal{F}_l \to \Sigma$ is linear, $\tilde{A}(\chi_\emptyset) = 1$ and $\tilde{A}$ does not respect the monomial basis. Then at least a $3/4$ fraction of the functions in $\mathcal{F}_l$ are detectors for $\tilde{A}$.

Proof: Let $N = 2^l$. We define a pair of N-by-N matrices whose rows and columns are indexed by the subsets of $[l]$. Specifically, for $S,T \subseteq [l]$, we set

$$\begin{aligned} M_1[S,T] &= \tilde{A}(\chi_S)\cdot\tilde{A}(\chi_T) \\ M_2[S,T] &= \tilde{A}(\chi_{S\cup T})\,. \end{aligned}$$

Clearly, both M_1 and M_2 are symmetric, and they agree on the diagonal. Using $\tilde{A}(\chi_\emptyset) = 1$ we have, for every $T \subseteq [l]$,

$$M_1[\emptyset,T] = \tilde{A}(\chi_\emptyset)\cdot\tilde{A}(\chi_T) = 1\cdot\tilde{A}(\chi_T) = M_2[\emptyset,T] \tag{3.4}$$

By the hypothesis that $\tilde{A}$ does not respects the monomial basis it follows that $M_1 \neq M_2$. Our aim is to relate the inequality of the above matrices to the existence of detectors for $\tilde{A}$. We first express the condition $\tilde{A}(fg) = \tilde{A}(f)\cdot\tilde{A}(g)$ in terms of these matrices.

Recall that $\mathcal{C}\colon \mathcal{F}_l \to \Sigma^{2^l}$ is the transformation which to any $f \in \mathcal{F}_l$ associates the vector $(\,C_f(S)\,)_{S\subseteq[l]}$ whose entries are the coefficients of f in its monomial series. Using the linearity of $\tilde{A}$ we note that

$$\begin{aligned} \tilde{A}(f)\cdot\tilde{A}(g) &= \tilde{A}\left(\textstyle\sum_S C_f(S)\cdot\chi_S\right)\cdot\tilde{A}\left(\textstyle\sum_T C_g(T)\cdot\chi_T\right) \\ &= \left[\textstyle\sum_S C_f(S)\cdot\tilde{A}(\chi_S)\right]\cdot\left[\textstyle\sum_T C_g(T)\cdot\tilde{A}(\chi_T)\right] \\ &= \textstyle\sum_{S,T} C_f(S)\cdot\tilde{A}(\chi_S)\cdot\tilde{A}(\chi_T)\cdot C_g(T) \\ &= \mathcal{C}(f)M_1\mathcal{C}(g)\,. \end{aligned}$$

For the next step we first need the following.

Fact. Let $f,g \in \mathcal{F}_l$ and $U \subseteq [l]$. Then $C_{fg}(U) = \sum_{S\cup T=U} C_f(S)\cdot C_g(T)$.

Using this fact (and the linearity of $\tilde{A}$) we have:

$$\begin{aligned} \tilde{A}(fg) &= \tilde{A}\left(\textstyle\sum_U C_{fg}(U)\cdot\chi_U\right) \\ &= \textstyle\sum_U C_{fg}(U)\cdot\tilde{A}(\chi_U) \\ &= \textstyle\sum_U\sum_{S\cup T=U} C_f(S)\cdot C_g(T)\cdot\tilde{A}(\chi_U) \\ &= \textstyle\sum_{S,T} C_f(S)\cdot C_g(T)\cdot\tilde{A}(\chi_{S\cup T}) \\ &= \mathcal{C}(f)M_2\mathcal{C}(g)\,. \end{aligned}$$

Since $\tilde{A}$ is linear and $\tilde{A}(\bar{1}) = 1$ (as $\bar{1} = \chi_\emptyset$), we can rephrase the condition $A(f'\cdot g) \neq 0$, where $f' = f$ if $\tilde{A}(f) = 0$ and $f' = f + \bar{1}$ otherwise, as $A(f'\cdot g) \neq A(f')\cdot A(g)$. Thus, for every f (setting f' as above), we conclude that

$$A(f'\cdot g) \neq A(f')\cdot A(g) \ \text{ if and only if } \ \mathcal{C}(f')M_2\mathcal{C}(g) \neq \mathcal{C}(f')M_1\mathcal{C}(g)\,.$$

A key observation is that $\mathcal{C}(f)$ and $\mathcal{C}(f')$ are identical in all entries except, possibly, for the entry corresponding to $\emptyset$ (i.e., $C_f(S) = C_{f'}(S)$ for all $S \neq \emptyset$). On the other hand, by Eq. (3.4), we have $M_1[\emptyset,\cdot] = M_2[\emptyset,\cdot]$. Thus,

$$A(f'\cdot g) \neq A(f')\cdot A(g) \ \text{ if and only if } \ \mathcal{C}(f)M_2\mathcal{C}(g) \neq \mathcal{C}(f)M_1\mathcal{C}(g)\,.$$

Now we note that $\mathcal{C}$ is a bijection, so that if h is uniformly distributed in $\mathcal{F}_l$ then $\mathcal{C}(h)$ is uniformly distributed in Σ^{2^l}. Fixing any $f \in \mathcal{F}_l$ and setting f' as above, we have, for $x = \mathcal{C}(f)$,

$$\begin{aligned} \Pr_{g \xleftarrow{R} \mathcal{F}_l}\left[\tilde{A}(f') \cdot \tilde{A}(g) \;=\; \tilde{A}(f'g)\right] &= \Pr_{g \xleftarrow{R} \mathcal{F}_l}\left[\mathcal{C}(f)M_1\mathcal{C}(g) \;=\; \mathcal{C}(f)M_2\mathcal{C}(g)\right] \\ &= \Pr_{y \xleftarrow{R} \Sigma^{2^l}}\left[xM_1y \;=\; xM_2y\right] . \end{aligned}$$

where the latter probability is $1/2$ if $xM_1 \neq xM_2$ and zero otherwise. Invoking Lemma 3.5.4 we conclude that the first case, which coincides with f being a detector for $\tilde{A}$, holds for at least $3/4$ fraction of the $f \in \mathcal{F}_l$. The lemma follows. ▮

Lemma 3.5.6 suggests that if we knew A was linear we could test that it respects the monomial basis by picking f, g at random and testing whether $A(f'g) = 0$, where $f' = f$ if $A(f) = 0$ and $f' = f + \bar{1}$ otherwise. The lemma asserts that in case A is linear and does not respect the monomial basis we will have

$$\Pr_{f,g \xleftarrow{R} \mathcal{F}_l}\left[A(f'g) \neq 0\right] \geq \frac{3}{4} \cdot \frac{1}{2}$$

where $3/4$ is a lower bound on the probability that f is a detector for A and $\Pr_{g \xleftarrow{R} \mathcal{F}_l}\left[A(f'g) \neq 0\right] \geq \frac{1}{2}$ for any detector f (by definition). However, we only know that A is close to linear. Still we can perform an approximation of the above test via self-correction of the value $A(f'g)$. This, indeed, is our test as indicated in Figure 3.2.

The RMB test. We are interested in lower bounding the probability $1 - \text{MBPass}(A)$ that the test rejects when f_1, f_2, f_3 are chosen at random, as a function of the distance of A to a linear function $\tilde{A}$, given that $\tilde{A}$ does not respect the monomial basis. We assume that A satisfies $A(f + \bar{1}) = A(f) + 1$ (for all $f \in \mathcal{F}_l$), as is the case in all our applications (since we use verifiers which access a $(\bar{1}, 1)$-folded function). The first item of the following lemma is in spirit of previous analysis of analogous tests. The second item is somewhat unusual and will be used only in our construction of verifiers of free-bit complexity 2 (cf., Section 3.9).

Lemma 3.5.7 (RMB test – final analysis): Let $A, \tilde{A}\colon \mathcal{F}_l \to \Sigma$ with $\tilde{A}$ linear but not respecting the monomial basis and let $x = \text{Dist}(A, \tilde{A})$. Suppose that the function A satisfies $A(f + \bar{1}) = A(f) + 1$, for all $f \in \mathcal{F}_l$. Then

1. $1 - \text{MBPass}(A) \;\geq\; \Gamma_{\text{RMB}}(x) \stackrel{\text{def}}{=} \frac{3}{8} \cdot (1 - 2x)$.
2. $\Pr_{f_1, f_3 \xleftarrow{R} \mathcal{F}_l}\left[\exists f_2 \in \mathcal{F}_l \;\text{ s.t. }\; \mathbf{MBTest}(A; f_1, f_2, f_3) = 1\right] \;\geq\; 2 \cdot \Gamma_{\text{RMB}}(x)$.

In particular, the lemma holds for $A_{(h,0),(\bar{1},1)}$, where $A\colon \mathcal{F}_l \to \Sigma$ is arbitrary and $h \in \mathcal{F}_l$. We will consider the linear function closest to $A_{(h,0),(\bar{1},1)}$, denoted $\tilde{A}$, and the case in which $\tilde{A}$ does not respect the monomial basis. (In this case $\text{Dist}(A_{(h,0),(\bar{1},1)}, \tilde{A})) = \text{Dist}(A_{(h,0),(\bar{1},1)}, \text{Lin}) \leq 1/2$.)

Proof: As a preparation to using Lemma 3.5.6, we first show that $\tilde{A}(\bar{1}) = 1$. For $x < 1/2$ this is justified by Corollary 3.5.2 (using the hypothesis $A(f + \bar{1}) = A(f) + 1$, $\forall f \in \mathcal{F}_l$). Otherwise (i.e., in case $x \geq 1/2$) the claimed lower bound (i.e., $\frac{3}{8} \cdot (1 - 2x) \leq 0$) holds vacuously.

Using Lemma 3.5.6 and Lemma 3.5.1 we lower bound the rejection probability of the test as follows:

$$\begin{aligned} 1 - \text{MBPass}(A) \;\;\geq\;\; & \Pr_{f_1 \xleftarrow{R} \mathcal{F}_l}\left[f_1 \text{ is a detector for } \tilde{A}\right] \\ & \cdot \min_{f \text{ is a } \tilde{A}\text{-detector}} \left\{\Pr_{f_2, f_3 \xleftarrow{R} \mathcal{F}_l}\left[\mathbf{MBTest}(A; f, f_2, f_3) = 1\right]\right\} \end{aligned}$$

$$\begin{aligned}
&\geq \frac{3}{4} \cdot \min_{f \text{ is a } \tilde{A}\text{-detector}} \left\{ \Pr_{f_2, f_3 \xleftarrow{R} \mathcal{F}_l} [A(f' f_2 + f_3) \neq A(f_3)] \right\} \\
&\geq \frac{3}{4} \cdot \min_{f \text{ is a } \tilde{A}\text{-detector}} \left\{ \Pr_{f_2, f_3 \xleftarrow{R} \mathcal{F}_l} \left[0 \neq \tilde{A}(f' f_2) \;=\; A(f' f_2 + f_3) - A(f_3) \right] \right\} \\
&\geq \frac{3}{4} \cdot \frac{1}{2} \cdot \min_{f' \text{ and } g \text{ s.t. } \tilde{A}(f' g) \neq 0} \left\{ \Pr_{f_3 \xleftarrow{R} \mathcal{F}_l} \left[\tilde{A}(f' \cdot g) \;=\; A(f' \cdot g + f_3) - A(f_3) \right] \right\} \\
&\geq \frac{3}{8} \cdot (1 - 2x)
\end{aligned}$$

where the second inequality uses Lemma 3.5.6, the fourth inequality follows by the definition of a detector for $\tilde{A}$ (by which $\Pr_{g \xleftarrow{R} \mathcal{F}_l} \left[\tilde{A}(f'g) \neq 0 \right] \geq 1/2$), and the last inequality follows by Lemma 3.5.1. This concludes the proof of Part (1). Part (2) is proven analogously with the exception that we don't lose a factor of two in the fourth inequality (since here f_2 is not selected at random but rather set existentially). ∎

Remark: An RMB test for arbitrary A's (rather than ones satisfying $A(f + \bar{1}) = A(f) + 1, \forall f \in \mathcal{F}_l$) can be derived by augmenting the above test with a test of $A(f + \bar{1}) = A(f) + 1$ for uniformly chosen $f \in \mathcal{F}_l$. The analysis of the augmented part is as in the circuit test (below).

3.5.3 Atomic projection test

The final test checks that the second function A_1 is not too far from the evaluation operator E_{a_1} where $a_1 = \sigma(a)$ is a function of the string a whose evaluation operator is close to A. Here, unlike previous works (for instance [BeSu]), σ may be an arbitrary mapping from Σ^l to Σ^{l_1} rather than being a projection (i.e., satisfying $\sigma(x) = x^{(i_1)} \ldots x^{(i_{l_1})}$ for some sequence $1 \leq i_1 < \cdots < i_{l_1} \leq l$ and all $x \in \Sigma^l$). Thus, the name "projection test" is adopted for historical reasons.

Lemma 3.5.8 Let $A: \mathcal{F}_l \to \Sigma$ and let $\sigma: \Sigma^l \to \Sigma^{l_1}$ be a function. Let $a \in \Sigma^l$ and let $x = \text{Dist}(A, E_a)$. Let $a_1 = \sigma(a) \in \Sigma^{l_1}$. Then $1 - \text{ProjPass}_\sigma(A, A_1) \geq \text{Dist}(A_1, E_{a_1}) \cdot (1 - 2x)$.

Proof: We lower bound the rejection probability as follows:

$$\begin{aligned}
&\Pr_{f \xleftarrow{R} \mathcal{F}_l \,;\, g \xleftarrow{R} \mathcal{F}_{l_1}} [A_1(g) \neq A(g \circ \sigma + f) - A(f)] \\
&\quad \geq \Pr_{f \xleftarrow{R} \mathcal{F}_l \,;\, g \xleftarrow{R} \mathcal{F}_{l_1}} [A_1(g) \neq E_a(g \circ \sigma) \text{ and } A(g \circ \sigma + f) - A(f) = E_a(g \circ \sigma)] \\
&\quad \geq \Pr_{g \xleftarrow{R} \mathcal{F}_{l_1}} [A_1(g) \neq E_a(g \circ \sigma)] \cdot (1 - 2x) \,.
\end{aligned}$$

Here we used Lemma 3.5.1 in the last step. Now we note that $E_a(g \circ \sigma) = E_{a_1}(g)$. Hence the first term in the above product is just

$$\Pr_{g \xleftarrow{R} \mathcal{F}_{l_1}} [A_1(g) \neq E_{a_1}(g)] \;=\; \text{Dist}(A_1, E_{a_1}) \,.$$

This concludes the proof. ∎

3.5.4 Atomic circuit test

For sake of elegancy, we present also a Circuit Test, denoted $\mathbf{CircTest}_h(A; f)$. The test consists of checking whether $A(h+f) = A(f)$ and it outputs 0 if equality holds and 1 otherwise. Assuming that A is close to some evaluation operator E_a, the *atomic circuit test* (above) uses self-correction [BLR] to test that a given function h has value 0 at a. As explained above, this test is not needed since all our proof systems will use a $(h, 0)$-folding (of A) and thus will impose $h(a) = 0$. The analysis lower bounds the rejection probability, as a function of the distance of A from linear, given that $h(a) = 1$.

Lemma 3.5.9 Let $A\colon \mathcal{F}_l \to \Sigma$ and let $a \in \Sigma^l$. Let $h \in \mathcal{F}_l$ and $x = \mathrm{Dist}(A, E_a)$. If $h(a) = 1$ then $1 - \textsc{CircPass}_h(A) \geq 1 - 2x$, where

$$\textsc{CircPass}_h(A) \stackrel{\text{def}}{=} \Pr_{f \stackrel{R}{\leftarrow} \mathcal{F}_l} [\, \mathbf{CircTest}_h(A; f) = 0 \,]$$

Proof: Follows directly from Lemma 3.5.1. ▮

3.6 The MAX SNP verifier

In this section we present a simple verifier which performs one of two simple checks, each depending on only three queries. This verifier will be the basis for the non-approximability results regarding Max-3-SAT, Max-2-SAT and MaxCUT (presented in Section 3.7 and Section 3.8, respectively).

3.6.1 The inner verifier

Figure 3.3 describes an inner verifier. Our verifier is adaptive; that is, some of its queries are determine as a function of answers to previous queries. (The adaptivity is not obvious from Figure 3.3; it is rather 'hidden' in the RMB Test — see Section 3.5.2). We remark that this is the first time that one takes advantage on adaptivity in the construction of verifiers.

The inner verifier takes the usual length parameters l, l_1 as well as additional (probability) parameters p_1, p_2 and p_3 such that $p_1 + p_2 + p_3 = 1$. It performs just one test: with probability p_1 the linearity test; with probability p_2 the respect of monomial basis test; and with probability p_3 the projection test. Formally, this is achieved by picking p at random and making cases based on its value.[2] To improve the results, we perform the tests on a folding of A over both $(h, 0)$ and $(\bar{1}, 1)$ (i.e., on $A_{(h,0),(\bar{1},1)}$). We stress that $A_{(h,0),(\bar{1},1)}$ is a virtual oracle which is implemented by the verifier which accesses the actual oracle A (on points determined by the definition of folding). We now examine the goodness of V_{SNPinner}. Recall the definitions of $\Gamma_{\mathrm{lin}}(x)$ (specifically, note that $\Gamma_{\mathrm{lin}}(x) \geq x$) and $\Gamma_{\mathrm{RMB}}(x) = \frac{3}{8}(1-2x)$, for all x.

Informally, the following lemma considers all the possible strategies of a "dishonest" prover and indicates the probability (denoted $1-\rho$) with which the verifier detects an error (when run against such strategies). The three cases correspond to the events that

(1) the function $A_{(h,0),(\bar{1},1)}$ may be very far from being linear;

[2] For simplicity p is depicted as being chosen as a random real number between 0 and 1. Of course we cannot quite do this. But we will see later that the values of p_1, p_2, p_3 in our final verifiers are appropriate constants. So in fact an appropriate choice of p can be made using $O(1)$ randomness, which is what we will implicitly assume.

(2) the function $A_{(h,0),(\bar{1},1)}$ is x-close to linear, for some $x < \frac{1}{2} - \delta_1$, but is not x-close to a valid codeword (i.e., to a linear function which respect the monomial basis); and

(3) the function $A_{(h,0),(\bar{1},1)}$ is x-close to linear but the encoding of $\sigma(E^{-1}(A_{(h,0),(\bar{1},1)}))$ is very far from the function A_1.

Lemma 3.6.1 (soundness of V_{SNPinner}): Suppose $\delta_1, \delta_2 > 0$ and $l, l_1 \in \mathcal{Z}^+$. Suppose $p_1, p_2, p_3 \in [0,1]$ satisfy $p_1 + p_2 + p_3 = 1$. Then the (l, l_1)-canonical inner verifier V_{SNPinner} is $(\rho, \delta_1, \delta_2)$-good, where $1 - \rho = \min(T_1, T_2, T_3)$ and

(1) $T_1 \stackrel{\mathrm{def}}{=} p_1 \cdot (\frac{1}{2} - \delta_1)$

(2) $T_2 \stackrel{\mathrm{def}}{=} \min_{x \le 1/2 - \delta_1} [\, p_1 \cdot \Gamma_{\mathrm{lin}}(x) + p_2 \cdot \Gamma_{\mathrm{RMB}}(x) \,]$

(3) $T_3 \stackrel{\mathrm{def}}{=} \min_{x \le 1/2 - \delta_1} [\, p_1 \cdot \Gamma_{\mathrm{lin}}(x) + p_3 \cdot (\frac{1}{2} - \delta_2)(1 - 2x) \,]$.

Proof: We consider an arbitrary pair of oracles, (A, A_1), and the behavior of V_{SNPinner} when given access to this pair of oracles. Our analysis is broken up into cases depending on (A, A_1); specifically, the first case-partition depends on the distance of $A_{(h,0),(\bar{1},1)}$ (i.e., the folding of A) from linear functions. We show that, in each case, either the verifier rejects with probability bounded below by one of the three quantities (above) or the oracle pair is such that rejection is not required.

Let $x = \mathrm{Dist}(A_{(h,0),(\bar{1},1)}, \textsc{Lin})$.

The Max-SNP inner verifier. Given functions $h \in \mathcal{F}_l$ and $\sigma \colon \Sigma^l \to \Sigma^{l_1}$, the verifier has access to oracles for $A \colon \mathcal{F}_l \to \Sigma$ and $A_1 \colon \mathcal{F}_{l_1} \to \Sigma$. In addition it takes three $[0,1]$ valued parameters p_1, p_2 and p_3 such that $p_1 + p_2 + p_3 = 1$.

Pick $p \stackrel{R}{\leftarrow} [0,1]$.

Case: $p \le p_1$:
Pick $f_1, f_2 \stackrel{R}{\leftarrow} \mathcal{F}_l$.
LinTest$(A_{(h,0),(\bar{1},1)}; f_1, f_2)$.

Case: $p_1 < p \le p_1 + p_2$:
Pick $f_1, f_2, f_3 \stackrel{R}{\leftarrow} \mathcal{F}_l$.
MBTest$(A_{(h,0),(\bar{1},1)}; f_1, f_2, f_3)$.

Case: $p_1 + p_2 < p$:
Pick $f \stackrel{R}{\leftarrow} \mathcal{F}_l$ and $g \stackrel{R}{\leftarrow} \mathcal{F}_{l_1}$.
ProjTest$_\sigma(A_{(h,0),(\bar{1},1)}, A_1; f, g)$.

Remark: access to $A_{(h,0),(\bar{1},1)}(f)$ is implemented by accessing either $A(f)$, $A(f+h)$, $A(f+\bar{1})$ or $A(f + h + \bar{1})$.

Figure 3.3: *The Max-SNP inner verifier* V_{SNPinner}

Case 1: $x \geq \frac{1}{2} - \delta_1$

Lemma 3.5.3 implies that $1 - \textsc{LinPass}(A_{(h,0),(\bar{1},1)}) \geq \Gamma_{\rm lin}(x) \geq x \geq \frac{1}{2} - \delta_1$. (The second inequality follows from the fact that $\Gamma_{\rm lin}(x) \geq x$ for all x.) Since $V_{\rm SNPinner}$ performs the atomic linearity test with probability p_1 we have

$$1 - \texttt{ACC}[\, V^{A,A_1}_{\rm SNPinner}(\sigma, h)\,] \;\geq\; p_1 \cdot (\frac{1}{2} - \delta_1) \;\geq\; 1 - \rho \tag{3.5}$$

Case 2: $x \leq \frac{1}{2} - \delta_1$

Lemma 3.5.3 implies that $1 - \textsc{LinPass}(A_{(h,0),(\bar{1},1)}) \geq \Gamma_{\rm lin}(x)$ and so the probability that $V_{\rm SNPinner}$ performs the linearity test and rejects is at least $p_1 \cdot \Gamma_{\rm lin}(x)$. Now let $\tilde{A}$ be a linear function such that $\mathrm{Dist}(A_{(h,0),(\bar{1},1)}, \tilde{A}) = x$. We consider the following sub-cases.

Case 2.1: $\tilde{A}$ does not respect the monomial basis

In this case Part (1) of Lemma 3.5.7 implies that $1 - \textsc{MBPass}(A_{(h,0),(\bar{1},1)}) \geq \Gamma_{\rm RMB}(x)$. So the probability that $V_{\rm SNPinner}$ performs the atomic respect of monomial basis test and rejects is at least $p_2 \cdot \Gamma_{\rm RMB}(x)$. Since the event that the verifier performs a linearity test and the event that it performs a respect of monomial basis test are mutually exclusive, we can add the probabilities of rejection and thus get

$$1 - \texttt{ACC}[\, V^{A,A_1}_{\rm SNPinner}(\sigma, h)\,] \;\geq\; p_1 \cdot \Gamma_{\rm lin}(x) + p_2 \cdot \Gamma_{\rm RMB}(x) \;\geq\; 1 - \rho \tag{3.6}$$

Case 2.2: $\tilde{A}$ respects the monomial basis

By Proposition 3.3.2, $\tilde{A}$ is an evaluation operator. So there exists $a \in \Sigma^l$ such that $\tilde{A} = E_a$. So $\mathrm{Dist}(A_{(h,0),(\bar{1},1)}, E_a) = x$. Let $a_1 = \sigma(a)$. The proof splits into two further sub-cases.

Case 2.2.1: $d \stackrel{\rm def}{=} \mathrm{Dist}(A_1, E_{a_1}) \geq \frac{1}{2} - \delta_2$

By Lemma 3.5.8 we have $1 - \textsc{ProjPass}_\sigma(A_{(h,0),(\bar{1},1)}, A_1) \geq d \cdot (1 - 2x) \geq (1/2 - \delta_2) \cdot (1 - 2x)$. So the probability that $V_{\rm SNPinner}$ performs the projection test and rejects is at least $p_3 \cdot (1/2 + \delta_2)(1 - 2x)$. Thus, adding probabilities as in Case (2.1), we get

$$1 - \texttt{ACC}[\, V^{A,A_1}_{\rm SNPinner}(\sigma, h)\,] \;\geq\; p_1 \cdot \Gamma_{\rm lin}(x) + p_3 \cdot (1/2 - \delta_2)(1 - 2x) \;\geq\; 1 - \rho \tag{3.7}$$

Case 2.2.2: Else–

In this case, we have $x = \mathrm{Dist}(A_{(h,0),(\bar{1},1)}, E_a) \leq 1/2 - \delta_1$ and $\mathrm{Dist}(A_1, E_{a_1}) < 1/2 - \delta_2$. Thus the functions $A_{(h,0),(\bar{1},1)}$ and A_1 satisfy conditions (2.1) and (2.2) in Definition 3.4.3.

Observe that the only case which does not yield $1 - \texttt{ACC}[\, V^{A,A_1}_{\rm PCPinner}(\sigma, h)\,] \geq 1 - \rho$ is Case (2.2.2). However, Case (2.2.2) satisfies conditions (2.1) and (2.2) of Definition 3.4.3. Thus, $V_{\rm PCPinner}$ satisfies condition (2) of Definition 3.4.3. Clearly, $V_{\rm PCPinner}$ also satisfies condition (1) of Definition 3.4.3, and thus the lemma follows. ∎

The upper bound on the soundness error of $V_{\rm SNPinner}$, provided by Lemma 3.6.1, is somewhat complicated to grasp. Fortunately, using $\Gamma_{\rm RMB}(x) = \frac{3}{8}(1 - 2x)$ and $\Gamma_{\rm lin}(x) \geq x$, for all $x \leq 1/2$, we can simplify the expression as follows.

Claim 3.6.2 Let T_1, T_2 and T_3 be as in Lemma 3.6.1, $\delta = \max(\delta_1, \delta_2) > 0$ and $p_1, p_2, p_3 \in [0, 1]$ satisfy $p_1 + p_2 + p_3 = 1$. Then, $T_2 \geq \min\{\frac{1}{2}p_1, \frac{3}{8}p_2\}$, $T_3 \geq \min\{\frac{1}{2}p_1, \frac{1}{2}p_3\} - \delta$, and

$$\min\{T_1, T_2, T_3\} \geq \min\{\frac{1}{2}p_1, \frac{3}{8}p_2, \frac{1}{2}p_3\} - \delta$$

Interestingly, this lower bound is tight (see Claim 3.14.1).

Proof: Clearly, $T_1 = (\frac{1}{2} - \delta_1)p_1 \geq \frac{1}{2}p_1 - \delta$. To analyze T_2, let $h(x) \stackrel{\rm def}{=} p_1 \cdot \Gamma_{\rm lin}(x) + p_2 \cdot \Gamma_{\rm RMB}(x)$.

Fact 1: $\min_{x \leq 1/2}\{h(x)\} = \min\{\frac{3}{8}p_2, \frac{1}{2}p_1\} = \min\{h(0), h(1/2)\}$.

proof: by considering two cases and using $\Gamma_{\rm lin}(x) \geq x$ and $g(x) = \frac{3}{8} - \frac{3}{4}x$.

case 1: $p_1 \geq \frac{3}{4}p_2$

$$h(x) \geq p_1 x + \frac{3}{8}p_2 - \frac{3}{4}p_2 x = \frac{3}{8}p_2 + (p_1 - \frac{3}{4}p_2) \cdot x \geq \frac{3}{8}p_2$$

case 2: $p_1 \leq \frac{3}{4}p_2$

$$h(x) \geq p_1 x + \frac{3}{8}p_2 - \frac{3}{4}p_2 x = \frac{1}{2}p_1 + (\frac{3}{4}p_2 - p_1) \cdot (\frac{1}{2} - x) \geq \frac{1}{2}p_1$$

The fact follows by observing that $h(0) = \frac{3}{8}p_2$ and $h(1/2) = \frac{1}{2}p_1$. $\Box$

Thus, we have $T_2 = \min_{x \leq 1/2 - \delta_1}[h(x)] \geq \min\{\frac{1}{2}p_1, \frac{3}{8}p_2\}$. The term T_3 is analyzed similarly, by (re-)defining $h(x) \stackrel{\rm def}{=} p_1 \cdot \Gamma_{\rm lin}(x) + p_3 \cdot (1 - 2x)/2$, and using the following fact.

Fact 2: $\min_{x \leq 1/2}\{h(x)\} = \min\{\frac{1}{2}p_3, \frac{1}{2}p_1\} = \min\{h(0), h(1/2)\}$.

proof: by considering two cases and using $\Gamma_{\rm lin}(x) \geq x$.

case 1: $p_1 \geq p_3$

$$h(x) \geq p_1 x + \frac{1}{2}p_3 - p_3 x = \frac{1}{2}p_3 + (p_1 - p_3) \cdot x \geq \frac{1}{2}p_3$$

case 2: $p_1 \leq p_3$

$$h(x) \geq p_1 x + \frac{1}{2}p_3 - p_3 x = \frac{1}{2}p_1 + (p_3 - p_1) \cdot (\frac{1}{2} - x) \geq \frac{1}{2}p_1$$

The fact follows by observing that $h(0) = \frac{1}{2}p_3$ and $h(1/2) = \frac{1}{2}p_1$. $\Box$

Thus, we have $T_3 \geq \min_{x \leq 1/2 - \delta_2}[h(x)] - \delta \geq \min\{\frac{1}{2}p_1, \frac{1}{2}p_3\} - \delta$. The claim follows. ∎

3.6.2 Main application: the MaxSNP verifier

We are now ready to state the main result of this section. It is a simple verifier for NP which achieves soundness error approaching 85% while performing one of two very simple tests.

Proposition 3.6.3 (The MaxSNP Verifier): For any $\gamma > 0$ and for any language $L \in \mathrm{NP}$, there exists a verifier $V_{\rm SNP}$ for L such that

- $V_{\rm SNP}$ uses logarithmic randomness and is perfectly complete;
- $V_{\rm SNP}$ has soundness error $\frac{17}{20} + \gamma$; and
- on access to an oracle π (and according to the outcome of the verifier's coin tosses), the verifier $V_{\rm SNP}$ performs one of the following actions:

  (1) Parity check: $V_{\rm SNP}$ makes three queries q_1, q_2 and q_3, and rejects if $\pi(q_1) \oplus \pi(q_2) \neq \pi(q_3)$.

(2) RMB check: $V_{\rm SNP}$ makes three out of four determined queries q_1, q_2, q_3 and q_4, and rejects if either $(\pi(q_1) = 0) \wedge (\pi(q_2) \neq \pi(q_4))$ or $(\pi(q_1) = 1) \wedge (\pi(q_3) \neq \pi(q_4))$.
That is, the verifier inspect $\pi(q_1)$ and consequently checks either $\pi(q_2) = \pi(q_4)$ or $\pi(q_3) = \pi(q_4)$.

Furthermore, the probability (over its coin tosses) that $V_{\rm SNP}$ performs a parity check is $\frac{3}{5}$ (and the probability that $V_{\rm SNP}$ performs a RMB check is $\frac{2}{5}$).

Proof: Set $\delta_1 = \delta_2 = \gamma/2$ and $\epsilon = \frac{\gamma}{2} \cdot (16\delta_1^2\delta_2^2) = \frac{\gamma^5}{2} > 0$. Now, let l and l_1 be integers such that the outer verifier, $V_{\rm outer}$, guaranteed by Lemma 3.4.2 is (l, l_1)-canonical and ϵ-good for L. Consider the (l, l_1)-canonical inner verifier $V_{\rm SNPinner}$, working with the parameters p_1, p_2 and p_3 set to minimize its error. Obviously this calls for setting $\frac{1}{2}p_1 = \frac{3}{8}p_2 = \frac{1}{2}p_3$, which yields

$$p_1 = \frac{3}{10} \;\; ; \;\; p_2 = \frac{4}{10} \;\; ; \;\; p_3 = \frac{3}{10} \tag{3.8}$$

Let $V_{\rm SNP}$ be the verifier obtained by composing $V_{\rm outer}$ with $V_{\rm SNPinner}$.

We start by analyzing the soundness error of $V_{\rm SNP}$. By Lemma 3.6.1 and Claim 3.6.2, we know that the inner verifier $V_{\rm SNPinner}$, with p_i's as in Eq. (3.8), is $(\rho, \delta_1, \delta_2)$-good, for

$$\begin{aligned} \rho &\leq 1 - \frac{1}{2} \cdot p_3 + \delta_1 \\ &= 1 - \frac{3}{20} + \frac{1}{2} \cdot \gamma \end{aligned}$$

Invoking Theorem 3.4.5, we upper bound the soundness error of $V_{\rm SNP}$ by $1 - \frac{3}{20} + \frac{1}{2} \cdot \gamma + \frac{\epsilon}{16\delta_1^2\delta_2^2}$ which by the setting of ϵ yields the claimed bound.

It is left to observe that the projection test, performed by $V_{\rm SNPinner}$, amounts to a Parity Check on answers taken from two different oracles (which can actually be viewed as one oracle). It is clear that $V_{\rm SNP}$ uses logarithmic randomness, has perfect completeness, and performs the Parity Checks with probability $p_1 + p_3 = \frac{3}{5}$ and an RMB check with probability $p_2 = \frac{2}{5}$. ∎

A tedious remark: The probability that verifier $V_{\rm SNP}$, of the above proposition, makes two identical queries is negligible. Specifically, it can be made smaller than γ (mentioned in the proposition). Thus, we can ignore this case[3] in the next two sections and assume, without loss of generality, that all queries are distinct.

In the following sections we use the verifier of Proposition 3.6.3 to obtain hardness results for various variants of MaxSAT as well as for Max-CUT. The hardness results are obtained by constructing an instance of the given problem which represent the verifier's computation on input x. The primary aspect of the reduction is the construction of gadgets which reflect the result of the verifier's computation (i.e., accept/reject) after performing one of the two types of checks, i.e., parity check or RMB check. We define a performance measure of a gadget and then relate the final hardness result achieved to the performance measure obtained by the gadgets used. Given that the performance of the various gadgets might be different for the different checks, one might suspect that it might have been a better idea to first construct the gadgets and then to optimize the soundness of $V_{\rm SNP}$ keeping in mind the relative performance measures of the two kinds of gadgets being employed. Surprisingly enough it turns out (see Claim 3.14.2) that the optimization is not a function of the

[3] Formally, suppose that when it occurs the verifier performs some standard check on fixed different queries. This modification increases the soundness error by at most γ which tends to zero anyhow.

performance of the gadgets and indeed the choice of parameters p_1, p_2 and p_3 as in Equation (3.8) is optimal for the following reductions.

SOURCES OF OUR IMPROVEMENTS. The explicit statement of a generic verifier for deriving Max SNP hardness results is a novelty of our paper. Thus, a quantitative comparison to previous works is not readily available. Certainly, we improve over these works thanks to the use of the new long code based inner verifier, the atomic tests and their analysis in Section 3.5, the new idea of folding, and the improved analysis of linearity testing due to [BCHKS].

3.6.3 Another application: minimizing soundness error in 3-query pcp

As a direct corollary to Proposition 3.6.3, we obtain

Theorem 3.6.4 For any $s > 0.85$, $\mathrm{NP} \subseteq \mathrm{PCP}_{1,s}[\log, 3]$.
Furthermore, the free-bit complexity of the verifier is 2.

3.7 Satisfiability problems: Max-3-SAT and Max-2-SAT

In this section we mainly deal with DNF formulae, however the last subsection deals formulae consisting of a conjunction of PARITY (rather than OR) clauses.

3.7.1 Definitions

A *formula* is a set of clauses (i.e., or-clauses) over some set of literals. We consider various classes of formulae. In particular, 3-SAT formulae (at most three literals in each clause), E3-SAT formulae (exactly three different literals in each clause) and 2-SAT formulae (at most two literals in each clause). We use the generic notation X-SAT to stand for some unspecified class; thus the above correspond to $\mathrm{X} \in \{3, \mathrm{E3}, 2\}$.

Let φ be a formula. We let $|\varphi|$ denote the number of clauses in φ. We let $\mathsf{MaxSAT}(\varphi)$ denote the maximum number of clauses in S that are simultaneously satisfiable. (That is, the maximum, over all assignments to the variables, of the number of clauses satisfied). We also let $\overline{\mathsf{MaxSAT}}(\varphi) = \mathsf{MaxSAT}(\varphi)/|\varphi|$ denote the maximum fraction of simultaneously satisfiable clauses. Max-X-SAT is the problem, given a X-SAT instance φ, of finding $\mathsf{MaxSAT}(\varphi)$.

An approximation algorithm A for Max-X-SAT achieves a ratio, or factor, of $\alpha \in [1, \infty]$ if $(1/\alpha) \cdot \mathsf{MaxSAT}(\varphi) \le A(\varphi) \le \mathsf{MaxSAT}(\varphi)$ for all X-SAT instances φ.

Remark. As this definition indicates, we adopt the convention that the approximation factor is a number at least 1. Sometimes Max-SNP approximation is discussed in terms of factors at most 1 (e.g. [GoWi2, FeGo]) but obviously the two are equivalent via an inversion of the factor.

We are interested in promise versions of Max-X-SAT which exhibit a gap in the $\overline{\mathsf{MaxSAT}}(\cdot)$ value between YES and NO instances.

Definition 3.7.1 (MaxSAT promise problems): For any $0 \le s \le c \le 1$ we let the promise problem $\mathsf{Gap}\text{-X-SAT}_{c,s}$ be the pair (A, B), where–

(1) A is the set of all X-SAT instances φ satisfying $\overline{\mathsf{MaxSAT}}(\varphi) \ge c$, and

(2) B is the set of all X-SAT instances φ satisfying $\overline{\mathsf{MaxSAT}}(\varphi) \le s$.

The gap of this problem is defined to be c/s.

Our goal is to find such promise problems having gap as large as possible while being NP-hard. This will imply that the Max-X-SAT problem is hard to approximate within a factor equal to the reciprocal of the gap, unless P = NP.

Due to	Assuming	Factor	Technique
[ALMSS]	$\mathrm{P} \neq \mathrm{NP}$	some constant	$\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}[\log, O(1)]$; Reduction of this to Max-3-SAT.
[BGLR]	$\widetilde{\mathrm{P}} \neq \widetilde{\mathrm{NP}}$	94/93	Framework; better analyses; uses proof systems of [LaSh, FeLo].
[BGLR]	$\mathrm{P} \neq \mathrm{NP}$	113/112	New four-prover proof systems.
[FeKi]	$\mathrm{P} \neq \mathrm{NP}$	94/93	New two-prover proof systems.
[BeSu]	$\widetilde{\mathrm{P}} \neq \widetilde{\mathrm{NP}}$	66/65	Canonicity and some optimizations.
[BeSu]	$\mathrm{P} \neq \mathrm{NP}$	73/72	Canonicity and some optimizations.
This paper	$\mathrm{P} \neq \mathrm{NP}$	27/26	Long code and new proof systems.

Figure 3.4: *Non-approximability results for Max-3-SAT indicating the factor shown hard and the assumption under which this was done.*

3.7.2 Previous work

Approximation algorithms. Max-3-SAT is the canonical Max-SNP complete problem [PaYa]. A polynomial-time algorithm due to Yannakakis [Ya] approximates it to within a factor of $4/3 < 1.334$ (see Goemans and Williamson [GoWi1] for an alternate algorithm). Currently the best known polynomial-time algorithm for Max-3-SAT achieves a factor of 1.258 (and is due to Sorkin et. al. [SSTW] which in turn build on Goemans and Williamson [GoWi2]). For Max-E3-SAT, which is also Max-SNP complete, a very simple algorithm achieves an approximation of $8/7 \leq 1.143$ (where $7/8$ is the expected fraction of clauses satisfied by a uniformly chosen assignment).

Max-2-SAT is also Max-SNP complete [GJS, PaYa]. This problem is particularly interesting because it has been the focus of recent improvements in the approximation factor attainable in polynomial-time. Specifically, Goemans and Williamson [GoWi2] exhibited a polynomial time algorithm achieving an approximation factor of $\frac{1}{0.878} \approx 1.139$, and consequently Feige and Goemans [FeGo] exhibited an algorithm achieving $\frac{1}{0.931} \approx 1.074$.

Non-approximability. Non-approximability results for Max-SNP problems begin with [ALMSS] who proved that there exists a constant $\epsilon > 0$ such that $\mathsf{Gap}\text{-3-SAT}_{1,1-\epsilon}$ is NP-hard. They did this by providing a reduction from a given NP language L to the promise problem in question, constructed by encoding as a 3-SAT instance the computation of a $\mathrm{PCP}_{1,1/2}[\log, O(1)]$ verifier for an NP-complete language, the variables in the instance corresponding to bits in the proof string. The basic paradigm of their reduction has been maintained in later improvements.

Figure 3.4 depicts the progress. Improvements (in the constant value of the non-approximability factor) begin with [BGLR]. They used Hadamard code based inner verifiers following [ALMSS]. They also introduced a framework for better analysis, and improved some previous analyses; we exploit in particular their better analyses of linearity testing (cf. Section 3.5) and of Freivalds's matrix multiplication test (cf. Lemma 3.5.4). The improvement of Feige and Kilian [FeKi] was obtained via new proof systems; that of [BeSu] by use of the canonicity property of constant prover proofs and some optimizations. (See Section 3.4 for a discussion of the role of constant-prover proofs in this context).

Garey, Johnson and Stockmeyer [GJS] had provided, as early as 1976, a reduction of Max-3-SAT to Max-2-SAT which showed that if the former is non-approximable within $(k+1)/k$ then the latter is non-approximable within $(7k+1)/(7k)$. With the best previous non-approximability factor for Max-3-SAT (namely 66/65) we would only get a 456/455 factor non-approximability for Max-2-SAT. In fact, even using our new Max-3-SAT result we would only get only a hardness factor of 185/184.

3.7.3 New Results

A consequence of the following theorem is that, assuming $\mathrm{P} \neq \mathrm{NP}$ there is no polynomial time algorithm to approximate: (1) Max-3-SAT within a factor of 1.038; (2) Max-E3-SAT within a factor of 1.038; (3) Max-2-SAT within a factor of 1.013.

Theorem 3.7.2 (MaxSAT non-approximability results): The following promise problems are NP-hard –

(1) $\mathsf{Gap}\text{-3-SAT}_{c,s}$ with $c = 1$ and $s = 26/27$.
(2) $\mathsf{Gap}\text{-E3-SAT}_{c,s}$ with $c = 1$ and $s = 26/27$.
(3) $\mathsf{Gap}\text{-2-SAT}_{c,s}$ for some $0 < s < c < 1$ satisfying $c > 0.9$ and $c/s = 74/73$.

Actually, items (1) and (2) hold for any $s > 1 - \frac{3}{80}$ whereas item (3) holds as long as $\frac{c}{s} < 1 + \frac{3}{217}$. Item (1) is implied by item (2) so we will prove only the latter. The value of c for item (3) can be determined from our proof.

Sources of our improvements. The principal part of our improvement for Max-3-SAT comes from the use of the Max SNP verifier of the previous section. The latter verifier benefits from the use of the new long code based inner verifiers and the atomic tests and their analysis in Section 3.5. We also gain by using the new idea of folding and the improved analysis, due to [BCHKS], of the linearity test. Our Max-2-SAT result is based on the above as well as a new reduction which directly encodes the computation of the verifier in 2-SAT instances. Finally, for both Max-3-SAT and Max-2-SAT, an important feature of the optimization is explicit 3-SAT and 2-SAT expressions for the different tests which use as few clauses as possible. The expressions used for Max-3-SAT are in fact of E3-SAT form thus yielding the result for Max-E3-SAT.

3.7.4 Gadgets and the Hardness of MaxSAT

We need to implement two types of checks: the Parity Check (checking that $a+b=c$ for a, b and c obtained from the oracle) and the RMB-Check (checking that $b_a = c$ for a, b_0, b_1 and c obtained from the oracle). Accordingly a Parity Check (PC) gadget, $\mathrm{PC}(a,b,c,x_1,x_2,\ldots,x_n)$, is a set of clauses over three *distinguished* variables a, b, c and n auxiliary variables $x_1,\ldots,x_n$. It is an (α,β)-PC gadget if the following is true: If $a+b=c$ then $\mathsf{MaxSAT}(\mathrm{PC}(a,b,c,x_1,x_2,\ldots,x_n)) = \alpha$; else it is at most $\alpha - \beta$. Similarly a Respect-Monomial-Basis Check (RMBC) gadget, $\mathrm{RMBC}(a,b_0,b_1,c,x_1,\ldots,x_n)$, is a set of clauses over four *distinguished* variables a, b_0, b_1, c and n auxiliary variables $x_1,\ldots,x_n$. It is an (α,β)-RMBC gadget if the following is true: If $b_a = c$ then $\mathsf{MaxSAT}(\mathrm{RMBC}(a,b_0,b_1,c,x_1,x_2,\ldots,x_n)) = \alpha$; else it is at most $\alpha - \beta$. We stress that in both cases the maximum number of clauses which are simultaneously satisfied is at most α. A gadget is said to be a X-SAT gadget if, as a formula, it is a X-SAT formula.

The following lemma describes how a gadget of the above form can be used to obtain the hardness of MaxSAT.

Lemma 3.7.3 (MaxSAT implementation of a verifier): Let V be a verifier for L of logarithmic randomness, with perfect completeness and soundness s, such that V performs either a single Parity

Check (with probability q) or a single RMB check (with probability $1-q$). Furthermore, suppose that in either case, the verifier never makes two identical queries. If there exists an (α_1, β)-Parity-Check X-SAT gadget containing m_1 clauses and an (α_2, β)-RMBC X-SAT gadget containing m_2 clauses then L reduces to Gap-X-SAT$_{c',s'}$ for

$$c' = \frac{\alpha_1 q + \alpha_2(1-q)}{m_1 q + m_2(1-q)}$$

$$s' = \frac{\alpha_1 q + \alpha_2(1-q) - (1-s)\beta}{m_1 q + m_2(1-q)}$$

In particular $\frac{c'}{s'} \geq 1 + \frac{(1-s)\beta}{\alpha_1 q + \alpha_2(1-q) - (1-s)\beta}$.

Remark: In the above lemma, we have assumed that both the PC and RMBC gadgets have the same second parameter β. This assumption is not really a restriction since we can transform a pair of a (α_1, β_1)-PC gadget and (α_2, β_2)-RMBC gadget into a pair of a $(\alpha_1\beta_2, \beta_1\beta_2)$-PC gadget and a $(\alpha_2\beta_1, \beta_1\beta_2)$-RMBC gadget, thereby achieving this feature. (Actually, what really matters are the fractions α_i/β.)

Proof: Let PC$(a, b, c, x_1, \ldots, x_{n_1})$ denote the Parity Check gadget and let RMBC$(a, b, c, d, x_1, \ldots, x_{n_2})$ denote the RMBC gadget. We encode V's computation on input x by a CNF formula φ_x. Corresponding to every bit $\pi[q]$ of the proof (oracle) accessed by the verifier V we create a variable $y[q]$. In addition we create some auxiliary variables $y_{\mathrm{Aux}}[R, i]$ for each random string R used by the verifier V and i going from 1 to $\max(n_1, n_2)$. For each such R we will construct a formula φ_R which encodes the computation of the verifier when its coins are R. The union of all these formulae will be our φ_x.

On random string R if the verifier performs a parity check on bits $\pi[q_1], \pi[q_2]$ and $\pi[q_3]$, then φ_R consists of the clauses $PC(y[q_1], y[q_2], y[q_3], y_{\mathrm{Aux}}[R, 1], \ldots, y_{\mathrm{Aux}}[R, n_1])$. On the other hand if the verifier performs a RMB check on bits $\pi[q_1], \pi[q_2], \pi[q_3], \pi[q_4]$, then φ_R consists of the clauses $RMBC(y[q_1], y[q_2], y[q_3], y[q_4], y_{\mathrm{Aux}}[R, 1], \ldots, y_{\mathrm{Aux}}[R, n_2])$.

Let N denote the number of possible random strings used by V. Observe that the number of clauses in φ_x equals $m_1 \cdot qN + m_2 \cdot (1-q)N$. We now analyze the value of $\mathsf{MaxSAT}(\varphi_x)$.

If $x \in L$ then there exists an oracle π such that $V^\pi(x)$ always accepts. Consider the assignment $y[q] = \pi[q]$ (i.e., $y[q]$ is true iff $\pi[q] = 1$). Then for every R, there exists an assignment to the variables $y_{\mathrm{Aux}}[R, i]$'s such that the number of clauses of φ_R that are satisfied by this assignment is α_1 if R corresponds to a Parity Check and α_2 if R corresponds to a RMB-check. Since qN of the gadgets are PC-gadgets and $(1-q)N$ of the gadgets are RMBC-gadgets, we have $\mathsf{MaxSAT}(\varphi_x) \geq qN\alpha_1 + (1-q)N\alpha_2$, and the expression for c' follows.

Now consider the case when $x \notin L$. We claim that if there exists an assignment which satisfies $qN\alpha_1 + (1-q)N\alpha_2 - (1-s)N\beta$ clauses of φ_x, then there exists an oracle π such that $V^\pi(x)$ accepts with probability at least s. Since we know this can not happen we conclude that $\mathsf{MaxSAT}(\varphi_x) < qN\alpha_1 + (1-q)N\alpha_2 - (1-s)N\beta = s'|\varphi_x|$.

To prove the claim, we convert any assignment to the variables y's into an oracle π in the natural way, i.e., $\pi[q] = 1$ iff $y[q]$ is true. Now by the property of the gadgets if a PC gadget $PC(y[q_1], y[q_2], y[q_3], y_{\mathrm{Aux}}[R, 1], \ldots)$ has more than $\alpha_1 - \beta$ clauses satisfied then $\pi[q_1] \oplus \pi[q_2] = \pi[q_3]$. In turn this implies that the verifier V accepts π on random string R. A similar argument can be made about the random strings R which correspond to RMB checks. We also use the property that a PC (resp., RMB) gadget cannot have more than α_1 (resp., α_2) satisfied clauses, even if the claim it checks does hold. Thus, if an assignment satisfies $qN \cdot (\alpha_1 - \beta) + (1-q)N \cdot (\alpha_2 - \beta) + sN\beta$ clauses, then there must exist sN random strings R on which V accepts. This proves the claim and the lemma follows. ∎

Figure 3.5 describes gadgets which will be used for our Max-E3-SAT construction: notice they are *exact*-3-SAT gadgets. We have a $(4,1)$-PC gadget PC_3 consisting of 4 clauses and a $(4,1)$-RMB gadget RMBC_3 consisting of 4 clauses in which all the clauses have exactly three variables. Both gadgets have no auxiliary variables. The $\mathrm{PC}_3(a,b,c)$ gadget is merely the canonical 3CNF of the expression $a+b+c=0$. The first two clauses in the $\mathrm{RMBC}_3(a,b,b',c)$ gadget are the canonical 3CNF of the expression $(a=0) \Rightarrow (b=c)$, whereas the latter two clauses are the canonical 3CNF of the expression $(a=1) \Rightarrow (b'=c)$. Figure 3.6 similarly describes 2-SAT gadgets for our Max-2-SAT construction. We have a $(11,1)$-PC gadget PC_2 consisting of 12 clauses, and a $(11,1)$-RMB gadget RMBC_2 consisting of 12 clauses. Each gadget has four auxiliary variables. The auxiliary variable $x_{\tau\sigma}$ in the PC_2 gadget is supposed to be the indicator of the event $((a=\sigma)\wedge(b=\tau))$. Thus, $a+b=c$ allows to satisfy 11 clauses by appropriately setting the indicator variables (e.g., if $a=b=c=0$ then setting $x_{00}=1$ and the other $x_{\tau\sigma}$'s to 0 satisfies all clauses except the last one). The RMBC_2 gadget is composed of two parts; the first six clauses handle the expression $(a=0) \Rightarrow (b=c)$, whereas the latter six clauses are for the expression $(a=1) \Rightarrow (b'=c)$.

Lemma 3.7.4 (SAT gadgets): The following gadgets exist

- E3-SAT gadgets: a $(4,1)$-PC gadget of 4 clauses and a $(4,1)$-RMB gadget of 4 clauses.
- 2-SAT gadgets: a $(11,1)$-PC gadget of 12 clauses and a $(11,1)$-RMB gadget of 12 clauses.

Remark: a ratio of 4 between the number of clauses and the second parameter (i.e., β) is minimal for both E3-SAT gadgets. More generally, we claim that for E3-SAT, an (α,β)-gadget with m clauses for a test which holds with probability $1/2$ (for a random assignment to the distinguished variables) must satisfy $m \geq 4\beta$. Note that both the Parity test and the RMB test satisfy the condition of the claim. The claim is proven by considering the expected number of clauses satisfied by a random assignment to all variables of a gadget. We may assume, without loss of generality, that no clause is a tautology and thus no clause may contain different literals of the same variable. Thus, each clause contains three literals belonging to three different variables and is satisfied with probability $7/8$. It follows that the expected number of unsatisfied clauses under a random assignment which does not satisfy the test is at most $m/4$. Therefore there exists an assignment to the distinguished variables which does not satisfy the test and yet the auxiliary variables can be set to satisfy at least $\frac{3}{4}m$ of the clauses of the gadget. Thus, $\beta \leq m/4$ and if one wants to derive results for $\textsf{Gap}\text{-E3-SAT}_{1,s'}$

The Max-E3-SAT Gadgets.

$$\mathrm{PC}_3(a,b,c) = \{(a \vee b \vee \overline{c}),\ (a \vee \overline{b} \vee c),\ (\overline{a} \vee b \vee c),\ (\overline{a} \vee \overline{b} \vee \overline{c})\}$$

$$\mathrm{RMBC}_3(a,b,b',c) = \{(a \vee b \vee \overline{c}),\ (a \vee \overline{b} \vee c),\ (\overline{a} \vee b' \vee \overline{c}),\ (\overline{a} \vee \overline{b'} \vee c),\ \}$$

Figure 3.5: *The Max-E3-SAT Gadgets*

then $\alpha \geq 4\beta$ follows. Many questions arise. In particular, can one get below the $\alpha/\beta = 4$ ratio for 3-SAT (or even for E3-SAT when giving away the requirement that α equals the number of clauses). What about 2-SAT? In general, it will be interesting to find the best possible gadgets (in terms of lowest α/β ratio) for both tests and all formula classes and to prove that these gadgets are really the best possible.

Proof of Lemma 3.7.4: We use the gadgets presented in Figure 3.5 and Figure 3.6. The claim regarding E3-SAT follows from the motivating discussion above (i.e., by which these gadgets are merely the canonical 3CNF expressions for the corresponding conditions). Thus, the E3-SAT gadgets are satisfiable if and only if the corresponding condition (i.e., parity or RMB) holds, and the first part of the lemma follows.

We now turn to the 2-SAT gadgets in Figure 3.6, starting with the PC-gadget $\mathrm{PC}_2(a, b, c, x_{00}, x_{01}, x_{10}, x_{11})$.

- We first claim that if $a + b = c$ then we can satisfy 11 clauses. This is done by setting each $x_{\tau\sigma}$ to 1 if and only if both $a = \sigma$ and $b = \tau$. Clearly, this assignment satisfies the three clauses in which the variable x_{ab} appears (the first two by a and b and the last by x_{ba}). Out of the other 9 clauses, 6 (i.e., those in which an auxiliary variable appears negated) are satisfied by the 0-assignment to the other 3 auxiliary variables, and 2 (i.e., of the 3 in which an auxiliary variable appears unnegated) are satisfied by the variable c.
- We next claim that no assignment for which $a + b = c$ can satisfy all 12 clauses. Let $a = \sigma$, $b = \tau$ and $c = \sigma + \tau$ be an arbitrary partial assignment and consider the three clauses in which the variable $x_{\overline{\tau}\,\overline{\sigma}}$ appears. To satisfy any of the first two clauses we must have $x_{\overline{\tau}\,\overline{\sigma}} = 0$ but this cannot satisfy the third clause unless $c \neq \sigma + \tau$, in contradiction to our hypothesis.
- Finally, we show that no assignment for which $a + c \neq c$ can satisfy more than 10 clauses. Let $a = \sigma$, $b = \tau$ and $c = 1 + \sigma + \tau$ be an arbitrary partial assignment and consider the three clauses in which the variable $x_{\tau\overline{\sigma}}$ appears. To satisfy the first clause we must have $x_{\tau\overline{\sigma}} = 0$

The MAX 2SAT Gadgets.

$$\begin{aligned}
&\mathrm{PC}_2(a, b, c, x_{00}, x_{01}, x_{10}, x_{11}) = \\
&\quad \{(\overline{x_{00}} \vee \overline{a}), (\overline{x_{00}} \vee \overline{b}), (x_{00} \vee c), \\
&\qquad (\overline{x_{01}} \vee a), (\overline{x_{01}} \vee \overline{b}), (x_{01} \vee \overline{c}), \\
&\qquad (\overline{x_{10}} \vee \overline{a}), (\overline{x_{10}} \vee b), (x_{10} \vee \overline{c}), \\
&\qquad (\overline{x_{11}} \vee a), (\overline{x_{11}} \vee b), (x_{11} \vee c)\}
\end{aligned}$$

$$\begin{aligned}
&\mathrm{RMBC}_2(a, b, b', c, x_{00}, x_{11}, y_{00}, y_{11}) = \\
&\quad \{(\overline{x_{00}} \vee \overline{b}), (\overline{x_{00}} \vee \overline{c}), (a \vee x_{00}), \\
&\qquad (\overline{x_{11}} \vee b), (\overline{x_{11}} \vee c), (a \vee x_{11}), \\
&\qquad (\overline{y_{00}} \vee \overline{b'}), (\overline{y_{00}} \vee \overline{c}), (\overline{a} \vee y_{00}), \\
&\qquad (\overline{y_{11}} \vee b'), (\overline{y_{11}} \vee c), (\overline{a} \vee y_{11})\}.
\end{aligned}$$

Figure 3.6: *The Max-2-SAT Gadgets*

but this cannot satisfy the third clause unless $c = \sigma + \tau$, in contradiction to our hypothesis. Applying the same analysis to the clauses in which the variable $x_{\overline{\tau}\sigma}$ appears, the claim follows.

Finally, we consider the RMB-gadget $\mathrm{RMB}_2(a, b, b', c, x_{00}, x_{11}, y_{00}, y_{11})$. This gadget is the conjunction of two analogous 2CNF formulae, each consisting of six clauses. We first consider the first six clauses and the expression $(a = 0) \Rightarrow (b = c)$.

- Suppose $a = 1$. Then, regardless of the values of b and c, we can satisfy all six clauses by setting $x_{00} = x_{11} = 0$.
- Suppose $a = 0$ and $b = c = \sigma$, for $\sigma \in \{0, 1\}$. Then, we can satisfy five out of the six clauses by setting $x_{\sigma\sigma} = 1$ and $x_{\overline{\sigma}\overline{\sigma}} = 0$. On the other hand, it is not possible to satisfy all six clauses, since this requires setting $x_{00} = x_{11} = 1$ (to satisfy the 3rd and 6th clauses), which in turn requires setting both b and $\overline{b}$ to 1.
- Suppose $a = 0$ and $b \neq c$. In this case we claim that no truth assignment can satisfy more than 4 clauses. The claim is proven by contradiction. We already know that no truth assignment can satisfy all clauses. Suppose that some truth assignment satisfies five (or more) clauses. Then, for some $\sigma \in \{0, 1\}$, we must satisfy all clauses in which the variable $x_{\sigma\sigma}$ appears. This requires setting $x_{\sigma\sigma} = 1$ (to satisfy the 3rd or/and the 6th clause), which in turn forces us to set b and c to σ (to satisfy the other two clauses), in contradiction to the case hypothesis.

The last six clauses are analyzed analogously. We conclude that if the RMB condition holds then we can satisfy $6 + 5 = 11$ clauses and that we can satisfy at most 11 clauses. Furthermore, in case the RMB condition does not hold we can satisfy at most $4 + 6 = 10$ clauses. The lemma follows. ∎

Proof of Theorem 3.7.2: The theorem follows by applying Lemma 3.7.3 to the verifier of Proposition 3.6.3 and the gadgets of Lemma 3.7.4. Details follows.

Recall that by the remark following the proof of Proposition 3.6.3, we may assume that the verifier does not make two identical queries. Applying Lemma 3.7.3 to the verifier of Proposition 3.6.3 we obtain a reduction of any language in NP to $\mathsf{Gap}\text{-X-SAT}_{c',s'}$ for values of c' and s' determined as a function of the gadget parameters, the probability parameter q and the soundness s of the verifier of Proposition 3.6.3. Specifically, we observe that for E3-SAT we have $c' = 1$ (since $\alpha_i = m_i$ for $i = 1, 2$), whereas for 2-SAT we have $0.9 < c' < 1$ (since $\frac{\alpha_i}{m_i} = \frac{11}{12}$ for $i = 1, 2$). In both cases, $\beta = 1$ and the expression for c'/s' is given by

$$1 + \frac{1 - s}{q\alpha_1 + (1 - q)\alpha_2 - (1 - s)} \tag{3.9}$$

where s and q are determined by Proposition 3.6.3; that is (for every $\gamma > 0$)

$$s = 1 - \frac{3}{20} + \gamma \tag{3.10}$$

$$q = \frac{3}{5} \tag{3.11}$$

Substituting Eq. (3.10) and (3.11) in Eq. (3.9), and letting $\gamma \to 0$, we get

$$\frac{c'}{s'} \to 1 + \frac{3}{12\alpha_1 + 8\alpha_2 - 3}.$$

The bounds for E3-SAT and 2-SAT now follow by using the α_i's values of Lemma 3.7.4. In particular, for E3-SAT we get $s' \to 77/80$ and for 2-SAT we get $\frac{c'}{s'} \to 1 + \frac{3}{217}$. ∎

We conclude this subsection by presenting a variant of Lemma 3.7.3. This variant refers only to 3SAT formulii, but makes no restrictions on the verifier in the PCP system.

Lemma 3.7.5 (Max3SAT implementation of a generic verifier): Let $L \in \mathrm{PCP}_{1,1-\delta}[\log, 3]$, for some $0 < \delta < 1$. Then, L reduces to $\mathsf{Gap}\text{-}3\text{-}\mathrm{SAT}_{1,1-\frac{\delta}{4}}$.

Proof: Let V be a verifier as guaranteed by the hypothesis. Building on Lemma 3.7.3, it suffices to show that the computation of V on any possible random-tape can be captured by a 3CNF formula with at most 4 clauses. As a warm-up consider the special case in which V is non-adaptive. In case the canonical CNF has at most 4 clauses we are done. Otherwise, we let the first variable be indetermined and consider all 4 possible assignments to the other two variables. The formula may be indetermined in which case we introduce a single 3-clause or identically FALSE (resp., TRUE) in which case we introduce a single 2-clause (resp., no clause at all).

The general case, in which V is adaptive, is handled analogously. We consider the depth-3 branching program which describes the acceptance of V on a specific random tape. In case this tree has at most 4 rejecting leaves (i.e., marked FALSE) writing corresponding 3CNF clauses (which state that these paths are not followed) we are done. Otherwise, we consider the 4 depth-1 subtrees. For each such subtree we do the following. In case both leaves are marked FALSE we write a 2CNF clause (which states that this subtree is not reached at all). In case a single leaf is marked FALSE we write one 3CNF clause (as above), and if no leaf is marked FALSE we write nothing. ∎

3.7.5 Maximum Satisfiable Linear Constraints (Parity Clauses)

Analogously to the MaxSAT problems considered above, we consider parity/linear clauses rather than disjunctive clauses. In other words, we are given a system of linear equations over GF(2), and need to determine the maximum number of equations which may be simultaneously satisfied.

The maximization problem is known to be Max-SNP complete (see [BrNa] or [Pet]). Here we provide a stronger bound via a direct reduction from the MaxSNP verifier. Before continuing, we remark that the problem of *maximizing* the number of satisfiable equations should not be confused with the "complementary" problem of *minimizing* the number of violated constraints, investigated by Arora et. al. [ABSS]. Also the case of maximum satisfiable linear constraints over larger fields (of size q) has been considered by Amaldi and Kann [AmKa], who show that this problem is hard to approximate to within a factor of q^ϵ for some universal $\epsilon > 0$.

Theorem 3.7.6 Let $\mathsf{GapParity}_{c,s}$ be defined analogously to the above. Then, for $c = 6/7$ and $\frac{c}{s} < 8/7$, $\mathsf{GapParity}_{c,s}$ is NP-hard.

Proof: The theorem follows by constructing appropriate gadgets. A PC-gadget is straightforward here and so we have a $(1,1)$-PC gadget. We conclude by presenting a $(3,2)$-RMB gadget consisting of 4 equations. Specifically, for $\mathrm{RMB}(a, b_0, b_1, c)$ we present the equations $b_0 + c = 0$, $a + b_0 + c = 0$, $b_1 + c = 0$ and $a + b_1 + c = 1$.

We claim that these 4 equations are indeed a $(3,2)$-gadget for $b_a = c$. First observe that if $a = 0$ and $b_0 = c$ (resp., if $a = 1$ and $b_1 = c$) then the first (resp., last) two equations hold. On the other hand, if $a = 0$ and $b_0 \neq c$ (resp., if $a = 1$ and $b_1 \neq c$) then the first (resp., last) two equations are both violated. Finally, if $a = 0$ (resp., if $a = 1$) then, regardless of the values of b_0, b_1, c, exactly one of the last (resp., first) two equations hold. Thus, the claim holds. Observe that we can think of the RMB gadget as a $(1.5, 1)$-gadget with 2 clauses (or, equivalently, think of the parity gadget as a $(2,2)$-gadget with 2 clauses).

Proceeding as in the proof of Theorem 3.7.2, we obtain a hardness for $\mathsf{GapParity}_{c',s'}$, where

$$\frac{c'}{s'} \to 1 + \frac{3}{12\alpha_1 + 8\alpha_2 - 3} = 1 + \frac{3}{12 + 12 - 3} = \frac{8}{7}$$

and $c' = \frac{3\alpha_1+2\alpha_2}{3m_1+2m_2} = \frac{6}{7}$. ∎

3.8 Max-CUT

3.8.1 Definitions

A **cut** in a graph $G = (V, E)$ is a partition of the vertex set into sets S and $\overline{S}$. Given an assignment of weights $w : E \to \mathcal{R}^+$, the weight of a cut $(S, \overline{S})$ is the sum of the weights of the edges with one endpoint in S and the other in $\overline{S}$. We let $\mathsf{MaxCUT}(G, w)$ denote the maximum weight of any cut in G for a weight assignment w. Let $\overline{\mathsf{MaxCUT}}(G, w)$ denote the quantity $\mathsf{MaxCUT}(G, w)/\sum_e w(e)$. Max CUT is the problem whose instances are the pairs (G, w), where G is a graph and w a weight assignment on it, and one has to find $\mathsf{MaxCUT}(G, w)$. An approximation algorithm A for Max-CUTachieves a ratio of $\alpha \in [1, \infty)$ if $\mathsf{MaxCUT}(G, w)/\alpha \leq A(G, w) \leq \mathsf{MaxCUT}(G, w)$ for all instances (G, w). As usual, we capture the approximation problem by a promise problem –

Definition 3.8.1 (MaxCUT promise problem): For any $0 \leq s \leq c \leq 1$, we let the promise problem $\mathsf{Gap}\text{-}\mathrm{Cut}_{c,s}$ be the pair (A, B), where:

(1) A is the set of Max-CUTinstances satisfying $\overline{\mathsf{MaxCUT}}(G, w) \geq c$.

(2) B is the set of Max-CUTinstances satisfying $\overline{\mathsf{MaxCUT}}(G, w) \leq s$.

The **gap** of this problem is defined to be c/s.

3.8.2 Previous work

In 1976, Sahni and Gonzales [SaGo] gave a simple 2-approximation algorithm for this problem. Recently, in a breakthrough result, Goemans and Williamson [GoWi2] gave a new algorithm which achieves a ratio of $\frac{1}{0.878} = 1.139$ for this problem. On the other hand, [PaYa] give an approximation preserving reduction from Max-3-SAT to Max-CUT. Combined with [ALMSS] this shows that there exists a constant $\alpha > 1$ such that approximating Max-CUTwithin a factor of α is NP-hard. No explicit bounds were given since and even using the best known hardness results for MAX 3SAT, one suspects that the bound for Max-CUTwould not be very large, since the reduction uses constructions of constant degree expanders etc.

3.8.3 New Result

We get the first explicit lower bounds on the constant upto which approximating the Max-CUTproblem is NP-hard. We show in the following theorem that the Max-CUTproblem is NP-hard to approximate to within a factor of 1.012. The following theorem presents a non-approximability result for a weighted graph. We stress that it holds even when the weights are given in unary.

Theorem 3.8.2 (Max-CUTnon-approximability result): $\mathsf{Gap}\text{-}\mathrm{Cut}_{c,s}$ is NP-hard for some c, s satisfying $c > 0.6$ and $c/s > 66/65$.

Actually, the theorem holds for any $c/s < 1 + \frac{3}{193}$. A weaker result can be obtained for simple graphs without weights or parallel edges. In particular, one may reduce the Max-CUTproblem for graphs with parallel edges to Max-CUTfor simple graphs, by replacing every edge by a path of 3 edges. This causes a loss of a factor of 3 in the hardness factor; that is, we would get a hardness factor of 194/193 for the Max-CUTproblem restricted to simple graphs. A better reduction which preserves the non-approximation ratio has been recently suggested by Crescenzi et. al. [CST].

3.8.4 Gadgets and the hardness of Max-CUT

Gadgets will be used to express the verifier's computation in terms of cuts in graphs. A parity check gadget PC-CUT$(a, b, c, T; x_1, \ldots, x_n)$ is a weighted graph on $n + 4$ vertices. Of these three vertices a, b, c correspond to oracle queries made by the verifier. The vertex T will be a special vertex mapping cuts to truth values so that a vertex corresponding to an oracle query is considered set to 1 if it resides in the T-side of the cut (i.e., a is considered set to 1 by a cut $(S, \overline{S})$ iff either $a, T \in S$ or $a, T \in \overline{S}$). The gadget is an (α, β)-PC gadget if MaxCUT(PC-CUT$(a, b, c, T; x_1, \ldots, x_n)$) is exactly α when restricted to cuts which induce $a + b = c$ (i.e., either 0 or 2 of the vertices $\{a, b, c\}$ lie on the same side of the cut as T), and is at most $\alpha - \beta$ when restricted to cuts for which $a + b \neq c$. Similarly a weighted graph RMBC-CUT$(a, b_0, b_1, c, T; x_1, \ldots, x_n)$ is an (α, β)-RMBC gadget if it satisfies the property that MaxCUT(RMBC-CUT$(a, b_0, b_1, c, T; x_1, \ldots, x_n)$) is exactly α when restricted to cuts satisfying $b_a = c$ and is at most $\alpha - \beta$ otherwise. The following lemma (similar to Lemma 3.7.3) shows how to use the above forms of gadgets to derive a reduction from NP to Gap-Cut.

Lemma 3.8.3 (MaxCUT implementation of a verifier): Let V be a verifier for L of logarithmic randomness, with perfect completeness and soundness s, such that V performs either a single Parity Check (with probability q) or a single RMB check (with probability $1 - q$). Furthermore, suppose that in either case, the verifier never makes two identical queries. If there exists an (α_1, β)-PC gadget consisting of edges of total weight w_1 and an (α_2, β)-RMBC gadget consisting of edges of total weight w_2 then L reduces to Gap-$\mathrm{Cut}_{c',s'}$ for $c' = \frac{\alpha_1 q + \alpha_2(1-q)}{w_1 q + w_2(1-q)}$ and $s' = \frac{\alpha_1 q + \alpha_2(1-q) - (1-s)\beta}{w_1 q + w_2(1-q)}$. In particular $c'/s' \geq 1 + \frac{(1-s)\beta}{\alpha_1 q + \alpha_2(1-q) - (1-s)\beta}$.

Proof: Let PC-CUT$(a, b, c, T, x_1, \ldots, x_{n_1})$ denote the Parity Check gadget and RMBC-CUT$(a, b, c, d, T, x_1, \ldots, x_{n_2})$ denote the RMBC gadget.

We create a graph G_x and weight function w_x which encodes the actions of the verifier V on input x. The vertices of G_x are as follows:

(1) For every bit $\pi[q]$ of the proof queried by the verifier V, the graph G_x has a vertex $v_{\pi[q]}$.

(2) For every random string R tossed by the verifier V, we create vertices $v_{R,i}$, for i going from 1 to $\max\{n_1, n_2\}$.

(3) There will be one special vertex T.

The edges of G_x are defined by the various gadgets. We stress that the same edge may appear in different gadgets (and its weight in these gadgets may be different). The graph G_x is defined by taking all these edges and thus it is a graph (or multi-graph) with parallel edges and weights. The natural conversion of G_x into a graph with no parallel edges replaces the parallel edges between two vertices with a single edge whose weight is the sum of the weights of the original edges. Alternatively, since the weights are constants which do not depend on x, we can transform G_x into a unweighted graph with parallel edges.

Suppose that on random string R the verifier V queries the oracle for bits $\pi[q_1]$, $\pi[q_2]$ and $\pi[q_3]$, and then does a parity check on these three bits. Then corresponding to this random string we add the weighted edges of the graph G_R to the graph G_x where $G_R = \text{PC-CUT}(v_{\pi[q_1]}, v_{\pi[q_2]}, v_{\pi[q_3]}, T; v_{R,1}, \ldots, v_{R,n_1})$. Alternatively, if the verifier V performs a respect of monomial basis test on the bits $\pi[q_1]$, $\pi[q_2]$, $\pi[q_3]$ and $\pi[q_4]$, then we add the weighted edges of the graph $G_R = \text{RMBC-CUT}(v_{\pi[q_1]}, v_{\pi[q_2]}, v_{\pi[q_3]}, v_{\pi[q_4]}, T; v_{R,1}, \ldots, v_{R,n_2})$.

Let N denote the number of possible random strings used by V. Observe that the total weight of the edges of G_x is $w_1 qN + w_2(1-q)N$. We now analyze the value of $\mathsf{MaxCUT}(G_x)$.

If $x \in L$ then there exists an oracle π such that $V^\pi(x)$ always accepts. We define a cut $(S, \bar{S})$ in G_x in the following way: We place $T \in S$ and for every query q we place $v_{\pi[q]} \in S$ iff $\pi[q] = 1$. Then for each R, there exists an placement of the vertices $v_{R,i}$ so that the size of the cut induced in G_R is α_1 if R corresponds to V performing a Parity Check and α_2 if R corresponds to V performing an RMB check. The weight of the so obtained cut is $\alpha_1 qN + \alpha_2(1-q)N$.

Now consider $x \notin L$. We claim that if there exists a cut $(S, \bar{S})$ such that the weight of the cut is greater than $qN\alpha_1 + (1-q)N\alpha_2 - (1-s)N\beta$, then there exists an oracle π, such that $V^\pi(x)$ accepts with probability at least s. Since we know this can not happen we conclude that $\mathsf{MaxCUT}(G_x) < qN\alpha_1 + (1-q)N\alpha_2 - (1-s)N\beta$. To prove the claim, we convert any cut in G_x into an oracle π where $\pi[q] = 1$ iff T and $v_{\pi[q]}$ lie on the same side of the cut. Now by the property of the gadgets if a graph $G_R = \text{PC-CUT}(y[q_1], y[q_2], y[q_3], T; x_1, \ldots, x_{n_1})$ contributes more than a weight of $\alpha_1 - \beta$ to the cut, then V accepts π on random string R. (Similarly if the graph G_R is an RMBC-gadget and contributes more than $\alpha_2 - \beta$ to the cut then V accepts π on random string R.) Recall that no gadget can contribute more than the corresponding α to any cut. Thus if the total weight of the cut is more than $(\alpha_1 - \beta)qN + (\alpha_2 - \beta)(1-q)N + sN \cdot \beta$, then V accepts on at least sN random strings. This proves the claim and the lemma follows. ▮

We now turn to the construction of cut-gadgets. Our first gadget, denoted PC-CUT$(a, b, c, T; \text{Aux})$, is a complete graph defined on five vertices $\{a, b, c, T, \text{Aux}\}$. The weight function, w, assign the edge $\{u, v\}$ weight $w_u \cdot w_v$, where $w_a = w_b = w_c = w_T = 1$ and $w_{\text{Aux}} = 2$. The following claim shows how PC-CUT$(a, b, c, T; \text{Aux})$ functions as a parity check gadget.

Claim 3.8.4 (MaxCUT PC-gadget): PC-CUT$(a, b, c, T; \text{Aux})$ is a $(9, 1)$-parity check gadget consisting of edges of total weight 14.

Proof: Recall that the edges in the graph are of two types: (1) edges to Aux having weight 2; and (2) other edges having weight 1. Thus, the total weight of the edges is $4 \cdot 2 + 6 \cdot 1 = 14$. The weight function is decomposed as a product of vertices "weights" and so we can express the weight of a cut $(S, \overline{S})$ by the corresponding product $(\sum_{u \in S} w_u) \cdot (\sum_{v \in \overline{S}} w_v)$. It turns out that the weight of a cut is maximized when the weight of the vertices on both sides are equal and specifically equal $\frac{6}{2} = 3$. Thus, the maximum cut has weight $3^2 = 9$. Furthermore, a max-cut must have Aux and exactly one of the other vertices on one side. On the other hand, all other cuts (i.e., in which the vertex weights are not split evenly) have weight at most 8. Using the above characterization of a max-cut we conclude that the max-cut may have one of the two forms:

(1) Aux resides in the same side with T: since a, b and c are on the other side, the induced assignment is $a = b = c = 0$ which satisfies the parity condition.

(2) Aux resides in the same side with $x \in \{a, b, c\}$: this induces $x = 0$ and an assignment of 1 to the other two variables and thus the parity condition is satisfied again.

Thus a max-cut corresponds to an assignment which satisfies the parity condition and each such assignment (can be extended to) corresponds to a max-cut. The claim follows. ∎

The second gadget, denoted RMBC-CUT$(a, b_0, b_1, c, T; \text{AUX}_1, \text{AUX}_2, \text{AUX}_3, a')$, is composed of two graphs denoted G_1 and G_2, respectively. To motivate the construction we first observe that the condition $b_a = c$ (i.e., $(a = 0) \Rightarrow (b_0 = c)$ and $(a = 1) \Rightarrow (b_1 = c)$) is equivalent to the conjuction of $(b_0 = b_1) \Rightarrow (b_0 = c)$ and $(b_0 \neq b_1) \Rightarrow (a + b_0 + c = 0)$. The graph $G_1(b_0, b_1, c; \text{AUX}_1)$ will take care of the first implication. It consists of the vertex set $\{b_0, b_1, c, \text{AUX}_1\}$, the unit-weight edges $\{b_0, \text{AUX}_1\}$ and $\{b_1, \text{AUX}_1\}$, and a weight 2 edge $\{c, \text{AUX}_1\}$. The graph $G_2(a, b_0, b_1, c, T; \text{AUX}_2, \text{AUX}_3, a')$, taking care of the second implication, consists of two subgraphs PC-CUT$(a, b_0, c, T; \text{AUX}_2)$ and $\overline{\text{PC-CUT}}(a, b_1, c, T; \text{AUX}_3, a')$, where the latter is supposed to "check" $a + b_1 + c = 1$. Specifically, $\overline{\text{PC-CUT}}(a, b, c, T; \text{AUX}, a')$ consists of the graph PC-CUT$(a', b, c, T; \text{AUX})$ and a unit-weight edge $\{a, a'\}$. The following claim shows exactly how good this gadget is in "verifying" that $b_a = c$.

Claim 3.8.5 (MaxCUT RMB-gadget): RMBC-CUT$(a, b_0, b_1, c, T; \text{AUX}_1, \text{AUX}_2, \text{AUX}_3, a')$ is a $(22, 2)$-RMBC gadget consisting of edges of total weight 33.

Proof: Clearly, the total edge weight is $4 + 14 + (14 + 1) = 33$. We analyze the performance of each of the two sub-gadgets, G_1 and G_2, considering three cases. Recall that each of two sub-gadgets corresponds to a condition of the form $E \Rightarrow E'$, where both E and E' are linear conditions on two/three variables. The first case corresponds to both E and E' being satisfied (i.e., "good case"), the second case (called "neutral") corresponds to E not being satisfied, whereas the third case (called "bad") corresponds to E being satisfied and E' being violated. We start with G_1.

Fact 1: Consider the set of all cuts in $G_1(b_0, b_1, c; \text{AUX})$.

(1) *Good Case* $(b_0 = b_1 = c)$: If b_0, b_1 and c are all in same side of the cut then we can place AUX so that the cut has weight 4. On the other hand, there is no cut with weight more than 4.

(2) *Neutral Case* $(b_0 \neq b_1)$: If b_0 and b_1 are on opposite sides of the cut, we can always place AUX so that the weight of the cut is 3. On the other hand, 3 is the maximum cut-weight for such cuts.

(3) *Bad Case* $(b_0 = b_1 \neq c)$: If b_0 and b_1 are the same side of the cut and c is on the opposite side then, no matter where AUX is placed, the cut-weight is 2.

proof: The lower bounds for Items (1) and (2) are proven by placing AUX on the opposite side to c. The upper bounds for Items (1) and (2) are obvious (since 4 is the total edge weight in G_1 and since placing b_0 and b_1 on opposite sides does not allow placing AUX opposite to both of them). Item (3) is obvious as in each of the two cases we get a cut of weight 2. □

Fact 2: Consider the set of all cuts in $G_2(a, b_0, b_1, c, T; \text{AUX}, \text{AUX}', a')$.

(1) *Good Case* $(b_0 \neq b_1$ and $a + b_0 + c = 0)$: If b_0 and b_1 are on opposite sides and $a + b_0 + c = 0$ then we can place AUX, AUX$'$ so that the cut has weight 19. On the other hand, there is no cut with weight more than 19.

(2) *Neutral Case* $(b_0 = b_1)$: If b_0 and b_1 are on the same side of the cut, we can place AUX, AUX$'$ so that the weight of the cut is 18. On the other hand, 18 is the maximum cut-weight for such cuts.

(3) *Bad Case* $(b_0 \neq b_1$ and $a + b_0 + c \neq 0)$: If b_0 and b_1 are on opposite sides and $a + b_0 + c \neq 0$ then 17 is the maximum cut-weight for such cuts.

proof: Recall that $G_2(a, b_0, b_1, c, T; \text{AUX}, scAux', a')$ consists of the subgraphs PC-CUT$(a, b_0, c, T; \text{AUX})$ and PC-CUT$(a', b_1, c, T; \text{AUX}')$, and the edge $\{a, a'\}$.

- Item (1) follows by Claim 3.8.4 (where for the lower bound we place a' opposite to a and use $a' + b_1 = a + b_0$).
- The upper bound of Item (2) follows from Claim 3.8.4 by first observing that if both $a+b_0+c = 0$ and $a' + b_1 + c = 0$ then $a = a'$ (since in this case $b_0 = b_1$). Thus, either we obtain maximum weight of 9 in both PC gadgets (and lose the edge $\{a, a'\}$) or we do not obtain the weight 9 in both PC gadgets – either way the bound follows.
- The lower bound of Item (2) follows by first observing that when we place a' opposite to a, either $a + b_0 + c = 0$ or $a' + b_1 + c = 0$ holds. Extending the argument of Claim 3.8.4, we next observe that if the parity condition is not satisfied we can still place the auxiliary vertex to obtain a cut of weight 8. Thus, we obtain a cut of weight $1 + 8 + 9 = 18$ as claimed.
- Item (3) follows from Claim 3.8.4 by first observing that if $b_0 \neq b_1$ and $a + b_0 + c \neq 0$ then $a + b_1 + c = 0$. Thus, $\text{PC-CUT}(a, b_0, c, T; \text{AUX})$ contributes at most weight 8 to the cut (use Claim 3.8.4) whereas either $a = a'$ or $\text{PC-CUT}(a', b_1, c, T; \text{AUX}')$ also contributes at most 8. As above, in the former case (i.e., $a = a'$) the edge $\{a, a'\}$ is not in the cut. Thus in either cases the maximum cut weight is 17 (obtained by either $2 \cdot 8 + 1$ or $8 + 9$).

This concludes the proof of Fact 2. □

The Claim now follows by combining the two facts. First recall that the RMB condition is equivalent to the conjunction of $(b_0 = b_1) \Rightarrow (b_0 = c)$ and $(b_0 \neq b_1) \Rightarrow (a + b_0 + c = 0)$. If the RMB condition holds then we obtain the Good Case weight from one sub-gadget, say G_i, and the Neutral Case weight from the other (i.e., G_{3-i}). (The value of $i \in \{1, 2\}$ depends on whether $b_0 = b_1$ or not.) Thus, the total weight equals 22 (obtained by either $4 + 18$ or $3 + 19$). If the RMB condition does not hold then we obtain the Bad Case weight from one sub-gadget and the Neutral Case weight from the other. Thus, the total weight equals 20 (obtained by either $2 + 18$ or $3 + 17$). The claim follows. ▮

Proof of Theorem 3.8.2: The theorem follows by combining Proposition 3.6.3, Lemma 3.8.3, Claim 3.8.4 and Claim 3.8.5 (when regarding the RMB gadget as a $(11, 1)$-gadget rather than a $(22, 2)$-gadget). Details follows.

As in the proof of Theorem 3.7.2, when applying Lemma 3.8.3 to the verifier in Proposition 3.6.3, we obtain the same expression for the gap, c'/s', for which $\mathrm{NP} \leq_D^K \textsf{Gap}\text{-}\mathrm{Cut}_{c',s'}$; namely,

$$\begin{aligned} \frac{c'}{s'} &\to 1 + \frac{(1-s)\beta}{q \cdot \alpha_1 + (1-q) \cdot \alpha_2 - (1-s)\beta} \\ &= 1 + \frac{3}{12\alpha_1 + 8\alpha_2 - 3}. \end{aligned}$$

Substituting $\alpha_1 = 9$ and $\alpha_2 = 11$, the above simplifies to $1 + \frac{3}{193} > \frac{66}{65}$ and the bound on $\frac{c'}{s'}$ follows. As for c', it equals $\frac{3\alpha_1 + 2\alpha_2}{3m_1 + 2m_2} > 0.6$. ▮

3.9 Free bits and vertex cover

It is known that approximating the minimum vertex cover of a graph to within a $1 + \epsilon$ factor is hard, for some $\epsilon > 0$ [PaYa, ALMSS]. However, we do not know of any previous attempt to provide a lower bound for ϵ. An initial attempt may use VC-gadgets that implement the various tests in V_{SNPinner}, analogously to the way it was done in the previous sections for the Max SAT versions and

The Enhanced RMB Test. Again, $A \colon \mathcal{F}_l \to \Sigma$ is the object being tested, and the test take additional inputs or parameters $f_1, f_2 \in \mathcal{F}_l$.

EMBTest$(A; f_1, f_2)$ (Enhanced Monomial-Basis Test)
For every $f \in \mathcal{F}_l$, invoke **MBTest**$(A; f_1, f, f_2)$.
Output 0 if all invocations answered with 0, else output 1.

The Passing Probability:

$$\mathrm{EMBP{\scriptsize ASS}}(A) \quad = \quad \Pr_{f_1, f_2 \stackrel{R}{\leftarrow} \mathcal{F}_l} [\, \mathbf{EMBTest}(A; f_1, f_2) = 0 \,]$$

Figure 3.7: *The Enhanced RMB test and its passing probability.*

Max Cut. This yields a lower bound of $\epsilon > \frac{1}{43} > 0.023$. However, a stronger result is obtained via free-bit complexity. Specifically, we apply the FGLSS-reduction to a proof system (for NP) in which the free-bit complexity is the lowest one possible: which, by the results of Section 5.1, is 2 free-bits. Consequently, the clique size, in case the original input is in the language, is at least one fourth (1/4) of the size of the graph which means that translating clique-approximation factors to VC-approximation factors yields only a loss of one third. Since the FGLSS-transformation translates the completeness/soundness ratio to the gap-factor for approximating clique, our first goal is to construct for NP a proof system which uses two free-bits and has soundness error as low as possible. We remark that the proof system of subsection 3.10 uses 7 free-bits and achieves soundness error less than 1/2. The reader may observe that, following the above approach, it is not worthwhile to use the proof system of subsection 3.10 or any proof systems which achieves a soundness error of 1/2 at the cost of 5 free-bits or more.

3.9.1 Minimizing the error achievable with two free bits

The pcp system of Proposition 3.6.3 had free-bit complexity 2 (and query-complexity 3). However, a smaller soundness error can be achieved if we make more queries. Our starting point is Part (2) of Lemma 3.5.7 which suggests an RMB-test with twice bigger detection probability still using 2 free-bits (alas 2^l+2 rather than 3 queries). Specifically, we consider an *enhanced* RMB test which on input $f_1, f_2 \in \mathcal{F}_l$, goes over all $f \in \mathcal{F}_l$ invoking the Atomic RMB test with input functions f_1, f, f_2. The enhanced RMB Test, denoted **EMBtest**, is depicted in Figure 3.7. Further improvement is obtained by "packing" together the Linearity Test and the Enhanced RMB Test (in contrast to V_{SNPinner} in which these tests were performed exclusively). Both tests make three queries of which two are common, and the answers to these queries determine the answer to the third query (which is different in the two tests). The resulting inner verifier, denoted V_{2inner}, is depicted in Figure 3.8. As V_{SNPinner}, the verifier V_{2inner} works with functions/oracles A that are folded twice — once across $(h, 0)$ and once across $(\bar{1}, 1)$.

The following corollary is immediate from Part (2) of Lemma 3.5.7.

Corollary 3.9.1 (analysis of the Enhanced Monomial-Basis Test): Let $A, \tilde{A}: \mathcal{F}_l \to \Sigma$ with A satisfying $A(f+\bar{1}) = A(f)+1$ for all f and $\tilde{A}$ linear but not respecting the monomial basis. Let $x = \mathrm{Dist}(A, \tilde{A})$. Then

$$1 - \mathrm{EMBP{\small ASS}}(A) \;\geq\; \frac{3}{4} \cdot (1-2x)$$

The following lemma is analogous to Lemma 3.6.1. Loosely speaking, it considers three possible strategies of a "dishonest" prover and indicates the probability with which the verifier detects an error.

Lemma 3.9.2 (soundness of V_{2inner}): Let $\delta_1, \delta_2 > 0$, $0 \leq p \leq 1$ and $l, l_1 \in \mathcal{Z}^+$. Then the (l, l_1)-canonical inner verifier V_{2inner} (with parameter p) is $(\rho, \delta_1, \delta_2)$-good, where $1-\rho = \min(T_1, T_2, T_3)$ and

(1) $T_1 \stackrel{\mathrm{def}}{=} (\frac{1}{2} - \delta_1) \cdot p$

(2) $T_2 \stackrel{\mathrm{def}}{=} p \cdot \min_{x \leq 1/2-\delta_1} [\, \max(\Gamma_{\mathrm{lin}}(x)\,,\, \frac{3}{4} \cdot (1-2x))\,]$

(3) $T_3 \stackrel{\mathrm{def}}{=} \min_{x \leq 1/2-\delta_1} [\, p \cdot \Gamma_{\mathrm{lin}}(x) + (1-p) \cdot (\frac{1}{2} - \delta_2)(1-2x)\,]$.

Proof: The analysis is broken up into several cases as in the proof of Lemma 3.6.1. Let $x = \mathrm{Dist}(A_{(h,0),(\bar{1},1)}, \mathrm{L{\small IN}})$.

Case 1: $x \geq 1/2 - \delta_1$

Lemma 3.5.3 implies that $1 - \mathrm{L{\small IN}P{\small ASS}}(A_{(h,0),(\bar{1},1)}) \geq \Gamma_{\mathrm{lin}}(x) \geq x \geq 1/2 - \delta_1$. Since V_{2inner} performs the atomic linearity test with probability p, we have in this case

$$1 - \mathtt{ACC}\,[\, V_{\mathrm{2inner}}^{A,A_1}(\sigma, h)\,] \geq p \cdot (1/2 - \delta_1)$$

The two free-bit inner verifier. Given functions $h \in \mathcal{F}_l$ and $\sigma: \Sigma^l \to \Sigma^{l_1}$, the verifier has access to oracles for $A: \mathcal{F}_l \to \Sigma$ and $A_1: \mathcal{F}_{l_1} \to \Sigma$. In addition it takes a parameter $p \in [0,1]$.

Pick $q \stackrel{R}{\leftarrow} [0,1]$.

Case: $q \leq p$:
- Pick $f_1, f_2 \stackrel{R}{\leftarrow} \mathcal{F}_l$.
- **LinTest**$(A_{(h,0),(\bar{1},1)}; f_1, f_2)$.
- **EMBTest**$(A_{(h,0),(\bar{1},1)}; f_1, f_2)$.

Case: $q > p$:
- Pick $f \stackrel{R}{\leftarrow} \mathcal{F}_l$ and $g \stackrel{R}{\leftarrow} \mathcal{F}_{l_1}$.
- **ProjTest**$_\sigma(A_{(h,0),(\bar{1},1)}, A_1; f, g)$.

Remark: access to $A_{(h,0),(\bar{1},1)}(f)$ is implemented by accessing either $A(f)$ or $A(f+h)$ or $A(f+\bar{1})$ or $A(f+h+\bar{1})$.

Figure 3.8: *The two free-bit inner verifier* V_{2inner}

Case 2: $x < 1/2 - \delta_1$

Again, Lemma 3.5.3 implies that $1 - \text{LinPass}(A_{(h,0),(\bar{1},1)}) \geq \Gamma_{\text{lin}}(x)$ and

$$1 - \mathtt{ACC}\left[\, V_{\text{2inner}}^{A,A_1}(\sigma, h) \,\right] \;\geq\; p \cdot \Gamma_{\text{lin}}(x)$$

follows. Now let $\tilde{A}$ be a linear function such that $\text{Dist}(A_{(h,0),(\bar{1},1)}, \tilde{A}) = x$. We consider the following sub-cases.

Case 2.1: $\tilde{A}$ does not respect the monomial basis

In this case Corolary 3.9.1 implies that $1 - \text{EMBPass}(A_{(h,0),(\bar{1},1)}) \geq \frac{3}{4}(1 - 2x)$. So the probability that V_{2inner} rejects is at least $p \cdot \frac{3}{4}(1-2x)$. Combining the two lower bounds on $1 - \mathtt{ACC}\left[\, V_{\text{2inner}}^{A,A_1}(\sigma, h) \,\right]$, we get

$$1 - \mathtt{ACC}\left[\, V_{\text{2inner}}^{A,A_1}(\sigma, h) \,\right] \;\geq\; p \cdot \max(\Gamma_{\text{lin}}(x), \frac{3}{4}(1 - 2x))$$

Case 2.2: $\tilde{A}$ respects the monomial basis

By Proposition 3.3.2, $\tilde{A}$ is an evaluation operator. So there exists $a \in \Sigma^l$ such that $\tilde{A} = E_a$. So $\text{Dist}(A_{(h,0),(\bar{1},1)}, E_a) = x$. Let $a_1 = \sigma(a)$. The proof splits into two further sub-cases.

Case 2.2.1: $d \stackrel{\text{def}}{=} \text{Dist}(A_1, E_{a_1}) \geq 1/2 - \delta_2$

By Lemma 3.5.8 we have $1 - \text{ProjPass}_\sigma(A_{(h,0),(\bar{1},1)}, A_1) \geq d \cdot (1 - 2x) \geq (1/2 - \delta_2) \cdot (1 - 2x)$. So the probability that V_{2inner} performs the projection test and rejects is at least $(1-p) \cdot (1/2 - \delta_2)(1 - 2x)$. To this we add the probability of the exclusively disjoint event in which the verifier performs the Linearity Test and rejects, obtaining

$$1 - \mathtt{ACC}\left[\, V_{\text{2inner}}^{A,A_1}(\sigma, h) \,\right] \;\geq\; p \cdot \Gamma_{\text{lin}}(x) + (1 - p) \cdot (1/2 - \delta_2)(1 - 2x)$$

Case 2.2.2: Else –

In this case, we have $x = \text{Dist}(A_{(h,0),(\bar{1},1)}, E_a) < 1/2 - \delta_1$ and $\text{Dist}(A_1, E_{a_1}) < 1/2 - \delta_2$. Thus the functions $A_{(h,0),(\bar{1},1)}$ and A_1 satisfy conditions (2.1) and (2.2) in Definition 3.4.3.

Similarly to the proof of Lemma 3.6.1, we infer that the lower bound on $1 - \rho$ is as claimed and the lemma follows. ∎

We now simplify the soundness bound of the lemma. The proof of the first item uses the fact that $\Gamma_{\text{lin}}(x) \geq 45/128$ for all $x \geq 1/4$. The second item uses the fact that $\Gamma_{\text{lin}}(x) \geq x$ for all $x \leq 1/2$.

Claim 3.9.3 :

(1) $\min_{x \leq 1/2 - \delta_1} \left[\max(\Gamma_{\text{lin}}(x)\,,\, \frac{3}{4}(1 - 2x))\right] \geq \frac{45}{128}$.

(2) $\min_{x \leq 1/2 - \delta_1} \left[\, p \cdot \Gamma_{\text{lin}}(x) + (1 - p) \cdot (\frac{1}{2} - x) \,\right] \geq \frac{1}{2} \cdot \min(p\,,\, 1 - p)$.

(3) Let T_1, T_2 and T_3 be as in Lemma 3.9.2. Then

$$\min(T_1, T_2, T_3) \;\geq\; \min\left\{\frac{45}{128} \cdot p\;,\; \frac{1}{2} \cdot (1 - p)\right\} - \max(\delta_1, \delta_2)$$

Interestingly, the lower bound provided by Item (3) is tight (see Claim 3.14.3). Optimization calls for setting $\frac{45}{128} \cdot p = \frac{1}{2} \cdot (1-p)$, which yields $p = \frac{64}{109}$ and a soundness bound of $1 - \frac{45}{128}p + \max(\delta_1, \delta_2) = 1 - \frac{45}{218} + \max(\delta_1, \delta_2)$.

Proof: Towards proving Part (1) we consider two cases.

Case 1.1: $x \geq 1/4$. In this case, by definition of $\Gamma_{\rm lin}$, we have

$$\max(\Gamma_{\rm lin}(x)\,,\,\frac{3}{4}(1-2x)) \;\geq\; \Gamma_{\rm lin}(x) \;\geq\; \frac{45}{128}$$

Case 1.2: $x \leq 1/4$. In this case we have

$$\max(\Gamma_{\rm lin}(x)\,,\,\frac{3}{4}(1-2x)) \;\geq\; \frac{3}{4}(1-2x) \;\geq\; \frac{3}{8} \;>\; \frac{45}{128}$$

This establishes Part (1). Towards proving Part (2) we consider two different cases.

Case 2.1: $p \leq (1-p)$. In this case

$$p \cdot \Gamma_{\rm lin}(x) + (1-p)\cdot(\frac{1}{2}-x) \;\geq\; p\cdot x + p\cdot(\frac{1}{2}-x) \;=\; \frac{p}{2}$$

Case 2.2: $p \geq (1-p)$. In this case

$$p \cdot \Gamma_{\rm lin}(x) + (1-p)\cdot(\frac{1}{2}-x) \;\geq\; (1-p)\cdot x + (1-p)\cdot(\frac{1}{2}-x) \;=\; \frac{1-p}{2}$$

This establishes Part (2). To prove Part (3) use Parts (1) and (2) to lower bound T_2 and T_3, respectively, and get

$$\begin{aligned}\min(T_1,T_2,T_3) &\geq \min\left\{(\frac{1}{2}-\delta_1)\cdot p\,,\,\frac{45}{128}\cdot p\,,\,\frac{1}{2}\cdot\min(p\,,\,1-p)-\delta_2\right\}\\ &\geq \min\left\{\frac{45}{128}\cdot p\,,\,\frac{1}{2}\cdot(1-p)\right\}-\max(\delta_1,\delta_2)\end{aligned}$$

The claim follows. ▮

Composing the above inner verifier with an adequate outer verifier, we get

Theorem 3.9.4 $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,s}[\log,2]$ for any $s > \frac{173}{218} \approx 0.79357798$.
Furthermore, the verifier has constant query complexity.

Proof: Let $\delta = s - \frac{173}{218}$, $\delta_1 = \delta_2 = \delta/3$ and $\epsilon = \frac{\delta}{3}\cdot 16\delta_1^2\delta_2^2 = \frac{16\delta^5}{243}$. Now, let l and l_1 be integers such that the outer verifier, $V_{\rm outer}$, guaranteed by Lemma 3.4.2, is (l,l_1)-canonical and ϵ-good for $L \in \mathrm{NP}$. Consider the (l,l_1)-canonical inner verifier $V_{\rm 2inner}$ working with parameter $p = 64/109$. Using Lemma 3.9.2 and Claim3.9.3, we conclude that $V_{\rm 2inner}$ is (ρ,δ,δ)-good for $\rho = 1 - \frac{45}{218} + \max(\delta_1,\delta_2)$.

Composing $V_{\rm outer}$ and $V_{\rm 2inner}$ we obtain a verifier, $V_{\rm 2free}$, which by Theorem 3.4.5 has soundness error bounded above by $\frac{173}{218} + \max(\delta_1,\delta_2) + \frac{\epsilon}{16\delta_1^2\delta_2^2} = s$, as required. Furthermore, $V_{\rm 2free}$ uses logarithmically many coins. We claim that $V_{\rm 2free}$ has query complexity 2^l+2 and free-bit complexity 2. The claim is obvious in case $V_{\rm 2inner}$ performs the Projection test. Otherwise, $V_{\rm 2inner}$ performs a Linearity Test with parameters f_1 and f_2 and an enhanced RMB Tests with the same parameters. Clearly, the answers on f_1 and f_2 determine the acceptable (by Linearity Test) answer on f_1+f_2. The key observation is that the former two answers also determine all 2^l acceptable answers in the enhanced RMB test (i.e., for every $f \in \mathcal{F}_l$, the answer on $f_1'\cdot f + f_2$ should equal the answer on f_2, where $f_1' = f_1$ if the answer on f_1 is zero and $f_1' = f_1 + \bar{1}$ otherwise). ▮

By repeating the above proof system three times, we obtain

Corollary 3.9.5 $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,1/2}[\log, 6]$.
Furthermore, the verifier has constant query complexity.

Proof: There exists $\epsilon > 0$ such that $\left(\frac{173}{218} + \epsilon\right)^3 \leq \frac{1}{2}$. ∎

3.9.2 Hardness of vertex cover

PRELIMINARIES. A *vertex cover* of a graph $G = (V, E)$ is a set $V' \subseteq V$ such that $V' \cap \{u, v\} \neq \emptyset$ for every $\{u, v\} \in E$. We let $\mathsf{MinVC}(G)$ denote the size of a smallest vertex cover in G, and we let $\overline{\mathsf{MinVC}}(G) = \mathsf{MinVC}(G)/|V|$. Min-VC is the problem whose instances are graphs G and one has to find $\mathsf{MinVC}(G)$. An approximation algorithm A for Min-VC achieves a ratio, or factor, of $\alpha \in [1, \infty)$ if $\mathsf{MinVC}(G) \leq A(G) \leq \alpha \cdot \mathsf{MinVC}(G)$ for all graphs G. (Here we have adopted the convention by which for minimization problems the approximation factor is at least 1.) Again, we capture the approximation problem by a promise problem, but this time the parameter c referring to YES-instances is lower than the parameter s referring to NO-instances.

Definition 3.9.6 For any $0 \leq c \leq s \leq 1$ we let the promise problem $\mathsf{Gap}\text{-VC}_{c,s}$ be the pair (A, B), where –

(1) A is the set of all graphs G satisfying $\overline{\mathsf{MinVC}}(G) \leq c$, and

(2) B is the set of all graphs G satisfying $\overline{\mathsf{MinVC}}(G) \geq s$.

The gap of this problem is defined to be s/c.

KNOWN UPPER AND LOWER BOUNDS. There is a simple polynomial time algorithm to approximate Min-VC in unweighted graphs within a factor of 2. The algorithm, due to F. Gavril (cf. [GJ2]), consists of taking all vertices which appear in a maximal matching of the graph. For weighted graphs, Bar-Yehuda and Even [BaEv1] and Hochbaum [Hoc], gave algorithms achieving the same approximation factor. The best known algorithm today achieves a factor only slightly better, namely $2 - (\log\log|V|)/(2\log|V|)$ [BaEv, MoSp].

Evidence to the hardness of approximating Min-VC was given by Bar-Yehuda and Moran who showed that, for every $k \geq 2$ and $\epsilon > 0$, a $1 + \frac{1}{k} - \epsilon$ approximator for (finding) a minimum vertex cover would yield an algorithm for coloring $(k+1)$-colorable graphs using only logarithmically many colors [BaMo]. The version of Min-VC in which one restricts attention to graphs of degree bounded by a constant B, is Max-SNP complete for suitably large B [PaYa]. In particular they provide a reduction from Max-3-SAT. Combined with [ALMSS] this implies the existence of a constant $\delta > 0$ such that approximating Min-VC within a factor of $1 + \delta$ is hard unless $\mathrm{P} = \mathrm{NP}$. No explicit value of δ has been stated until now. Indeed, the value that could be derived, even using the best existing non-approximability results for Max-3-SAT, will be very small, because of the cost of the reduction of [PaYa], which first reduces Max-3-SAT to its bounded version using expanders, and then reduces this to Min-VC-B.

GOING FROM FREE BITS TO VC. Rather than reduce from Max-3-SAT, we will first use Theorem 3.9.4 to get gaps in Clique size. Then we apply the standard reduction.

Proposition 3.9.7 $\mathrm{FPCP}_{c,s}[\log, f] \leq_D^K \mathsf{Gap}\text{-VC}_{c',s'}$ for $c' = 1 - 2^{-f}c$ and $\frac{s'}{c'} = 1 + \frac{c-s}{2^f - c}$.

Proof: The FGLSS reduction says that $\mathrm{FPCP}_{c,s}[\log, f] \leq_D^K \mathsf{Gap}\text{-Clique}_{c'',s''}$ where $c'' = 2^{-f} \cdot c$ and $s'' = 2^{-f} \cdot s$. (See Section 3.12 for definition of Gap-Clique.) Now we apply the standard Karp reduction (of MaxClique to Min VC) which maps a graph G to its complement $\overline{G}$, noting that $\overline{\mathsf{MinVC}}(\overline{G}) = 1 - \overline{\mathsf{MaxClique}}(G)$. Thus $\mathsf{Gap}\text{-Clique}_{c'',s''} \leq_D^K \mathsf{Gap}\text{-VC}_{1-c'',1-s''}$, where $c' = 1 - c'' = 1 - 2^{-f}c$. Finally,

$$\frac{1-s''}{1-c''} = \frac{1-s2^{-f}}{1-c2^{-f}} = 1 + \frac{c-s}{2^f - c}$$

This completes the proof. ∎

Our results. We obtain the first explicit and reasonable constant factor non-approximability result for Min-VC. A consequence of the following theorem is that, assuming P $\neq$ NP there is no polynomial time algorithm to approximate Min-VC within a factor of 1.0688.

Theorem 3.9.8 $\mathsf{Gap}\text{-VC}_{c,s}$ is NP-complete for some c, s satisfying $s/c \geq 1.0688 > 16/15$. Moreover $c = 3/4$.

Proof: Follows immediately from Proposition 3.9.7 and Theorem 3.9.4. Namely, for any $s > 173/218$, $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,s}[\log, 2] \leq_D^K \mathsf{Gap}\text{-VC}_{c',s'}$ for $c' = 1 - 2^{-2} = \frac{3}{4}$ and $\frac{s'}{c'} = 1 + \frac{1-s}{2^2-1} = 1 + \frac{1-s}{3}$. Thus, $\frac{s'}{c'} = 1 + \frac{15}{218} - \frac{\epsilon}{3} > 1.068807 - \frac{\epsilon}{3}$, where $\epsilon \stackrel{\text{def}}{=} s - \frac{173}{218}$. ∎

We remark that a special case of Proposition 3.9.7 in which the statement is restricted to $f = 0$ would have suffices for proving the above theorem. The reason being that we could have applied Proposition 5.2.8 to Theorem 3.9.4 and obtain $\mathrm{NP} \subseteq \mathrm{FPCP}_{1/4,s/4}[\log, 0]$, for $s = 0.7936$, which by the special case of Proposition 3.9.7 is reducible to $\mathsf{Gap}\text{-VC}_{c',s'}$ with $c' = 1 - \frac{1}{4} = \frac{3}{4}$ and $\frac{s'}{c'} = 1 + \frac{(1/4)-(s/4)}{1-(1/4)} = 1 + \frac{1-s}{3}$ (as above). Interestingly, the special case of Proposition 3.9.7 can be "reversed": namely, $\mathsf{Gap}\text{-VC}_{c',s'}$ is reducible to $\mathrm{FPCP}_{c,s}[\log, 0]$ with $c = 1 - c'$, $s = 1 - s'$ and $\frac{s}{c} = \frac{1-s'}{1-c'}$ (which reverses $\frac{s'}{c'} = \frac{1-s}{1-c} = 1 + \frac{c-s}{1-c}$). The key fact in proving this "reverse reduction" is Corollary 4.1.5 which asserts that $\mathsf{Gap}\text{-Clique}_{c,s} \leq_D^K \mathrm{FPCP}_{c,s}[\log, 0]$. However, we do not know if it is possible to "reverse" the other step in the alternative proof; namely, whether $\mathrm{FPCP}_{c,s}[\log, 0]$ is reducible to $\mathrm{FPCP}_{4c,4s}[\log, 2]$ (our reverse transformation is weaker – see Proposition 5.2.6).

3.9.3 On using the MaxSNP verifier to establish Min VC hardness

Although our current VC-gadgets yield a hardness result which is inferior to what has been presented above, it may be the case that improved results can be obtained by a better implementation of the MaxSNP verifier. As in Sections 3.7 and 3.8, we first define problem-specific gadgets and establish a reduction of pcp systems to the promised problem at hand. The gadgets will be graphs with distinguished vertices corresponding to the two literals of each variable appearing in the test/check. Covers will induce truth assignments in the standard manner (i.e., a literal is set to 1 iff the corresponding vertex is in the cover). (Covers which contain none or both literals of the same variable are defined to set the variable to a special symbol $\perp$ which does not satisfy any equality.) Specifically, a Parity Check gadget $\text{PC-VC}(a, b, c, \overline{a}, \overline{b}, \overline{c}; x_1, \ldots, x_n)$ is a graph on $6 + n$ vertices where a, b, c correspond to oracle queries made by the verifier. The gadget is an (α, β)-PC gadget if $\mathsf{MinVC}(\text{PC-VC}(a, b, c, \overline{a}, \overline{b}, \overline{c}; x_1, \ldots, x_n))$ is exactly α when restricted to covers which induce $a+b = c$ (i.e., either 0 or 2 of the vertices $\{a, b, c\}$ are in the cover), and is at least $\alpha + \beta$ when restricted to covers for which $a+b \neq c$. Similarly a graph $\text{RMBC-VC}(a, b_0, b_1, c, \overline{a}, \overline{b_0}, \overline{b_1}, \overline{c}; x_1, \ldots, x_n)$ is an (α, β)-RMBC gadget if it satisfies the property that $\mathsf{MinVC}(\text{RMBC-VC}(a, b, c, d, \overline{a}, \overline{b_0}, \overline{b_1}, \overline{c}; x_1, \ldots, x_n))$ is exactly α when restricted to covers satisfying $b_a = c$ and is at least $\alpha + \beta$ otherwise. We stress that covers of minimal size must contain exactly one of the two vertices corresponding to the distinguished pair of literals. The following lemma (similar to Lemmas 3.7.3 and 3.8.3) shows how to use the above forms of gadgets to derive a reduction from NP to $\mathsf{Gap}\text{-VC}$.

Lemma 3.9.9 (MinVC implementation of a verifier): Let V be a verifier for L of logarithmic randomness, with perfect completeness and soundness s, such that V performs either a single Parity Check (with probability q) or a single RMB check (with probability $1-q$). Furthermore, suppose that in either case, the verifier never makes two identical queries. If there exists an (α_1, β)-PC gadget consisting of n_1 vertices and an (α_2, β)-RMBC gadget consisting of n_2 vertices then L reduces to $\mathsf{Gap}\text{-}\mathrm{VC}_{c',s'}$ for $c' = \frac{\alpha_1 q + \alpha_2(1-q)}{n_1 q + n_2(1-q)}$ and $s' = \frac{\alpha_1 q + \alpha_2(1-q)+(1-s)\beta}{n_1 q + n_2(1-q)}$. In particular $s'/c' \geq 1 + \frac{(1-s)\beta}{\alpha_1 q + \alpha_2(1-q)}$.

Proof: The reduction is analogous to the other two reductions presented above. Namely, for each possible random string R we introduce a graph G_R which is a copy of the corresponding gadget. All vertices and edges in these copies are distinct.[4] In addition, for each variable v (corresponding to an oracle location) we join by edges each occurrence of v with each occurrence of $\overline{v}$. Namely, if v is a query under both random strings R and R', then we join by an edge the vertex labeled v in G_R and the vertex labeled $\overline{v}$ in $G_{R'}$.

Letting N denote the number of possible random strings, we observe that the number of vertices in the resulting graph is $n_1 \cdot qN + n_2 \cdot (1-q)N$. Also, if $x \in L$ then the resulting graph has a cover with $\alpha_1 \cdot qN + \alpha_2 \cdot (1-q)N$ vertices (i.e., just use the cover corresponding to the oracle which always makes the prover accept). On the other hand, we claim that if $x \notin L$ then the resulting graph, denoted G_x, does not have a cover of size smaller than $\alpha_1 qN + \alpha_2(1-q)N + (1-s)N\beta$. Once the claim is proven the bound on s' follows.

Fixing an arbitrary cover of G_x, we first define an oracle, π, by setting $\pi(v) = 1$ if all copies of v are in the cover and $\pi(v) = 0$ otherwise. Using the edges joining all occurrences of v and $\overline{v}$, we conclude that in the latter case all copies of $\overline{v}$ are in the cover. Now, each copy of the PC-gadget (resp., RMB-gadget) having α_1 (resp., α_2) vertices in the cover corresponds to a random string which makes the verifier accept the oracle π. Using the soundness of the verifier, we conclude that at least $(1-s)N$ of the gadgets correspond to random strings on which the verifier rejects π and the claim follows. ▌

Hardness results for MinVC can be derived by combining Proposition 3.6.3 and Lemma 3.9.9. Namely, the existence of a $(\alpha_1, 1)$-PC gadget with n_1 vertices and a $(\alpha_2, 1)$-RMB gadget with n_2 vertices implies NP-hardness of $\mathsf{Gap}\text{-}\mathrm{VC}_{c',s'}$ with

$$\frac{s'}{c'} \;\rightarrow\; 1 + \frac{3}{12\alpha_1 + 8\alpha_2} \tag{3.12}$$

$$c' \;=\; \frac{12\alpha_1 + 8\alpha_2}{12n_1 + 8n_2} \tag{3.13}$$

We know how to construct a $(6,1)$-PC gadget with 10 vertices and a $(7,1)$-RMB gadget with 12 vertices. This yields a gap of $1 + \frac{3}{128} < \frac{43}{42}$. In order to beat the current hardness gap of 1.0688 (established by the reduction from 2 free-bit pcp) one would need to construct gadgets with α_i's (of weighted average) below 2.2 (i.e., $\frac{3}{5}\alpha_1 + \frac{2}{5}\alpha_2 < 2.2$). So it seems that this approach (i.e., of using the MaxSNP verifier to establish MinVC hardness) offers little hope for progress.

3.10 Minimizing the number of queries

The problem we consider here is to minimize the values of q (and q_{av}) for which we can construct PCPs for NP using q queries in the worst case (and q_{av} on the average) to achieve a soundness

[4] This is in contrast to the MaxCUT implementation where the same non-auxiliary vertices were used in all gadgets.

Due to	q	$q_{\rm av}$
[ALMSS]	some constant	some constant
[BGLR]	36	29
[FeKi]	32	24
This paper	11	10.9

Figure 3.9: *Worst case (q) and average ($q_{\rm av}$) number of queries needed to get 1/2 soundness with logarithmic randomness; that is, results of the form of Eq. (3.14).*

error of 1/2. We allow only logarithmic randomness. In other words we want results of the form:

$$\mathrm{NP} \;=\; \mathrm{PCP}_{1,1/2}[\,\mathsf{coins} = \log\,;\,\mathsf{query} = q\,;\,\mathsf{query}_{\rm av} = q_{\rm av}\,]\,. \tag{3.14}$$

Later in this paper we will return to this question by looking at lower bounds.

Previous work. It was shown by [ALMSS] that there are constants $q, q_{\rm av}$ for which (3.14) is achieved. Reductions in the values of these numbers obtained since then are depicted in Figure 3.9.

The interest of [BGLR] in these numbers was to improve non-approximability factors for Max Clique. But we now know that free-bits are a better measure towards this end [FeKi, BeSu]. Yet we remain interested in query bits for their own sake. Indeed, the number of bits queried remains a most natural measure, and it is an intruiging question as to how many bits of a proof you need to look at to detect an error with a given probability.

Sources of our improvements. The principal part of our improvement comes from the use of the new long code based inner verifier, the atomic tests and their analysis in Section 3.5, and the new idea of folding. By repeating the proof system of Theorem 3.6.4 five times, we obtain that Eq. (3.14) holds for $q = 15$. (Four repetitions yielding $q = 12$ do not suffice.) A straightforward implementation of the recycling technique of [BGLR] yields $q = 12$ and $q_{\rm av} = 11.74$ for which Eq. (3.14) is achieved. Using a more careful implementation of this technique, we reduce the query complexity by an additional bit.

3.10.1 The PCP inner verifier

Our result is based on the construction of the (l, l_1)-canonical inner verifier $V_{\rm PCPinner}$ depicted in Figure 3.10. In addition to its standard inputs h, σ it takes parameters $p_1, p_2, p_3 \geq 0$ so that $p_1 + p_2 + p_3 = 1$. The inner verifier $V_{\rm PCPinner}$ combines the atomic tests in three different ways.

(1) Some tests are performed independently (i.e., the main steps in Figure 3.10);

(2) Some tests are performed while re-using some queries (i.e., the tests in Step (2) re-use f_3);

(3) Some tests are performed in a mutual exclusive manner (i.e., the tests in Step (3));

As in previous sections, the tests are executed on the function $A_{(h,0),(\bar{1},1)}$ to which the verifier has an effective oracle access given his access to A. By inspection it is clear that the total number of accesses to the oracles for A and A_1 is $3+5+3 = 11$ (whereas the free-bit complexity is $2+3+2 = 7$). We now examine the goodness of $V_{\rm PCPinner}$. Recall the definitions of the functions $\Gamma_{\rm lin}(x)$ (from Lemma 3.5.3) and $\Gamma_{\rm RMB}(x) = \frac{3}{8}(1-2x)$ (from Lemma 3.5.7).

Lemma 3.10.1 (soundness of V_{PCPinner}): For any $0 < \delta_1, \delta_2 < 0.1$ and any l, l_1, p_1, p_2 and p_3, satisfy $p_1 + p_2 + p_3 = 1$ and $5p_1 = 2p_2$, the (l, l_1)-canonical inner verifier V_{PCPinner} is $(\rho, \delta_1, \delta_2)$-good, where $1 - \rho$ is the minimum of the following three quantities

(1) $\frac{1}{2} + \frac{p_1}{10} - \delta_1$;

(2) $[1 - \min(\frac{1+q}{6}, \frac{2-q}{8})]^3 + \frac{p_3}{1-p_3}$, where $q \stackrel{\mathrm{def}}{=} \frac{p_1}{1-p_3}$;

(3) $\min(\frac{1}{2} + \frac{p_3}{20} - \delta_2 \,, 1 - (0.55218507 + \delta_2) \cdot (1 - \frac{45}{128} p_1))$

Furthermore, if $p_1 > 10\delta_1$, $p_3 > 20\delta_2$ and $p_3 \leq 0.01$ then $1 - \rho > \frac{1}{2}$.

Proof: We split the analysis into several cases based on the value of $x = \mathrm{Dist}(A_{(h,0),(\bar{1},1)}, \textsc{Lin})$.

Case 1: $x \geq \frac{1}{2} - \delta_1$

Lemma 3.5.3 implies that $\textsc{LinPass}(A_{(h,0),(\bar{1},1)}) \leq 1 - \Gamma_{\mathrm{lin}}(x) \leq 1 - x \leq \frac{1}{2} + \delta_1$. Thus, in this case

$$\mathtt{ACC}[\, V^{A,A_1}_{\mathrm{PCPinner}}(\sigma, h)\,] \;\leq\; \rho_1 \stackrel{\mathrm{def}}{=} (1 - p_1) \cdot \left(\frac{1}{2} + \delta_1\right) + p_1 \cdot \left(\frac{1}{2} + \delta_1\right)^2 \;<\; \frac{1}{2} + \delta_1 - \frac{p_1}{10}$$

(The last inequality is due to $\delta_1 < 0.1$.) Using $p_1 > 10\delta_1$ we get $\rho_1 < 1/2$.

The PCP inner verifier. This (l, l_1)-canonical inner verifier is given functions $h \in \mathcal{F}_l$ and $\sigma \colon \Sigma^l \to \Sigma^{l_1}$, and has access to oracles for $A \colon \mathcal{F}_l \to \Sigma$ and $A_1 \colon \mathcal{F}_{l_1} \to \Sigma$. In addition it takes three non-negative parameters p_1, p_2 and p_3 which sum-up to 1.

Pick functions $f_1, \ldots, f_8 \stackrel{R}{\leftarrow} \mathcal{F}_l$ and $g_1, g_2 \stackrel{R}{\leftarrow} \mathcal{F}_{l_1}$.

Step 1: Linearity Test

LinTest$(A_{(h,0),(\bar{1},1)}; f_1, f_2)$.

Step 2: Combined RMB and Projection Test

MBTest$(A_{(h,0),(\bar{1},1)}; f_3, f_4, f_5)$.

ProjTest$_\sigma(A_{(h,0),(\bar{1},1)}, A_1; f_3, g_1)$.

Step 3: Invoking V_{SNPinner} with parameters p_1, p_2, p_3.

Pick $p \stackrel{R}{\leftarrow} [0, 1]$.

Case $p \leq p_1$: **LinTest**$(A_{(h,0),(\bar{1},1)}; f_6, f_7)$.

Case $p_1 < p \leq p_1 + p_2$: **MBTest**$(A_{(h,0),(\bar{1},1)}; f_6, f_7, f_8)$.

Case $p_1 + p_2 < p$: **ProjTest**$_\sigma(A_{(h,0),(\bar{1},1)}, A_1; f_6, g_2)$.

Accept iff all the above tests accept.

Remark: access to $A_{(h,0),(\bar{1},1)}(f)$ is implemented by accessing either $A(f)$, $A(f+h)$, $A(f+\bar{1})$ or $A(f + h + \bar{1})$.

Figure 3.10: *The PCP inner verifier* V_{PCPinner}

Case 2: $x < \frac{1}{2} - \delta_1$

Let $\tilde{A}: \mathcal{F}_l \to \Sigma$ be a linear function such that $\mathrm{Dist}(A_{(h,0),(\bar{1},1)}, \tilde{A}) = x$. The proof splits into two subcases.

Case 2.1: $\tilde{A}$ does not respect the monomial basis

In this case, by Lemmas 3.5.3 and 3.5.7 we have

$$\begin{aligned}
\texttt{ACC}[\,V^{A,A_1}_{\mathrm{PCPinner}}(\sigma,h)\,] &\leq (1-\Gamma_{\mathrm{lin}}(x))\cdot(1-\Gamma_{\mathrm{RMB}}(x))\cdot(1-p_1\Gamma_{\mathrm{lin}}(x)-p_2\Gamma_{\mathrm{RMB}}(x)) \\
&< (1-\Gamma_{\mathrm{lin}}(x))\cdot(1-\Gamma_{\mathrm{RMB}}(x)) \\
&\quad \cdot\left(1-\frac{p_1}{p_1+p_2}\cdot\Gamma_{\mathrm{lin}}(x)-\frac{p_2}{p_1+p_2}\cdot\Gamma_{\mathrm{RMB}}(x)+\frac{p_3}{1-p_3}\right) \\
&< \alpha\cdot\beta\cdot[q\alpha+(1-q)\beta]+\frac{p_3}{1-p_3}
\end{aligned}$$

where $q \stackrel{\mathrm{def}}{=} \frac{p_1}{p_1+p_2}$, $\alpha = 1-\Gamma_{\mathrm{lin}}(x)$ and $\beta = 1-\Gamma_{\mathrm{RMB}}(x)$. Using $p\cdot x+(1-p)\cdot y \geq x^p\cdot y^{1-p}$, we show that $\alpha\cdot\beta\cdot[q\alpha+(1-q)\beta] \leq [\frac{1+q}{3}\alpha+\frac{2-q}{3}\beta]^3$. Specifically,

$$\begin{aligned}
\left[\frac{1+q}{3}\alpha+\frac{2-q}{3}\beta\right]^3 &= \left[\frac{2}{3}\cdot(\frac{1}{2}\alpha+\frac{1}{2}\beta)+\frac{1}{3}\cdot(q\alpha+(1-q)\beta)\right]^3 \\
&\geq \left(\frac{1}{2}\alpha+\frac{1}{2}\beta\right)^{\frac{2}{3}\cdot 3}\cdot(q\alpha+(1-q)\beta)^{\frac{1}{3}\cdot 3} \\
&= \left[\frac{1}{2}\cdot\alpha+\frac{1}{2}\cdot\beta\right]^2\cdot(q\alpha+(1-q)\beta) \\
&\geq \alpha\cdot\beta\cdot(q\alpha+(1-q)\beta)
\end{aligned}$$

Combining the above with Claim 3.6.2 (i.e., the bound on T_2), we obtain (for every $x < 1/2$)

$$\begin{aligned}
\texttt{ACC}[\,V^{A,A_1}_{\mathrm{PCPinner}}(\sigma,h)\,] &< \left[1-\frac{1+q}{3}\cdot\Gamma_{\mathrm{lin}}(x)-\frac{2-q}{3}\cdot\Gamma_{\mathrm{RMB}}(x)\right]^3+\frac{p_3}{1-p_3} \\
&\leq \left[1-\min(\frac{1+q}{6},\frac{2-q}{8})\right]^3+\frac{p_3}{1-p_3}
\end{aligned}$$

Observe that $\min(\frac{1+q}{6},\frac{2-q}{8})$ is maximized at $q = 2/7$ where its value is $3/14$. Indeed this value of q is consistent with $p_1 = \frac{2}{7}\cdot(p_1+p_2)$ and so, in this case, we get

$$\texttt{ACC}[\,V^{A,A_1}_{\mathrm{PCPinner}}(\sigma,h)\,] \leq \rho_2 \stackrel{\mathrm{def}}{=} \left[\frac{11}{14}\right]^3+\frac{p_3}{1-p_3} < 0.48505832+\frac{p_3}{1-p_3}$$

Using $p_3 \leq 0.01$ we get $\rho_2 < 1/2$.

Case 2.2: $\tilde{A}$ respects the monomial basis

By Proposition 3.3.2, $\tilde{A}$ is an evaluation operator. So there exists $a \in \Sigma^l$ such that $\tilde{A} = E_a$. So $\mathrm{Dist}(A_{(h,0),(\bar{1},1)}, E_a) = x$. Let $a_1 = \sigma(a)$. The proof splits into two further sub-cases.

Case 2.2.1: $d \stackrel{\mathrm{def}}{=} \mathrm{Dist}(A_1, E_{a_1}) \geq 1/2 - \delta_2$

By Lemma 3.5.8 we have $\textsc{ProjPass}_\sigma(A_{(h,0),(\bar{1},1)}, A_1) \leq 1 - d \cdot (1-2x) < \frac{1}{2} + x + \delta_2$. Letting $\Gamma_{\rm PRJ}(x) \stackrel{\rm def}{=} \frac{1}{2} - x - \delta_2$, we get in this case

$$\texttt{ACC}[\,V^{A,A_1}_{\rm PCPinner}(\sigma,h)\,] \;\leq\; \rho_3 \stackrel{\rm def}{=} (1-\Gamma_{\rm lin}(x))\cdot(1-\Gamma_{\rm PRJ}(x))\cdot(1-p_1\Gamma_{\rm lin}(x)-p_3\Gamma_{\rm PRJ}(x))$$

We upper bound ρ_3 by considering three sub-cases (corresponding to the segments of $\Gamma_{\rm lin}$).

Case 2.2.1.1: $x \leq 1/4$. In this case we use $\Gamma_{\rm lin}(x) \geq 3x(1-2x)$ and obtain

$$\begin{aligned}
\rho_3 &< (1-\Gamma_{\rm lin}(x))\cdot(1-\Gamma_{\rm PRJ}(x))\cdot(1-p_3\Gamma_{\rm PRJ}(x)) \\
&< (1-3x(1-2x))\cdot(\frac{1}{2}+x+\delta_2)\cdot(1-\frac{p_3}{10}) \\
&< \frac{1}{2}\cdot[1-x+12x^3]\cdot\left[1-\frac{p_3}{10}\right]+\delta_2 \\
&\leq \frac{1}{2}\cdot\left[1-\frac{p_3}{10}\right]+\delta_2
\end{aligned}$$

where the last inequality uses the fact that the function $x - 12x^3$ is non-negative in the interval $[0,1/4]$. Using $p_3 > 20\delta_2$ we obtain $\rho_3 < 1/2$.

Case 2.2.1.2: $x \geq 1/4$ and $x \leq 45/125$. In this case we use $\Gamma_{\rm lin}(x) \geq 45/128 = \Gamma_{\rm lin}(45/128)$ and $\Gamma_{\rm PRJ}(x) \geq \Gamma_{\rm PRJ}(45/128)$ and obtain

$$\begin{aligned}
\rho_3 &< (1-\Gamma_{\rm lin}(x))\cdot(1-\Gamma_{\rm PRJ}(x))\cdot(1-p_1\Gamma_{\rm lin}(x)) \\
&\leq (1-\Gamma_{\rm lin}(45/128))\cdot(1-\Gamma_{\rm PRJ}(45/128))\cdot(1-p_1\Gamma_{\rm lin}(45/128)) \\
&< \frac{83}{128}\cdot\left(\frac{109}{128}+\delta_2\right)\cdot\left(1-p_1\frac{45}{128}\right) \\
&< (0.55218507+\delta_2)\cdot\left(1-p_1\frac{45}{128}\right)
\end{aligned}$$

Using $\delta_2 < \frac{p_3}{10} < 0.001$ and $p_1 \geq \frac{2}{7}\cdot 0.99 > 0.28$, we obtain $\rho_3 < 0.5532 \cdot 0.902 < 0.499$.

Case 2.2.1.3: $x \geq 45/128$. In this case we use $\Gamma_{\rm lin}(x) \geq x \geq 45/128$ and obtain

$$\begin{aligned}
\rho_3 &< (1-\Gamma_{\rm lin}(x))\cdot(1-\Gamma_{\rm PRJ}(x))\cdot(1-p_1\Gamma_{\rm lin}(x)) \\
&< (1-x)\cdot(\frac{1}{2}+x+\delta_2)\cdot(1-p_1\frac{45}{128})
\end{aligned}$$

The latter expression decreases in the interval $[\frac{45}{128}, \frac{1}{2}]$ and is hence maximized at $x = 45/128$. Thus we obtain the same expression as in Case 2.2.1.2, and the bound on ρ_3 follows identically.

We conclude that in Case (2.2.1) we have

$$\rho_3 \;<\; \max\left[\frac{1}{2}-\frac{p_3}{20}+\delta_2\,,\,(0.55218507+\delta_2)\cdot(1-p_1\frac{45}{128})\right]$$

Case 2.2.2: Else ($d < 1/2 - \delta_2$)

In this case, we have $x = \mathrm{Dist}(A_{(h,0),(\bar{1},1)}, E_a) \leq 1/2 - \delta_1$ and $\mathrm{Dist}(A_1, E_{a_1}) < 1/2 - \delta_2$. Thus the functions $A_{(h,0),(\bar{1},1)}$ and A_1 satisfy the properties required in conditions (2.1) and (2.2) of Definition 3.4.3.

Let $\rho \stackrel{\text{def}}{=} \max\{\rho_1, \rho_2, \rho_3\}$. We conclude that the only case which allows $\texttt{ACC}\,[\,V^{A,A_1}_{\text{PCPinner}}(\sigma, h)\,] > \rho$ is Case (2.2.2) which also satisfies conditions (2.1) and (2.2) of Definition 3.4.3. Thus, V_{PCPinner} satisfies condition (2) of Definition 3.4.3. Clearly, V_{PCPinner} also satisfies condition (1) of Definition 3.4.3, and thus the lemma follows. ▌

3.10.2 The new proof system

Combining the above inner verifier with an adequate outer verifier, we obtain a pcp system for NP with query complexity 11.

Theorem 3.10.2 : $\text{NP} = \text{PCP}_{1,1/2}[\,\mathsf{coins} = \log\,;\ \mathsf{query} = 11\,;\ \mathsf{query}_{\text{av}} = 10.89\,]$. Furthermore, the free-bit complexity of the proof system is 7.

Proof: We consider a canonical (l, l_1)-inner verifier V_{PCPinner} with parameters $p_3 = 0.001$, $p_1 = \frac{2}{7} \cdot 0.999$ and $p_2 = \frac{5}{7} \cdot 0.999$. By Lemma 3.10.1, V_{PCPinner} is $(\rho, \delta_1, \delta_2)$-good for $\delta_1 = \delta_2 = 0.00001$ and $\rho = 0.49999 > \max(0.499, 0.5 - \frac{p_3}{10} + \delta_2)$. We now choose an appropriate outer verifier. Let $\epsilon = 16 \cdot (0.5 - \rho)\delta_1^2\delta_2^2$. Lemma 3.4.2 provides us with l and l_1 such that an ϵ-good (l, l_1)-canonical outer verifier V_{outer} with randomness $O(\log n)$ exists. Let $V = \langle V_{\text{outer}}, V_{\text{PCPinner}} \rangle$ be the composition of V_{outer} and V_{PCPinner} according to the definitions in Section 3.4. This verifier has randomness $O(\log n)$. Apply Theorem 3.4.5 to see that V has completeness parameter 1 and soundness parameter $\rho + \epsilon/(16\delta_1^2\delta_2^2) = 1/2$. The query (and free-bit) complexity of V is the same as that of V_{PCPinner} above (i.e., 11 and 7, respectively).

To obtain the bound on the average query complexity, we observe that we can afford not to perform the RMB test with some small probability. Specifically, Case (2.1) in the proof of Lemma 3.10.1, which is the only case where the RMB test is used, yields error of $0.48505832 + \frac{p_3}{1-p_3}$. Thus, if we modify V_{PCPinner} so that, whenever the RMB test is invoked it is performed only with probability 0.973, we get that Case (2.1) detects violation with probability at least $(1 - 0.48505832 - 0.0010011) \cdot 0.973 > 0.50006$. Consequently, the modified inner verifier errs with probability bounded away from 1/2 and so does the composed verifier. The modification decreases the average query complexity by $(1 - 0.973) \cdot (2 + p_2 \cdot 3) > 0.027 \cdot 4.12 > 0.11$. (The reduction is both from Step (2) and the second case in Step (3).) The theorem follows. ▌

3.11 The iterated tests

The iterated tests will be used in our two free-bits proof system. We will be running each of the atomic tests many times, but, to keep the free-bit count low, these will not be independent repetitions. Rather, following [BeSu], we will run about $2^{O(m)}$ copies of each test in a way which is pairwise, or "almost" pairwise independent, to lower the error probability to $O(2^{-m})$. This will be done using $2m$ free-bits. Specifically, we will select uniformly m functions in $\mathcal{F}_l$ (and m functions in $\mathcal{F}_{l_1}$) and invoke the atomic tests with functions resulting from all possible linear combinations of the selected functions.

3.11.1 Linearity and randomness

We begin with some observations relating probabilistic to linear independence. Note that $\mathcal{L}_m$ is a sub-vector-space of $\mathcal{F}_m$, and in particular a vector space over Σ in its own right. So we can discuss the linear independence of functions in $\mathcal{L}_m$. We say that $\vec{L} = (L_1, \ldots, L_k) \in \mathcal{L}_m^k$ is linearly independent if $L_1, \ldots, L_k$ are linearly independent. Furthermore we say that $\vec{L}_1 = (L_{1,1}, \ldots, L_{1,k})$ and $\vec{L}_2 = (L_{2,1}, \ldots, L_{2,k})$ are *mutually linearly independent* if the $2k$ functions $L_{1,1}, L_{2,1}, \ldots, L_{1,k}, L_{2,k}$ are linearly independent.

Lemma 3.11.1 For $\vec{L} = (L_1, \ldots, L_k) \in \mathcal{L}_m^k$ let $J_{\vec{L}} \colon \mathcal{F}_l^m \to \mathcal{F}_l^k$ be defined by $J_{\vec{L}}(\vec{f}) = (L_1 \circ \vec{f}, \ldots, L_k \circ \vec{f})$, for $\vec{f} = (f_1, ..., f_m)$. Fix $\vec{L}$ and consider the probability space defined by having $f_1, ..., f_m$ be uniformly and independently distributed over $\mathcal{F}_l$. Regard the $J_{\vec{L}}$'s as random variables over the above probability space.

(1) If $\vec{L}$ is linearly independent then $J_{\vec{L}}$ is uniformly distributed in $\mathcal{F}_l^k$.

(2) If $\vec{L}_1, \vec{L}_2$ are mutually linearly independent then $J_{\vec{L}_1}$ and $J_{\vec{L}_2}$ are independently distributed.

The analysis of the Iterated Projection test (see Figure 3.11) can be done relatively straightforwardly, given the above, because the invoked projection test uses a single linear combination rather than several such combinations (as in the other iterated tests). Thus we begin with the iterated projection tests. The analysis of the other iterated tests, where the atomic tests are invoked on two/three linear combinations, require slightly more care. The corresponding lemmas could have been proven using the notion of "weak pairwise independence" introduced in [BeSu]. However, we present here an alternative approach.

3.11.2 Iterated projection test

The *iterated projection test* described in Figure 3.11 takes as input a vector $\vec{f} \in \mathcal{F}_l^m$ and also a linear function $L \in \mathcal{L}_m$. Note that $f = L \circ \vec{f}$ is in $\mathcal{F}_l$. The test is just the atomic projection test on this input. The following lemma says that if the passing probability $\text{ProjPass}_A^m()$, representing 2^m invocations of the atomic projection test, is even slightly significant and if A is close to E_a, then A_1 is close to the encoding of the projection of a.

Lemma 3.11.2 There is a constant c_3 such that the following is true. Let $\sigma \colon \Sigma^l \to \Sigma^{l_1}$ be a function. Let $a \in \Sigma^l$ be such that $\text{Dist}(E_a, A) \le 1/4$, and let $a_1 = \sigma(a) \in \Sigma^{l_1}$. If $\text{ProjPass}_\sigma^m(A, A_1) \ge c_3 \cdot 2^{-m}$ then $\text{Dist}(E_{a_1}, A_1) \le 0.1$.

Proof: The proof is similar to that of [BeSu, Lemma 3.5]. Let $\epsilon_1 = \text{Dist}(A_1, E_{a_1})$ and assume it is at least 0.1. We show that there is a constant c_3 such that $\text{ProjPass}_h^m(A) < c_3 \cdot 2^{-m}$.

Let $N = |\mathcal{L}_m^*| = 2^m - 1$. For $L \in \mathcal{L}_m^*$ let $X_L \colon \mathcal{F}_l^m \times \mathcal{F}_{l_1}^m \to \Sigma$ be defined by

$$X_L(\vec{f}, \vec{g}) \stackrel{\text{def}}{=} \mathbf{ProjTest}_\sigma^m(A, A_1; \vec{f}, \vec{g}, L) \;=\; \mathbf{ProjTest}_\sigma(A, A_1; L \circ \vec{f},\, L \circ \vec{g})\,.$$

Regard it as a random variable over the uniform distribution on $\mathcal{F}_l^m \times \mathcal{F}_{l_1}^m$. Let $X = \sum_{L \in \mathcal{L}_m^*} X_L$. It suffices to show that $\Pr[\, X = 0 \,] \le O(1/N)$.

Lemma 3.11.1 implies that $\{X_L\}_{L \in \mathcal{L}_m^*}$ are pairwise independent, identically distributed random variables. Let $L \in \mathcal{L}_m^*$ and let $p = \mathbf{E}[X_L]$. Again using Lemma 3.11.1 we have

$$\begin{aligned} p &= \Pr_{\vec{f} \stackrel{R}{\leftarrow} \mathcal{F}_l^m \,;\, \vec{g} \stackrel{R}{\leftarrow} \mathcal{F}_{l_1}^m} \left[\mathbf{ProjTest}_\sigma(A, A_1; L \circ \vec{f},\, L \circ \vec{g}) = 1 \right] \\ &= \Pr_{f \stackrel{R}{\leftarrow} \mathcal{F}_l \,;\, g \stackrel{R}{\leftarrow} \mathcal{F}_{l_1}} \left[\mathbf{ProjTest}_\sigma(A, A_1; f, g) = 1 \right]. \end{aligned}$$

But by Lemma 3.5.8, p is at least $\epsilon_1(1 - 2\epsilon) \ge 0.05$, since $\epsilon \stackrel{\text{def}}{=} \text{Dist}(E_a, A) \le 1/4$. We can conclude by applying Chebyshev's inequality. Namely,

$$\Pr[\, X = 0 \,] \;\le\; \Pr[\, |X - Np| \ge Np \,] \;\le\; \frac{Np}{(Np)^2} \;\le\; \frac{20}{N}$$

as desired. ▮

3.11.3 Technical claim

For analyzing the other two tests we will use the following simple claim.

Claim 3.11.3 Let $k \geq 1$ and $N = 2^m$. Then $\mathcal{L}_m^k$ contains a subset S of cardinality $\frac{N}{2^{2k}}$ such that every $\vec{L}_1 \neq \vec{L}_2 \in S$ are mutually linearly independent.

Proof: Let $\vec{L} \in \mathcal{L}_m^k$ be linearly independent. Then, the probability that L chosen uniformly in $\mathcal{L}_m$ is linearly independent of $\vec{L}$ is $1 - \frac{2^k}{N}$. Thus, the probability that a uniformly chosen $\vec{L}' \in \mathcal{L}_m^k$ is mutually linearly independent of $\vec{L}$ is greater than $1 - \sum_{i=1} k \frac{2^{k+i-1}}{N} > 1 - \frac{2^{2k}}{N}$. Now, consider a graph with vertex set $\mathcal{L}_m^k$ and edges connecting pairs of mutually linearly independent sequences (i.e., $\vec{L}_1$ and $\vec{L}_2$ are connected if and only they are mutually linearly independent). This graph has N^k vertices and every vertex which is linearly independent has degree greater than $(1 - \frac{2^{2k}}{N}) \cdot N^k$. Clearly this graph has a clique of size $\frac{N}{2^{2k}}$ (e.g., consider a greedy algorithm which pick a vertex of maximal degree among all vertices connected to the previously selected vertices). Noting that a clique corresponds to a set of mutually linear independent sequences, we are done. ▮

3.11.4 Iterated linearity test

The *iterated linearity test* described in Figure 3.11 takes as input a vector $\vec{f} \in \mathcal{F}_l^m$ and also linear functions $L_1, L_2 \in \mathcal{L}_m$. Note that $f_1 = L_1 \circ \vec{f}$ and $f_2 = L_2 \circ \vec{f}$ are in $\mathcal{F}_l$. The test is just the atomic linearity test on these inputs. The following lemma says that if the passing probability is even slightly significant, then A is almost linear.

Lemma 3.11.4 There is a constant c_1 such that if $\textsc{LinPass}^m(A) \geq c_1 \cdot 2^{-m}$ then $\mathrm{Dist}(A, \textsc{Lin}) \leq 0.1$.

Proof: Assume that $\epsilon \stackrel{\mathrm{def}}{=} \mathrm{Dist}(A, \textsc{Lin}) \geq 0.1$. We show that there is a constant c_1 such that $\textsc{LinPass}^m(A) < c_1 \cdot 2^{-m}$. Let $N = 2^m$. For $\vec{L} = (L_1, L_2) \in \mathcal{L}_m^2$ let $X_{\vec{L}} \colon \mathcal{F}_l^m \to \Sigma$ be defined by

$$X_{\vec{L}}(\vec{f}) \stackrel{\mathrm{def}}{=} \mathbf{LinTest}^m(A; \vec{f}, L_1, L_2) \;=\; \mathbf{LinTest}(A; L_1 \circ \vec{f},\, L_2 \circ \vec{f})\,.$$

Regard it as a random variable over the uniform distribution on $\mathcal{F}_l^m$. Let $S \subset \mathcal{L}_m^2$ be a set as guaranteed by Claim 3.11.3 and $X = \sum_{\vec{L} \in S} X_{\vec{L}}$. It suffices to show that $\Pr[\, X = 0 \,] \leq O(1/N)$. (Thus our analysis of $\textsc{LinPass}^m(A)$ is based only on a small fraction of all possible invocations of the iterated linear test; yet, this small fraction corresponds to a sufficiently large number of invocations.)

Using Lemma 3.11.1, it follows that the random variables $\{X_{\vec{L}}\}_{\vec{L} \in S}$ are pairwise independent and that for every $\vec{L} \in S$

$$p \stackrel{\mathrm{def}}{=} \Pr_{\vec{f} \stackrel{R}{\leftarrow} \mathcal{F}_l^m}\left[X_{\vec{L}}(\vec{f}) = 1\right] \;=\; \Pr_{f_1, f_2 \stackrel{R}{\leftarrow} \mathcal{F}_l}\left[\mathbf{LinTest}(A; f_1, f_2) = 1\right].$$

By Lemma 3.5.3, $p \geq \Gamma_{\mathrm{lin}}(\epsilon)$ and so $p \geq 3\epsilon - 6\epsilon^2$ if $\epsilon \leq 1/4$ and $p \geq 45/128$ otherwise. In either case, we get $p > 0.2$. Now by Chebyshev's inequality we have

$$\Pr[\, X = 0 \,] \leq \Pr[\, |X - N'p| \geq N'p \,] \leq O(1/N')$$

where $N' \stackrel{\mathrm{def}}{=} |S| = 2^m/16$. The lemma follows. ▮

The Iterated Tests. Here $A\colon \mathcal{F}_l \to \Sigma$ and $A_1\colon \mathcal{F}_{l_1} \to \Sigma$ are the objects being tested. The tests also take additional inputs or parameters: below $\vec{f} \in \mathcal{F}_l^m$; $\vec{g} \in \mathcal{F}_{l_1}^m$; $L, L_1, L_2, L_3 \in \mathcal{L}_m$; and $\sigma\colon \Sigma^l \to \Sigma^{l_1}$. The tests are specified in terms of the atomic tests of Figure 3.2.

$\mathbf{LinTest}^m(A; \vec{f}, L_1, L_2) = \mathbf{LinTest}(A; L_1 \circ \vec{f},\ L_2 \circ \vec{f})$.

$\mathbf{MBTest}^m(A; \vec{f}, L_1, L_2, L_3) = \mathbf{MBTest}(A; L_1 \circ \vec{f},\ L_2 \circ \vec{f},\ L_3 \circ \vec{f})$.

$\mathbf{ProjTest}_\sigma^m(A, A_1; \vec{f}, \vec{g}, L) = \mathbf{ProjTest}_\sigma(A, A_1; L \circ \vec{f},\ L \circ \vec{g})$.

The Passing Probabilities. These are the probabilities we are interested in:

$$\begin{aligned}
\textsc{LinPass}^m(A) &= \Pr_{\vec{f} \stackrel{R}{\leftarrow} \mathcal{F}_l^m}\left[\forall\, L_1, L_2 \in \mathcal{L}_m \,:\, \mathbf{LinTest}^m(A; \vec{f}, L_1, L_2) = 0\right] \\
\textsc{MBPass}^m(A) &= \Pr_{\vec{f} \stackrel{R}{\leftarrow} \mathcal{F}_l^m}\left[\forall\, L_1, L_2, L_3 \in \mathcal{L}_m \,:\, \mathbf{MBTest}^m(A; \vec{f}, L_1, L_2, L_3) = 0\right] \\
\textsc{ProjPass}_\sigma^m(A, A_1) &= \Pr_{\vec{f} \stackrel{R}{\leftarrow} \mathcal{F}_l^m\,;\, \vec{g} \stackrel{R}{\leftarrow} \mathcal{F}_{l_1}^m}\left[\forall\, L \in \mathcal{L}_m \,:\, \mathbf{ProjTest}_\sigma^m(A, A_1; \vec{f}, \vec{g}, L) = 0\right]
\end{aligned}$$

Figure 3.11: *The iterated tests and their passing probabilities.*

3.11.5 Iterated RMB test

The *iterated respect of monomial basis test* in Figure 3.11 takes an input $\vec{f}$ and also three linear functions $L_1, L_2, L_3 \in \mathcal{L}_m$. For simplicity of exposition, we assume that A is folded over $(\bar{1}, 1)$. (This assumption is justified by our usage of the test – see next subsection.) If the probability $\textsc{MBPass}^m(A)$ is significant, we can conclude that the linear function close to A respects the monomial basis.

Lemma 3.11.5 There is a constant c_2 such that the following is true. Let $A\colon \mathcal{F}_l \to \Sigma$ so that $A(f + \bar{1}) = A(f) + 1$, for every $f \in \mathcal{F}_l$. Let $\epsilon \le 0.1$ so that A is ϵ-close to a linear function $\tilde{A}$ and suppose that $\textsc{MBPass}^m(A) \ge c_2 \cdot 2^{-m}$. Then $\tilde{A}$ respects the monomial basis.

Proof: Assume that $\tilde{A}$ is linear but does not respect the monomial basis. We will show that there is a constant c_2 such that $\textsc{MBPass}^m(A) < c_2 \cdot 2^{-m}$.

Let $N = 2^m$. For $\vec{L} = (L_1, L_2, L_3) \in \mathcal{L}_m^3$ let $X_{\vec{L}}\colon \mathcal{F}_l^m \to \Sigma$ be defined by

$$X_{\vec{L}}(\vec{f}) \stackrel{\text{def}}{=} \mathbf{MBTest}^m(A; \vec{f}, L_1, L_2, L_3) \;=\; \mathbf{MBTest}(A; L_1 \circ \vec{f},\ L_2 \circ \vec{f},\ L_3 \circ \vec{f})\,.$$

Regard it as a random variable over the uniform distribution on $\mathcal{F}_l^m$. Again, let $S \subset \mathcal{L}_m^3$ be a set as guaranteed by Claim 3.11.3 and $X = \sum_{\vec{L} \in S} X_{\vec{L}}$. It suffices to show that $\Pr[\,X = 0\,] \le O(1/N)$.

Using Lemma 3.11.1, it follows that the random variables $\{X_{\vec{L}}\}_{\vec{L}\in S}$ are pairwise independent and that for every $\vec{L} \in S$

$$p \stackrel{\rm def}{=} \Pr_{\vec{f} \stackrel{R}{\leftarrow} \mathcal{F}_l^m}\left[X_{\vec{L}}(\vec{f}) = 1\right] \;=\; \Pr_{f_1,f_2,f_3 \stackrel{R}{\leftarrow} \mathcal{F}_l}\left[\mathbf{MBTest}(A; f_1, f_2, f_3) = 1\right].$$

By Lemma 3.5.7, $p \geq 3/8 - 7\epsilon/4 + 5\epsilon^2/2 - \epsilon^3$. Using $\epsilon \leq 0.1$, it follows that $p > 0.2$. Using Chebyshev's inequality we are done. ∎

Remark. For general A's (which are not folded over $(\bar{1}, 1)$) a similar result can be proven by augmenting the iterated RMB test so that on input A, $\vec{f}$ and $\vec{L} = (L_1, L_2, L_3)$ it also checks if $A((L_1 \circ \vec{f}) + \bar{1}) = A(L_1 \circ \vec{f}) + 1$.

3.11.6 Putting some things together

The last two lemmas above allow us to conclude that if $A_{(h,0),(\bar{1},1)}$ passes the first two tests with any significant probability then $A_{(h,0),(\bar{1},1)}$ is close to some evaluation operator E_a so that $h(a) = 0$. Thus, again, there is no need for a "circuit test".

Corollary 3.11.6 There is a constant c such that the following is true. Let $A\colon \mathcal{F}_l \to \Sigma$, and suppose $\textsc{LinPass}^m(A_{(h,0),(\bar{1},1)}) \geq c \cdot 2^{-m}$ and $\textsc{MBPass}^m(A_{(h,0),(\bar{1},1)}) \geq c \cdot 2^{-m}$. Then there is a string $a \in \Sigma^l$ such that $\mathrm{Dist}(E_a, A_{(h,0),(\bar{1},1)}) \leq 0.1$ and $h(a) = 0$.

Proof: Let c be the larger of the constants from Lemmas 3.11.4 and 3.11.5. By the first lemma there is a linear $\tilde{A}$ such that $\mathrm{Dist}(A_{(h,0),(\bar{1},1)}, \tilde{A}) < 0.1$. Now the second lemma implies that $\tilde{A}$ respects the monomial basis (using the fact that $A_{(h,0),(\bar{1},1)}(f+\bar{1}) = A_{(h,0),(\bar{1},1)}(f)+1$ for all f's). So Proposition 3.3.2 says $\tilde{A}$ is an evaluation function. Finally, by Proposition 3.3.3, we have $h(a) = 0$. ∎

3.12 Amortized free bits, Max Clique, and Coloring

3.12.1 Definitions

A *clique* in a graph $G = (V, E)$ is a subset S of the vertices such that any pair of vertices in S is connected by an edge. We let $\mathsf{MaxClique}(G) = \max\{\,|S| : S \text{ is a clique in } G\,\}$ denote the maximum clique size, and we let $\overline{\mathsf{MaxClique}}(G) = \mathsf{MaxClique}(G)/N$ be the ratio of the Max Clique size to the number of nodes $N = \|G\|$ in the graph. Max Clique is the problem whose instance is a graph G and one has to find $\mathsf{MaxClique}(G)$. An approximation algorithm A for Max Clique achieves a ratio of $\alpha \in [1, \infty)$ if $\mathsf{MaxClique}(G)/\alpha \leq A(G) \leq \mathsf{MaxClique}(G)$ for all graphs G. Here α is a function of the number N of nodes in G.

The chromatic number of G is the smallest number of colors with which the nodes of G can be colored so that no two adjacent vertices have the same color. It is denoted $\mathsf{ChromNum}(G)$, and as usual $\overline{\mathsf{ChromNum}}(G) = \mathsf{ChromNum}(G)/N$. Coloring is the problem, given G, of finding $\mathsf{ChromNum}(G)$. An approximation algorithm A for coloring achieves a ratio of $\alpha \in [1, \infty)$ if $\mathsf{ChromNum}(G) \leq A(G) \leq \alpha \cdot \mathsf{ChromNum}(G)$ for all graphs G.

Promise problems $\mathsf{Gap}\text{-}\mathrm{Clique}_{c,s}$ and $\mathsf{Gap}\text{-}\mathrm{ChromNum}_{c,s}$ corresponding to the approximation are defined analogously to our previous definitions for other problems. Here c, s are functions of N such that $0 \leq s(N) \leq c(N) \leq 1$.

3.12.2 Sources of our improvements

We adopt the basic framework of the construction of proof systems with low free-bit complexity as presented in [BeSu]. Our improvement comes from the use of the new long code instead of the Hadamard code as a basis for the construction of inner verifiers. This allows us to save one bit in the amortized free-bit complexity. The reason being that the long code contains explicitly all functions of the encoded string whereas the Hadamard code contains only linear combinations of the bits of the string. Typically, we need to check that the verifier accepts a string and this condition is unlikely to be expressed by a linear combination of the bits of the string. Thus, one needs to keep also the linear combinations of all two-bit products and using these extra combinations (via self-correcting) increases the amortized free-bit by one. Instead, as seen above, the long code allows us to directly handle any function. The fact that we take linear combinations of these functions should not confuse the reader; these are linear combinations of random functions rather than being linear combinations of random linear functions (as in [BeSu]).

3.12.3 Construction and results

Our construction of a proof systems with amortized free-bit complexity of two bits is obtained by composing the (l, l_1)-canonical outer verifier of Lemma 3.4.2 with a (l, l_1)-canonical inner verifier, denoted $V_{\text{free-in}}$, which is depicted in Figure 3.12. The inner verifier $V_{\text{free-in}}$ consists of invoking the three iterated tests of Figure 3.11. In addition, $V_{\text{free-in}}$ also applies the linearity test to the oracle A_1. This is not done in order to improve the rejection probability of $V_{\text{free-in}}$ (in case the oracles A and A_1 are far from being fine), but rather in order to decrease the number of accepting configurations (and consequently the free-bit complexity). We also remark that $V_{\text{free-in}}$ invokes the iterated tests while providing them with access to a double folding of A (i.e., $A_{(h,0),(\bar{1},1)}$) rather than to A itself. This eliminates the need for checking that A encodes a string which evaluates to zero under h and simplifies the iterated RMB test (see remark at the end of subsection 3.11.5). However, unlike in previous subsections, these simplifications do not buy us anything significant

The free inner verifier. Given functions $h \in \mathcal{F}_l$ and $\sigma\colon \Sigma^l \to \Sigma^{l_1}$, the verifier has access to oracles for $A\colon \mathcal{F}_l \to \Sigma$ and $A_1\colon \mathcal{F}_{l_1} \to \Sigma$. It also takes an integer parameter m.

Random choices:	$\vec{f} \stackrel{R}{\leftarrow} \mathcal{F}_l^m$; $\vec{g} \stackrel{R}{\leftarrow} \mathcal{F}_{l_1}^m$
$\forall\, L_1, L_2 \in \mathcal{L}_m$:	$\mathbf{LinTest}^m(A_{(h,0),(\bar{1},1)}; \vec{f}, L_1, L_2)$
$\forall\, L_1, L_2, L_3 \in \mathcal{L}_m$:	$\mathbf{MBTest}^m(A_{(h,0),(\bar{1},1)}; \vec{f}, L_1, L_2, L_3)$
$\forall L \in \mathcal{L}_m$:	$\mathbf{ProjTest}_\sigma^m(A_{(h,0),(\bar{1},1)}, A_1; \vec{f}, \vec{g}, L)$
$\forall\, L_1, L_2 \in \mathcal{L}_m$:	$\mathbf{LinTest}^m(A_1; \vec{g}, L_1, L_2)$

Remark: access to $A_{(h,0),(\bar{1},1)}(f)$ is implemented by accessing either $A(f)$, $A(f+h)$, $A(f+\bar{1})$ or $A(f+h+\bar{1})$.

Figure 3.12: *The free inner verifier* $V_{\text{free-in}}$

(here), since the additional testing could have been done without any additional cost in free-bits.

Lemma 3.12.1 There exists a constant c such that the following is true. Let l, l_1, m be integers. Then the (l, l_1)-canonical inner verifier $V_{\text{free-in}}$ with parameter m is $(\rho, \delta_1, \delta_2)$-good, where $\rho = c \cdot 2^{-m}$ and $\delta_i = 0.4$, for $i = 1, 2$.

Proof: Here the analysis can be less careful than in analogous statements such as in Lemmas 3.6.1 and 3.9.2. Using Corollary 3.11.6, with respect to the oracle $A_{(h,0),(\bar{1},1)}$, we conclude that if $A_{(h,0),(\bar{1},1)}$ passed both the iterated Linearity and RMB Tests with probability at least $c \cdot 2^{-m}$ then there exists a string $a \in \Sigma^l$ such that $\text{Dist}(E_a, A_{(h,0),(\bar{1},1)}) \leq 0.1 = \frac{1}{2} - \delta_1 < 1/4$ and $h(a) = 0$. Using Lemma 3.11.2, we conclude that if $(A_{(h,0),(\bar{1},1)}, A_1)$ passed the iterated Projection Test, with probability at least $c_3 \cdot 2^{-m}$, then $\text{Dist}(E_{\sigma(a)}, A_1) < 0.1 = \frac{1}{2} - \delta_2$. Setting $\rho = c' \cdot 2^{-m}$, where $c' = \max\{c, c_3\}$, we conclude that $V_{\text{free-in}}$ satisfies condition (2) of Definition 3.4.3. Clearly, $V_{\text{free-in}}$ also satisfies condition (1) and the lemma follows. ▮

Proposition 3.12.2 Let l, l_1, m be integers. Then the (l, l_1)-canonical inner verifier $V_{\text{free-in}}$ with parameter m uses $2m$ free-bits.

Proof: We consider only accepting computations of $V_{\text{free-in}}$. We start by observing that all oracle values obtained from A, during the iterated Linearity Test (on $A_{(h,0),(\bar{1},1)}$), are determined by the values of A in locations $f'_1, f'_2, ..., f'_m$, where each f'_i is either f_i or $f_i + h$. Likewise, all oracle values obtained from A, during the iterated RMB Test, are determined by the values of A in these locations $f'_1, f'_2, ..., f'_m$. Finally, all oracle values obtained from A, during the iterated Projection Test, are determined by the values of A_1 in locations $L \circ \vec{g}$ (for all L's) and the values of A in the locations $f'_1, f'_2, ..., f'_m$.

Now we use the fact that $V_{\text{free-in}}$ applies an iterated Linearity Test to the oracle A_1. It follows that all oracle values obtained from A_1, in accepting computations of $V_{\text{free-in}}$, are determined by the values of A_1 in locations $g_1, g_2, ..., g_m$.

We conclude that, in accepting computations of $V_{\text{free-in}}$, all values obtained from the oracles are determined by $2m$ bits (i.e., $A(f'_1), ..., A(f'_m)$ and $A_1(g_1), ..., A_1(g_m)$). ▮

Composing the canonical outer verifier of Lemma 3.4.2 and the canonical inner verifier $V_{\text{free-in}}$, we get the following

Theorem 3.12.3 There is a constant c such that the following is true. Let $L \in \text{NP}$ and m an integer. Then $L \in \text{PCP}_{1,s}[\,\textsf{coins} = \log\,;\ \textsf{free} = 2m\,]$ with $s = c \cdot 2^{-m}$.

Proof: Given an NP language L and an integer m, we use Lemma 3.4.2 to construct a 2^{-m}-good outer verifier, denoted V_{outer}, for L. Recall that this outer verifier uses logarithmic randomness (actually the randomness depends linearity on m which is a constant). Next, compose V_{outer} with the inner verifier $V_{\text{free-in}}$, where $V_{\text{free-in}}$ uses m as its integer parameter. The composed verifier has free-bit complexity $2m$ (as inherited from $V_{\text{free-in}}$ by Proposition 3.12.2). By Theorem 3.4.5 the soundness error of the composed verifier is at most $(c+1) \cdot 2^{-m}$, where $c \cdot 2^{-m}$ is the soundness error of $V_{\text{free-in}}$ (due to Lemma 3.12.1). The theorem follows. ▮

By selecting m to be sufficiently large (i.e., $m = (2+\epsilon)\log_2 c/\epsilon$, where c is the constant above), we get

Theorem 3.12.4 For any $\epsilon > 0$ it is the case that $\text{NP} \subseteq \overline{\text{FPCP}}[\log, 2+\epsilon]$.

Using the FGLSS-transformation, we get

Theorem 3.12.5 For any $\epsilon > 0$
(1) NP $\leq_R^K$ Gap-Clique$_{c,s}$ for $s(N) = N^\epsilon$ and $c(N) = N^{1/3}$.
(2) NP $\leq_D^K$ Gap-Clique$_{c,s}$ for $s(N) = N^\epsilon$ and $c(N) = N^{1/4}$.

Proof: For Part (1) we use Corollary 5.2.3 (below), with $r = O(\log n)$ and $k = \frac{r}{\epsilon}$. We get that NP is randomly reducible to a pcp system with randomness $r + k + O(1)$, free-bit complexity $(2+\epsilon)k$ and error probability 2^{-k}. The FGLSS-graph corresponding to the resulting pcp system has size $N = 2^{(r+k+O(1))+(2+\epsilon)k}$ and a gap in clique size of factor 2^k, which can be rewritten as $N^{1/(1+2+2\epsilon)}$. The clique size in case of input not in the language is 2^r which can be rewritten as N^ϵ. Substituting ϵ for $\epsilon/2$, the claim of Part (1) follows. For Part (2) we use Corollary 5.2.5, and get a pcp system for NP with randomness $r + (2+\epsilon)k$, free-bit complexity $(2+\epsilon)k$ and error probability 2^{-k}. Using the FGLSS-construction on this system, the claim of Part (2) follows. ▮

Combining the above with a recent reduction of Furer [Fu], which in turn improved the reductions of [LuYa, KLS, BeSu], we get

Theorem 3.12.6 For any $\epsilon > 0$
(1) NP $\leq_R^K$ Gap-ChromNum$_{c,s}$ for $s(N)/c(N) = N^{1/5-\epsilon}$.
(2) Gap-ChromNum$_{c,s}$ is NP-complete for $s(N)/c(N) = N^{1/7-\epsilon}$.

3.12.4 Previous work

Max Clique. Prior to 1991, no non-approximability results on Max Clique were known. In 1991 the connection to proofs was made by Feige et. al. [FGLSS]. The FGLSS reduction says that $\mathrm{PCP}_{1,e}[\,\mathsf{coins} = r\,;\,\mathsf{query} = q\,]$ Karp reduces to Gap-Clique$_{c,s}$ via a reduction running in time poly(2^{r+q}), and with the gap c/s being a function of $(r, q$ and) the error e. In applying it one works with PCP classes containing NP. One obtains a result saying Max Clique has no polynomial time approximation algorithm achieving a certain factor, under an assumption about the deterministic time complexity of NP (the time complexity depends on r, q and the factor on these, but, most importantly, on the error e). In particular, these authors were able to "scale-down" the proof system of [BFL] to indicate strong non-approximability factors of $2^{\log^\epsilon N}$ for some $\epsilon > 0$, assuming NP is not in quasi-polynomial deterministic time. They also initiated work on improving the factors and assumptions via better proof systems. The best result in their paper is indicated in Figure 3.13.

Arora and Safra [ArSa] reduced the randomness complexity of a PCP verifier for NP to logarithmic — they showed NP $= \mathrm{PCP}_{1,1/2}[\,\mathsf{coins} = \log\,;\,\mathsf{query} = \sqrt{\log N}\,]$. They also observed that random bits can be recycled for error-reduction via the standard techniques [AKS, CW, ImZu]. The consequence was the first NP-hardness result for Max Clique approximation. The corresponding factor was $2^{\sqrt{\log N}}$.

Arora et. al. showed that NP $= \mathrm{PCP}_{1,1/2}[\,\mathsf{coins} = \log\,;\,\mathsf{query} = O(1)\,]$, which implied that there exists an $\epsilon > 0$ for which approximating Max Clique within N^ϵ was NP-complete. The number of queries was unspecified, but indicated to be $\approx 10^4$, so $\epsilon \approx 10^{-4}$. Later work has focused on reducing the constant value of ϵ in the exponent.[5]

In later work a slightly tighter form of the FGLSS reduction due to [BeSc, Zu] has been used. It says that $\mathrm{PCP}_{1,1/2}[\,\mathsf{coins} = r\,;\,\mathsf{query}_{\mathrm{av}} = q_{\mathrm{av}}\,]$ reduces, via a randomized Karp reduction, to

[5] The value $\epsilon = 10^{-4}$ means that the size N of the graph must be at least 2^{1000}, which is more than the number of particles in the universe, before the factor N^ϵ exceeds 2!

Due to	Factor	Assumption
[FGLSS]	$2^{\log^{1-\epsilon} N}$ for any $\epsilon > 0$	NP $\not\subseteq \widetilde{\mathrm{P}}$
[ArSa]	$2^{\sqrt{\log N}}$	P $\neq$ NP
[ALMSS]	N^ϵ for some $\epsilon > 0$	P $\neq$ NP
[BGLR]	$N^{1/25}$	NP $\not\subseteq$ co$\widetilde{\mathrm{RP}}$
[BGLR]	$N^{1/30}$	NP $\neq$ coRP
[FeKi]	$N^{1/15}$	NP $\neq$ coRP
[BeSu]	$N^{1/4}$	NP $\not\subseteq$ co$\widetilde{\mathrm{RP}}$
[BeSu]	$N^{1/6}$	P $\neq$ NP
This paper	$N^{1/4}$	P $\neq$ NP
This paper	$N^{1/3}$	NP $\neq$ coRP

Figure 3.13: *Some Milestones in the project of proving non-approximability of the Clique number: Approximation Factor (in terms of the graph size N) which is infeasible to achieve under an indicated Assumption. In stating results from [BGLR] on, we ignore N^ϵ terms in which $\epsilon > 0$ can be arbitrary small.*

Gap-Clique$_{c,s}$ for some c, s satisfying $c(N)/s(N) = N^{1/(1+q_{\text{av}})}$, and with the running time of the reduction being poly(2^r). (We assume $q_{\text{av}} = O(1)$ for simplicity.) (We omit factors of N^ϵ where $\epsilon > 0$ can be arbitrarily small, here and in the following.) Thus the hardness factor was tied to the (average) number of queries required to get soundness error 1/2. Meanwhile the assumption involved the probabilistic, rather than deterministic time complexity of NP– it would be NP $\not\subseteq$ co$\widetilde{\mathrm{RP}}$ if $r = \text{polylog}(n)$ and NP $\neq$ coRP if $r = \log(n)$.

New proof systems of [BGLR] were able to obtain significantly smaller query complexity: they showed NP $\subseteq \mathrm{PCP}_{1,1/2}[\,\textsf{coins} = \text{polylog}\,;\,\textsf{query} = 24\,]$ and NP $\subseteq \mathrm{PCP}_{1,1/2}[\,\textsf{coins} = \log\,;\,\textsf{query} = 29\,]$. This leads to their hardness results shown in Figure 3.13. However, significantly reducing the (average) number of bits queried seemed hard.

However, as observed by Feige and Kilian, the performance of the FGLSS reduction actually depends on the free-bit complexity which may be significantly smaller than the query complexity [FeKi]. Namely, the factor in the above mentioned reduction is $N^{1/(1+f)}$ where f is the free-bit complexity. They observed that the proof system of [BGLR] has free-bit complexity 14, yielding a $N^{1/15}$ hardness of approximation factor.

The notion of amortized free-bits was introduced in [BeSu]. They observed that the performance of the reduction depended in fact on this quantity, and that the factor was $N^{1/(1+\bar{f})}$ where $\bar{f}$ is the amortized free bit complexity. They then showed that NP $\subseteq \overline{\mathrm{FPCP}}[\text{polylog}, 3]$. This lead to a $N^{1/4}$ hardness factor assuming NP $\neq$ co$\widetilde{\mathrm{RP}}$.

Chromatic Number. The first hardness result for the chromatic number is due to Garey and Johnson [GJ1]. They showed that if P $\neq$ NP then there is no polynomial time algorithm that can achieve a factor less than 2. This remained the best result until the connection to proofs, and the above mentioned results, emerged.

Now hardness results for the chromatic number are obtained via reduction from Max Clique. A

N^{ϵ} factor hardness for Max Clique translates into a N^{δ} factor hardness for the Chromatic number[6], with δ a function of ϵ. To discuss the quality of reductions, let us, following [BeSu], define an (a,b)-reduction to be one that achieves $\delta = \frac{1}{a-b+(b/\epsilon)} = \frac{\epsilon}{b+(a-b)\epsilon}$.

The first reduction, namely that of Lund and Yannakakis [LuYa], was a $(1,5)$-reduction. Via the Max Clique hardness results of [ArSa, ALMSS] this implies the chromatic number is hard to approximate within N^{δ} for some $\delta > 0$. But, again, δ is very, very small. Improvements to δ are a function both of improvements to ϵ and the values a, b for which (a,b)-reductions are available.

A subsequent reduction of Khanna, Linial and Safra [KLS] is simpler but in fact slightly less efficient, being a $(6,5)$-reduction. However a more efficient reduction is given by [BeSu]– they present a $(1,3)$-reduction. Our $N^{1/3}$ hardness for Clique would yield, via this, a $N^{1/7}$ hardness for the chromatic number. But more recently an even more efficient reduction has become available, namely that of Furer [Fu]. It is a $(1,2)$-reduction, and thereby we get our $N^{1/5}$ hardness.

Randomized and de-randomized error reduction. As mentioned above, randomized and de-randomized error reduction techniques play an important role in obtaining the best Clique hardness results via the FGLSS method. Typically, one first reduces the error so that its logarithm relates to the query (or free-bit) complexity and so that the initial randomness cost can be ignored (as long as it were logarithmic). (Otherwise, one would have needed to construct proof systems which minimize also this parameter; i.e., the constant factor in the logarithmic randomness complexity.)

The randomized error reduction method originates in the work of Berman and Schnitger [BeSc] were it is applied to the Clique Gap promise problem. An alternative description is given by Zuckerman [Zu]. Another alternative description, carried out in the proof system, is presented in Section 5.2.

The de-randomized error reduction method consists of applying general, de-randomized, error-reduction techniques to the proof system setting. The best method knows as the "Expander Walk" technique is due to Ajtai, Komlos and Szemeredi [AKS] (see also [CW, ImZu]). It is easy to see that this applies in the pcp context. (The usage of these methods in the pcp context begins with [ArSa].) It turns out that the (constant) parameters of the expander, specifically the ratio $\rho \stackrel{\rm def}{=} \frac{\log_2 d}{\log_2 \lambda}$, where d is the degree of the expander and λ is the second eigenvalue (of its adjacency matrix), play an important role here. In particular, $\rho-1$ determines how much we lose with respect to the randomized error reduction (e.g., $\mathrm{NP} \in \overline{\mathrm{FPCP}}[\log, f]$ translates to a hardness factor of $N^{\frac{1}{1+f}}$ under $\mathrm{NP} \not\subseteq \mathrm{BPP}$ and to a hardness factor of $N^{\frac{1}{\rho+f}}$ under $\mathrm{NP} \neq \mathrm{P}$). Thus the Ramanujan Expander of Lubotzky, Phillips and Sarnak [LPS] play an important role yielding $\rho \approx 2$ (cf. Proposition 5.2.4), which is the best possible.

3.13 The coding theory bound

We provide here the coding theory bound used in the proof of Lemma 3.4.4. It is a slight extension of bounds in [MaSl, Ch. 17] which consider only vectors of weight exactly w rather than at most w. For sake of completeness, we include a proof of this bound. In discussing binary vectors, the weight is the number of ones in the vector and the distance between two vectors is the number of places in which they disagree.

[6]Actually all the reductions presented here, make assumptions regarding the structure of the graph and hence do not directly yield the hardness results stated here. However, as a consequence of some results from this paper, we are able to remove the assumptions made by the earlier papers and hence present those results in a simpler form. See Section 4.1.3 for details.

Lemma 3.13.1 Let $B = B(n, d, w)$ be the maximum number of binary vectors of length n, each with weight at most w, and any two being distance at least d apart. Then $B \leq (1-2\beta)/(4\alpha^2 - 2\beta)$, provided $\alpha^2 > \beta/2$, where $\alpha = (1/2) - (w/n)$ and $\beta = (1/2) - (d/n)$.

Proof: Consider an arbitrary sequence, $v_1, ..., v_M$, of n-vectors which are at mutual distance at least $n/2$. Let us denote by $v_{i,j}$ the j^{th} entry in the i^{th} vector, by w_i the weight of the i^{th} vector, and by $\overline{w}$ the average value of the w_i's. Define

$$S \stackrel{\text{def}}{=} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{k=1}^{n} v_{i,k} v_{j,k}$$

Then, on one hand

$$\begin{aligned} S &= \sum_{i=1}^{M} \sum_{k=1}^{n} v_{i,k}^2 + \sum_{1 \leq i \neq j \leq M} \sum_{k=1}^{n} v_{i,k} v_{j,k} \\ &\leq \sum_i w_i + \sum_{1 \leq i \neq j \leq M} \frac{w_i + w_j - d}{2} \\ &= M\overline{w} + M(M-1) \cdot (\overline{w} - (d/2)) \end{aligned}$$

where the inequality follows from observing that, for $i \neq j$,

$$\begin{aligned} w_i + w_j &= 2|\{k : v_{i,k} = v_{j,k} = 1\}| + |\{k : v_{i,k} \neq v_{j,k}\}| \\ &\geq 2 \sum_{k=1}^{n} v_{i,k} v_{j,k} + d \end{aligned}$$

On the other hand $S = \sum_{k=1}^{n} |\{i : v_{i,k} = 1\}|^2$. This allows to lower bound S by the minimum of $\sum_k x_k^2$ subject to $\sum_k x_k = M\overline{w}$. The minimum is obtained when all x_k's are equal and yields

$$S \geq n \cdot \left(\frac{M\overline{w}}{n}\right)^2$$

Confronting the two bounds, we get

$$\frac{M \cdot \overline{w}^2}{n} \leq M \cdot \overline{w} - (M-1) \cdot (d/2)$$

which yields $(\frac{\overline{w}^2}{n} - \overline{w} + \frac{d}{2})M \leq \frac{d}{2}$. Letting $\overline{\alpha} = (1/2) - (\overline{w}/n)$ and using $\overline{\alpha}^2 \geq \alpha^2 > \beta/2$, we get

$$M \leq \frac{1 - 2\beta}{4\overline{\alpha}^2 - 2\beta}$$

and the lemma follows by observing that the bound maximizes when $\alpha = \overline{\alpha}$. ∎

3.14 On the optimality of some choices in our analysis

In this section we demonstrate the optimality of several of the choices made in the analysis in previous sections.

CHOICE OF THE PROBABILITY PARAMETERS FOR $V_{\rm SNPinner}$. We start by proving that the choice of probabilities for $V_{\rm SNPinner}$ (i.e., requiring the p_i's to satisfy Eq. (3.8)) is optimal for minimizing the soundness upper bound provided by Lemma 3.6.1. Actually, we show that no matter how one selects these probabilities, the expression given in Lemma 3.6.1 is at least 0.85 (i.e., the upper bound provided by Claim 3.6.2).

Claim 3.14.1 For any choice of the parameters $p_1, p_2, p_3 > 0$ so that $p_1 + p_2 + p_3 \leq 1$ one of the following three expressions, $\frac{1}{2} \cdot p_1$, $\min_{x\leq 1/2} [\, p_1 \cdot \Gamma_{\rm lin}(x) + p_2 \cdot \Gamma_{\rm RMB}(x) \,]$ and $\min_{x\leq 1/2} [\, p_1 \cdot \Gamma_{\rm lin}(x) + p_3 \cdot \frac{1}{2}(1-2x) \,]$, is at most 0.15. Furthermore, the minimum of the above expressions is bounded above by

$$\min(\frac{1}{2} \cdot p_1, \frac{3}{8} \cdot p_2, \frac{1}{2} \cdot p_3)$$

Proof: The furthermore clause follows by setting $x = 0$. This expression is maximized at

$$\frac{1}{2} \cdot p_1 \;=\; \frac{3}{8} \cdot p_2 \;=\; \frac{1}{2} \cdot p_3$$

which yields $p_1 = p_3 = 0.3$ and $p_2 = 0.4$, as in Eq. (3.8). Indeed, at this optimum the value is 0.15. ▮

We remark that the setting of x represents plausible existence of oracles for which the proof of Lemma 3.6.1 provides the soundness bounds appearing in the claim. Specifically, Case (1) corresponds to having a first oracle, A, which is far away (i.e., at distance $\approx 1/2$) from being linear. Case (2) corresponds to having A linear but not respecting the monomial basis (and thus at distance $1/2$ from the long code). Finally, Case (3) corresponds to having $A = E_a$ (i.e., a codeword for a) and A_1 be at distance $1/2$ from $E_{\sigma(a)}$.

EVALUATING $V_{\rm SNPinner}$ INDEPENDENTLY OF THE GADGETS. Lemmas 3.7.3 and 3.8.3 (as well as Lemma 3.9.9) provide hardness results for a factor $\frac{1-s}{\alpha_1 \cdot q + \alpha_2 \cdot (1-q)}$, where s and q depend on the verifier being used whereas α_1 and α_2 depend on the gadgets. Specifically, s is the soundness error for a verifier (based on $V_{\rm SNPinner}$) which performs a parity check with probability q and an RMB check with probability $1 - q$. Our approach was to select the probabilities for $V_{\rm SNPinner}$ so to minimize s and this in turn determines $q = 1 - p_2$. A natural question is whether it is not better to allow greater error, s, so to obtain a smaller value for $1 - q$. This is natural since α_1 is likely to be smaller than α_2 (e.g., in SAT gadgets, a Parity Check for $a + b_0 + c = 0$ is obtained as a special case of the RMB check for a, b_0, b_1, c, when setting $b_0 \neq b_1$).

Claim 3.14.2 Let $a_1 < \alpha_2$, $p_1, p_2, p_3 > 0$ s.t. $p_1 + p_2 + p_3 = 1$ and let $s(p_1, p_2, p_3) \stackrel{\rm def}{=} \min(\frac{1}{2}p_1, \frac{3}{8}p_2, \frac{1}{2}p_3)$ be the soundness upper bound provided by Lemma 3.6.1 (and Claim 3.14.1). Then $\frac{1-s(p_1,p_2,p_3)}{\alpha_1 \cdot (1-p_2) + \alpha_2 \cdot p_2}$ is maximized at p_i's satisfying Eq. (3.8) (i.e., $p_1 = p_3 = 0.3$ and $p_2 = 0.4$).

Proof: Let $p_1^* = p_3^* = 0.3$ and $p_2^* = 0.4$ be as in Eq. (3.8) and factor(p_1, p_2, p_3) be the hardness factor obtained by implementing $V_{\rm SNPinner}$ by corresponding gadgets; that is

$$\text{factor}(p_1, p_2, p_3) \stackrel{\rm def}{=} \frac{1 - s(p_1, p_2, p_3)}{\alpha_1 + (\alpha_2 - \alpha_1) \cdot p_2}$$

We consider two cases.

Case 1: $p_2 \geq p_2^*$. In this case either $p_1 < p_1^*$ or $p_3 < p_3^*$. Without loss of generality we assume that $p_1 < p_1^*$. Using Claims 3.6.2 (in the equality) and Claim 3.14.1 (in the inequality), we get

$$\begin{aligned} \text{factor}(p_1^*, p_2^*, p_3^*) &= \frac{\frac{1}{2} \cdot p_1^*}{\alpha_1 + (\alpha_2 - \alpha_1) \cdot p_2^*} \\ \text{factor}(p_1, p_2, p_3) &\leq \frac{\frac{1}{2} \cdot p_1}{\alpha_1 + (\alpha_2 - \alpha_1) \cdot p_2} \end{aligned}$$

and $\text{factor}(p_1^*, p_2^*, p_3^*) > \text{factor}(p_1, p_2, p_3)$ follows easily.

Case 2: $p_2 < p_2^*$. By Claim 3.6.2,

$$\begin{aligned} \text{factor}(p_1^*, p_2^*, p_3^*) &= \frac{\frac{3}{8} \cdot p_2^*}{\alpha_1 + (\alpha_2 - \alpha_1) \cdot p_2^*} \\ &= \frac{3/8}{(\alpha_1/p_2^*) + (\alpha_2 - \alpha_1)} \end{aligned}$$

On the other hand, using Claim 3.14.1, we get

$$\begin{aligned} \text{factor}(p_1, p_2, p_3) &\leq \frac{\frac{3}{8} \cdot p_2}{\alpha_1 + (\alpha_2 - \alpha_1) \cdot p_2} \\ &= \frac{3/8}{(\alpha_1/p_2) + (\alpha_2 - \alpha_1)} \end{aligned}$$

Observing that for $p_2 < p_2^*$ we have $(\alpha_1/p_2) + (\alpha_2 - \alpha_1) > (\alpha_1/p_2^*) + (\alpha_2 - \alpha_1)$, we obtain $\text{factor}(p_1^*, p_2^*, p_3^*) > \text{factor}(p_1, p_2, p_3)$ here too. ▮

Analysis of V_{2inner}. We now show that the choice of the probability parameter for V_{2inner} (i.e., the setting p) is optimal for minimizing the soundness upper bound provided by Lemma 3.9.2. Actually, we show that no matter how one selects this parameter, the expression given in Lemma 3.9.2 is at least 173/218 (i.e., the upper bound provided by Theorem 3.9.4).

Claim 3.14.3 For any choice of the parameter $p \in [0, 1]$ one of the following three expressions, $\frac{1}{2} \cdot p$, $p \cdot \min_{x \leq 1/2 - \delta_1} [\max(\Gamma_{\text{lin}}(x), \frac{3}{4}(1-2x))]$, and $\min_{x \leq 1/2 - \delta_1} [\, p \cdot \Gamma_{\text{lin}}(x) + (1-p) \cdot (\frac{1}{2} - x)\,]$, is at most 45/218. Furthermore, the minimum of the above expressions is bounded above by

$$\min \left[\frac{45}{128} \cdot p \ , \ \frac{1}{2} \cdot (1-p) \right]$$

Proof: Observe that

$$\begin{aligned} T_1 &\stackrel{\text{def}}{=} \min_{x \leq 1/2 - \delta_1} \left[\max(\Gamma_{\text{lin}}(x) \, , \, \frac{3}{4}(1-2x)) \right] \\ &\leq \max \left(\Gamma_{\text{lin}}(45/128), \frac{3}{4}(1 - 2 \cdot \frac{45}{128}) \right) \\ &= \frac{45}{128} \end{aligned}$$

and that

$$\begin{aligned}
T_2 &\stackrel{\mathrm{def}}{=} \min_{x \le 1/2-\delta_1} \left[p \cdot \Gamma_{\mathrm{lin}}(x) + (1-p) \cdot (\frac{1}{2} - x) \right] \\
&\le \min_{45/128 \le x \le 1/2-\delta_1} \left[p \cdot \Gamma_{\mathrm{lin}}(x) + (1-p) \cdot (\frac{1}{2} - x) \right] \\
&= \min_{45/128 \le x \le 1/2-\delta_1} \left[p \cdot x + (1-p) \cdot (\frac{1}{2} - x) \right] \\
&= \frac{1}{2} \cdot (1-p)
\end{aligned}$$

The Furthermore clause follows by observing that $\min[\frac{p}{2}, p \cdot T_1, T_2] \le \min[\frac{45}{128} \cdot p, \frac{1}{2} \cdot (1-p)]$. The latter expression is maximized at $\frac{45}{128} \cdot p = \frac{1}{2} \cdot (1-p)$ which yields $p = 64/109$ (as in the proof of Theorem 3.9.4). Indeed, at this optimum the value is $\frac{45}{128} \cdot \frac{64}{109} = \frac{45}{218}$. ▮

Chapter 4

Proofs and approximation: Potential and limitations

We have seen in the last chapter that non-approximability results are getting steadily stronger, particularly for Max Clique. How far can they go? This chapter is about answering this kind of question.

The first Section describes our "reverse connection" indicating the necessity of proof checking techniques to the derivation of non-approximability results for Max Clique, and pointing to amortized free bits as the crucial parameter. The second Section focuses on lower bounds on amortized free bits which will indicate that our two free bit result of the last section is tight in the light of current techniques. The two together indicate that one needs new techniques to prove better than a $N^{1/3}$ hardness for Max Clique.

4.1 The reverse connection and its consequences

Feige et al. [FGLSS] describe a procedure which takes a verifier V, and an input x and constructs a graph, which we denote $\mathcal{G}_V(x)$, whose vertices correspond to possible accepting transcripts in V's computation and edges corresponding to consistent/non-conflicting computations. They then show the following connection between the maximum (over all possible oracles) acceptance probability of the verifier and the clique size in the graph. Recall that $\texttt{ACC}[\,V(x)\,] = \max_\pi \Pr_R[V^\pi(x;R)=0]$ is the maximum accepting probability. Also recall that $\mathsf{MaxClique}(G)$ is the maximum clique size.

Theorem 4.1.1 ([FGLSS]) If, on input x, a verifier V tosses r coins then the following relationship holds:

$$\texttt{ACC}[\,V(x)\,] = \frac{\mathsf{MaxClique}(\mathcal{G}_V(x))}{2^r}\,.$$

In this section we essentially show an inverse of their construction.

4.1.1 The Clique-Gap Verifier

We stress that by the term *graph* we mean an undirected simple graph (i.e., no self-loops or parallel edges).

Theorem 4.1.2 (Clique verifier of ordinary graphs): There exists a verifier, denoted W, of logarithmic randomness-complexity, logarithmic query-length and zero free-bit complexity, that, on input an N-node graph G, satisfies

$$\mathtt{ACC}[\,W(G)\,] = \frac{\mathsf{MaxClique}(G)}{N}\ .$$

Furthermore, $\mathcal{G}_W(G)$ is isomorphic to G where the isomorphism is easily computable. Lastly, given a proof/oracle π we can construct in polynomial-time a clique of size pN in G, where p is the probability that W accepts G with oracle access to π.

Proof: On input a graph G on N nodes, the verifier W works with proofs of length $\binom{N}{2} - |E(G)|$. The proof π is indexed by the edges in $\overline{G}$ (i.e., non-edges in G). For clarity of the proof we assume that the binary value $\pi(\{u,v\})$ is either u or v. This is merely a matter of encoding (i.e., consider a 1-1 mapping of the standard set of binary values, $\{0,1\}$, to the set $\{u,v\}$). On input G and access to oracle π, the verifier W acts as follows:

- Picks uniformly a vertex u in the vertex set of G.
- For every $\{u,v\} \in E(\overline{G})$, the verifier W queries the oracle at $\{u,v\}$ and rejects if $\pi(\{u,v\}) \neq u$.
- If the verifier did not reject by now (i.e., all queries were answered by u), it accepts.

Properties of W. Clearly, W tosses $\log_2 N$ coins. Also, once W picks a vertex u, the only pattern it may accepts is $(u, u, \ldots, u)$. Thus the free-bit complexity of W is 0. To analyze the probability that W accepts the input G, when given the best oracle access, we first prove the following:

Claim. The graphs $\mathcal{G}_W(G)$ and G are isomorphic.

Proof. The proof is straightforward. One needs first to choose an encoding of accepting transcripts of the computation of W on input G. We choose to use the "full transcript" in which the random coins as well as the entire sequence of queries and answers is specified. Thus, a generic accepting transcript has the form

$$T_u \stackrel{\rm def}{=} (u, (\{u,v_1\}, u), ..., (\{u,v_d\}, u))$$

where u is the random vertex selected by the verifier and $\{v_1, ..., v_d\}$ the set of non-neighbors of u. We stress that T_u is the only accepting transcript in which the verifier has selected the vertex u. Also, for each vertex u, the transcript T_u is accepting. Thus, we may consider the 1-1 mapping, ϕ, that maps T_u to u. We claim that ϕ is an isomorphism between $\mathcal{G}_W(G)$ and G.

Suppose that T_u and T_v are adjacent in $\mathcal{G}_G(W)$. Then, by definition of the FGLSS graph, these transcripts are consistent. It follows that the same query can not appear in both (accepting) transcripts (otherwise it would have been given conflicting answers). By definition of W we conclude that (u,v) is not a non-edge; namely, $(\phi(T_u), \phi(T_v)) = (u,v) \in E(G)$. Suppose, on the other hand, that $(u,v) \in E(G)$. It follows that the query $\{u,v\}$ does not appear in either T_u or T_v. Since no other query may appear in both transcript, we conclude that the transcripts are consistent and thus T_u and T_v are adjacent in $\mathcal{G}_G(W)$. $\square$

By Theorem 4.1.1 it now follows that the probability that W accepts on input G, given the best oracle, is $\mathsf{MaxClique}(\mathcal{G}_W(G))/N$ which by the above equals $\mathsf{MaxClique}(G)/N$. Furthermore, given a proof π which makes W accept G with probability p, the accepting random strings of W constitute a clique of size pN in $\mathcal{G}_W(G)$. These accepting random strings can be found in polynomial-time and they encode vertices of G (which form a clique in G). ▮

We now generalize the above construction to get verifiers which indicate the existence of large cliques in layered graphs. An (L, M, N)-layered graph is an N-vertex graph in which the vertices are arranged in L layers so that there are no edges between vertices in the same layer and there are at most M vertices in each layer. We use a convention by which, whenever a layered graph is given to some algorithm, a partition into layers is given along with it (i.e., is implicit in the encoding of the graph).

Theorem 4.1.3 (Clique verifier for layered graphs): There exists a verifier, denoted W, of logarithmic randomness-complexity and logarithmic query-length that, on input an (L, M, N)-layered graph G has free-bit complexity $\log_2 M$, average free-bit complexity $\log_2(N/L)$ and satisfies

$$\mathtt{ACC}\,[\,W(G)\,] = \mathsf{MaxClique}(G)/L\;.$$

Furthermore, $\mathcal{G}_W(G)$ is isomorphic to G where the isomorphism is easily computable. Lastly, given a proof/oracle π we can construct in polynomial-time a clique of size pL in G, where p is the probability that W accepts G with oracle access to π.

Proof: On input a (L, M, N)-layered graph G, the verifier W works with proofs consisting of two parts. The first part assigns every layer (i.e., every integer $i \in [L]$) a vertex in the layer (i.e., again we use a redundant encoding by which the answers are vertex names rather then an index between 1 and the number of vertices in the layer). The second part assigns pairs of non-adjacent (in G) vertices, a binary value, which again is represented as one of the two vertices. On input G and access to oracle π, the verifier W acts as follows:

- Picks uniformly a layer i in $\{1, ..., L\}$.
- Queries π at i obtaining as answer a vertex u. If u is not in the i^{th} layer of G then the verifier rejects. (Otherwise, it continues as follows.)
- For every $\{u, v\} \in E(\overline{G})$, the verifier W queries the oracle at $\{u, v\}$ and rejects if $\pi(\{u, v\}) \neq u$. (Actually, it is not needed to query the oracle on pairs of vertices belonging to the same layer.)
- If the verifier did not reject by now (i.e., all queries were answered by u), it accepts.

Properties of W. Here W tosses $\log_2 L$ coins. Once the first query of W is answered, specifying a vertex u, the only pattern it may accept in the remaining queries is $(u, u, \ldots, u)$. Thus, the free-bit complexity of W is $\log_2 M$, accounting for the first query which may be answered arbitrarily in $\{1, ..., m\}$, where $m \leq M$ is the number of vertices in the chosen layer. The average free-bit complexity is $\log_2(N/L)$ (as N/L is the average number of vertices in a layer of the graph G). Again, we can prove that $\mathcal{G}_W(G) = G$ and the theorem follows.

Proof. Here, the accepting transcripts of W, on input G, correspond to a choice of a layer, i, and a vertex in the i^{th} layer (since once a vertex is specified by the first answer there is only one accepting way to answer the other queries). Thus, a generic accepting transcript has the form

$$T_u \stackrel{\text{def}}{=} (i, (i, u), (\{u, v_1\}, u), ..., (\{u, v_d\}, u))$$

where i is the layer selected by the verifier, u is a vertex in the i^{th} layer of G and $\{v_1, ..., v_d\}$ the set of non-neighbors of u. Again, T_u is the only accepting transcript in which the verifier has selected the vertex u, and for each vertex u, the transcript T_u is accepting. Again, we consider the 1-1 mapping, ϕ, that maps T_u to u, and show that it is an isomorphism between $\mathcal{G}_W(G)$ and G.

Suppose that T_u and T_v are adjacent in $\mathcal{G}_G(W)$. Then, by definition of the FGLSS graph, these transcripts are consistent. We first note that u and v cannot appear in the same layer of G (otherwise the first query in the transcript would yield conflicting answers). Again, the same two-vertex query can not appear in both (accepting) transcripts, and we conclude that $(\phi(T_u), \phi(T_v)) = (u, v) \in E(G)$. Suppose, on the other hand, that $(u, v) \in E(G)$. Clearly, u and v belong to different layers and as before the query (u, v) does not appear in either T_u or T_v. Since no other two-vertex query may appear in both transcripts, we conclude that the transcripts are consistent and thus T_u and T_v are adjacent in $\mathcal{G}_G(W)$. □

The theorem follows as before. ▮

Remark. The clique verifier W is adaptive: the answer to its first query determines (all) the other queries. We wonder if it is possible to construct a non-adaptive clique verifier with properties as claimed in the theorem.

4.1.2 Main Consequences

We are interested in problems exhibiting a gap in Max-Clique size between positive and negative instances. Recall that $\overline{\mathsf{MaxClique}}(G) = \mathsf{MaxClique}(G)/N$ is the fraction of nodes in a maximum clique of N-node graph G. Also recall the Gap-$\text{Clique}_{c,s}$ promise problem:

Definition 4.1.4 For any $0 \le s(\cdot) \le c(\cdot) \le 1$ we let the promise problem Gap-$\text{Clique}_{c,s}$ be the pair (A, B), where–

(1) A is the set of all graphs G with $\overline{\mathsf{MaxClique}}(G) \ge c(N)$, and

(2) B is the set of all graphs G with $\overline{\mathsf{MaxClique}}(G) \le s(N)$.

The *gap* of this problem is defined to be c/s.

As a direct consequence of Theorem 4.1.2, we get

Corollary 4.1.5 For all functions $c, s \colon \mathcal{Z}^+ \to [0, 1]$ we have Gap-$\text{Clique}_{c,s} \in \text{FPCP}_{c,s}[\log, 0, \text{poly}]$.

The above corollary transforms the gap in the promise problem into a gap in a pcp system. However, the accepting probabilities in this pcp system are very low (also on yes-instances). Below, we use Theorem 4.1.3 to obtain pcp systems with perfect (resp., almost-perfect) completeness for this promise problem. We start by presenting two randomized reductions of the promise problem to a layer version. Alternative methods are presented in Section 5.2 (cf., Theorem 5.2.6).

Proposition 4.1.6 (Layering the clique promise problem):

(1) (Obtaining a perfect layering): There exists a polynomial-time randomized transformation, T, of graphs into layered graphs so that, on input a graph G, integers C and L, outputs a subgraph $H = T(G, C, L)$ of G in L layers such that if $\mathsf{MaxClique}(G) \ge C$ then

$$\Pr[\,\mathsf{MaxClique}(H) < L\,] < L \cdot 2^{-\frac{C}{2L}}$$

Furthermore, with probability $1 - L \cdot 2^{-N/3L}$, no layer of H contains more than $2 \cdot \frac{N}{L}$ nodes.

(2) (Using logarithmic randomness): There exists a polynomial-time randomized transformation, T, of graphs into layered graphs so that, on input a graph G, integers C and L, outputs a subgraph $H = T(G, C, L)$ of G in L layers such that if $\mathsf{MaxClique}(G) \ge C$ then

$$\Pr[\,\mathsf{MaxClique}(H) \le (1 - \epsilon) \cdot L\,] < \frac{L}{\epsilon C}$$

for every $\epsilon \in [0, 1]$. Furthermore, the transformation uses logarithmically many coins. Also, with probability $1 - \frac{L}{\epsilon N}$, at most ϵL layers of H contains more than $2 \cdot \frac{N}{L}$ nodes.

Proof: The *first transformation* consists of assigning to each vertex of G a randomly chosen layer of H. Namely, we construct the graph H which is a subgraph of G by uniformly selecting for each vertex v a layer $l(v) \in [L]$ and copying only the edges of G which connect vertices placed in different layers (of H). The construction can be carried out in random polynomial-time and we show that if the original graph has a clique of size C then with high probability the resulting graph has a clique of size L, provided $L \ll C/2\log_2 L$.

Claim 1. Suppose that G has a clique of size C denoted S. Then, the probability that all vertices in S were placed in less than L layers is at most $L \cdot 2^{-\frac{C}{2L}}$.

Proof. We start by bounding, for each i, the probability that no vertex of S is placed in the i^{th} layer. For each $v \in S$, we introduce the 0-1 random variable ζ_v so that $\zeta_v = 1$ if v is placed in the i^{th} layer (i.e., $l(v) = i$) and $\zeta_v = 0$ otherwise. Let $t \stackrel{\text{def}}{=} C/L$. Then, $\mathbf{E}[\sum_{v\in S} \zeta_v] = t$. Using a multiplicative Chernoff bound [MoRa], we get

$$\begin{aligned}
\Pr[\,\forall v \in S : l(v) \neq i\,] &= \Pr\left[\sum_{v\in S} \zeta_v = 0\right] \\
&< 2^{-\frac{t}{2}}
\end{aligned}$$

Call the i^{th} layer *bad* if no vertex of S is placed in it. By the above, the probability that there exists a bad layer is smaller than $L \cdot 2^{-t/2}$, and the claim follows. □

It is left to bound the probability that a particular layer contains more than twice the expected number of vertices. Using again a multiplicative Chernoff bound, this probability is at most $2^{-N/3L}$ and the first part of the proposition follows.

The *second transformation* consists of selecting randomly a Universal$_2$ Hashing function (a.k.a., pairwise independent hash function) mapping the vertices of the graph G into the layer-set $[L]$. Namely, suppose that the function h was chosen, then we construct the graph H which is a subgraph of G by placing a vertex v (of G) in layer $h(v)$ of H, and copying only the edges of G which connect vertices placed in different layers (of H). The construction can be carried out in polynomial-time using only logarithmic randomness (for the selection of the hashing function). We show that if the original graph has a clique of size C then with high probability the resulting graph has a clique of size almost L, provided $L \ll C$.

Claim 2. Suppose that G has a clique of size C denoted S. Then, the probability that all vertices in S were placed in less than $(1-\epsilon) \cdot L$ layers is at most $\frac{L}{\epsilon C}$.

Proof. Again, we bound, for each i, the probability that no vertex of S is placed in the i^{th} layer. For each $v \in S$, we introduce the 0-1 random variable ζ_v so that $\zeta_v = 1$ if $h(v) = i$ and $\zeta_v = 0$ otherwise. Let $t \stackrel{\text{def}}{=} C/L$ and $\zeta \stackrel{\text{def}}{=} \sum_{v\in S} \zeta_v$. Then, $\mathbf{E}[\zeta] = t$ (which is greater than 1, otherwise the claim holds vacuously). Using the pairwise independence of h and Chebyshev's inequality, we get

$$\begin{aligned}
\Pr[\,\forall v \in S : h(v) \neq i\,] &= \Pr[\,\zeta = 0\,] \\
&\leq \frac{\mathbf{Var}[\sum_{v\in S} \zeta_v]}{t^2} \\
&< \frac{C/L}{t^2} = \frac{1}{t}
\end{aligned}$$

Call the i^{th} layer *bad* if no vertex of S is placed in it. By the above, the expected number of bad layers is smaller than $L \cdot \frac{1}{t}$, so by Markov inequality the probability that more than ϵL layers are bad is at most $1/\epsilon t$. The claim follows. □

Again, it is left to bound the probability that a particular layer contains more than $M \stackrel{\text{def}}{=} 2N/L$. Using Chebyshev's inequality again, this probability is at most L/N. Thus, the expected number of layers having more than M vertices is at most L^2/N and it follows that the probability that ϵL layers contain more than M vertices each is at most $\frac{L^2/N}{\epsilon L} = \frac{L}{\epsilon N}$. The second part of the proposition follows. ∎

Combining Theorem 4.1.3 and Proposition 4.1.6, we obtain

Proposition 4.1.7 For any polynomial-time computable functions c, s, ϵ: $\mathcal{Z}^+ \to [0,1]$ we have
(1) (Randomized reduction to a pcp with perfect completeness):

$$\textsf{Gap}\text{-Clique}_{c,s} \leq_R^K \text{FPCP}_{1,s'}[\log, f']$$

where $f'(N) \stackrel{\text{def}}{=} \log_2(1/c(N)) + \log_2\log_2 N + 2$ and $s'(N) \stackrel{\text{def}}{=} 2\log_2 N \cdot \frac{s(N)}{c(N)}$.
(2) (A pcp with almost-perfect completeness):

$$\textsf{Gap}\text{-Clique}_{c,s} \in \text{FPCP}_{1-4\epsilon,s'}[\log, f']$$

where $f'(N) \stackrel{\text{def}}{=} 1 + \log_2(1/c(N)) + 2\log_2(1/\epsilon(N))$ and $s'(N) \stackrel{\text{def}}{=} \frac{1}{\epsilon(N)^2} \cdot \frac{s(N)}{c(N)}$.

Proof: For the *second part*, we construct a verifier for the promise problem proceeds as follows. On input an N-vertex graph G, the verifier computes $C \stackrel{\text{def}}{=} N \cdot c(N)$, $\epsilon \stackrel{\text{def}}{=} \epsilon(N)$ and $L \stackrel{\text{def}}{=} \epsilon^2 C$. It invokes the second transformation of Proposition 4.1.6, obtaining a (L, N, N)-layered graph $H = T(G, C, L)$. (We stress that this transformation requires only logarithmically many coin tosses.) Next, the verifier modifies H into H' by omitting (the minimum number of) vertices so that no layer of H' has more than $2N/L$ vertices. Finally, the verifier invokes the clique-verifier W of Theorem 4.1.3 on input H'.

The free-bit complexity of the verifier constructed above is $\log_2(2N/L) = 1 + \log_2(1/c(N)) + 2\log_2(1/\epsilon(N))$. Suppose that G is a no-instance of the promise problem. Using $\textsf{MaxClique}(H') \leq \textsf{MaxClique}(G)$ and Theorem 4.1.3, it follows that the constructed verifier accepts G with probability at most $\frac{\textsf{MaxClique}(H')}{L} \leq \frac{s(N)}{\epsilon^2(N)\cdot c(N)}$. Suppose, on the other hand, that G is a yes-instance of the promise problem. Then, with probability at least $1 - \frac{L}{\epsilon C} = 1 - \epsilon$ we have $\textsf{MaxClique}(H) \geq (1-\epsilon)\cdot L$, and with probability at least $1 - \frac{L}{\epsilon N} > 1 - \epsilon$ we have $\textsf{MaxClique}(H') \geq \textsf{MaxClique}(H) - \epsilon L$. Thus, with probability at least $1 - 2\epsilon$, we have $\textsf{MaxClique}(H') \geq (1-2\epsilon)\cdot L$. It follows that the constructed verifier, when given oracle access to an appropriate proof, accepts G with probability at least $1 - 4\epsilon$.

For the *first part*, we define a promise problem which refers to gaps in cliques of layered graphs. Specifically,

Definition. For any function $\ell : \mathcal{Z}^+ \to \mathcal{Z}^+$ and $s : \mathcal{Z}^+ \to [0,1]$, we define the promise problem $\textsf{Gap}-\text{LG}_{\ell,s}$ be the pair (A, B), where–

(1) A is the set of all $(\ell(N), \frac{2N}{\ell(N)}, N)$-layered graphs G with $\textsf{MaxClique}(G) = \ell(N)$, and
(2) B is the set of all $(\ell(N), \frac{2N}{\ell(N)}, N)$-layered graphs G with $\textsf{MaxClique}(G) \leq s(N)\cdot\ell(N)$.

The *gap* of this problem is defined to be $1/s$.

Using the first transformation of Proposition 4.1.6, we obtain $\textsf{Gap}\text{-Clique}_{c,s} \leq_R^K \textsf{Gap}-\text{LG}_{\ell,s'}$, where $\ell(N) = \frac{c(N)\cdot N}{2\log_2 N}$ and $s'(N) = \frac{s(N)\cdot N}{\ell(N)} = 2\log_2 N \cdot \frac{s(N)}{c(n)}$. On the other hand, Theorem 4.1.3 asserts that $\mathsf{Gap}\text{-}\mathrm{LG}_{\ell,s'} \in \mathrm{FPCP}_{1,s'}[\log, f']$, where $f'(N) \stackrel{\mathrm{def}}{=} \log_2(2N/\ell(N))$. Observing that $f'(N) = 1 + \log_2 \frac{2\log_2 N}{c(N)}$ (which equals $\log_2(1/c(N)) + \log_2 \log_2 N + 2$), the proposition follows. ∎

Each of the two parts of Proposition 4.1.7 shows that the well-known method of obtaining clique-approximation results from efficient pcp systems (cf., [FGLSS, BeSc, Zu, FeKi, BeSu]) is "complete" in the sense that if clique-approximation can be shown NP-hard then this can be done via this method. The following is a more precise version of Theorem 1.4.1 in that the role of $\epsilon > 0$ is made explicit. The restriction that f be a constant is only for notational simplicity. (The issue is that f in one case must be measured as a function of $n = |x|$ and in the other case as a function of $N = \|G\|$.)

Theorem 4.1.8 Let f be a constant. Then the following statements are equivalent:
(1) For all $\epsilon > 0$ it is the case that NP reduces to $\mathsf{Gap}\text{-Clique}_{c,s}$ with gap $c(N)/s(N) = N^{1/(1+f+\epsilon)}$.
(2) For all $\epsilon > 0$ it is the case that NP reduces to $\overline{\mathrm{FPCP}}[\log, f+\epsilon]$.
In both items the reduction is randomized. Furthermore the equivalence holds both for Karp and for Cook reductions.

Proof: The direction (2) $\Rightarrow$ (1) follows by first amplifying the gap of the verifier for NP (cf., Corollary 5.2.3) and then by applying the FGLSS-reduction [FGLSS] to the amplified gap verifier. Specifically, we first obtain $\mathrm{NP} \leq_R \mathrm{FPCP}_{1,2^{-t}}[(1+\epsilon)\cdot t, f\cdot t]$, where $t(n) = \gamma \log_2 n$ (with the constant γ determined by the constant $\epsilon > 0$). The FGLSS-reduction now yields a graph of size $N \stackrel{\mathrm{def}}{=} 2^{(1+\epsilon+f)\cdot t(n)}$ with gap $2^{-t(n)}$ (which can be written as $N^{\frac{1}{1+\epsilon+f}}$).

For the reverse direction, we will use the first part of Proposition 4.1.7 and show that the resulting verifier has a small amortized free bit complexity. Let $\mathsf{Gap}\text{-Clique}_{c,s}$ be NP-hard for some functions $c(N)$ and $s(N)$ satisfying $s(N) \geq 1/N$ and $c(N)/s(N) \geq N^{\frac{1}{1+f+\epsilon}}$. Thus, $c(N) \geq N^{\frac{1}{1+f+\epsilon}}/N$ and $1/c(N) \leq N^{\frac{f+\epsilon}{1+f+\epsilon}}$.

Let $\alpha(N) \stackrel{\mathrm{def}}{=} 2\log_2 N$, $f'(N) \stackrel{\mathrm{def}}{=} \log_2(1/c(N)) + \log_2 \alpha(N)$ and let $s'(N) \stackrel{\mathrm{def}}{=} \alpha(N)\cdot\frac{s(N)}{c(N)}$. By invoking Proposition 4.1.7 (Part 1) we find that $\mathsf{Gap}\text{-Clique}_{c,s} \leq_R \mathrm{FPCP}_{1,s'}[\log, f']$ and $\mathsf{Gap}\text{-Clique}_{c,s} \leq_R \overline{\mathrm{FPCP}}[\log, \overline{f'}]$, for $\overline{f'} = \frac{f'}{\log(1/s')}$, follows. It now remains to argue that for any $\delta > 0$, $\overline{f'} \leq f + \epsilon + \delta$.

Using the lower bounds on $c(N)$ and $c(N)/s(N)$, we obtain $f'(N) \leq \frac{f+\epsilon}{1+f+\epsilon}\log_2 N + \log_2 \alpha(N)$ and $\log(1/s'(N)) \geq \frac{1}{1+f+\epsilon}\cdot\log_2 N - \log_2\alpha(N)$. Selecting a sufficiently small $\delta' > 0$ and using $\log_2 \alpha(N) < \delta' \cdot \log_2 N$, we get

$$\begin{aligned}
\overline{f'} &\leq \frac{\frac{f+\epsilon}{1+f+\epsilon}\log N + \log_2 \alpha(N)}{\frac{1}{1+f+\epsilon}\log N - \log_2\alpha(N)} \\
&< \frac{\frac{f+\epsilon}{1+f+\epsilon} + \delta'}{\frac{1}{1+f+\epsilon} - \delta'} \\
&< f + \epsilon + \frac{\delta'}{1-\delta'}\cdot(1 + 2(f+\epsilon))
\end{aligned}$$

and the theorem follows. ∎

An alternative statement is provided by the following theorem. Here the second item (existence of pcp systems with certain parameters) is weaker than in the previous theorem, but this allows the (1) $\Rightarrow$ (2) direction to be proven via a deterministic reduction (instead of the randomized reduction used in the analogous proof above). Interestingly, the FGLSS-reduction used to establish the other direction is insensitive to the gap location and in particular to the fact that we no longer use proof systems of perfect completeness. Recall that $\overline{\mathrm{FPCP}}_{1-o(1)}[\,\cdot,\, f\,]$ is the class of problems having a proof system with almost-perfect completeness (i.e., $c = 1 - o(1)$) and amortized free-bit complexity f.

Theorem 4.1.9 Let f be a constant. Then the following statements are equivalent:

(1) For all $\epsilon > 0$ it is the case that NP reduces to Gap-Clique$_{c,s}$ with gap $c(N)/s(N) = N^{1/(1+f+\epsilon)}$.

(2) For all $\epsilon > 0$ it is the case that NP reduces to $\overline{\mathrm{FPCP}}_{1-o(1)}[\log,\, f+\epsilon\,]$.

In both items the reduction is randomized and the equivalence holds both for Karp and for Cook reductions. Furthermore, if item (1) holds with respect to deterministic reductions so does item (2). It follows that in case item (1) holds with a deterministic Karp reduction then NP $\subseteq \overline{\mathrm{FPCP}}_{1-o(1)}[\log,\, f+\epsilon\,]$.

Proof: The direction (2) $\Rightarrow$ (1) follows essentially as in the proof of the previous theorem. Specifically, item (2) asserts that, for some function m, NP $\leq_R$ $\mathrm{FPCP}_{c,2^{-m}\cdot c}[\log, m\cdot f]$, for $c(n) = 1 - o(1)$ (but we are not going to use the bound on c). Using Proposition 5.2.1 and Proposition 5.2.2 (Part 2), we first obtain NP $\leq_R$ $\mathrm{FPCP}_{c',2^{-t}\cdot c'}[(1+\epsilon)\cdot t, f\cdot t]$, where $c'(n) = c(n)^{t(n)/m(n)}$ and $t(n) = \gamma \log_2 n$ (with the constant γ determined by the constant $\epsilon > 0$). The FGLSS-reduction now yields a graph of size $N \stackrel{\mathrm{def}}{=} 2^{(1+\epsilon+f)\cdot t(n)}$ with gap $2^{-t(n)}$ as in the analogous proof above. (The gap is in a different location but this does not matter.)

For the reverse direction, we will use the second part of Proposition 4.1.7 and show that the resulting verifier has a small amortized free bit complexity. Let Gap-Clique$_{c,s}$ be NP-hard for some functions $c(N)$ and $s(N)$ satisfying $s(N) \geq 1/N$ and $c(N)/s(N) \geq N^{\frac{1}{1+f+\epsilon}}$. As in the analogous proof above, this implies that $1/c(N) \leq N^{\frac{f+\epsilon}{1+f+\epsilon}}$.

Let α be a slowly decreasing function s.t. $\alpha(N) = o(1)$ but $\log_2(1/\alpha(N)) = o(\log N)$. Let $f'(N) \stackrel{\mathrm{def}}{=} \log_2(1/c(N)) + 2\log_2(1/\alpha(N))$ and let $s'(N) \stackrel{\mathrm{def}}{=} \frac{1}{\alpha(N)^2} \cdot \frac{s(N)}{c(N)}$. By invoking Proposition 4.1.7 (Part 2) we get Gap-Clique$_{c,s} \in \mathrm{FPCP}_{1-\alpha,s'}[\log, f']$. Since $\alpha(N) = o(1)$, we conclude that Gap-Clique$_{c,s} \in \overline{\mathrm{FPCP}}_{1-o(1)}[\log,\, \overline{f'}\,]$ for $\overline{f'} = \frac{f'}{\log_2(1/s')}$. It now remains to argue that for any $\delta > 0$, $\overline{f'} \leq f + \epsilon + \delta$.

We use the lower bound on $c(N)$ and $c(N)/s(N)$, we obtain $f'(N) \leq \frac{f+\epsilon}{1+f+\epsilon}\log_2 N - 2\log_2 \alpha(N)$ and $\log_2(1/s'(N)) = 2\log_2 \alpha(N) + \frac{1}{1+f\epsilon}\log_2 N$. Selecting a sufficiently small $\delta' > 0$ and using $\log_2(1/\alpha(N)) < \delta' \cdot \log_2 N$, we get

$$
\begin{aligned}
\overline{f'} &\leq \frac{\frac{f+\epsilon}{1+f+\epsilon}\log_2 N + 2\log_2(1/\alpha(N))}{\frac{1}{1+f+\epsilon}\log_2 N - 2\log_2(1/\alpha(N))} \\
&< \frac{\frac{f+\epsilon}{1+f+\epsilon} + \delta'}{\frac{1}{1+f+\epsilon} - \delta'} \\
&< f + \epsilon + \frac{\delta'}{1-\delta'} \cdot (1 + 2(f+\epsilon))
\end{aligned}
$$

and the theorem follows. ∎

4.1.3 More Consequences

The equivalence between clique and fpcp described above turns out be a useful tool in the study of the hardness of the clique and chromatic number problems. Here we describe some applications. The first application is a non-technical one which simply allows us to rephrase the many known reductions from the Max Clique problem to the Chromatic number problem in a simpler and more convenient way. The remaining applications use the fact that the equivalence between fpcp and Max Clique allows us to easily shift gaps, in the Max Clique problem, from one place to another. Loosely speaking, these applications use the fact that the complexity of the promise problem $\textsf{Gap}\text{-Clique}_{c,s}$ remains unchanged when changing the parameters c and s so the $\frac{\log_2 c(N)}{\log_2 s(N)}$ remains invariant. We stress that the ratio $\frac{c(N)}{s(N)}$ does not remain invariant.

Rephrasing known reductions from Max Clique to Chromatic Number Starting with the work of Lund and Yannakakis [LuYa], there have been several works on showing the hardness of approximating the Chromatic number, which reduce the Max Clique problem to the Chromatic number problem. Yet none of these results could be stated cleanly in terms of a reduction from Max Clique to Chromatic Number without loss of efficiency - i.e., the theorems could not be stated as saying "If approximating Max Clique to within a factor of N^α is NP-hard, then approximating Chromatic Number to within a factor of $N^{h(\alpha)}$ is NP-hard." The reason for the lack of such a statement is that these reductions use the structure of the graph produced by applying an FGLSS-reduction to a FPCP result, and are hence really reductions from FPCP to Chromatic Number rather than reductions from Max Clique to Chromatic Number. However now we know that FPCP and Max Clique are equivalent, so we can go back and rephrase the old statements. Thus results of [LuYa, KLS, BeSu] can be summarized as:

> For every $\gamma > 0$, if approximating Max Clique to within N^α is NP-hard then approximating Chromatic Number to within $N^{h(\alpha)-\gamma}$ is also NP-hard, where:
>
> **(1)** $h(\alpha) = \min\{\frac{1}{6}, \frac{\alpha}{5-4\alpha}\}$ [LuYa].
> **(2)** $h(\alpha) = \min\{\frac{1}{11}, \frac{\alpha}{5+\alpha}\}$ [KLS].
> **(3)** $h(\alpha) = \min\{\frac{1}{4}, \frac{\alpha}{3-2\alpha}\}$ [BeSu].
> **(4)** $h(\alpha) = \min\{\frac{1}{3}, \frac{\alpha}{2-\alpha}\}$ [Fu].

(Our discussion of Furer's results [Fu] reflects only the best current understanding we have of them, since it is on-going work.) We note that it is an open problem whether one can get a reduction in which $h(\alpha) \to 1$ as $\alpha \to 1$. We also note that Furer's reduction is randomized while the rest are deterministic.

Reductions among Max Clique Problems Next we present an invariance of the Gap Clique problem with respect to shifting of the gaps. The following result has also been independently observed by Feige [Fei], where he uses a randomized graph product to show the result. Our description uses the properties of fpcp and its equivalence to clique approximation.

Theorem 4.1.10 Let $k, \epsilon_1, \epsilon_2$ be real numbers such that $k \geq 1$ and $0 \leq \epsilon_1 < \epsilon_2 \leq 1$. Then the following hold:

(1) $\textsf{Gap}\text{-Clique}_{N^{-\epsilon_2}, N^{-k\epsilon_2}} \leq_D^K \textsf{Gap}\text{-Clique}_{N^{-\epsilon_1}, N^{-k\epsilon_1}}$. (Deterministic reduction.)

(2) $\textsf{Gap}\text{-Clique}_{N^{-\epsilon_1}, N^{-k\epsilon_1}} \leq_R^K \textsf{Gap}\text{-Clique}_{\frac{1}{2}\cdot N^{-\epsilon_2}, 2\cdot N^{-k\epsilon_2}}$.

Proof: Part (1) is proved via a well-known graph theoretic trick. Let G be an instance of Gap-Clique$_{N^{-\epsilon_2},N^{-k\epsilon_2}}$ with N nodes. We take the graph-product of G with a complete graph on m nodes, to get a graph H on $M = mN$ nodes. (By a graph-product of two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ we mean a graph with vertex set $V_1 \times V_2$ where vertices (u_1, u_2) and (v_1, v_2) are connected iff $(u_i, v_i) \in E_i$ for both $i = 1, 2$.) We choose m so that if G has a clique of size $N^{1-\epsilon_2}$, then H has a clique of size $M^{1-\epsilon_1}$. Specifically, setting $m = N^{\frac{\epsilon_2-\epsilon_1}{\epsilon_1}}$, the requirement is satisfied (as a clique of size $N^{1-\epsilon_2}$ in G yields a clique of size $m \cdot N^{1-\epsilon_2} = N^{\frac{\epsilon_2-\epsilon_1}{\epsilon_1}+1-\epsilon_2} = M^{\frac{\epsilon_1}{\epsilon_2}\cdot\frac{\epsilon_2(1-\epsilon_1)}{\epsilon_1}}$ in H.) Under this choice of m we will show that if G has no cliques of size $N^{1-k\epsilon_2}$ then H has no cliques of size $M^{1-k\epsilon_1}$. This will complete the proof of part (1).

Suppose H has a clique of size $M^{1-\epsilon_1}$. Then, by construction, G must have a clique of size

$$\begin{aligned} \frac{M^{1-\epsilon_1}}{m} &= \frac{N^{1-\epsilon_1}}{m^{\epsilon_1}} \\ &= N^{1-\epsilon_1-\frac{\epsilon_2-\epsilon_1}{\epsilon_1}\cdot\epsilon_1} \end{aligned}$$

and the claim follows.

For part (2) we use the equivalence between FPCP and gaps in MaxClique and apply amplification properties of FPCP. Let $c(N) = N^{-\epsilon_1}$ and $s(N) = N^{-k\epsilon_1}$. Then, using Corollary 4.1.5 (for line 1), Proposition 5.2.1 (for line 2) and Part (2) of Proposition 5.2.2 (for line 3), we get

$$\begin{aligned} \text{Gap-Clique}_{N^{-\epsilon_1},N^{-k\epsilon_1}} &\in \mathrm{FPCP}_{c,s}[\log_2 N, 0, N^2] \\ &\subseteq \mathrm{FPCP}_{c^t,s^t}[t\cdot\log_2 N, 0, N^2] \quad \text{(for any integer constant } t \geq 1.) \\ &\leq_R^K \mathrm{FPCP}_{\frac{1}{2}\cdot c^t,2\cdot s^t}[\log_2(N^2/s^t), 0, N^2] \end{aligned}$$

The choice of the integer t will be determined later.

Now, we go back to the clique-gap promised problem. Applying the FGLSS-reduction to the pcp class $\mathrm{FPCP}_{\frac{1}{2}\cdot c^t,2\cdot s^t}[\log_2(N^2/s^t), 0, N^2]$ we obtain an instance of Gap-Clique$_{\frac{1}{2}N^{-\epsilon_1 t},2N^{-k\epsilon_1 t}}$ on an M-vertex graph, where $M = \frac{N^2}{s^t} = N^{2+k\epsilon_1 t}$. To clarify the last assertion and the rest of the proof, we introduce the notation Gap-Clique$_{\alpha(N),\beta(N)}(N)$ which makes explicit the size parameter to which the promise problem refers. Thus, letting $\gamma \stackrel{\text{def}}{=} \frac{t}{2+tk\epsilon_1}$, we have obtained

$$\text{Gap-Clique}_{N^{-\epsilon_1},N^{-k\epsilon_1}}(N) \leq_R^K \text{Gap-Clique}_{\frac{1}{2}M^{-\gamma\epsilon_1},2M^{-k\cdot\gamma\epsilon_1}}(M)$$

(with M polynomial in N). Now, part (2) follows by setting t so that $\gamma = \frac{t}{2+tk\epsilon_1} \geq \frac{\epsilon_2}{\epsilon_1}$ and $t = \lceil \frac{2\epsilon_2}{(1-k\epsilon_2)\epsilon_1} \rceil$ will do. (Actually, we get Gap-Clique$_{N^{-\epsilon_1},N^{-k\epsilon_1}}(N) \leq_R^K$ Gap-Clique$_{\frac{1}{2}M^{-\epsilon_2'},2M^{-k\epsilon_2'}}(M)$, for $\epsilon_2' \geq \epsilon_2$, but this can be corrected by invoking item (1).) ▮

The following theorem, was first shown by Blum [Bl], using the technique of randomized graph products. It essentially uses the gap-shifting idea to show that a seemingly very weak approximator to the clique (say, $N^{1-\epsilon}$-approximation algorithm for some $\epsilon > 0$), can be used to obtain a very good approximator to the clique number in graphs which are guaranteed to have very large cliques. In particular, using such an algorithm, if a graph has a clique of size $\frac{N}{k}$, then a clique of size $\frac{N}{k^{\frac{1}{\epsilon}}}$ can be found in such a graph in polynomial time. As observed by Blum, this can be translated into significantly better algorithms for approximate coloring of a three colorable graph than known currently (see Item (1) in Corollary 4.1.12 below). Here we derive the theorem using FPCP and the gap-shifting techniques. The parameters are generalized so as to be able to conclude, say, that even if we have a $\frac{N}{2^{\sqrt{\log_2 N}}}$-approximation (for Max Clique), then we can obtain non-trivially good algorithms for 3-coloring (see Item (2) in Corollary 4.1.12).

Theorem 4.1.11 Let $\alpha \in [0,1]$, $\beta \in [0,1/2)$ and $k > 1$. Define $\epsilon : \mathcal{Z}^+ \to \mathcal{R}^+$, $c \in \mathcal{R}^+$ and $g : \mathcal{Z}^+ \to \mathcal{R}^+$ so that

$$\begin{aligned} \epsilon(N) &= \frac{\alpha}{\log_2^\beta N} \\ c &= \frac{2}{\log_2 k} \\ \text{and } \log_2 g(N) &= \left(\frac{c^\beta \log_2 k}{\alpha}\right)^{1/(1-\beta)} \log_2^{\beta/(1-\beta)} N. \end{aligned}$$

Then there is a randomized poly$(N^{2+c\log_2 g(N)})$-time reduction of instances of Gap-Clique$_{1/k,1/g}$ to M-vertex instances of Gap-Clique$_{\frac{1}{2}M^{-\epsilon(M)},2M^{-1+\epsilon(M)}}$.
Remark: Observe that $g(N) = N^{o(1)}$. Also, for $\beta = 0$ we have $\epsilon(N) = \alpha$ and $g(N) = k^{\frac{1}{\alpha}}$. Thus, the theorem states that given a $\frac{1}{4}M^{1-2\alpha}$ approximator for clique one can one can solve Gap-Clique$_{1/k,1/k'}$ in polynomial-time, where $k' = k^{1/\alpha}$.

Proof: As usual we first reduce Gap-Clique to FPCP and then amplify.

$$\begin{aligned} \mathsf{Gap}\text{-Clique}_{1/k,1/g} &\in \mathrm{FPCP}_{1/k,1/g}[\log_2 N, 0, N^2] \\ &\subseteq \mathrm{FPCP}_{(1/k)^t,(1/g)^t}[t\log_2 N, 0, N^2] \text{ (for any function } t : \mathcal{Z}^+ \to \mathcal{Z}^+.) \\ &\leq_R^K \mathrm{FPCP}_{\frac{1}{2}(1/k)^t,2(1/g)^t}[\log_2 N^2 g^t, 0, N^2] \end{aligned}$$

We now show that by setting $t = c\log_2 N$ and using the FGLSS-reduction, the above reduces in poly(M)-time to Gap-Clique$_{\frac{1}{2}M^{-\epsilon},2M^{-\epsilon+1}}$ in an M vertex graph, where $M = N^2 g(N)^t$.

In case the graph is a no-instance the size of the clique is most $2(1/g(N))^t \cdot M = 2N^2$. In the case the graph is a yes-instance then the clique size is at least $\frac{1}{2}(1/k)^t \cdot M$. Thus it suffices to show that $2N^2 \leq 2M^{\epsilon(M)}$ and $2k^t \leq 2M^{\epsilon(M)}$, respectively. Taking logs in both cases it suffices to show that

$$2\log_2 N \leq \epsilon(M)\log_2 M \tag{4.1}$$

$$t\log_2 k \leq \epsilon(M)\log_2 M \tag{4.2}$$

We first lower bound the right hand side of both equations.

$$\begin{aligned} \epsilon(M)\log_2 M &= \alpha\log_2^{1-\beta} M \\ &\geq \alpha\log_2^{1-\beta}(g(N)^t) \\ &\geq \alpha t^{1-\beta}\log_2^{1-\beta} g(N) \\ &= \alpha\cdot(c\log_2 N)^{1-\beta}\cdot\left(c^\beta\frac{\log_2 k}{\alpha}\log_2^\beta N\right) \\ &= c\log_2 N\log_2 k \end{aligned}$$

Inequality (4.1) now follows from the fact that and $c\log_2 k = 2$. Inequality (4.2) follows from the fact that $t = c\log_2 N$. ∎

The following result was derived as a corollary by Blum [Bl] and shows the application of the above theorem to coloring graphs with low-chromatic number with relatively small number of colors. We warn the reader that the corollary does not follow directly from the above theorem; this is because it uses a Levin-reduction[1] from the search version of chromatic number to the search version of the clique problem. However, it is possible to define search versions of all the gap problems above appropriately and verify that all the reductions work for the search problems as well (i.e., they are in fact Levin-reductions). Thus the following can be derived as a corollary to the above.

Corollary 4.1.12 Let $k < \infty$.

(1) For $\epsilon > 0$, given an $N^{1-\epsilon}$ approximator to the clique, one can color any k-colorable graph on M nodes with $O(k^{1/\epsilon} \log M)$ colors in polynomial time.

(2) For $\epsilon(N) = \omega((\log N)^{-1/2})$, given an $N^{1-\epsilon(N)}$ approximator to the clique, one can color any k-colorable graph on M nodes with $M^{o(1)}$-colors in time $M^{O(\log M)}$.

4.2 On the Limitations of Some Common Approaches

In this section we provide lower bounds on the free-bit complexity of two tasks which are central to all existing ("low-complexity") probabilistically checkable proofs. Specifically, we consider the task of checking that a string (given by oracle access) is "close" to a valid codeword and the task of checking that one oracle is an encoding of a projection of a string encoded by a second oracle. Here a string is considered **close** to the code if its distance from some codeword is less than half the distance of the code. Loosely speaking, we show that each of these tasks has amortized free-bit complexity of at least *one* (and this is tight by the codes and tests presented in Section 3.12). Furthermore, we show that the amortized free-bit complexity of performing both tasks (with respect to the same given oracles) is at least *two* (and also this is tight by Section 3.12).

We consider the lower bound on the complexity of the projection test to be more robust since avoiding such a test (which is in essence a "consistency" test) requires departing significantly from the known paradigms of constructing pcp systems. On the other hand, the lower bound on the complexity of the codeword test relies on the particular interpretation of 'closeness' used above (i.e., being at distance less than *half* the distance of the code). This requirement is not essential as can be seen in Section 3.4, where we show that also relaxed codeword tests, in which closeness means approximately the distance of the code, suffice. Furthermore, as shown by Hastad in [Has], pcp systems of amortized free-bit complexity 1 (for NP) can be constructed by using a relaxed form of the codeword test, In a sense the lower bound on the complexity of the codeword test (and of the combined test) justify Hastad's relaxation of the requirements from the codeword test.

4.2.1 The tasks

Our definitions of the various tasks/tests are quite minimal and do not suffice for the applications. However, as we are proving lower bounds this only makes our results stronger.

Loosely speaking, the first task consists of testing that an oracle encodes a valid codeword, or is "close" to a valid codeword, with respect to an error-correcting code of non-trivial distance (i.e., distance greater than 1). The condition regarding the distance of the code is essential since the task is easy with respect to the identity map (which is a code of distance 1). We remark that

[1] A Levin-reduction is a polynomial-time many-to-one reduction which is augmented by corresponding polynomial-time witness transformations.

testing "closeness" to codewords with respect to codes of large distance is essential in all known pcp constructions [BFLS, FGLSS, ArSa, ALMSS, BGLR, FeKi, BeSu].

The **absolute distance** between two words $w, u \in \{0,1\}^n$, denoted $\Delta(w,u)$, is the number of bits on which w and u disagree. We say that the code $E : \{0,1\}^* \mapsto \{0,1\}^*$ has **absolute distance** d if for every m and every $x \neq y \in \{0,1\}^m$ the absolute distance between $E(x)$ and $E(y)$ is at least $d(m)$. The absolute distance between a word w and a code E, denoted $\Delta_E(w)$, is defined as the minimum absolute distance between w and a codeword of E.

Definition 4.2.1 (codeword test): Let $E : \{0,1\}^m \to \{0,1\}^n$ be a code of absolute distance $d > 1$. A **codeword test** (with respect to E) is an oracle machine, T, such that $T^{E(a)}(R)$ accepts for all a, R. The error probability of T is defined as the maximum accepting probability of T over oracles A of absolute distance at least $\lfloor d/2 \rfloor$ from the code E; namely,

$$\max_{A \in \{0,1\}^n \text{ s.t. } \Delta_E(A) \geq \lfloor d/2 \rfloor} \left\{ \Pr_R \left[T^A(R) \text{ accepts} \right] \right\}$$

(Nothing is required with respect to non-codewords which are "close" to the code.)

We do not know if our lower bounds apply to a more relaxed definition in which the codeword test is required to reject only strings which are at distance d or more from the code; namely, when the error probability of T is defined as

$$\max_{A \in \{0,1\}^n \text{ s.t. } \Delta_E(A) \geq d} \left\{ \Pr_R \left[T^A(R) \text{ accepts} \right] \right\}$$

We propose the determination of the amortized free-bit complexity of such a relaxed codeword test as an open problem. The relevance of this problem was discussed in the introduction.

The second task is defined with respect to a "projection function" π and a pair of codes, E_1 and E_2. Loosely speaking, the task consists of checking if the string E_1-encoded by the first oracle is mapped by π to the string that is E_2-encoded by the second oracle.

Definition 4.2.2 (projection test): Let E_1: $\{0,1\}^m \to \{0,1\}^n$ and E_2: $\{0,1\}^k \to \{0,1\}^{n'}$ be two codes and let $\pi : \{0,1\}^m \to \{0,1\}^k$ be a function. A **projection test** (with respect to the above) is a two-oracle machine, T, such that $T^{E_1(a), E_2(\pi(a))}(R)$ accepts for all a, R. The error probability of T is defined as the maximum accepting probability of T over oracles pairs $(E_1(a), E_2(b))$ where $b \neq \pi(a)$; namely,

$$\max_{a,b \text{ s.t. } \pi(a) \neq b} \left\{ \Pr_R \left[T^{E_1(a), E_2(b)}(R) \text{ accepts} \right] \right\}$$

(Nothing is required with respect to non-codewords.)

Finally, we consider a test T which combines the two tests above; namely, T takes two oracles A and B and performs a codeword test on A and a projection test on the pair (A, B).

Definition 4.2.3 (combined test): Let E_1: $\{0,1\}^m \to \{0,1\}^n$ be a code of absolute distance $d > 1$ and E_2: $\{0,1\}^k \to \{0,1\}^{n'}$ be two codes and let $\pi : \{0,1\}^m \to \{0,1\}^k$ be a function. A **combined test** for (E_1, E_2, π) is a two-oracle machine T such that $T^{E_1(a), E_2(\pi(a))}(R)$ accepts on all a, R. The error probability of T is defined as the maximum accepting probability of T over oracles pairs (A, B) where either $\Delta_{E_1}(A) \geq \lfloor d/2 \rfloor$ or $A = E_1(a)$, $B = E_2(b)$ but $\pi(a) \neq b$; namely,

$$\max_{(A,B) \in S} \left\{ \Pr_R \left[T^{A,B}(R) \text{ accepts} \right] \right\}.$$

where $S \stackrel{\text{def}}{=} \{(A, B) : (\Delta_{E_1}(A) \geq \lfloor d/2 \rfloor) \text{ or } (\exists a, b \text{ s.t. } A = E_1(a) \text{ and } B = E_2(b) \text{ and } \pi(a) \neq b)\}$.

(Nothing is required with respect to non-codeword pairs, (A, B), which are "close" to some pair $(E_1(a), E_2(b))$ with $\pi(a) \neq b$.)

Conventions and Notations

The **pattern** of test T on access to oracle A (resp., oracles A and B) when using coin-sequence R consists of (R and) the sequence of queries and answers made by T. Namely, this pattern, denoted $\mathsf{pattern}_T(A;R)\}$ (resp., $\mathsf{pattern}_T(A,B;R)\}$), is defined as the sequence $(R,q_1,a_1,...,q_t,a_t)$ where q_i is the i^{th} query made by T on coin-sequence R and after receiving the answers $a_1,...,a_{i-1}$. We include the queries in the pattern for sake of clarity (but they can be easily reconstructed from the coin-sequence and the answers). In case T uses two oracles, we may assume that the queries specify to which oracle they are addressed. For simplicity, we assume in the rest of this subsection that the test has access to one oracle, denoted A.

The set $\mathsf{Acc}_T(R)$ is defined to be the set of accepting patterns of T on coin-sequence R. Clearly,

$$\mathsf{Acc}_T(R) = \{\mathsf{pattern}_T(A;R) : T^A(R) \text{ accepts}\}$$

Recall that T is said to have **free-bit complexity** f if for each possible coin-sequence R it holds that $|\mathsf{Acc}_T(R)| \leq 2^f$. We say that T has *average free-bit complexity* f_{av} if $\mathbf{E}_R[|\mathsf{Acc}_T(R)|] \leq 2^{f_{\text{av}}}$, when the expectation is taken uniformly over all possible coin-sequences. The amortized free-bit complexity of a test is defined as $\frac{f_{\text{av}}}{\log_2(1/\epsilon)}$, where f_{av} is the average free-bit complexity of the test and ϵ is its error probability.

4.2.2 Lower Bound for the Codeword Test

Proposition 4.2.4 For any code of absolute distance greater than 1, the Codeword Test has amortized free-bit complexity of at least $1-o(1)$.

The amortization in the above proposition is to be understood as taking place on a fixed number of free-bits whereas the length of the oracle grows. Actually, we can allow both the oracle-length and the free-bit count to grow, provided that the logarithm of the number of codewords grows faster than the free-bit complexity. Alternatively, we can consider a fixed oracle length and a fix bound on the number of free-bits. Actually, this is done in the following technical lemma from which the above proposition follows.

Lemma 4.2.5 Let $E : \{0,1\}^m \mapsto \{0,1\}^n$ be a code of absolute distance $d > 1$, and let T be a codeword test with respect to E having average free-bit complexity f_{av}. Then, T has error probability at least $\frac{1}{F} - \frac{1}{M}$, where $F = 2^{f_{\text{av}}}$ and $M = 2^m$. Furthermore, T has error probability at least $2 - 2^{f_{\text{av}}}$.

Proof: Fix an arbitrary coin-sequence R, and let F_R denote the cardinality of the set $\mathsf{Acc}_T(R)$.

Let a_1, a_2 be selected independently and uniformly in $\{0,1\}^m$, and consider the codewords $E(a_1)$ and $E(a_2)$. With probability $\frac{1}{M}$ we have $a_1 = a_2$ and otherwise $\Delta(E(a_1), E(a_2)) \geq d$. From a_1 and a_2, we construct an oracle $A(a_1, a_2)$ as follows: If $a_1 = a_2$, then $A = E(a_1)$. Otherwise, we construct $A(a_1, a_2)$ so that it agrees with the value of the bits of both $E(a_i)$'s whenever they are the same and is at distance $\lceil d/2 \rceil$ from $E(a_1)$. This can be done as follows: let S be the set of positions on which $E(a_1)$ and $E_2(a_2)$ disagree and let S' be a subset of S of cardinality $\lceil d/2 \rceil$. Then $A(a_1, a_2)$ equals $E(a_1)$ on all positions not in S' (and equals $E(a_2)$ on the positions in S').

We claim that, when $a_1 \neq a_2$, the oracle $A \stackrel{\text{def}}{=} A(a_1, a_2)$ is at distance at least $\lfloor d/2 \rfloor$ from the code (i.e., $\Delta_E(A) \geq \lfloor d/2 \rfloor$). This can be proved as follows: Consider any $a \in \{0,1\}^m$ and observe that by the triangle inequality

$$\Delta(A, E(a)) \geq \Delta(E(a_1), E(a)) - \Delta(E(a_1), A) \geq d - \lceil d/2 \rceil = \lfloor d/2 \rfloor$$

We now claim that
$$\Pr_{a_1,a_2}\left[T^{A(a_1,a_2)}(R) \text{ accepts}\right] \geq \frac{1}{F_R}$$
where the probability is taken uniformly over all possible choices of $a_1, a_2 \in \{0,1\}^m$. The key observation is that if $\mathsf{pattern}_T(E(a_1);R)$ equals $\mathsf{pattern}_T(E(a_2);R)$, then $\mathsf{pattern}_T(A(a_1,a_2);R)$ will be equal to $\mathsf{pattern}_T(E(a_1);R)$ (since no query of $T(R)$ falls in the set S – defined above). Thus, since $T^{E(a_1)}(R)$ accepts, $T^{A(a_1,a_2)}(R)$ must accept too. This suggests to lower bound the probability that $T^{A(a_1,a_2)}(R)$ accepts by the probability that $\mathsf{pattern}_T(E(a_1);R) = \mathsf{pattern}_T(E(a_2);R)$. Consider an enumeration, $\alpha_1, ..., \alpha_{F_R}$, of the patterns in $\mathsf{Acc}_T(R)$ and denote by p_i the probability that $\mathsf{pattern}_T(E(a);R)$ equals the i^{th} pattern in this enumeration, when a is uniformly selected in $\{0,1\}^m$ (i.e., $p_i \stackrel{\text{def}}{=} \Pr_a[\mathsf{pattern}_T(E(a);R) = \alpha_i]$). Thus, when a_1 and a_2 are picked at random, the probability that $\mathsf{pattern}_T(E(a_1);R) = \mathsf{pattern}_T(E(a_2);R)$ is $\sum_{i=1}^{F_R} p_i^2$. Subject to the condition $\sum_i p_i = 1$, the quantity $\sum_{i=1}^{F_R} p_i^2$ is lower bounded by $\frac{1}{F_R}$ (with an equality occurring when the p_i's are equal).

The following observations now bound the error of T:
$$\begin{aligned}\Pr_{a_1,a_2}\left[T^{A(a_1,a_2)}(R) \text{ accepts and } a_1 \neq a_2\right] &\geq \Pr_{a_1,a_2}\left[T^{A(a_1,a_2)}(R) \text{ accepts}\right] - \Pr_{a_1,a_2}[a_1 = a_2] \\ &\geq \frac{1}{F_R} - \frac{1}{M}\end{aligned}$$

All the above holds for any coin-sequence R. Now, we let R be uniformly chosen and get
$$\begin{aligned}\Pr_{R,a_1,a_2}\left[T^{A(a_1,a_2)}(R) \text{ accepts and } a_1 \neq a_2\right] &\geq \mathbf{E}_R\left[\frac{1}{F_R}\right] - \frac{1}{M} \\ &\geq \frac{1}{F} - \frac{1}{M}\end{aligned}$$

(The last inequality follows by Jensen's inequality.) Thus there must exist oracles a_1 and a_2 with $a_1 \neq a_2$ such that
$$\Pr_R\left[T^{A(a_1,a_2)}(R) \text{ accepts}\right] \geq \frac{1}{F} - \frac{1}{M}$$
But the oracle $A(a_1,a_2)$ above satisfies $\Delta_E(A(a_1,a_2)) \geq \lfloor d/2 \rfloor$ implying that the error of T is at least $\frac{1}{F} - \frac{1}{M}$.

For the "furthermore" part observe that if $F_R = 1$ for some coin-sequence R then $\mathsf{pattern}_T(E(a_1);R) = \mathsf{pattern}_T(E(a_2);R)$, for every two $a_1, a_2 \in \{0,1\}^m$. It follows that, for every $a_1 \neq a_2$, given access to the oracle $A(a_1,a_2)$ and using coin-sequence R, the test T accepts (and is wrong in doing so). Thus, for every $a_1 \neq a_2$,
$$\Pr_R\left[T^{A(a_1,a_2)}(R) \text{ accepts}\right] \geq \Pr_R[F_R = 1]$$
and the Furthermore Claim follows by using Markov's Inequality (i.e., $\Pr_R[F_R > 1] \leq \mathbf{E}_R[F_R - 1]$). ∎

Proof of Proposition 4.2.4: Let T be a test for the code $E : \{0,1\}^* \to \{0,1\}^*$ so that E maps m-bit strings into $n(m)$-bit strings. Suppose that T has average free-bit complexity $f(m)$ and error $\epsilon(m)$, as a function of m (the length of strings encoded by the oracle). We first assume that $f(m) \geq 1$. Using Lemma 4.2.4 (and letting $\rho(m) \stackrel{\text{def}}{=} 2^{f(m)-m}$), we lower bound the amortized free-bit complexity of T as follows

$$\begin{aligned}
\frac{f(m)}{\log_2(1/\epsilon(m))} &\geq \frac{f(m)}{-\log_2(\frac{1}{2^{f(m)}} - \frac{1}{2^m})} \\
&= \frac{f(m)}{f(m) - \log_2(1-\rho(m))} \\
&> \frac{f(m)}{f(m) + \rho(m)} \\
&> 1 - \rho(m)
\end{aligned}$$

(For the last inequality, we have assumed $f(m) \geq 1$.) Thus, for this case, the proposition follows by our convention that the number of codewords (denoted 2^m) grows faster than exponential in the free-bit complexity $f(m)$ (i.e., $\rho(m) = \frac{2^{f(m)}}{2^m} \to 0$ with $n \to \infty$). Finally, we need to address the case in which $f(m) \geq 1$ does not hold. We consider two sub-cases. In the first sub-case, we assume that $f(m) \to 0$ for some subsequence of the m's. For these m's the Furthermore-part of Lemma 4.2.4 guarantees that $\epsilon(m) \geq 2 - 2^{f(m)}$. Setting $g(m) \stackrel{\text{def}}{=} 2^{f(m)} - 1$, we lower bound the amortized free-bit complexity by

$$\begin{aligned}
\frac{f(m)}{\log_2(1/\epsilon(m))} &\geq \frac{\log_2(1+g(m))}{-\log_2(1-g(m))} \\
&\to \frac{g(m)}{g(m)}
\end{aligned}$$

For the other sub-case, we have $f(m) \geq t$, for some constant $t > 0$. Applying T for t times we get a test T' with average free-bit complexity $t \cdot f(m) \geq 1$ and error $\epsilon'(m) = \epsilon(m)^t$, which maintains the amortized free-bit complexity of T (since $\frac{f(m)}{-\log_2 \epsilon(m)} = \frac{t \cdot f(m)}{-\log_2 \epsilon'(m)}$). Applying the above analysis to T', the proposition follows. ▮

4.2.3 Lower Bound for the Projection Test

A *projection function* is a function $\pi : \{0,1\}^* \mapsto \{0,1\}^*$ having the property that for every m there exists a k so that π maps $\{0,1\}^m$ onto $\{0,1\}^k$.

Proposition 4.2.6 For any pair of codes used in the two oracles and any projection function, the Projection Test has amortized free-bit complexity of at least $1 - o(1)$.

Again, the proposition is proved by the following technical lemma. Actually, the lemma refers to any function $\pi : \{0,1\}^m \mapsto \{0,1\}^k$ and its conclusion depends on the cardinality of the range of π (which in case of a projection function equals 2^k). Abusing notations we let $\pi(S) \stackrel{\text{def}}{=} \{\pi(a) : a \in S\}$.

Lemma 4.2.7 Let $E_1 : \{0,1\}^m \mapsto \{0,1\}^n$, $E_2 : \{0,1\}^k \mapsto \{0,1\}^{n'}$ and $\pi : \{0,1\}^m \mapsto \{0,1\}^k$ be as in Definition 4.2.2, and T be a projection test with respect to them having average free-bit complexity f_{av}. Then, T has error probability at least $\frac{1}{F} - \frac{1}{K}$, where $K = |\pi(\{0,1\}^m)|$ and $F = 2^{f_{\text{av}}}$. Furthermore, if $K > 1$ then T has error probability at least $2 - 2^{f_{\text{av}}}$.

Proof: Fixing an arbitrary coin-sequence R, let $F_R \stackrel{\rm def}{=} |\{\mathsf{Acc}_T(R)\}|$. We consider the behavior of the test T when given oracle access to a pair of randomly and independently selected codewords. Specifically, let $S \subset \{0,1\}^m$ be a set of K strings such that for every $b \in \pi(\{0,1\}^m)$ there exists an $a \in S$ satisfying $\pi(a) = b$. We consider the behavior of T when given access to the oracles $E_1(a)$ and $E_2(\pi(a'))$, where a and a' are independently and uniformly selected in S. With probability $\frac{1}{K}$, we have $\pi(a) = \pi(a')$. On the other hand we claim that, given access to such pair of random oracles, T accepts with probability at least $\frac{1}{F_R}$. Once the claim is proven, the lemma follows (as in the proof of the previous lemma).

Consider the set of all F_R possible accepting patterns of T on access to oracles, $E_1(a)$ and $E_2(\pi(a))$, where $a \in S$. Each such pattern consists of a pair (α, β), where α (resp., β) denotes the transcript of the test's interaction with $E_1(a)$ (resp., $E_2(\pi(a))$). Enumerating all possible F_R patterns, we denote by p_i the probability that the $i^{\rm th}$ pattern occurs, when T is given access to the oracle-pair $(E_1(a), E_2(\pi(a))$ where a is uniformly selected in S. Namely,

$$p_i \stackrel{\rm def}{=} \Pr_{a\in S}\left[\mathsf{pattern}_T(E_1(a), E_2(\pi(a)); R) = (\alpha_i, \beta_i)\right]$$

where (α_i, β_i) is the $i^{\rm th}$ accepting pattern for $T(R)$. Clearly,

$$\Pr_{a,a'\in S}\left[\mathsf{pattern}_T(E_1(a), E_2(\pi(a)); R) = \mathsf{pattern}_T(E_1(a'), E_2(\pi(a')); R) = (\alpha_i, \beta_i)\right] = p_i^2 \qquad (4.3)$$

We now claim that the probability that a pair of independently chosen random oracles (i.e., $(E_1(a), E_2(b))$ selected by uniformly selecting $a, a' \in S$ and setting $b = \pi(a')$) leads to the $i^{\rm th}$ pattern is at least p_i^2; namely,

$$\Pr_{a,a'\in S}\left[\mathsf{pattern}_T(E_1(a), E_2(\pi(a')); R) = (\alpha_i, \beta_i)\right] \geq p_i^2 \qquad (4.4)$$

Eq. (4.4) is proven by a cut-and-paste argument: Suppose $\mathsf{p} \stackrel{\rm def}{=} \mathsf{pattern}_T(E_1(a), E_2(\pi(a)); R)$ equals $\mathsf{p}' \stackrel{\rm def}{=} \mathsf{pattern}_T(E_1(a'), E_2(\pi(a')); R)$ and consider a computation of $T^{E_1(a),E_2(\pi(a'))}(R)$. Proceeding by induction, and assuming that the first t queries are answered as in p, we conclude that the $t+1^{\rm st}$ query (in our "mixed" computation) is identical to the $t+1^{\rm st}$ query in $\mathsf{p} = \mathsf{p}'$. If this query is directed to the fist oracle then it is answered by $E_1(a)$ (as in p) and otherwise it is answered by $E_2(\pi(a'))$ (as in p'). In both cases the answer matches the $t+1^{\rm st}$ answer in $\mathsf{p} = \mathsf{p}'$. We conclude that whenever $\mathsf{p} = \mathsf{p}'$, the computation of $T^{E_1(a),E_2(\pi(a'))}(R)$ encounters the same pattern (p). Thus, the probability that the computation of $T^{E_1(a),E_2(\pi(a'))}(R)$ encounters the $i^{\rm th}$ pattern is lower bounded by the expression in Eq. (4.3), and Eq. (4.4) follows. (We remark that for non-adaptive tests, the probability that the $i^{\rm th}$ pattern is encountered equals $\sum_{i=1}^{F_R} p_i' p_i''$, where p_i' (resp., p_i'') is the sum of all p_j's satisfying $\alpha_j = \alpha_i$ (resp., $\beta_j = \beta_i$). Actually, the same holds for any test which selects its queries for each oracle independently of answers obtained from the other oracle.)

Using Eq. (4.4), we get

$$\begin{aligned} \Pr_{a,a'\in S}\left[\mathsf{pattern}_T(E_1(a), E_2(\pi(a')); R) \in \mathsf{Acc}_T(R)\right] &\geq \sum_{i=1}^{F_R} p_i^2 \\ &\geq \frac{1}{F_R} \end{aligned}$$

and the main part of the lemma follows. Again, the furthermore part follows by observing for $F_R = 1$, $\mathsf{pattern}_T(E_1(a), E_2(\pi(a)); R) = \mathsf{pattern}_T(E_1(a'), E_2(\pi(a')); R)$, for every two $a, a' \in \{0,1\}^m$. Again, this implies that, for every $a_1 \neq a_2$, given access to the oracle-pair $(E_1(a), E_2(\pi(a')))$ and using coin-sequence R, the test T (wrongly) accepts. ∎

4.2.4 Lower Bound for the Combined Test

Proposition 4.2.8 For any pair of codes used in the two oracles, so that the first code has absolute distance greater than 1, and for any projection function, the Combined Test has amortized free-bit complexity of at least $2 - o(1)$.

Again, the proposition is proved by the following technical lemma. Loosely speaking, the lemma asserts that a combined test of free-bit complexity $2f$ must have error probability at least $\frac{1}{8} \cdot 2^{-f}$. The lower bound extends to the case where $2f$ is a bound on the average free-bit complexity; the error probability in this case can be lower bounded by $\frac{3}{64} \cdot 2^{-f}$ – see details below. It follows that the amortized free-bit complexity of such a test must be at least $\frac{2f}{f+5} \approx 2$ (for large f's). The restriction to large f's does not really weaken the result. Suppose on the contrary that there exists a test with amortized free-bit complexity $f_{\rm am}$. Then, for any sufficient large t, we can obtain a test with free-bit complexity $2f \stackrel{\rm def}{=} t \cdot f_{\rm am}$ and error 2^{-t}. By the above $\frac{t \cdot f_{\rm am}}{t} \geq \frac{2f}{f+5} \approx 2$ (as f is now large).

Lemma 4.2.9 Let $E_1 : \{0,1\}^m \mapsto \{0,1\}^n$ be a code of absolute distance greater than 1, $E_2 : \{0,1\}^k \mapsto \{0,1\}^{n'}$, and $\pi : \{0,1\}^m \mapsto \{0,1\}^k$ be a projection function. Suppose that T is a combined codeword and projection test with respect to the above having free-bit complexity $2f$. Then, T has error probability at least $\frac{1}{8F} - \frac{1}{2K} - \frac{1}{4M}$, where $K = 2^k$, $F = 2^f$, and M is the minimum, over all $b \in \{0,1\}^k$, of the number of $a \in \{0,1\}^m$ projected by π to b (i.e., $M \stackrel{\rm def}{=} \min_{b \in \{0,1\}^k} \{|\{a : \pi(a) = b\}|\}$). Furthermore, if $2f < 1$ and $\max\{M, K\} > 1$ then T has error probability 1.

Proof: The "furthermore" part follows immediately by any of the furthermore parts of Lemma 4.2.5 or Lemma 4.2.7 (as 2^{2f} must be an integer and so $2f < 1$ implies $f = 0$). The proof of the main part of the lemma uses both strategies employed in the proofs of Lemmas 4.2.5 and 4.2.7. We consider two cases. The first case is that for some $E_2(b)$, half of the possible (coin-sequences) R's have at most F accepting patterns with respect to the coin-sequence R and second oracle $B = E_2(b)$. In this case we employ the strategy used in the proof of Lemma 4.2.5, restricted to oracles constructed by combining two uniformly selected codewords $E_1(a_i)$'s satisfying $\pi(a_i) = b$. The second case is that for every $b \in \{0,1\}^k$, for half of the possible (coin-sequences) R's, the number of accepting patterns with respect to the coin-sequence R and second oracle $B = E_2(b)$ is at least F. In this case we show that many possible B's must fit into fewer than $\frac{F^2}{F}$ accepting patterns and we may employ the strategy used in the proof of Lemma 4.2.7. Details follow.

In the sequel $\delta \in [0,1]$ is a constant to be determined later. (In the above motivating discussion we have used $\delta = \frac{1}{2}$ but a better bound follows by letting δ be larger.)

Case 1: there exists $b \in \{0,1\}^k$ so that for at least $(1-\delta)$ fraction of the possible (coin-sequences) R's, hereafter called good, the number of accepting patterns with respect to the coin-sequence R and second oracle (fixed to) $B = E_2(b)$ is at most F.

Fixing this b, we consider M possible a's satisfying $\pi(a) = b$. Employing the argument of Lemma 4.2.5, we get that for each of these *good* R's, a random oracle A (constructed using two uniformly chosen a's as above) is wrongly accepted with probability at least $\frac{1}{F} - \frac{1}{M}$. By an averaging argument, it follows that there exists a pair of oracles (A, B) on which T errs with probability at least

$$(1-\delta) \cdot \left(\frac{1}{F} - \frac{1}{M}\right) \tag{4.5}$$

Case 2: for every $b \in \{0,1\}^k$, for at least a δ fraction of the possible (coin-sequences) R's, the number of accepting patterns with respect to the coin-sequence R and second oracle $B = E_2(b)$ is at least F.

Let $\gamma < \delta$ be a parameter to be determined later. By a counting argument, for at least a $\frac{\delta-\gamma}{1-\gamma}$ fraction of the possible R's, hereafter called **good**, there exists a set, denoted Π_R, of at least $\gamma \cdot 2^k$ possible $b \in \{0,1\}^k$ so that there are at least F accepting patterns which are consistent with coin-sequence R and second oracle fixed to $B = E_2(b)$. (Namely, let g denote the fraction of good R's. Then $g + (1-g) \cdot \gamma \geq \delta$ and $g \geq \frac{\delta-\gamma}{1-\gamma}$ follows.)

Let $S \subset \{0,1\}^m$ be a set of 2^k strings, defined as in the proof of Lemma 4.2.7, so that π maps S onto $\{0,1\}^k$. Fixing a good coin-sequence R, we adapt the strategy used in the proof of Lemma 4.2.7 as follows. We consider a set $S_R \subseteq S$ of $|\Pi_R|$ strings so that π maps S_R onto Π_R, and enumerate the accepting patterns which occur when the test, using coins R, is given access to a oracle-pair $(E_1(a), E_2(\pi(a)))$, where a is uniformly chosen in S_R. We first claim that there are at most F such patterns. Namely,

Claim: For any good R, $|\{\mathsf{pattern}_T(E_1(a), E_2(\pi(a)); R) : a \in S_R\}| \leq F$.

Proof: By definition of Π_R, for each $b \in \Pi_R$, there are at least F accepting patterns consistent with the coin-sequence R and the second oracle $E_2(b)$ (and out of them only one fits the first oracle $E_1(a)$ where $a \in S_R$ and $\pi(a) = b$). By a cut-and-paste argument, if (R, α, β) and (R, α', β) are accepting patterns for second-oracle $E_2(b)$ and if (R, α, β) is an accepting pattern for second-oracle $E_2(b')$ then (R, α', β) is also an accepting pattern for second-oracle $E_2(b')$. It follows that the accepting patterns of two $E_2(b)$'s either collide or do not intersect. Thus, the number of accepting patterns for the various $(E_1(a), E_2(\pi(a)))$'s, where $a \in S_R$, is at most $\frac{F^2}{F} = F$ and the claim follows. □

Now we consider what happens if one selects independently and uniformly $a, a' \in S$. Following the proof of Lemma 4.2.7, with probability $\frac{1}{K}$, we have $\pi(a) = \pi(a')$ (and otherwise $\pi(a) \neq \pi(a')$). On the other hand, given access to such pair of random oracles, the test accepts with probability at least $\gamma^2 \cdot \frac{1}{F}$. (The γ^2 factor is due to the probability that $a, a' \in S_R$, whereas the $\frac{1}{F}$ factor corresponds to the analysis which supposes that a and a' are uniformly selected in S_R).

The above analysis holds for any good coin-sequence R. Using the lower bound on the fraction of good R's, it follows that for a $\frac{\delta-\gamma}{1-\gamma}$ fraction of the R's, the probability that the test errs, on coin-sequence R when given access to a random pair of oracles (selected as above), is at least $\frac{\gamma^2}{F} - \frac{1}{K}$. By an averaging argument, there exists a pair of oracles for which the test errs with probability

$$\frac{\delta-\gamma}{1-\gamma} \cdot \left(\frac{\gamma^2}{F} - \frac{1}{K}\right) \tag{4.6}$$

It is left to select δ and γ so to maximize the minimum among the expressions in Equations (4.5) and (4.6). (But why bother?) Setting $\delta = \frac{3}{4}$ and $\gamma = \frac{1}{2}$ we lower bound these expressions by $\frac{1}{4F} - \frac{1}{4M}$ and $\frac{1}{8F} - \frac{1}{2K}$, respectively, and the (current statement of the) lemma follows. ∎

To prove a bound for the case of average free-bit complexity F^2, we first apply Markov's Inequality and conclude that all but an ϵ fraction of the coin-sequences have at most $G^2 \stackrel{\text{def}}{=} \frac{F^2}{\epsilon}$ accepting patterns (in which this fixed coin-sequence appears). (We can use any $0<\epsilon<1$.) We then consider only those coin sequences (and apply the same argument as above to each of them). The averaging argument at the end of the above proof then yields that there exists an oracle-pair on which T errs on at least a $\frac{1}{8G} - \frac{1}{2K} - \frac{1}{4M}$ fraction of these coin-sequences. It follows that this oracle makes T err with probability at least $(1-\epsilon)\cdot(\frac{1}{8G}-\frac{1}{2K}-\frac{1}{4M})$ (which equals $(1-\epsilon)\cdot(\frac{\sqrt{\epsilon}}{8F}-\frac{1}{2K}-\frac{1}{4M})$). Using $\epsilon=\frac{1}{4}$, we get a lower bound of $\frac{3}{64F}-\frac{3}{8K}-\frac{3}{16M}$.

CHAPTER 5

PCP: Properties and Transformations

5.1 The Complexity of PCP and FPCP

In this section we present several results regarding the complexity of languages acceptable by probabilistically checkable proofs having, respectively, small query complexity, small amortized-query complexity and small free-bit complexity. Thus, in the current section, notations such as $\mathrm{PCP}_{c,s}[r,q]$ stand for classes of languages. The results can be extended to classes of promise problems having such probabilistically checkable proofs.

5.1.1 Query complexity and amortized query complexity

In this subsection, $\mathrm{MIP}_{c,s}[r,p]$ denotes the class of languages accepted by a (one-round) p-prover interactive proof system in which r is the randomness complexity, c is a lower bound on the probability of accepting yes-instances and s is an upper bound on the probability of accepting no-instances. The corresponding class for probabilistically checkable proofs is $\mathrm{PCP}_{c,s}[r,q]$, where q denotes the number of queries. In both classes only binary queries are allowed (indeed this is less standard for MIP). The first part of the following lemma is folklore and is stated here for sake of completeness.

Lemma 5.1.1 For all admissible functions c, s, r, p.
(1) $\mathrm{MIP}_{c,s}[r,p] \subseteq \mathrm{PCP}_{c,s}[r,p]$.
(2) $\mathrm{MIP}_{c,s}[r,p] \subseteq \mathrm{MIP}_{c,2s}[r,p-1]$.

Proof: Part (1) follows from the definition of PCP and MIP. Part (2) is shown as follows. Let V be an (r,p)-restricted MIP verifier. We define V' – an $(r,p-1)$-restricted verifier who on input x behaves as follows:

- V' tosses coins $\overline{c}$ for V.
- V' refers the first $p-1$ queries of V to the corresponding $p-1$ provers obtaining answers (bits) $a_1, \ldots, a_{p-1}$, respectively.
- V' accepts if and only if there exists $a_p \in \{0,1\}$ such that V would accept answers $a_1, \ldots, a_p$ on input x and random string $\overline{c}$.

Suppose that provers $P_1, \ldots, P_p$ convince V to accept x with probability δ. Then, the provers $P_1, \ldots, P_{p-1}$ convince V' to accept x with probability at least δ (because if $V(x)$ accepts the transcript $(\overline{c}, a_1, ..., a_p)$ then $V'(x)$ will accept the transcript $(\overline{c}, a_1, ..., a_{p-1})$). This justifies the bound on the completeness probability of V'. Suppose, on the other hand, that provers $P_1, \ldots, P_{p-1}$ cause V' to accept x with probability δ. Consider a uniformly selected strategy for another prover, denoted P_p (i.e., choose a random response for every question). Then, the probability that provers $P_1, \ldots, P_p$ cause V to accept input x is at least $\frac{1}{2} \cdot \delta$ (because if $V'(x)$ accepts the transcript $(\overline{c}, a_1, ..., a_{p-1})$ then there exists a value $a_p \in \{0,1\}$ so that $V'(x)$ will accept the transcript $(\overline{c}, a_1, ..., a_p)$ and with probability one half P_p answer equals this a_p). This justifies the bound on the soundness probability of V'. ∎

The following proposition explores the limitations of probabilistically checkable proof systems which use logarithmic randomness and upto three queries. Some of the qualitative assertions are well-known; for example, when considering perfect completeness, 3 queries are the minimum needed (and sufficient [ALMSS]) to get above P.

Proposition 5.1.2 (PCP systems with logarithmic randomness and at most 3 queries):

(1) (PCP with 1 query is weak): For all admissible functions $s, c : \mathcal{Z}^+ \to [0,1]$, so that s is strictly smaller than c, $\mathrm{PCP}_{c,s}[\log, 1] = \mathrm{P}$.

(2) (One-sided error pcp with 2 queries is weak): For all admissible functions $s : \mathcal{Z}^+ \to [0,1]$ strictly less than 1, $\mathrm{PCP}_{1,s}[\log, 2] = \mathrm{P}$.

(3) (Two-sided error pcp with 2 queries is not weak): On the other hand, there exists $0 < s < c < 1$ so that $\mathrm{PCP}_{c,s}[\log, 2] = \mathrm{NP}$. Furthermore, this holds for $c > 0.9$ and $s < \frac{73}{74}c$.

(4) (One-sided error pcp with 3 queries is not weak): $\mathrm{PCP}_{1,0.85-\epsilon}[\log, 3] = \mathrm{NP}$, $\forall \epsilon > 0$.

(5) (One-sided error pcp with 3 queries is not very strong): $\forall s < 0.18$, $\mathrm{PCP}_{1,s}[\log, 3] = \mathrm{P}$. Furthermore, $\forall s \leq 0.299$, $\mathbf{na}\mathrm{PCP}_{1,s}[\log, 3] = \mathrm{P}$, where $\mathbf{na}\mathrm{PCP}$ is a restriction of PCP in which the verifier is required to be non-adaptive.

Proof of Proposition 5.1.2, Part (1): Part (1) is obvious since an oracle π maximizing the acceptance probability can be constructed by scanning all possible random pads and setting $\pi(q)$ so that it "satisfies" the majority of random-pads for which the verifier makes query q. ∎

Proof of Proposition 5.1.2, Part (2): The folklore proof commonly deals only with the non-adaptive case. In general, the verifier V, demonstrating that $L \in \mathrm{PCP}_{1,s}[\log, 2]$, may be adaptive. We assume, without loss of generality, that V always makes at least one query. Thus, after making the first query, V decides whether to accept, reject or make an additional query and accept only a specific answer for it. Thus, the computation of V on input x, random pad $\overline{c}$ and access to a generic oracle can be captured by two Horn clauses, each corresponding to a different answer-value for the first query. Specifically, suppose that V queries the oracle at location i and upon receiving value σ accepts iff location j have value τ. Then, we write the Horn clause $\pi_i^\sigma \to \pi_j^\tau$. (In case V always accepts (resp., rejects) after obtaining value σ from oracle location i, we write the clause $\pi_i^\sigma \to \mathbf{T}$ (resp., $\pi_i^\sigma \to \mathbf{F}$).) In addition, for every i, we write the Horn clauses $\pi_i^0 \to (\neg\pi_i^1)$ and $(\neg\pi_i^0) \to \pi_i^1$. Thus, the computation of V, on input x and access to a generic oracle, can be captured by a Horn formula, denoted ϕ_x, in which Horn clauses correspond to the various (polynomially many) possible (random-pad,first-answer) pairs. Furthermore, ϕ_x can be constructed in polynomial-time given x (and V). Using a (polynomial-time) decision procedure for satisfiability of Horn Formulae, we are done. (Alternatively, we can use the linear-time decision procedure for 2-SAT due to Even et. al. [EIS].) ∎

Proof of Proposition 5.1.2, Part (4): To see that $\mathrm{PCP}_{1,s}[\log, \mathrm{poly}] \subseteq \mathrm{NP}$, for every $s < 1$, consider a non-deterministic machine which tries to guess an oracle which makes the verifier (of the above system) always accept. The other direction (of Part (4)) is shown in Theorem 3.6.4. ∎

Proof of Proposition 5.1.2, Part (3): To see that $\mathrm{PCP}_{c,s}[\log, \mathrm{poly}] \subseteq \mathrm{NP}$, for every $s < c$, consider a non-deterministic machine which tries to guess an oracle which makes the verifier accept with probability at least c. The $\mathrm{NP} \subseteq \mathrm{PCP}_{c,s}[\log, 2]$ result follows from the hardness of approximating Max2SAT. Specifically, suppose that the following promise problem is NP-hard (via Karp reductions): given a 2CNF formula decide whether there exists a truth assignment which satisfies at least a c fraction of the clauses or any truth assignment satisfies at most a s fraction of its clauses, where $0 < s < c < 1$ are fixed constants. Then we can present a $\mathrm{PCP}_{c,s}[\log, 2]$ system for any $L \in \mathrm{NP}$. On input x, the verifier in this system, performs the reduction (of L to the promise problem) obtaining a 2CNF formula ϕ_x, next it uniformly selects a clause of ϕ_x and queries the oracle for the values of the variables in this clause (accepting accordingly). Using the result in Section 3.7, we can set $c > 0.9$ and $s < \frac{73}{74} \cdot c$.

Remark: It may be possible to increase the ratio c/s by implementing the inner verifier used to establish the NP-Hardness of Max2SAT using arbitrary 2-literal clauses, rather than 2CNF clauses. ∎

Proof of Proposition 5.1.2, Part (5): The result for general verifiers follows from Lemma 3.7.5 and the fact that MaxSAT can be approximated to within a $0.795 = 0.75 + \frac{0.18}{4}$ factor in polynomial-time (cf., [SSTW]). The rest of the proof is devoted to the non-adaptive case. Let $L \in \mathbf{na}\mathrm{PCP}_{1,s}[\log, 3]$, and let V be a (non-adaptive) verifier demonstrating this fact. Without loss of generality, we may assume that V always makes 3 different queries. As a mental experiment we define, for every set $Q \subseteq \{0,1\}^*$, a "verifier" V_Q who on input x acts as follows:

- V_Q uniformly selects a random pad $\overline{c}$ for V.
- Let q_1, q_2 and q_3 be the three queries of V, on input x and randomness $\overline{c}$. (The hypothesis that V is non-adaptive is crucial for the definition of q_2 and q_3.)
- If all three (desired) queries are in Q then V_Q accepts (without making any query!).
- Otherwise (i.e., not all q_j's are in Q), then V_Q makes only the queries which lie in Q. Specifically, for every j such that $q_j \in Q$, the verifier V_Q makes query q_j, obtaining an answer denoted a_j.
- V_Q accepts x if and only if there exists a triple (b_1, b_2, b_3) so that
  - $b_j = a_j$ for each $q_j \in Q$; and
  - V accepts the input x on randomness $\overline{c}$ and oracle answers (b_1, b_2, b_3).

It is clear that for every set Q, the verifier V_Q uses logarithmic randomness and makes at most two queries. At this point we don't consider the issue of implementing V_Q. The probability that V_Q accepts x (given access to oracle π) is greater or equal to the probability that V accepts x (given access to oracle π). Thus, if V can be led (by an appropriate oracle) to always accept the input x, so can V_Q. We now show that, for every $x \notin L$, provided some condition (specified below) on Q holds, V_Q accepts x with probability strictly less than 1.

Claim. Fix any $x \notin L$ and any set Q. For $i = 0, 1, 2, 3$, let $p_i^x(Q)$ denote the probability (taken over V's coin tosses) that V, on input x, generates i queries in the set Q. (Since V is non-adaptive this is well defined.) Suppose that

$$p^x(Q) \stackrel{\text{def}}{=} \sum_{i=0}^{2} \frac{1}{2^{3-i}} \cdot p_i^x(Q) > s$$

Then, given access to any oracle, V_Q accepts x with probability strictly less than 1.

that V_Q, when given oracle access to π, always accepts input $x \notin L$ (i.e., accepts with probability 1). We will show that there exists a proof π' such that V, when given access to oracle π', accepts input $x \notin L$ with probability $p^x(Q) > s$, contradicting the soundness of V.

We start by considering a random oracle, denoted ρ, defined as follows. For every $q \in Q$, we set $\rho(q) \stackrel{\text{def}}{=} \pi(q)$. For every $q \notin Q$, we let $\rho(q)$ be uniformly and independently distributed in $\{0,1\}$, We now lower bound the accepting probability of V when given access to ρ, using the hypothesis that V_Q always accepts. Let $\overline{c}$ be a random-pad for V and suppose that V using random pad $\overline{c}$ makes $m > 0$ queries outside Q. Then, using the random-pad $\overline{c}$, the verifier V_Q accepts while refraining from making m queries. It follows that V, using random pad $\overline{c}$ and given oracle access to ρ, accepts with probability at least 2^{-m}, where the probability is taken over the choice of ρ. Since V_Q, given access to π, always accepts x it follows that V, on access to a random ρ, accepts x with probability at least $\sum_{m=1}^{3} p_{3-m}^x(Q) \cdot \frac{1}{2^m} = p^x(Q)$. It follows that there exists an oracle π' so that, given access to π', the verifier V accepts x with probability at least $p^x(Q)$. Since $x \notin L$ we conclude, using the soundness of V, that $p^x(Q) \leq s$ in contradiction to the hypothesis of the claim. □

Next we present a polynomial-time algorithm that, given $x \in \{0,1\}^*$, efficiently constructs a set Q with high $p^x(Q)$. Note that there are only polynomially many queries to consider for membership in Q (specifically these appearing in all possible computations of V on input x). We first consider a randomized construction of a set Q, in which each such query is included in Q with probability q independently of all other queries, where q is a fixed parameter. Now, the expected value of $p^x(Q)$ equals $\sum_{i=0}^{2} \frac{1}{2^{3-i}} \cdot q_i^x$, where q_i^x is the probability that V on input x makes i queries which hit the random set Q (the probability is taken over V's coin tosses and the random choice of Q). Clearly, $q_i^x = \binom{3}{i} q^i (1-q)^{3-i}$. Thus, the expected value of $p^x(Q)$ equals

$$p(q) \stackrel{\text{def}}{=} \sum_{i=0}^{2} \frac{1}{2^{3-i}} \cdot \binom{3}{i} q^i (1-q)^{3-i} = \frac{3}{2}(1-q)q^2 + \frac{3}{4}(1-q)^2 q + \frac{1}{8}(1-q)^3$$

Using the method of conditional probabilities [ASE], given x, we can construct in (deterministic) polynomial-time a set Q satisfying $p^x(Q) \geq p(q)$. In the construction we use the fact that, given a partial specification of a set Q, we can compute the expected value of $p^x(Q)$ where the expectation is taken over all random extension of Q. (Specifically, this is done by considering all random pads for V and considering for each such pad the number of queries which are yet unspecified. Each such unspecified query is in Q with probability q.) Thus, we obtain a polynomial-time verifier V_q which uses logarithmically many coins and two queries. Furthermore, V_q accepts any $x \in L$ with probability 1 and, provided $p(q) > s$, accepts $x \notin L$ with probability strictly less than 1. We conclude that if $p(q) > s$ then, for some $s' < 1$, $L \in \mathrm{PCP}_{1,s'}[\log, 2] = \mathrm{P}$ (where the equality is due to Part (2)).

To conclude the proof we need to select q so to maximize $p(q)$. Numerical experiments show that there exists q so that $p(q) > 0.299$ and $\mathrm{PCP}_{1,0.299}[\log, 3] \subseteq \mathrm{PCP}_{1,s'}[\log, 2] = \mathrm{P}$ follows (for some $s' < 1$). This completes the proof of Part (5). ▮

The (stronger) bound obtained in Lemma 5.1.2.5, let alone that it is restricted to the non-adaptive case, is weaker than what can be proven for MIP proof systems (see next corollary). This contrast may perhaps provide a testing ground to separate PCP from MIP, a question raised by [BGLR]. The following corollary is obtained by combining Lemma 5.1.1 and Proposition 5.1.2.2.

Corollary 5.1.3 For $s < 1/2$, $\mathrm{MIP}_{1,s}[\mathsf{coins} = \log, \mathsf{provers} = 3] = \mathrm{P}$.

A general result which relates the query complexity of a probabilistically checkable proof system and the ratio between the acceptance probabilities of yes-instances and no-instances, follows –

Lemma 5.1.4 For all admissible functions c, s, q, r, l such that $\frac{c}{s} > 2^q$,

$$\mathrm{PCP}_{c,s}[r, q] \subseteq \mathrm{RTIME}\left(\mathrm{poly}\left(n, \frac{1}{c - 2^q s}\right)\right)$$

Furthermore, $\mathrm{PCP}_{c,s}[r, q] \subseteq \mathrm{PSPACE}$, and if r and q are both logarithmically bounded then $\mathrm{PCP}_{c,s}[r, q] = \mathrm{P}$.

Proof: Let $L \in \mathrm{PCP}_{c,s}[r, q]$ and V be a verifier demonstrating this fact. Observe that for $x \in L$, the probability that V accepts x, given access to a random oracle, is at least $\frac{c}{2^q}$. On the other hand, for $x \notin L$, the probability that V accepts x, given access to any oracle, is at most $s < \frac{c}{2^q}$. Thus, we can decide if x is in L by simulating the execution of V with access to a random oracle and estimating the acceptance probability, over V's random choices and all possible oracles. In particular, we can estimate this probability upto an $\epsilon \stackrel{\mathrm{def}}{=} s - \frac{c}{2^q}$ additive term, with very high probability, by taking $\mathrm{poly}(1/\epsilon)$ samples. Alternatively, we can compute this probability in polynomial-space. Finally, in case r and q are both logarithmically bounded, we can (exactly) compute the probability that V accepts x, given access to a random oracle. To this end we loop through all possible random-pads for V and for each pad consider all possibilities of setting the oracle bits examined by V. Thus, for $s < \frac{c}{2^q}$, we get a deterministic polynomial-time decision procedure. ▮

The last assertion in the above lemma (i.e., $\mathrm{PCP}_{c,s}[\log, q] = \mathrm{P}$ for $\frac{c}{s} > 2^q$). cannot be strengthen by omitting the (logarithmic) bound on q since $\mathrm{NP} = \mathrm{PCP}_{1,0}[0, \mathrm{poly}]$. On the other hand, recalling the definition of $\overline{\mathrm{PCP}}$ we immediately get

Corollary 5.1.5 Let $\epsilon : \mathcal{Z}^+ \to [0, 1]$ be an admissible function strictly greater than 0. Then, for every admissible function $c : \mathcal{Z}^+ \to [0, 1]$,

$$\overline{\mathrm{PCP}}_c[\,\log n,\ 1 - \epsilon\,] = \mathrm{P}$$

In particular, this holds for $c = 1$.

Proof: $L \in \overline{\mathrm{PCP}}_c[\,\log,\ 1 - \epsilon\,]$ implies that for some logarithmically bounded function m, we have $L \in \mathrm{PCP}_{c,2^{-m}\cdot c}[\log, (1 - \epsilon) \cdot m]$ and the corollary follows. ▮

The above results are focused on pcp systems with logarithmic randomness. However, proof systems with unrestricted randomness (as considered in the next proposition) may also provide some indication to the effect of very low query complexity. The results we obtain are somewhat analogous to those of Proposition 5.1.2. Recall that $\mathrm{PCP}_{1,\frac{1}{2}}[\mathrm{poly}, \mathrm{poly}]$ equals NEXPT (Non-deterministic exponential time) [BFL]. Thus, the power of pcp systems with polynomial randomness has to be compared against NEXPT.

Proposition 5.1.6 (general PCP systems with at most 3 queries):

(1) (PCP with 1 query is relatively very weak): For all admissible functions $s, c : \mathcal{Z}^+ \to [0,1]$, so that $c(n) - s(n)$ is non-negligible[1]

$$\mathrm{PCP}_{c,s}[\mathrm{poly}, 1] \subseteq \mathrm{AM}$$

where AM is the class of languages having one-round Arthur-Merlin proof systems (cf., [Bab]).

(2) (One-sided error pcp with 2 queries is relatively weak): For all admissible functions $s : \mathcal{Z}^+ \to [0,1]$ strictly less than 1, $\mathrm{PCP}_{1,s}[\mathrm{poly}, 2] \subseteq \mathrm{PSPACE}$.

(3) (Two-sided error pcp with 2 queries is not weak): On the other hand, there exists $0 < s < c < 1$ so that $\mathrm{PCP}_{c,s}[\mathrm{poly}, 2] = \mathrm{NEXPT}$.

(4) (One-sided error pcp with 3 queries is not weak): $\mathrm{PCP}_{1,0.85-\epsilon}[\mathrm{poly}, 3] = \mathrm{NEXPT}$, $\forall \epsilon > 0$.

(5) (One-sided error pcp with 3 queries is not very strong): $\forall s < \frac{1}{8}$, $\mathrm{PCP}_{1,s}[\mathrm{poly}, 3] = \mathrm{PSPACE}$. Furthermore, $\forall s \leq 0.299$, $\mathbf{na}\mathrm{PCP}_{1,s}[\mathrm{poly}, 3] = \mathrm{PSPACE}$.

The first part of the proposition may be hard to improve since, as indicated in Proposition 5.1.7 Part (6), Graph Non-Isomorphism is in $\mathrm{PCP}_{1,\frac{1}{2}}[\mathrm{poly}, 1]$.

Proof of Proposition 5.1.6, Part (1): We first observe that a 1-query pcp system is actually a one-round interactive proof system (cf., [GMR]). (The completeness and soundness bounds are as in the pcp system.) Using well-known transformations we obtain the claimed result. Specifically, we first reduce the error of the interactive proof by parallel repetition, next transform it into an Arthur-Merlin interactive proof [GS], and finally transform it into an Arthur-Merlin interactive proof of perfect completeness [FGMSZ]. We stress that all the transformations maintain the number of rounds upto a constant and that the constant-round Arthur-Merlin hierarchy collapses to one-round [Bab]. ∎

Proof of Proposition 5.1.6, Parts (3) and (4): For these parts we observe that the proof systems used in the corresponding items of the proof of Proposition 5.1.2, do "scale-up". Specifically, it is easy to see that the outer verifier used for all proof systems in this paper does scale-up, yielding a canonical outer verifier of randomness complexity $O(\log(T(n))$ fo any language in $\mathrm{Ntime}(T(n))$, provided $n < T(n) < 2^{\mathrm{poly}(n)}$. Furthermore, all inner-verifiers used in the paper operate on constant sized oracles and so the composed verifier maintains the time and randomness complexities of the outer verifier. In particular, the verifier used for establishing Theorem 3.6.4 can be scaled-up to yield Part (4). The same holds for the verifier used for establishing Part (3) of Proposition 5.1.2. (Note that although the exposition of the proof in Proposition 5.1.2 is in terms of reducing NP to Max2SAT, what actually happens is that the verifier used to establish the NP-hardness of Max2SAT (cf., Section 3.7) is implemented by a verifier which makes only two queries (out of a constant number of possibilities).) ∎

Proof of Proposition 5.1.6, Part (2): Following the strategy of the proof of the analogous part in Proposition 5.1.2, we obtain a polynomial-space reduction of $L \in \mathrm{PCP}_{1,s}[\mathrm{poly}, 2]$ to the set of satisfiable 2-Horn formulae (i.e., Horn formulae in which each clause has at most 2 literals). Namely, on input x, the reduction uses space $\mathrm{poly}(|x|)$ and produces a Horn formula ϕ_x (of size exponential in $|x|$) so that $x \in L$ iff ϕ_x is satisfiable. Using a poly-logarithmic decision procedure for satisfiability of 2-Horn formulae[2], we can decide if ϕ_x is satisfiable using $\mathrm{poly}(|x|)$-space. ∎

---

[1] A function $f : \mathcal{Z}^+ \to \mathcal{Z}^+$ is called non-negligible if there exists a positive polynomial p so that $\forall n : \; f(n) > \frac{1}{p(n)}$.

[2] For example, consider the following procedure. Given a 2-Horn formula, we construct a directed graph in which the vertices are the literals of the formula and there is an directed edge from literal x to literal y if the formula

Proof of Proposition 5.1.6, Part (5): The result for non-adaptive verifiers follows from Part (2) by using the same strategy as in the analogous proof in Proposition 5.1.2. The result for general verifiers follows by the Furthermore-part of Lemma 5.1.4 (i.e., $\mathrm{PCP}_{c,s}[\mathrm{poly}, q] = \mathrm{PSPACE}$ for $\frac{c}{s} > 2^q$). ▮

5.1.2 Free-bit complexity

The class $\mathrm{FPCP}_{c,s}[r, f]$ is defined analogously to the class $\mathrm{PCP}_{c,s}[r, q]$, except that we consider the free-bit complexity (denoted f) instead of the query complexity (denoted q). The following proposition demonstrates the limitations of probabilistically checkable proof systems with free-bit complexity bounded by 1. We do not believe that similar limitations hold for amortized free-bit complexity.[3] The first three items refer to proof systems with logarithmic randomness. The very first item shows that proof systems with two-sided error (non-perfect completeness) having amortized free-bit complexity zero (and logarithmic randomness) suffice for $\mathcal{NP}$. The third item asserts that the second item cannot be strengthened neither with respect to increasing the free-bit complexity nor with respect to referring to two-sided error. However, proof systems with unrestricted randomness (as considered in the other items) may also provide some indication to the effect of very low free-bit complexity. The last item can be viewed as (weak) evidence that the result in the fourth item cannot be "drastically improved" (e.g., to yield $\mathrm{FPCP}_{1,s}[\mathrm{poly}, 0] \subseteq \mathrm{BPP}$).

Proposition 5.1.7 (PCP systems with low free-bit complexity): Let $s : \mathcal{Z}^+ \to [0, 1]$ be an admissible function strictly smaller than 1. Then,

(1) (PCP with logarithmic randomness and 0 free-bit):
There exists $s < 0.794$ so that $\mathrm{NP} \subseteq \mathrm{FPCP}_{\frac{1}{4},\frac{s}{4}}[\log, 0]$. Thus, $\mathrm{NP} = \overline{\mathrm{FPCP}}_{\frac{1}{4}}[\log, 0]$.

(2) (Limitations of PCP with logarithmic randomness and 1 free-bit):
$\mathrm{FPCP}_{1,s}[\log, 1] = \mathrm{P}$. Also, $\mathrm{FPCP}_{1,1-(1/\mathrm{poly})}[\mathsf{coins} = \mathrm{poly}\ ;\ \mathsf{free} = 1\ ;\ \mathsf{pflen} = \mathrm{poly}] \subseteq \mathrm{BPP}$.

(3) ("Tightness" of Item 2): There exists $s < 0.794$ so that
- $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,s}[\log, 2]$;
- $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,\frac{1+s}{2}}[\log, f]$ where $f = \log_2 3$ (i.e., $2^f = 3$);
- $\mathrm{NP} \subseteq \mathrm{FPCP}_{\frac{1}{2},\frac{s}{2}}[\log, 1]$.

(4) (General pcp with 0 free-bit): $\mathrm{FPCP}_{1,s}[\mathrm{poly}, 0] \subseteq \mathrm{coNP}$.

(5) (general pcp with 1 free-bit): $\mathrm{FPCP}_{1,s}[\mathrm{poly}, 1] \subseteq \mathrm{PSPACE}$.

(6) (Examples for pcp with 0 free-bit): Graph Non-Isomorphism, `GNI`, has a PCP system with perfect completeness and soundness bound $\frac{1}{2}$, in which the verifier makes a single query and this query is free. Namely,

$$\mathtt{GNI} \in \mathrm{FPCP}_{1,\frac{1}{2}}[\mathsf{coins} = \mathrm{poly}\ ;\ \mathsf{free} = 0\ ;\ \mathsf{query} = 1]$$

The same holds for `QNR` ("Quadratic Non-Residuosity" (cf., [GMR])) the set of integer pairs (x, N) so that x is a non-residue modulo N.

---

contains the clause $x \to y$. One can easily verify that the formula is not satisfied iff there exists a variable for which every truth assignment yields a contradiction (i.e., "forcing paths" to contradicting values – cf., [EIS]). Thus, a non-deterministic logspace machine can guess this variable and check that both possible truth assignments (to it) yield contradictions. The latter checking reduces to guessing the variable for which a conflicting assignment is implied and verifying the conflict via s-t directed connectivity. Since the latter task is in $\mathcal{NL}$, we are done. (Actually, 2SAT is complete for co$\mathcal{NL}$; see [JLL].)

[3]The conjecture is stated for systems with perfect completeness. For systems with two-sided error probability, we know that they can recognize $\mathcal{NP}$ languages using zero free-bits – see below.

Proof of Proposition 5.1.7, Part (4): Here we consider proofs with zero free-bits. Let $L \in \mathrm{PCP}_{1,s}[\mathrm{poly}, 0]$ and V be a verifier demonstrating this fact. By definition, for every possible sequence of coin tosses for V, there exists at most one accepting configuration (of oracle answers to the queries made by V). Furthermore, by definition, this accepting configuration (if it exists) can be generated in polynomial time, from the coin-sequence. Following is a non-deterministic procedure that accepts $\overline{L}$. It starts by guessing two sequences of coin tosses for V, generating the corresponding accepting configurations and checking whether they are consistent. Clearly, if $x \in L$ then for all possible pairs of coin-sequences these configurations exist and are consistent (since an oracle which always makes V accept x does exist). On the other hand, if all pairs of coin-sequences yield accepting and mutually consistent configurations then an oracle which always makes V accept x emerges. ∎

Proof of Proposition 5.1.7, Parts (2) and (5): Here we consider proofs with free-bit complexity 1. Thus, for each possible sequence of coin tosses, there exist at most two accepting configurations (which again can be efficiently found given the coin-sequence). We refer to these two possible accepting configuration as to the 1-configuration and the 2-configuration of the coin-sequence. In case a specific coin-sequence has less than two accepting configurations, we introduce dummy configurations so that now each coin-sequence has two associated configurations. Given an input x to such a pcp system, we consider the following 2CNF formula representing all possible computations of the verifier with a generic oracle. For each possible sequence of coin tosses, $\overline{c}$, we introduce a pair of Boolean variables, $\pi^1_{\overline{c}}$ and $\pi^2_{\overline{c}}$, representing which of the two associated configurations is encountered (e.g., $\pi^1_{\overline{c}} = T$ means that the 1-configuration is encountered). To enforce that a single configuration is encountered we introduce the clauses $(\pi^1_{\overline{c}} \vee \pi^2_{\overline{c}})$ and $((\neg\pi^1_{\overline{c}}) \vee (\neg\pi^2_{\overline{c}}))$. In addition, in case the σ-configuration of $\overline{c}$ is not accepting (but rather a dummy configuration) we introduce the clause $(\neg\pi^\sigma_{\overline{c}})$ thus "disallowing" a computation in which it is encountered. Finally, for each pair of coin-sequences we introduce clauses disallowing inconsistencies. Namely, suppose that the σ-configuration of $\overline{c}$ is inconsistent with the τ-configuration of $\overline{c}'$, then we introduce the clause $((\neg\pi^\sigma_{\overline{c}}) \vee (\neg\pi^\tau_{\overline{c}'}))$, which is logically equivalent to $\neg(\pi^\sigma_{\overline{c}} \wedge \pi^\tau_{\overline{c}'})$. The resulting 2CNF formula, ϕ_x, is satisfiable if and only if there exists an oracle which causes V to accept x with probability 1. Thus, given x, we need to test if ϕ_x is satisfiable. We consider two cases.

(1) In case V uses logarithmically many coins, the 2CNF formula ϕ_x can be generated from x in polynomial-time. Using a polynomial-time decision procedure for satisfiability of 2CNF formulae, we conclude that $\mathrm{FPCP}_{1,s}[\log, 1] = \mathrm{P}$. Using Proposition 5.2.2, we can randomly reduce $\mathrm{FPCP}_{1,1-(1/\mathrm{poly})}[\mathrm{poly}, \mathsf{free} = 1, \mathsf{pflen} = \mathrm{poly}]$ to $\mathrm{FPCP}_{1,1-(1/\mathrm{poly})}[\log, \mathsf{free} = 1]$, and $\mathrm{FPCP}_{1,1-(1/\mathrm{poly})}[\mathrm{poly}, \mathsf{free} = 1, \mathsf{pflen} = \mathrm{poly}] \subseteq \mathrm{BPP}$ follows. This establishes Part (2).

(2) In general (V may make polynomially many coin tosses), the 2CNF formula ϕ_x may have exponential (in $|x|$) length and yet can be generated from x in polynomial-space. Using a poly-logarithmic-space decision procedure for satisfiability of 2CNF formulae[4], we can decide if ϕ_x is satisfiable using $\mathrm{poly}(|x|)$-space. Part (5) (i.e., $\mathrm{FPCP}_{1,s}[\mathrm{poly}, 1] \subseteq \mathrm{PSPACE}$) follows.

∎

Proof of Proposition 5.1.7, Parts (3) and (1): The first claim of Part 3 is justified by Theorem 3.9.4. Applying Proposition 5.2.9 to this verifier (which indeed satisfies the condition of this proposition), yields the second claim of Part 3. Applying Proposition 5.2.8 to the same verifier

[4]For example, note that 2CNF formulae can be written in Horn form and use the procedure described in the proof of Proposition 5.1.6 Part (2).

(with $k = 1 < f = 2$), the third claim of Part 3 follows. Finally, applying Proposition 5.2.8 with $k = f = 2$, Part 1 follows. ▮

Proof of Proposition 5.1.7, Part (6): We merely note that the interactive proof presented in [GMW] for Graph Non-Isomorphism[5] constitute a 1-query pcp system with perfect completeness and soundness bound $\frac{1}{2}$. Furthermore, the query made by the verify has a unique acceptable answer and thus the free-bit complexity of this system is zero. The same holds for the interactive proof presented in [GMR] for Quadratic Non-Residuosity QNR, which is actually the inspiration to the proof in [GMW]. ▮

5.1.3 Query complexity versus free-bit complexity

The following proposition quantifies the intuition that not all queries are "free" (i.e., that the free-bit complexity is lower than the query complexity). Furthermore, as a corollary we obtain that the amortized (average) free-bit complexity is at least 1 unit less than the amortized query complexity.

Proposition 5.1.8 For admissible functions c, s, r, q such that $r(n), q(n) = O(\log n)$.

$$\mathrm{PCP}_{c,s}[r, q] \subseteq \mathrm{PCP}_{c,s}[\,\mathsf{coins} = r\ ;\ \mathsf{free}_{\mathrm{av}} = q - \log_2(1/s)\,]$$

Furthermore, for every admissible function t, $\mathrm{PCP}_{c,s}[r, q] \subseteq \mathrm{FPCP}_{c,(2^t+1)s}[r, q - t]$.

Proof: Let $L \in \mathrm{PCP}_{c,s}[r, q]$ and let V be the verifier demonstrating this. Fix an input $x \in \Sigma^n$, and let $r = r(n), q = q(n), s = s(n)$ For a random string $R \in \{0,1\}^r$, let F_R^x denote the number of accepting patterns of V, i.e., $F_R^x = |\mathsf{pattern}_V(x; R)|$. We first claim that if $\mathbf{E}_R[F_R^x] > 2^q \cdot s$, then $x \in L$. This is true since a random oracle π is accepted with probability $\mathbf{E}_R[F_R^x \cdot 2^{-q}]$ and in case the claim does not hold we reach contradiction to the soundness condition (i.e., $x \notin L$ is accepted with probability strictly larger than s).

We now construct a verifier, denoted V', witnessing $L \in \mathrm{FPCP}^{\mathrm{av}}_{c,s}[r, q - \log_2(1/s)]$. On input x, the verifier first computes $\mathbf{E}_R[F_R^x]$ (by scanning all possible R's and generating all accepting patterns for each of them). If $\mathbf{E}_R[F_R^x]) > 2^q \cdot s$, then V' accepts x (without querying the oracle). Otherwise (i.e., if $\mathbf{E}_R[F_R^x]) \leq 2^q \cdot s$), then V' simulates V and accepts if V accepts. It follows that the average free-bit complexity of V' on input x equals the corresponding quantify for V, provided the latter is at most $q - \log_2(1/s)$, and equals zero otherwise. The first part of the proposition follows.

To establish the second part, for some $t = t(n)$, we construct a verifier V'' which, on input x, proceeds as follows. First, V'' computes $q \stackrel{\mathrm{def}}{=} \mathbf{E}_R[F_R^x]$ and accepts if $q > s2^q$ (just as V'). In case $q \leq s2^q$, the new verifier proceeds differently: it randomly selects R as V does and computes F_R^x. If $F_R^x > 2^{q-t}$ then V'' accepts and otherwise it invokes V on input x and coins R. Clearly, this guarantees that the free-bit complexity of V'' is at most $q - t$. To analyze the soundness of V'', note that when $\mathbf{E}_R[F_R^x] \leq s2^q$, it follows that $\Pr_R[F_R^x > 2^{q-t}] \leq 2^t s$ (Markov Inequality). Thus, the soundness error of V'' is at most $s + 2^t s$ and the second part follows. ▮

By computing the amortized average free bit complexity of the class of languages in the right hand side of the containment above, we obtain the following consequence.

[5] On input a pair of graphs, G_0 and G_1, the verifier uniformly selects $i \in \{0,1\}$ and generates a random isomorphic copy of G_i, denoted H. This graph H is the single query made by the verifier, which accepts if and only if the answer equals i.

Corollary 5.1.9 For admissible functions c, r, q with $r(n), q(n) = O(\log n)$,

$$\overline{\mathrm{PCP}}_c[\,r,\,q\,] \subseteq \overline{\mathrm{FPCP}}_c^{\mathrm{av}}[\,r,\,q-1\,].$$

where $\overline{\mathrm{FPCP}}_{\cdot}^{\mathrm{av}}[\,\cdot,\,f\,]$ denotes a class analogous to $\overline{\mathrm{FPCP}}_{\cdot}[\,\cdot,\,f\,]$ in which average free-bit complexity is measured instead of (worst-case) free-bit complexity.

The above corollary clinches the argument that the amortized query complexity is incapable of capturing the approximability of the clique function. Previously we had argued thus based on the assumption that the clique number may be hard to approximate to within $N^{\frac{1}{2}}$ (i.e., establishing such a clique NP-hardness would require showing that NP $\subseteq \overline{\mathrm{PCP}}[\,\log,\,1-\epsilon\,]$, for every $\epsilon > 0$, which is impossible[6] as we've shown that $\overline{\mathrm{PCP}}[\,\log,\,1-\epsilon\,] \subseteq$ P). Now, we can remove this assumption also. Suppose that, for some g (e.g., $g = \frac{3}{2}$), MaxClique is NP-hard to approximate to within a $N^{1/(1+g)}$ factor, but it can be approximated to within a $N^{1/(1+g-\delta)}$ factor in polynomial-time, for every $\delta > 0$ (actually, we can handle any $\delta \leq 1$). Furthermore, supposed that the hardness result is demonstrated by showing that NP $\subseteq \overline{\mathrm{PCP}}[\,\log,\,g-\epsilon\,]$, for every $\epsilon > 0$. Then, using the above corollary, we get NP $\subseteq \overline{\mathrm{FPCP}}^{\mathrm{av}}[\,\log,\,g-1-\epsilon\,]$, for every $\epsilon > 0$. and an NP-hardness result of clique approximation[7] upto a $N^{1/(1+(g-1-\epsilon)+\epsilon)} = N^{1/g}$ follows, in contradiction to our hypothesis that such approximations could be achieved in polynomial time. To summarize, attempts to establish the factor $N^{1/g}$ for which it is NP-hard to approximate MaxClique via amortized query complexity will always fall at least one unit away from the truth; whereas amortized free-bit complexity will yield the right answer.

5.2 Transformations of FPCP Systems

We present several useful transformations which can be applied to pcp systems. These fall to two categories:

(1) Transformations which amplifies the (completeness versus soundness) gap of the proof system, while preserving (or almost preserving) its amortized free-bit complexity.

(2) Transformations which move the gap location (or, equivalently, the completeness parameter). The gap itself is almost preserved but the moving it changes the free-bit complexity (and thus the amortized free bit complexity is not preserved). Specifically, moving the gap 'up' requires increasing the free-bit complexity, whereas moving the gap 'down' allows to decrease the free-bit complexity.

All these transformations are analogous to transformations which can be applied to graphs with respect to the max-clique problem. In view of the relation between FPCP and the clique promise problem (shown in Section 4.1), this analogy is hardly surprising.

In this section, we use a more extensive FPCP notation which refers to promise problems (rather than to languages) and introduce an additional parameter – the proof length. Specifically, $\mathrm{FPCP}_{c,s}[r, f, l]$ refers to randomness complexity r, free-bit complexity f and proof-length l.

5.2.1 Gap amplification maintaining amortized free-bit complexity

We start by stating the simple fact that the ratio between the completeness and soundness bounds (also referred to as gap) is amplified (i.e., raise to the power k) when one repeats the pcp system (k times). Note, however, that if the original system is not perfectly complete then the completeness bound in the resulting system gets decreased.

[6]The entire discussion assumes P $\neq$ NP. The discussion is anyhow mute otherwise.

[7] Here we use the observation that the FGLSS-reduction works also for amortized *average* free-bit complexity.

Proposition 5.2.1 (simple gap amplification): For all $c, s : \mathcal{Z}^+ \to [0,1]$ and $k : \mathcal{Z}^+ \to \mathcal{Z}^+$,

$$\mathrm{FPCP}_{c,s}[r,f,l] \subseteq \mathrm{FPCP}_{c^k,s^k}[kr,kf,l].$$

Proof: Let $(Y,N) \in \mathrm{FPCP}_{c,s}[r,f,l]$ and let V be a verifier witnessing this with query complexity $q : \mathcal{Z}^+ \to \mathcal{Z}^+$. Given $k : \mathcal{Z}^+ \to \mathcal{Z}^+$, we define a verifier $V^{(k)}$ as follows: On input $x \in \{0,1\}^n$, let $r = r(n), k = k(n), f = f(n), l = l(n)$ and $q = q(n)$.

- $V^{(k)}$ picks k random strings $\overline{c}^{(1)}, \ldots, \overline{c}^{(k)}$ uniformly and independently in $\{0,1\}^r$.
- For $i = 1$ to k, verifier $V^{(k)}$ simulates the actions of V on input x and random string $\overline{c}^{(i)}$. Verifier $V^{(k)}$ accepts if V accepts on each of these k instances.

Clearly, $V^{(k)}$ tosses kr coins and examines the l-bit long oracle in at most kq bits, where at most kf of these are free. For every x, if the probability that V accepts x, given access to oracle π, is p then the probability that $V^{(k)}$ accepts x, given access to π is exactly p^k. Thus, $(Y,N) \in \mathrm{FPCP}_{c^k,s^k}[kr,kf,l]$, and oracles can be transformed (by identity) from one pcp system to the other. ∎

Next, we show that in some sense the randomness-complexity of a proof system need not be higher than logarithmic in the length of the proofs/oracles employed. Specifically, we show how to randomly reduce languages proven by the first kind of systems into languages proven by the second kind. Thus, whenever one is interested in the computational complexity of languages proven via pcp systems, one may assume that the system is of the second type. Recall that $\leq_R^K$ denotes a randomized Karp reduction.

Proposition 5.2.2 (reducing randomness): There exists a constant $\gamma > 0$ so that

(1) (for perfect completeness): For every two admissible functions $s, \epsilon : \mathcal{Z}^+ \to [0,1]$,

$$\mathrm{FPCP}_{1,s}[r,f,l] \leq_R^K \mathrm{FPCP}_{1,s'}[r',f,l]$$

where $s' = (1+\epsilon)\cdot s$ and $r' = \gamma + \log_2(l/\epsilon^2 s)$.

(2) (for two-sided error): For every four admissible functions $c, s, \epsilon_1, \epsilon_2 : \mathcal{Z}^+ \to [0,1]$,

$$\mathrm{FPCP}_{c,s}[r,f,l] \leq_R^K \mathrm{FPCP}_{c',s'}[r',f,l]$$

where $c' = 1 - (1+\epsilon_1)\cdot(1-c) \geq c - \epsilon_1,\ \ s' = (1+\epsilon_2)\cdot s$ and $r' = \gamma + \max\{-\log_2(\epsilon_1^2(1-c))\ ,\ \log_2(l) - \log_2(\epsilon_2^2 s)\}$.

Proof: The proof is reminiscent of Adleman's proof that $\mathcal{RP} \subseteq \mathrm{P}/\mathrm{poly}$ [Ad]. Suppose we are given a pcp system for which we want to reduce the randomness complexity. The idea is that it suffices to choose the random pad for the verifier out of a relatively small set of possibilities (instead than from all 2^r possibilities). Furthermore, most small sets (i.e., sets of size linear in l) are good for this purpose. This suggest randomly mapping an input x for the original pcp system into an input (x,R) for the new system, where R is a random set of $m = O(l)$ possible random-pads for the original system. The new verifier will select a random-pad uniformly in R, thus using only $\log_2|R|$ random coins, and run the original verifier using this random-pad. Details follow.

We start with the simpler case stated in Part (1). Let $(Y,N) \in \mathrm{FPCP}_{1,s}[r,f,l]$ and V be a verifier demonstrating this fact. The random reduction maps $x \in \{0,1\}^n$ to (x,R), where R is a uniformly chosen m-multi-subset of $\{0,1\}^r$ for $l \stackrel{\mathrm{def}}{=} l(n)$, $r \stackrel{\mathrm{def}}{=} r(n)$, $s \stackrel{\mathrm{def}}{=} s(n)$, $\epsilon \stackrel{\mathrm{def}}{=} \epsilon(n)$ and $m \stackrel{\mathrm{def}}{=} \frac{\gamma l}{\epsilon^2 s}$. (The constant γ is chosen to make the Chernoff bound, used below, hold.) On input (x, R), the new verifier V' uniformly selects $\overline{c} \in R$ and invokes V with input x and random-pad $\overline{c}$. Clearly, the complexities of V' are as claimed above. Also, assuming that V always accepts x, when given access to an oracle π then, for every possible pair (x, R) to which x is mapped, V' always accepts (x, R) when given access to the oracle π. It remains to upper bound, for each $x \notin L$ and most R's, the probability that V' accepts (x, R) when given access to an arbitrary oracle.

Fixing any $x \notin L$ and any oracle π, we bound the probability that V', give access to π, accepts (x, R) for most R's. A set R is called *bad* for x with respect to π if for more than a s' fraction of the $\overline{c} \in R$ the verifier V accepts x when given access to π and random-pad $\overline{c}$. Let $R = (\overline{r}^{(1)}, ..., \overline{r}^{(m)})$ be a uniformly selected multi-set. For every $i \in [m]$ (a possible random choice of V'), we define a 0-1 random variable ζ_i so that it is 1 iff V on random-pad $\overline{r}^{(i)}$ and access to oracle π accepts the input x. Clearly, the ζ_i's are mutually independent and each equals 1 with probability $\delta \leq s$. Using a multiplicative Chernoff Bound (cf. [MoRa, Theorem 4.3]), the probability that a random R is bad (for x w.r.t. π) is bounded by

$$\Pr\left[\sum_{i=1}^{m} \zeta_i \geq (1+\epsilon)\cdot ms\right] < 2^{-\Omega(\epsilon^2 \cdot ms)}$$

Thus, by the choice of m, the probability that a random R is bad for x, with respect to any fixed oracle, is smaller than $\frac{1}{2}\cdot 2^{-l}$. Since they are only 2^l relevant oracles, the first part of the proposition follows.

For the second part of the proposition, we repeat the same argument, except that now we need to take care of the completeness bound in the resulting pcp system. This is done similarly to the way we dealt with the soundness bound, except that we do not need to consider all possible oracles – it suffices to consider the best oracle for any $x \in Y$. When applying the multiplicative Chernoff bound it is important to note that, since we are interested in the rejection-event, the relevant expectation is $m \cdot (1-c)$ (and not $m \cdot c$). Thus, as long as $m \geq \frac{2\gamma}{\epsilon_1^2(1-c)}$, at least $\frac{3}{4}$ of the possible sets R cause V' to accept $x \in Y$ with probability at least $1-(1+\epsilon_1)\cdot(1-c) = c-(1-c)\epsilon_1$. The second part of the proposition follows. ▌ Combining Propositions 5.2.1 and 5.2.2, we obtain the a

randomized reduction of pcp systems which yields the effect of Proposition 5.2.1 at much lower (and in fact minimal) cost in the randomness complexity of the resulting pcp system. This reduction is analogous to the well-known transformation of Berman and Schnitger [BeSc]. The reduction (in either forms), plays a central role in deriving clique approximation results via the FGLSS method: applying the FGLSS-reduction to proof systems obtained via the second item (below), one derives graphs of size $N \stackrel{\text{def}}{=} 2^{(1+\epsilon+f)\cdot t}$ with clique-gap 2^t (which can be rewritten as $N^{1/(1+f+\epsilon)}$).

Corollary 5.2.3 (probabilistic gap amplification at minimal randomness cost):

(1) (Combining the two propositions): For every admissible $k : \mathcal{Z}^+ \to \mathcal{Z}^+$,

$$\mathrm{FPCP}_{1,\frac{1}{2}}[r, f, l] \leq_R^K \mathrm{FPCP}_{1,2^{-k+1}}[r + \log_2 q + O(1) + k, kf, l]$$

where q is the query complexity of the first proof system.

(2) (using amortized free-bit complexity): For every $\epsilon > 0$ there exists a constant c so that

$$\overline{\mathrm{FPCP}}[\log, f] \leq_R^K \mathrm{FPCP}_{1,2^{-t}}[(1+\epsilon)\cdot t, f\cdot t, l]$$

where $t(n) = c\log_2 n$.

Proof: Suppose that $(Y, N) \in \text{FPCP}_{1,1/2}[r, f, l]$. Clearly, $l \leq 2^r \cdot q$, where $q(n) = \text{poly}(n)$ is the query complexity of the verifier. Then, applying Proposition 5.2.1, we get $(Y, N) \in \text{FPCP}_{1,1/2^k}[kr, kf, 2^r \cdot q]$. Applying Part (1) of Proposition 5.2.2, we obtain $(Y, N) \leq_R^K \text{FPCP}_{1,\frac{1}{2^{k-1}}}[r', kf]$, where $r' = O(1) + \log_2(2^r q/2^{-k}) = O(1) + r + k + \log_2 q$. The first part of the corollary follows.

Suppose now that a language L has a proof system as in the hypothesis of the second part. Then, there exists a logarithmically bounded function m so that $L \in \text{FPCP}_{1,1/2^m}[r, mf, l]$, where $r(n) \leq \alpha \cdot \log_2 n$ and $l(n) \leq n^\beta$ for some constants α and β. Invoking a similar argument (to the above), we get $L \leq_R^K \text{FPCP}_{1,\frac{1}{2^{km-1}}}[r', k \cdot mf]$, where $r'(n) = O(1) + km + (\alpha + \beta) \cdot \log_2 n$. Now, setting $k(n)$ so that $k(n) \cdot m(n) \geq \frac{\alpha+\beta}{\epsilon} \cdot \log_2 n$, the corollary follows. ▌ An alternative gap amplification procedure which does not employ randomized reductions is presented below. This transformation increases the randomness complexity of the pcp system more than the randomized reduction presented above (specifically, by a factor of 2). The transformation is used to obtain in-approximability results under the assumption $\text{P} \neq \text{NP}$ (rather than under $\text{NP} \not\subseteq \text{BPP}$). It is stated here only for the one-sided error case:

Proposition 5.2.4 (deterministic gap amplification at low randomness cost): For every $\epsilon, s > 0$ and every admissible function $k : \mathcal{Z}^+ \to \mathcal{Z}^+$

$$\text{FPCP}_{1,s}[r, f, l] \subseteq \text{FPCP}_{1,s^k}[O(r) + (2+\epsilon)k, (1+\epsilon)kf, l].$$

Actually, the constant in the O-notation is $\min\{1, \frac{2+(4/\epsilon)}{\log_2(1/s)}\}$.

The use of random walks on expander graphs for error reduction was suggested by Ajtai, Komlos and Szemeredi [AKS] (cf., [CW]). The use of random walks on expander graphs for gap amplification in the context of pcp originates in [ArSa]. The value of the constant multiplier of k in the randomness complexity of the resulting pcp system, depends on the expander graph used. Specifically, using a degree d expander graph with second eigenvalue λ yields a factor of $\frac{\log_2 d}{1+\log_2 \lambda}$. Thus, it is essential to use Ramanujan graphs [LPS] in order to obtain the claimed constant $2(1+\epsilon)$.

Proof of Proposition 5.2.4: For simplicity assume $s = 1/2$. The idea is to use a "pseudorandom" sequence generated by a random walk on an expander graph in order to get error reduction at moderate randomness cost. Specifically, we will use a Ramanujan expander graph of constant degree d and second eigenvalue $\lambda \approx 2\sqrt{d}$ (cf., [LPS]). The constant d will be determined so that $d > 2^{4+\frac{8}{\epsilon}}$ (and $d < 2^{6+\frac{8}{\epsilon}}$). It is well-known by now, that a random walk of length t in an expander avoids a set of density ρ with probability at most $(\rho + \frac{\lambda}{d})^t$ (cf., [AKS, Kah]). Thus, as a preparation step, we reduce the error probability of the pcp system to

$$p \stackrel{\text{def}}{=} \frac{\lambda}{d} = \frac{2}{\sqrt{d}} \tag{5.1}$$

This is done using the trivial reduction of Proposition 5.2.1. We derive a proof system with error probability p, randomness complexity

$$r' \stackrel{\text{def}}{=} r \cdot \log_2(1/p) = r \cdot \log_2(\sqrt{d}/2) = O(r) \tag{5.2}$$

and free-bit complexity

$$f' \stackrel{\text{def}}{=} f \cdot \log_2(1/p) = f \cdot \log_2(\sqrt{d}/2) \tag{5.3}$$

(In case we start with soundness error s, where $s > p$, the multiplier will be $\log_{1/s}(1/p)$ instead of $\log_2(1/p)$.) Now we are ready to apply the expander walk technique. Using an expander walk of length t, we transform the proof system into one in which the randomness complexity is $r' + (t-1) \cdot \log_2 d$, the free-bit complexity is $tf' = tf \cdot \log_2(\sqrt{d}/2)$ and the error probability is at most $(2p)^t = (4/\sqrt{d})^t = 2^{-k}$, where $k \stackrel{\text{def}}{=} t \cdot \log_2(\sqrt{d}/4)$. Using $\log_2 d > \frac{8}{\epsilon} + 4$, we can bound the randomness complexity by

$$\begin{aligned} r' + t\log_2 d &= r' + \frac{\log_2 d}{\frac{1}{2} \cdot (\log_2 d) - 2} \cdot k \\ &< r' + (2+\epsilon) \cdot k \end{aligned}$$

and the free-bit complexity by

$$\begin{aligned} tf \cdot \log_2(\sqrt{d}/2) &= \frac{\frac{1}{2} \cdot (\log_2 d) - 1}{\frac{1}{2} \cdot (\log_2 d) - 2} \cdot kf \\ &< (1+\epsilon) \cdot kf \end{aligned}$$

The proposition follows. ∎

Using Proposition 5.2.4, we obtain the following corollary which is used in deriving clique inapproximability results under the P $\neq$ NP assumption, via the FGLSS method: applying the FGLSS-reduction to proof systems obtained via this corollary, one derives graphs of size $N \stackrel{\text{def}}{=} 2^{(2+\epsilon+f)\cdot t}$ with clique-gap 2^t (which can be rewritten as $N^{1/(2+f+\epsilon)}$).

Corollary 5.2.5 For every $\epsilon > 0$ there exists a constant c so that

$$\overline{\mathrm{FPCP}}[\log,\, f] \subseteq \mathrm{FPCP}_{1,2^{-t}}[(2+\epsilon)\cdot t, (1+\epsilon)f \cdot t, l]$$

where $t(n) = c\log_2 n$.

5.2.2 Trading-off gap location and free-bit complexity

The following transformation is analogous to the randomized layering procedure for the clique promise problem (i.e., Proposition 4.1.6). The transformation increases the acceptance probability bounds at the expense of increasing the free-bit complexity.

Proposition 5.2.6 (increasing acceptance probabilities):

(1) (using a randomized reduction which preserves the randomness of the proof system): For all admissible functions $c, s : \mathcal{Z}^+ \to [0,1]$, and $r, f, m : \mathcal{Z}^+ \to \mathcal{Z}^+$,

$$\mathrm{FPCP}_{c,s}[r,f] \leq_R^K \mathrm{FPCP}_{c',s'}[r, f + \log_2 m]$$

where $c' = 1 - 4(1-c)^m$ and $s'_2 = m \cdot s$.
Note that if $c' > 1 - 2^{-r}$ then $c' = 1$.

(2) (inclusion which moderately increases the randomness of the proof system): For all admissible functions $c, s : \mathcal{Z}^+ \to [0,1]$, and $r, f, m : \mathcal{Z}^+ \to \mathcal{Z}^+$,

$$\mathrm{FPCP}_{c,s}[r,f] \subseteq \mathrm{FPCP}_{c',s'}[r', f + \log_2 m]$$

- where if $m \leq 1/c$ then $r' = 2 \cdot \max\{r, \log m\}$, $c' = \frac{m}{2} \cdot c$ and $s' = m \cdot s$;
- and otherwise (i.e., for $m > 1/c$), $r' = O(\max\{r, \log m\} + mc)$, $c' = 1 - 2^{-\Theta(mc)}$ and $s' = m \cdot s$.

Proof: Suppose we are given a pcp system for which we want to increase the acceptance probability bound in the completeness condition. The idea is to allow the new verifier to select m random-pads for the original verifier and query the oracle as to which pad to use. A straightforward implementation of this idea will increase the randomness complexity of the verifier too much. Instead, we use two alternative implementations, which yield the two parts of the proposition. In both implementations the free-bit complexity increases by $\log_2 m$ and the soundness bound increases by a factor of m.

The first implementation employs a technique introduced by Lautemann (in the context of $\mathcal{BPP}$) [Lau]. Using a randomized reduction, we supply the new verifier with a sequence of m possible "shifts" that it may effect. The new verifier selects one random-pad for the original verifier and generates m shifts of this pad. Now, the new verifier queries the oracle as to which of these shifts it should use as a random-pad for the original verifier. Details follow.

We first present a random reduction mapping $x \in \{0,1\}^n$ to (x, S), where S is a uniformly chosen m-multi-subset of $\{0,1\}^r$, for $r \stackrel{\rm def}{=} r(n)$. On input (x, S), the new verifier V' uniformly selects $\overline{c} \in \{0,1\}^r$ and queries the oracle on $(x, \overline{c})$ receiving an answer $i \in [m]$. Intuitively, V' asks which shift of the random-pad to use. Finally, V' invokes V with input x and random-pad $\overline{c} \oplus \overline{s}_i$, where $\overline{s}_i$ is the $i^{\rm th}$ string in S. Clearly, the complexities of V' are as claimed above. Also, assuming that V accepts x with probability δ, we get that, for every S, verifier V' accepts (x, S) with probability at most $m \cdot \delta$. On the other hand suppose that, when given access to oracle π, verifier V accepts x with probability δ. It follows that there exists a set R of $\delta 2^r$ random-pads for V so that if V uses $\overline{c} \in R$ (and queries oracle π) then it accepts x. Fixing any $\overline{c} \in \{0,1\}^r$, we ask what is the probability, for a uniformly chosen $S = \{\overline{s}_i : i \leq m\}$, that there exists an $i \in [m]$ so that $\overline{c} \oplus \overline{s}_i \in R$. Clearly, the answer is $1 - (1-\delta)^m$. Thus, by Markov Inequality, with probability at least $\frac{3}{4}$, a uniformly chosen $S = \{\overline{s}_i\}$ has the property that for at least $1 - 4 \cdot (1-\delta)^m$ of the $\overline{c}$'s (in $\{0,1\}^r$) there exists an $i \in [m]$ so that $\overline{c} \oplus \overline{s}_i \in R$. Part (1) of the proposition follows.

To prove Part (2) of the proposition, we use an alternative implementation of the above idea, which consists of letting the new verifier V' generate a "pseudorandom" sequence of possible random-pads by itself. V' will then query the oracle as to which random-pad to use, in the simulation of V, and complete its computation by invoking V with the specified random-pad. To generate the "pseudorandom" sequence we use the sampling procedure of [BGG]. Specifically, for $m \leq 1/c$ this merely amounts to generating a pairwise independent sequence of uniformly distributed strings in $\{0,1\}^r$, which can be done using randomness $\max\{2r, 2\log_2 m\}$. Otherwise (i.e., for $m > 1/c$) the construction of [BGG] amounts to generating $\Theta(cm)$ such related sequences, where the sequences are related via a random walk on a constant degree expander. Part (2) follows. ▌

The following corollary exemplifies the usage of the above proposition. In case $c(n) = n^{-\alpha}$ and $r(n) = O(\log n)$, the gap is preserved (upto a logarithmic factor) and the free-bit complexity increases by a $\log_2 1/c$ additive term. Thus, the corollary provides an alternative way of deriving the reverse-FGLSS transformation (say, Proposition 4.1.7) from the simple clique verifier of Theorem 4.1.2. Specifically, one may apply the following corollary to the simple clique verifier of Theorem 4.1.2, instead of combining the layered-graph verifier[8] (of Theorem 4.1.3), and the graph-layering process of Proposition 4.1.6.

Corollary 5.2.7 For all admissible $r, f : \mathcal{Z}^+ \to \mathcal{Z}^+$, so that $\forall n : \; r(n) \geq 2$,

$$\mathrm{FPCP}_{c,s}[r, f] \leq_R^K \mathrm{FPCP}_{1, \frac{r}{c} \cdot s}[r, f + \log_2 r + \log_2(1/c)]$$

[8] which generalizes the simple clique verifier

We conclude with another transformation which is reminiscent to an assertion made in Section 4.1. The following transformation has an opposite effect than the previous one, reducing the free-bit complexity at the expense of lowering the bounds on acceptance probability. The transformation can be effected provided each possible random-pad in the original pcp system has enough free bits.

Proposition 5.2.8 (decreasing acceptance probabilities): For all admissible functions $c, s : \mathcal{Z}^+ \to [0,1]$, and $r, f, k : \mathcal{Z}^+ \to \mathcal{Z}^+$ so that $k \le f$, if $L \in \mathrm{FPCP}_{c,s}[r,f]$ then $L \in \mathrm{FPCP}_{\frac{c}{2^k},\frac{s}{2^k}}[r+k, f-k]$. Furthermore, the average free-bit complexity of the resulting system is $\max\{0, f_{\mathrm{av}} - k\}$, where f_{av} is the average free-bit complexity of the original system.

Proof: Let V be a verifier satisfying the condition of the proposition. We construct a new verifier V' that on input $x \in \{0,1\}^n$, setting $r = r(n)$, $k = k(n)$ and $f = f(n)$, acts as follows. Verifier V' uniformly selects a random-pad $\overline{c} \in \{0,1\}^r$ for V, and generates all possible accepting configurations with respect to $V(x)$ and random-pad $\overline{c}$. In case there are less than 2^k accepting configurations we add *dummy configurations* to reach the 2^k count. We now partition the set of resulting configurations (which are accepting and possibly also dummy) into 2^k parts of about the same size (i.e., some parts may have one configuration more than others). Actually, if we only care about average free-bit complexity then any partition of the accepting configurations into 2^k non-empty parts will do. The new verifier, V', uniformly selects $i \in [2^k]$ thus specifying one of these parts, denoted A_i. Next, V' invokes V with random-pad $\overline{c}$ and accepts if and only if the oracle's answers form an accepting configuration which is in A_i (i.e., resides in the selected portion of the accepting configurations). (We stress that in case $\overline{c}$ has less than 2^k accepting configurations and the selected A_i does not contain any accepting configuration then V' rejects on coins $(i, \overline{c})$.) Clearly, the randomness complexity of the new verifier is $r + k$.

To analyze the other parameters of V', we fix any $x \in \{0,1\}^n$. For sake of simplicity, we first assume that the number of accepting configurations of V for any random-pad is a power of 2. Then the number of accepting configurations of V' for any random-pad $(\overline{c}, i) \in \{0,1\}^r \times [2^k]$ is 2^{m-k}, where 2^m is the number of accepting configurations of V on random-pad $\overline{c}$. Thus, the free-bit complexity of V' is $f - k$. Finally, we relate the acceptance probability of V' to that of V. This is done by reformulating the execution of V' with oracle π as consisting of two steps. First V' invokes V with access to π. If V reaches a rejecting configuration then V' rejects as well; otherwise (i.e., when V reaches an accepting configuration), V accepts with probability 2^{-k} (corresponding to uniformly selecting $i \in [2^k]$). It follows that on input x and access to oracle π, the verifier V' accepts with probability $\frac{\delta}{2^k}$, where δ denotes the probability that V accepts input x when given access to oracle π.

In general, our simplifying assumption that the number of accepting configurations of V is a power of 2, may not hold and the analysis becomes slightly more cumbersome. Firstly, the number of accepting configurations of V' for a random-pad $(\overline{c}, i)$ is either $\lceil M/2^k \rceil$ or $\lfloor M/2^k \rfloor$, where M is the number of accepting configurations of V on random-pad $\overline{c}$. Thus, in the worse-case the number of accepting configurations for V' (on random-pad $(\overline{c}, i)$) is $\lceil M/2^k \rceil$ and it follows that the free-bit complexity of V' is $\log_2 \lceil 2^f/2^k \rceil = f - k$. Furthermore, the expected number of accepting configurations (for a fixed $\overline{c}$ and uniformly chosen $i \in [2^k]$) is exactly $M/2^k$ (even if $M < 2^k$) and so the free-bit complexity of V' equals $f_{\mathrm{av}} - k$. Finally, observe that the argument regarding the acceptance probabilities remains unchanged (and actually it does not depend on the partition of the accepting configurations into 2^k non-empty parts). The proposition follows. ∎

We conclude with a transformation which reduces the free-bit complexity. Unlike Proposition 5.2.8, the following does not decrease the completeness parameters. Furthermore, the transformation increases the soundness parameter and does not preserve the gap (between the completeness and soundness parameters).

Proposition 5.2.9 (decreasing free-bit complexity): Let $c, s : \mathcal{Z}^+ \to [0,1]$ be admissible functions and $r, f, k : \mathcal{Z}^+ \to \mathcal{Z}^+$. Suppose $L \in \mathrm{FPCP}_{c,s}[r, f]$ with a verifier for which the first k oracle-answers for each random-pad allow at most 2^{f-k} accepting configurations. Then $L \in \mathrm{FPCP}_{c',s'}[r+k, f']$, where $c' = 1 - \frac{1-c}{2^k}$, $s' = 1 - \frac{1-s}{2^k}$, and $f' = \log_2(2^{f-k} + 2^k - 1)$.

The above can be further generalized; yet the current paper only utilizes the special case in which $c = 1$ (specifically, in the proof of Part 3.2 in Proposition 5.1.7, we use $f = 2$ and $k = 1$ obtaining $f' = \log_2 3$ and $c' = 1$ and $s' = \frac{1+s}{2}$).

Proof: The proof is similar to the proof of Proposition 5.2.8. Again, we consider a verifier V as guaranteed by the hypothesis and let A_i be the set of (at most 2^{f-k}) accepting configurations which are consistent with the i^{th} possibility of k oracle-answers to the first k queries. Denote the i^{th} possibility by α_i (i.e., all configurations in A_i start with α_i). We construct a new verifier, V', which uniformly selects a random-pad $\overline{c}$ for V and $i \in [2^k]$ (specifying a part A_i as above). The verifier V' makes the first k queries of V and if the answers differ from α_i then V' halts and accepts. Otherwise, V' continues the emulation of V and accepts iff V accepts.

Clearly, V' uses $r + k$ coin-tosses. The accepting configurations of v' on random-pad $(\overline{c}, i)$ are those in A_i as well as the "truncated V configurations" α_j, for $j \neq i$. Thus, there are $2^{f-k} + 2^k - 1$ accepting configurations. Suppose $V^\pi(x)$ accepts with probability p, then V' accepts input x with oracle access to π with probability $(1 - 2^{-k}) + 2^{-k} \cdot p = 1 - \frac{1-p}{2^k}$. The proposition follows. ∎

Bibliography

[Ad] L. Adleman. Two theorems on random polynomial time. *Proceedings of the Nineteenth Annual Symposium on the Foundations of Computer Science,* IEEE, 1978.

[AKS] M. Ajtai, J. Komlos and E. Szemeredi. Deterministic Simulation in Logspace. *Proceedings of the Nineteenth Annual Symposium on the Theory of Computing,* ACM, 1987.

[ASE] N. Alon, J. Spencer and P. Erdos. The Probabilistic Method. John Wiley and Sons, 1992.

[AmKa] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, vol. 147, pages 181–210, 1995.

[Ar] S. Arora. Reductions, Codes, PCPs and Inapproximability. Manuscript, May 1995.

[ABSS] S. Arora, L. Babai, J. Stern and Z. Sweedyk. The hardness of approximate optima in lattices, codes and linear equations. FOCS, 1993.

[ALMSS] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and intractability of approximation problems. *Proceedings of the Thirty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1992.

[ArSa] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Proceedings of the Thirty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1992.

[Bab] L. Babai. Trading Group Theory for Randomness. *Proceedings of the Seventeenth Annual Symposium on the Theory of Computing,* ACM, 1985.

[BFL] L. Babai, L. Fortnow and C. Lund. Non-deterministic Exponential time has two-prover interactive protocols. *Proceedings of the Thirty First Annual Symposium on the Foundations of Computer Science,* IEEE, 1990.

[BFLS] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing,* ACM, 1991.

[BaEv1] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. In *Jour. of Algorithms* Vol. 2, 1981, pages 198–201.

[BaEv] R. Bar-Yehuda and S. Even. A local ratio theorem for approximating the weighted vertex cover problem. In *Analysis and Design of Algorithms for Combinatorial Problems* Vol. 25 of Annals of Discrete Math, Elsevier, 1985.

[BaMo] R. Bar-Yehuda and S. Moran. On approximation problems related to the independent set and vertex cover problems. *Discrete Applied Mathematics* Vol. 9, 1984, pages 1–10.

[Be] M. Bellare. Interactive proofs and approximation: reductions from two provers in one round. *Proceedings of the Second Israel Symposium on Theory and Computing Systems,* 1993.

[BCHKS] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi and M. Sudan. Linearity testing in characteristic two. Manuscript, November 1994.

[BGG] M. Bellare, O. Goldreich and S. Goldwasser. Randomness in interactive proofs. *Proceedings of the Thirty First Annual Symposium on the Foundations of Computer Science,* IEEE, 1990.

[BGS] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability — Towards Tight Results (Version 2). August 1995. TR95-024 of ECCC, the *Electronic Colloquium on Computational Complexity,* `http://www.eccc.uni-trier.de/eccc/`.

[BGLR] M. Bellare, S. Goldwasser, C. Lund and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing,* ACM, 1993. (See also Errata sheet in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994).

[BeRo] M. Bellare and P. Rogaway. The complexity of approximating a quadratic program. *Complexity of Numerical Optimization*, ed. P. M. Pardalos, World Scientific, 1993.

[BeSu] M. Bellare and M. Sudan. Improved non-approximability results. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[BGKW] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover interactive proofs: How to remove intractability assumptions. *Proceedings of the Twentieth Annual Symposium on the Theory of Computing,* ACM, 1988.

[BeSc] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation* **96**, 77–94 (1992).

[Bl] A. Blum. Algorithms for approximate graph coloring. Ph. D Thesis, Dept. of Computer Science, MIT, 1991.

[BLR] M. Blum, M. Luby and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences* Vol. 47, pp. 549–595, 1993.

[BrNa] J. Bruck and M. Naor. The hardness of decoding with preprocessing. *IEEE Transactions on Information Theory*, Vol. 36, No. 2, pp. 381–385, 1990.

[BoHa] R. Boppana and M. Haldórsson. Approximating maximum independent sets by excluding subgraphs. BIT, Vol. 32, No. 2, 1992.

[CrKa] P. Crescenzi and V. Kann, A compendium of NP optimization problems. Technical Report, Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", SI/RR-95/02, 1995. The list is updated continuously. The latest version is available by anonymous ftp from `nada.kth.se` as `Theory/Viggo-Kann/compendium.ps.Z`.

[CST] P. Crescenzi, R. Silvestri and L. Trevisan. To Weight or not to Weight: Where is the Question? Manuscript, October 1995.

[CW] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science,* IEEE, 1989.

[Co] S. A. Cook. The Complexity of Theorem-Proving Procedures. *Proceedings of the Third Annual Symposium on the Theory of Computing,* ACM, 1971.

[EIS] S. Even, A. Itai and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. on Computing* Vol. 5, 691–703, 1976.

[ESY] S. Even, A. Selman and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control* Vol. 2, 159–173, 1984.

[Fei] U. Feige. Randomized graph products, chromatic numbers, and the Lovász theta function. *Proceedings of the Twenty Seventh Annual Symposium on the Theory of Computing,* ACM, 1995.

[FeGo] U. Feige and M. Goemans. Approximating the value of two prover proof systems, with application to Max-2SAT and Max-DICUT. *Proceedings of the Third Israel Symposium on Theory and Computing Systems*, IEEE, 1995.

[FGLSS] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. *Proceedings of the Thirty Second Annual Symposium on the Foundations of Computer Science,* IEEE, 1991.

[FeKi] U. Feige and J. Kilian. Two prover protocols – Low error at affordable rates. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994.

[FeLo] U. Feige and L. Lovász. Two-prover one round proof systems: Their power and their problems. *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing,* ACM, 1992.

[FRS] L. Fortnow, J. Rompel and M. Sipser. On the power of multiprover interactive protocols. Proceedings of the 3rd Structures, IEEE, 1988.

[Fu] M. Furer. Improved hardness results for approximating the chromatic number. Manuscript, 1994.

[FGMSZ] M. Furer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On Completeness and Soundness in Interactive Proof Systems. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pp. 429–442, 1989.

[GJ1] M. Garey and D. Johnson. The complexity of near optimal graph coloring. *Journal of the ACM* Vol. 23, No. 1, 43–49, 1976.

[GJ2] M. Garey and D. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, 1979.

[GJS] M. Garey, D. Johnson and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science* **1**, pp. 237–267, 1976.

[GoWi1] M. Goemans and D. Williamson. New 3/4-approximation algorithm for MAX SAT. *Proceedings of the 3rd Mathematical Programming Society Conference on Integer Programming and Combinatorial Optimization*, 1993.

[GoWi2] M. Goemans and D. Williamson. .878 approximation algorithms for Max-CUT and Max-2SAT. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[GMW] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. *Proceedings of the Twenty Seventh Annual Symposium on the Foundations of Computer Science,* IEEE, 1986.

[GMR] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proofs. *SIAM J. Computing* Vol 18, No. 1, 186–208, 1989.

[GS] S. Goldwasser and M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. *Proceedings of the Eighteenth Annual Symposium on the Theory of Computing,* ACM, 1986.

[Has] J. Håstad. private communication, September 1995.

[Hoc] D. Hochbaum. Efficient algorithms for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, Vol 6, pages 243–254, 1983.

[ImZu] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science*, IEEE, 1989.

[JLL] N.D. Jones, Y.E. Lien and W.T. Laaser. New problems complete for nondeterministic log space. *Math. Systems Theory* Vol. 10, 1976, pages 1–17.

[Kah] N. Kahale. On the second eigenvalue and linear expansion of regular graphs. *Proceedings of the Thirty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1992.

[KKLP] V. Kann, S. Khanna, J. Lagergren and A. Panconesi. On the hardness of approximating MAX k-CUT and its dual. Technical Report of the Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, TRITA-NA-P9505, 1995.

[KMS] D. Karger, R. Motwani and M. Sudan. Approximate graph coloring by semidefinite programming. *Proceedings of the Thirty Fifth Annual Symposium on the Foundations of Computer Science,* IEEE, 1994.

[Ka] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations,* Miller and Thatcher (eds.), Plenum Press, New York (1972).

[KLS] S. Khanna, N. Linial and S. Safra. On the hardness of approximating the chromatic number. *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, 1993.

[LaSh] D. Lapidot and A. Shamir. Fully Parallelized Multi-prover protocols for NEXP-time. *Proceedings of the Thirty Second Annual Symposium on the Foundations of Computer Science,* IEEE, 1991.

[Lau] C. Lautemann. BPP and the Polynomial Hierarchy. *Information Processing Letters*, Vol. 17 (4), pages 215–217, 1983.

[Lev] L.A. Levin. Universal'nyĭe perebornyĭe zadachi (universal search problems : in russian). *Problemy Peredachi Informatsii*, 9 (3), pages 265–266, 1973.

[LPS] A. Lubotzky, R. Phillips and P. Sarnak. Explicit Expanders and the Ramanujan Conjectures. *Proceedings of the Eighteenth Annual Symposium on the Theory of Computing,* ACM, 1986.

[LuYa] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, vol. 41, pages 960–981, 1994.

[LFKN] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *Proceedings of the Thirty First Annual Symposium on the Foundations of Computer Science,* IEEE, 1990.

[MaSl] F. MacWilliams and N. Sloane. The Theory of Error-Correcting Codes. North-Holland, 1981.

[MoRa] R. Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, 1995.

[MoSp] Monien and Speckenmeyer. Some further approximation algorithms for the vertex cover problem. *Proceedings of CAAP 83*, Lecure Notes in Computer Science Vol. 159, Springer-Verlag, 1983.

[PaYa] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences* **43**, pp. 425–440, 1991.

[Pet] E. Petrank. The Hardness of Approximations: Gap Location. TR–754, Department of Computer Science, Technion – Israel Institute of Technology, 1992.

[PoSp] A. Polishchuk and D. Spielman. Nearly-linear size holographic proofs. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994.

[Raz] R. Raz. A parallel repetition theorem. *Proceedings of the Twenty Seventh Annual Symposium on the Theory of Computing*, ACM, 1995.

[SaGo] S. Sahni and T. Gonzales. P-complete approximation problems. *J. of the ACM*, 23:555–565, 1976.

[Ta] G. Tardos. Multi-prover encoding schemes and three prover proof systems. *Proceedings of the Ninth Annual Conference on Structure in Complexity Theory*, IEEE, 1994.

[Sh] A. Shamir. IP=PSPACE. *Proceedings of the Thirty First Annual Symposium on the Foundations of Computer Science*, IEEE, 1990.

[SSTW] G. Sorkin, M. Sudan, L. Trevisan and D. Williamson. In preparation. 1995.

[Ya] M. Yannakakis, On the approximation of maximum satisfiability. *Journal of Algorithms*, vol. 17, pages 475–502, 1994.

[Zu] D. Zuckerman. NP-Complete Problems have a version that is hard to Approximate. *Proceedings of the Eighth Annual Conference on Structure in Complexity Theory*, IEEE, 1993.