Reductions, Codes, PCPs, and Inapproximability

Sanjeev Arora*
Princeton University

Abstract

Many recent results show the hardness of approximating NP-hard functions. We formalize, in a very simple way, what these results involve: a code-like Levin reduction.

Assuming a well-known complexity assumption, we show that such reductions cannot prove the NP-hardness of the following problems, where ϵ is any positive fraction: (i) achieving an approximation ratio $n^{1/2+\epsilon}$ for Clique, (ii) achieving an approximation ratio $1.5+\epsilon$ for Vertex Cover, and (iii) coloring a 3-colorable graph with $O(\log n)$ colors. In fact, we explain why current reductions cannot prove the NP-hardness of coloring 3-colorable graphs with 9 colors.

Our formalization of a code-like reduction, together with our justification of why such reductions are natural, also clarifies why current proofs of inapproximability results use error-correcting codes.

1 Introduction

Combinatorial optimization arises naturally in computational tasks. Computing optimum solutions in many cases is NP-hard ([Coo71, Kar72, Lev73]). A recent body of results shows that computing even approximately optimum solutions for problems such as Clique, Chromatic Number, Set Cover, etc., is NP-hard. These results, which rely upon new probabilistic characterizations of NP, have greatly increased our understanding of the complexity of approximation.

There has been great progress in nailing down the exact approximation ratio for which approximation becomes hard. For instance, in the case of Clique, this ratio has gone from $2^{\log^{1-\delta} n}$ to $n^{1/3+\epsilon}$ in four years [FGL+91, AS92, ALM+92, BGLR93, FK94, BS94, BGS95]. Chromatic Number has seen similar progress [LY93b, KLS93, BS94, BGS95, F$\ddot{9}$4]. In the case of Set Cover, we now even know a *threshold result.* Feige[Fei95b], improving upon the work of [LY93b, BGLR93], recently showed that achieving a ratio $(1-\epsilon)\ln n$ for Set Cover is hard for any fixed $\epsilon > 0$, whereas a well-known polynomial-time algorithm ([Joh74, Lov75]) achieves a ratio $1+\ln n$.

But Set Cover is an anomaly; threshold results for most interesting optimization problems remain out of reach. For instance, the provably hard ratio for Clique is $\approx \sqrt[3]{n}$, which is far below the ratio $O(n/\log^2 n)$ achieved by the best polynomial-time approximation algorithm. (Furthermore, the hopes of improving the algorithmic result for Clique — to achieve, say, a ratio $n^{1-\epsilon}$ for some fixed $\epsilon > 0$ — were recently dashed by the results in [Fei95a], which show that the Lovász θ function does not help in this task.) As Table 1 shows, a similar gap between algorithmic results and hardness results exists for Vertex Cover and Chromatic Number. (Further, most MAX-SNP hard problems [PY91] display a gap similar to Vertex Cover's.)

But given the ingeniousness of the constructions used already in the hardness results, there might seem no *a priori* reason why further improvements could not lead to threshold results for these problems, particularly in view of Feige's threshold result for Set Cover.

This paper attempts to quantify the limitations of recent techniques used in proving inapproximability results. We identify a technique that is central to all the results: a type of reduction that we name a *code-like Levin reduction.* The definition of this reduction is elementary, and unlike the inapproximability results themselves, does not involve polynomials, probabilistic proof checking, etc.. Furthermore, the definition conforms to intuitions about reductions developed over the past couple of decades.

We also show, assuming the well-known conjecture NP $\neq$ *few*P, that code-like Levin reductions cannot prove the NP-hardness of the following problems for any $\epsilon > 0$: (i) achieving an approximation ratio $n^{0.5+\epsilon}$ for Clique, (ii) coloring a 3-colorable graph with $O(\log n)$ colors, and (iii) achieving a ratio $1.5+\epsilon$ for Vertex Cover.

The impossibility result for Clique also extends — although we don't discuss it in this abstract — with almost no modification to maximum subgraph problems discussed in [LY93a]. We suspect that it will extend to many other problems. On the other hand, there *are* problems for which code-like reductions suffice to prove threshold results. For instance, Feige's result on Set-Cover uses such reductions. Thus our impossibility results depend in an essential way on the combinatorial structure of the problem at hand.

By showing that code-like Levin reductions conform to standard intuitions about reductions, this paper also hopes to clarify the issue of why recent inapproximability results use error-correcting codes. Such codes play an important role in proving new probabilistic characterizations of NP, such as NP = PCP($\log n$, 1) [AS92, ALM+92], which underlie recent inapproximability results. No formal justification had earlier been given for why (or if) this coding-theoretic approach is a *natural* way to prove inapproximability results. We provide the following justification: in-

*e-mail: arora@cs.princeton.edu. Research supported by NSF CAREER award CCR-9502747.

Problem	Performance of best poly-time algorithm	Best Hardness Result	Limits of Code-like Reductions (this paper)
Clique	$O(n/\log^2 n)$[BH92]	$n^{1/3-\epsilon}$ [BGS95]	$n^{1/2+\epsilon}$
Chromatic Number	$O(n/\log^3 n)$ [BH92]	$n^{1/5-\epsilon}$ [F94]	$n^{1/2+\epsilon}$
Coloring 3-colorable graphs	Uses $O(n^{0.25})$ colors [KMS94]	5 colors required	Cannot prove $O(\log n)$ colors are required
Vertex Cover	$2-o(1)$ [Hoc82]	≈ 1.01 ([PY91, BGS95])	$1.5+\epsilon$

Table 1: Summary of known approximation properties of some well-known problems. The last column shows hardness results that are beyond the reach of current techniques (as shown in this paper).

approximability results use codes to make reductions code-like.

Comparison with related work. We summarise here the work of Bellare, Goldreich and Sudan [BGS95], specifically, the part in which they study the inherent limitations of current techniques in the context of Clique. They describe their work in terms of *free bits*, but we state it in in the graph-theoretic language of this paper (the two terminologies are equivalent; see Section 4.1). The authors observe that current reductions produce a special clique instance — an r-partite graph, for some integer r, whose clique number is either r or r/β, for some gap β. They consider whether this feature inherently handicaps the reduction. Their conclusion is that it does not, since they can modify *every* instance of clique, in an approximation-preserving fashion, to such a special instance. As a consequence, they can modify *every* clique reduction to produce only these special instances. The authors also show that the techniques of their paper (which are representative of the techniques used in other papers too) cannot make $\beta > n^{1/3}$.

Our characterization of "current reductions" goes somewhat deeper. We observe that they produce graphs with superpolynomially many large cliques with small pair-wise intersections. By analyzing such graphs, we arrive at the opposite conclusion: current reductions do inherently handicap themselves. For instance, we can show that they cannot reduce the *free bit* parameter of PCP verifiers below 1.

Also, our ideas extend to problems other than clique.

1.1 Overview

Strictly speaking, an NP-completeness reduction (by which we will mean the usual *many-one* reduction) between two languages L_1 and L_2 is required to do very little: map members of L_1 to members of L_2, and non-members to non-members. But in actual practice, the proof of the reduction's correctness ends up doing more: it defines a polynomial-time map from witnesses to witnesses. We call reductions with such associated maps *Levin reductions*, since they were first explicitly defined in [Lev73]. (In Section 7.1 of the appendix we give a formal reason why known reductions are Levin.)

In fact, in all known reductions the forward map from witnesses to witnesses is injective: different witnesses map to different witnesses. For example, if the reduction is from SAT to Clique, then every two different satisfying assignments in a formula get mapped to distinct cliques. The reason for this *injectivity* property should be intuitively clear: the polynomial-time reduction, since it does not "know" enough about the satisfying assignments – or even whether one exists – cannot possibly coalesce different assignments into one witness. Section 2.1 gives a complexity-theoretic justification for this intuition.

When the reduction is trying to prove the hardness of approximation, the Levinness and injectivity properties get suitably generalized. The generalization of injectivity is particularly interesting. To illustrate, we use the current reduction to Clique as an example.

Example 1 To prove the hardness of achieving an approximation ratio β for Clique (currently $\beta = n^{1/3-\epsilon}$), the current reduction maps every boolean formula φ to a graph $f(\varphi)$ such that

$$\begin{aligned} \varphi \in \text{SAT} \quad &\Rightarrow \quad \omega(f(\varphi)) \geq K(|\varphi|) \\ \varphi \notin \text{SAT} \quad &\Rightarrow \quad \omega(f(\varphi)) < \frac{K(|\varphi|)}{\beta}, \end{aligned}$$

where ω denote the clique number and K is some function.

Note that even a clique of size $K(|\varphi|)/\beta$ in $f(\varphi)$ is a *witness* that φ is satisfiable. Thus Levin-ness means that there exists a polynomial-time mapping from satisfying assignments of φ to cliques of size $K(|\varphi|)$ in $f(\varphi)$, and from cliques of size $\geq K(|\varphi|)/\beta$ to satisfying assignments.

Note that the obvious generalization of the injectivity property should be: "A witness of satisfiability, in other words, a clique of size $K(|\varphi|)/\beta$, corresponds to a unique satisfying assignment." Current reductions ensure this; they ensure that every two cliques of size $K(|\varphi|)$ that are produced from two different satisfying assignments intersect in fewer than $K(|\varphi|)/\beta$ vertices. $\square$

The phenomenon of different satisfying assignments mapping to "well-separated" witnesses occurs in all recent inaproximability results. (Unlike in the case of Levinness and injectiveness, we cannot give any formal justification for this phenomenon besides the intuition already given in Example 1.) We call a Levin reduction with this property of "well-separated" witnesses a *code-like Levin reduction*, where the notion of "well-separated witnesses" is defined individually for the problem at hand. (Typically, well-separatedness means the witnesses have large symmetric difference.) Sections 3.1 to 3.3 define code-like reductions for Clique, Vertex Cover, and Chromatic Number, and Section 4.2 defines them for MAX-3SAT.

Note that code-like reductions produce very special instances of the optimization problem: many of the optimal solutions (cliques, vertex covers, colorings) are well-separated. Moreover, the well-known complexity conjecture NP $\neq$ fewP implies that the number of such solutions is superpolyomial in the worst case. Thus the reduction needs to pack a lot of well-separated solutions into the problem instance. Our impossibility results follow either by showing that the combinatorial structure of the problem forbids this packing (this happens in the case of Clique), or that it allows it only on instances that are easy cases for a polynomial-time approximation algorithm (this happens in the case of Vertex Cover).

Notes: *(i)* The reason current reductions are code-like is that they use the result NP = PCP$(\log n, 1)$, whose proof involves error-correcting codes; see Section 4. *(ii)* A reduction, by definition, must work correctly for *every* problem instance. Thus an impossibility result like ours only needs to argue that the reduction fails on *some* instance. This means we can prove our result under somewhat weaker assumptions (see Section 5). *(iii)* The discussion upto this point applies only to deterministic polynomial-time reductions. However, many current reductions run in quasi-polynomial time. Some are also *nonuniform.* (Their discoverers state them as randomized reductions, but it is easily checked in each case that the randomness can be hardwired into a circuit.) An obvious modification of the assumption NP $\neq$ fewP extends our impossibility results to nonuniform quasipolynomial-time reductions. True randomized reductions – those that cannot be viewed as circuits – do not currently appear in any inapproximability results. However, we indicate in the appendix how to extend our impossibility results to them, by using a complexity assumption related to NP $\neq$ fewP (but not as standard).

2 Levin Reductions

For convenience, we define Levin reductions only for maximization problems. An *NP maximization problem* consists of an integer-valued, polynomial-time function Value$(\cdot,\cdot)$. Given input x, the objective is to find a solution y, $|y| \leq |x|^c$, that maximizes Value(x, y). The integer $\max_y$ Value(x, y) is denoted by $OPT(x)$.

Definition 1 Let Π be an NP maximization problem. A *Levin reduction from SAT to* Π is a triple (f, g, h) of polynomial-time maps. Map f transforms instances of SAT to instances of Π. For every input size m, there are integers K, K' such that $K' < K$ and $\forall \varphi \in \{0,1\}$:

$\varphi \in \text{SAT} \Rightarrow OPT(f(\varphi)) \geq K.$

$\varphi \notin \text{SAT} \Rightarrow OPT(f(\varphi)) < K'.$

Maps g, h are called *witness transformations.* Map h is allowed to be partial, and it transforms satisfying assignments of φ into solutions (in instance $f(\varphi)$) of value at least K. Map g transforms solutions of value at least K' into satisfying assignments of φ.

The *gap* produced by the reduction, denoted β, is the ratio K/K' (measured as a function of the size of the new instance $f(\varphi)$).

Do reductions need to be Levin? Certainly, all those in [GJ79] are, as are all those used in recent inapproximability results. Typically, the Levin-ness is never demonstrated explicitly, but *falls out of the proof of the reduction's correctness.* As Section 7.1 of the Appendix shows, this is no coincidence, since the correctness of these reductions is proved using "constructive mathematics."

Note that a reduction could work in devious (that is, non-Levin) ways on instances of SAT that are solvable in polynomial time. Clearly, such reductions are easily modified to be Levin. Also, a reduction could use cryptographic primitives to hide witnesses and thus become non-Levin. But in all such examples we know of, the cryptographic primitive serves no useful pupose and can be removed without affecting the reduction'c correctness.

Proposition 1 *(i) All known NP-completeness reductions are either Levin or easily modified to be Levin. (ii) The forward witness transformation h in all of them is injective.*

2.1 Naturalness of the Injectiveness of h

We give a complexity-theoretic reason why the forward witness transformation h has to be injective, at least on worst-case inputs.

We define a new notion: *witness intractable functions.* Let $p = \{p_n\}_{n \in Z^+}$ be a family of functions computable by polynomial-size circuits, where $p_n : \{0,1\}^n \to \{0,1\}$. Function p is *witness intractable* if every polynomial-time algorithm fails to perform the following task for almost all n: Given an x such that $p(x) = 1$, find another y such that $p(y) = 1$. Witness intractable functions exist if one-way functions exist. The reason is that one-way functions can be used to construct signature schemes secure against chosen plaintext attack [R90]. For such a signature scheme and a public key k, let p_n be the polynomial-time algorithm that verifies, given x, whether it is a message that has been signed using key k. By definition of the signature scheme, p_n is witness intractable.

Lemma 2 *Let (f, g, h) be the reduction mentioned in Definition 1. If witness-intractable functions exist, then there exist formulae φ such that for every satisfying assignment s of φ, either $h(s)$ is not defined, or else $g(h(s)) = s$.*

In particular, for every two distinct satisfying assignments s_1, s_2 of φ, we have $h(s_1) \neq h(s_2)$.

Proof: Let p be a witness-intractable function. Use the Cook-Levin theorem to construct a boolean formula φ whose satisfying assignments are in one-to-one correspondence with the set of y such that $p_n(y) = 1$.

We claim that formula φ has the properties mentioned in the lemma statement. For, suppose assignment s satisfies φ. Then $g(h(s))$ is also a satisfying assignment that is constructible in polynomial time from s. The witness intractability of p implies that $g(h(s)) = s$. $\Box$

2.2 Relevance of NP $\neq$ fewP

Recall from Definition 1 that the forward witness transformation h is allowed to be a partial map. (Such a map occurs for example in the well-known Valiant-Vazirani reduction [VV86].) Thus h could conceivably be defined only on very few (say, polynomially many) satisfying assignments. Now we show that if the conjecture NP $\neq$ few P is true, then this cannot always happen.

Allender's class few P [AR88] contains every language that is accepted by an NP machine with at most polynomially many accepting branches. (This class is a generalization of Valiant's clas s UP [Val76], which contains every language that is accepted by an NP machine with at most one accepting branch.) It is conjectured that NP $\neq$ fewP. Notice, P $\subseteq$ fewP, so NP $\neq$ fewP $\Rightarrow$ P $\neq$ NP.

Proposition 3 *Let (f, g, h) be the Levin reduction in Definition 1. If NP $\neq$ fewP, there is a satisfiable formula φ such that superpolynomially many satisfying assignments of φ have images under h.*

Proof: For, if not, then the following reduction reduces SAT to a language in fewP (thus implying NP = fewP): Given a boolean formula ψ, use the Cook-Levin theorem to construct a boolean formula whose satisfying assignments are in one-to-one correspondence with those satisfying assignments of ψ that have an image under h. $\Box$

The following corollary of Proposition 3 shows how to construct the formula φ so that it is in addition a hard case for any given polynomial-time heuristic for SAT. The corollary will be useful in Section 3.3.

Corollary 4 *Let (f, g, h) be the Levin reduction in Proposition 3, and p be any polynomial-time heuristic for SAT. If NP $\neq$ fewP, then there exists a satisfiable formula φ with the same property as in Proposition 3, and on which p gives the wrong answer.* $\Box$

3 Code-like Reductions and their limitations

This section defines code-like reductions and exhibits their limitations in proving good inapproximability results. The following lemma, a folk-theorem in combinatorics, is crucial.

Lemma 5 (Packing Lemma) *For any positive integer n and fraction ρ, let $S_1, S_2, \ldots,$ be any N subsets of $[1..n]$, where each $|S_i| \geq \rho n$ and the number of subsets $N > 2n/\rho$. Then some i, j satisfy: $|S_i \cap S_j| \geq \rho^2 n$.*

Proof: (sketch) By contradiction. Pick one of the n elements in the set randomly, and denote by X the number of sets that contain it. If every pair of sets intersect in fewer than $\rho^2 n$ elements and $N \geq 2n/\rho$, then X has negative variance. $\Box$

3.1 Clique

Let (f, g, h) be a Levin reduction from SAT to Clique that proves the hardness of achieving a ratio β for Clique. Suppose it maps satisfiable formulae to graphs with Clique number at least ρn, and unsatisfiable formula to graphs with clique number less than $\rho n/\beta$, where n is the number of vertices in the graph and ρ, β may depend upon n.

Definition 2 The reduction in the previous paragraph is *code-like* if, for every formula φ and every two distinct satisfying assignments s_1, s_2 for which $h(s_1)$ and $h(s_2)$ are both defined, we have

$$|h(s_1) \cap h(s_2)| < \frac{\rho n}{\beta} \qquad (1)$$

(i.e., the intersection of the two cliques is smaller than the clique number in the case the formula is unsatisfiable).

Theorem 6 *Assuming NP $\neq$ fewP, the following are true for the code-like reduction described in the previous paragraphs: (i) $\beta < 1/\rho$. In other words, the reduction maps some unsatisfiable formula to a graph with clique number at least $\rho^2 n$. (ii) $\beta \leq n^{0.5+\epsilon}$ for every $\epsilon > 0$.*

Proof: (i) If NP $\neq$ fewP, Proposition 3 implies that there is some satisfiable formula in which superpolynomially many satisfying assignments map to cliques of size ρn. The packing lemma implies that some pair of cliques intersect in at least $\rho^2 n$ vertices. If the reduction is code-like, this size should be less than the clique number in the case the formula is unsatisfiable. Hence some unsatisfiable formula must map to a graph with clique number $\geq \rho^2 n$. (ii) Note that for β to be $n^{0.5+\epsilon}$, ρ must be $\Omega(n^{-\frac{1}{2}+\epsilon})$. But then part (i) restricts the gap β to $1/\rho = O(n^{\frac{1}{2}-\epsilon})$. $\Box$

3.2 Coloring

First we note, in light of Theorem 6, why current reductions to Chromatic Number cannot be expected to prove the hardness of coloring a 3 colorable graph with 9 colors. The reason is [LY93b, KLS93, F$\ddot{9}$4] that they are actually reductions to independent set (to which Theorem 6 also applies). They use the following fact: If $\chi(G), \alpha(G)$ denote, respectively, the chromatic number and independent set number of graph G, then $\chi(G) \geq |V(G)|/\alpha(G)$.

To show the hardness of coloring a K-colorable graph with βK colors (for any $\beta > 1$), these reductions map satisfiable formulae to graphs with $\chi(G) \leq K$ (note that $\alpha(G) \geq n/K$ in this case) and unsatisfiable formulae to graphs with $\alpha(G) < n/(\beta K)$ (note that in this case $\chi(G) > \beta K$). Now if the reduction to independent set is code-like, and NP $\neq few$P, then Theorem 6 implies that $\beta \leq K$. Hence such reductions cannot show the hardness of coloring 3 colorable graphs with 9 colors.

Now we consider a more general class of reductions (of which no examples are known) and show that so long as they conform to the intuitions about reductions mentioned earlier in Example 1 (i.e., witnesses corresponding to unique witnesses), they are also very limited. To motivate this result, we recall a folk-theorem (cf. [Blu91]): Any reduction that shows the hardness of coloring a K-colorable graph with $K^2 \log n$ colors also shows the hardness of finding, in a graph with $\alpha \geq n/K$, an independent set of size n/K^2. To prove the folk-theorem, assume there is an algorithm for the above-mentioned subcase of independent set. Since a K-colorable graph has an independent set of size n/K, we can use the algorithm to find an independent set of size n/K^2, use up a color on it, remove it from the graph, and continue. This way, we will color the graph with t colors where $(1-1/K^2)^t < 1/n$. Thus $K^2 \log n$ colors suffice.

The folk-theorem motivates the following definition. A K^2*-sequence* in a graph on n vertices is a sequence $\sigma = (S_1, S_2, \ldots,)$, where each S_i is an independent set of size n_i/K^2 in the graph obtained by removing $S_1, \ldots, S_{i-1}$ from G, and $n_i = n - \sum_{j<i} |S_i|$. (Formally, we should use $n_i(\sigma)$ instead of n_i, but σ will be understood from context.) The *length* of the sequence is the number of sets in it.

Suppose some Levin reduction maps satisfiable formulae to graphs with $\chi \leq K$ and unsatisfiable ones to graphs with $\chi > K^2 \log n$. Let j_{max} denote the maximum length of a K^2-sequence in any graph generated using an unsatisfiable formula. As already noted, $j_{max} < K^2 \log n$ if $\chi > K^2 \log n$. Thus a K^2-sequence of length j_{max} in the graph is not a witness that the formula was satisfiable, whereas a K^2-sequence of size $j_{max}+1$ is.

Since the reduction is Levin, some of the satisfying assignments gives rise to a K-coloring. Thus in fact there is a polynomial-time way, given a K^2-sequence $(S_1, \ldots, S_{j_{max}})$ and such a satisfying assignment s, to generate an independent set of size $n_{j_{max}+1}/K$ in the graph $G \setminus \cup_{i \leq j_{max}} S_i$. Let $h(\sigma, s)$ denote this independent set (Note that there might be more than one way to pick this independent set since there are K colors; assume for simplicity that $h(\sigma, s)$ is picked randomly from among these choices.)

Note, however, that even an independent set of size $n_{j_{max}+1}/K^2$ in $G \setminus \cup_{i \leq j_{max}} S_i$ is a witness of satisfiability. The following definition is therefore quite natural, given the intuition in Example 1.

Definition 3 The reduction described in the previous paragraphs is *code-like* if the map $h(.,.)$ described above satisfies, for every K^2-sequence σ and every two satisfying assignments s_1, s_2,

$$h(\sigma, s_1) \cap h(\sigma, s_2) \leq n_{j_{max}+1}/K^2.$$

Theorem 7 *If NP $\neq few$P, then the code-like reduction to Chromatic Number described above does not exist.*

Proof: (sketch) NP $\neq few$P implies that there is a formula in which superpolynomially many satisfying assignments map to different K-colorings. A little thought shows that in fact this is true for *partial K-colorings.* A *partial coloring* of the graph is a coloring of $n - (1-1/K)n_{j_{max}+1}$ vertices of the graph using at most K independent sets each of size at least $n_{j_{max}+1}/K$. (Note that even a partial coloring is a witness of satisfiability.)

Now we show – via a probabilistic argument – that there exists a K^2-sequence σ such that the set $\{h(\sigma, s) : s \text{ a satisfying assignment}\}$ has superpolynomial size. Pick an assignment randomly from among those that map to distinct K-colorings. This assignment gives us a partial coloring. Now pick a K^2-sequence using this partial coloring, randomly picking an independent set of the appropriate size at each step. If two assignments s_1, s_2 mapped to different partial colorings, the probability is at least $1/\text{poly}(n)$ that $h(\sigma, s_1) \neq h(\sigma, s_2)$.

Hence the expected size of the set $\{h(\sigma, s) : s \text{ a satisfying assignment}\}$ is superpolynomial. In particular, there exists a K^2-sequence σ for which this size is superpolynomial.

□

Thus proving the hardness of coloring a K-colorable graph with $K^2 \log n$ colors seems difficult with current techniques.

3.3 Vertex Cover

Note that the complement of a vertex cover is an independent set. Thus a reduction to vertex cover is also a reduction to independent set. We define a reduction to vertex cover to be *code-like* if the associated reduction to independent set is code-like. Known reductions are code-like according to this definition.

Suppose some reduction proves the hardness of approximating $VC_{\min}$ within a factor $1.5 + \epsilon$ for some fixed $\epsilon > 0$. In other words, for some fraction c, it maps formulae to graphs with $VC_{\min}$ either $\leq cn$ or $\geq (1.5+\epsilon)cn$, depending upon whether or not the formula is satisfiable. So α is either $\geq n(1-c)$ or $< n(1-(1.5+\epsilon)c)$. We will show – assuming NP $\neq few$P – that the reduction is not code-like.

First assume $c \geq 1/2$. If NP $\neq few$P, Theorem 6 implies that there exist formulae such that the independent sets produced from some pair of satisfying assignments intersect in a set of size $\geq (1-c)^2 n$, which is more than $n(1-(1.5+\epsilon)c)$. Hence the reduction is not code-like.

The proof for the case $c \leq 1/2$ is more complicated, and in this abstract we give a simpler version of it by allowing the constant 1.6 instead of 1.5. Let $k = 1.6+\delta$ for any $\delta > 0$. In what follows, we express the sizes of independent sets and matchings as fractions of the number of vertices in the graph.

Claim: If a graph has a matching with $\geq kc$ vertices, then among every $O(1/kc)$ independent sets of size

$1 - c$, there are two that have an intersection of size more than $1 - kc$.

Proof: Let S be the set of vertices in the matching. Assume (wlog, since the remaining argument is only helped by the presence of a larger matching) that $|S| = kc$. Note that every independent set intersects S in at most $kc/2$ vertices. Assume (wlog, since the remaining argument is only helped otherwise) that each of the independent sets mentioned in the hypothesis intersects S in a set of size $kc/2$.

The Packing Lemma implies there are two independent sets I_1 and I_2 such that $I_1 \cap S$ and $I_2 \cap S$ intersect in $\geq \frac{1}{2^2} kc$ vertices. Furthermore, $I_1 \setminus S$ and $I_2 \setminus S$ each have size $1 - c - kc/2$, and $\overline{S}$ has size $1 - kc$. So $I_1 \setminus S$ and $I_2 \setminus S$ must intersect in $\geq 1 - 2c$ vertices. Hence $|I_1 \cap I_2| \geq 1 - 2c + kc/4$. For the value of k we chose, this is $> 1 - kc$. $\square$

Now assume the reduction maps every unsatisfiable formula to a graph with $\alpha < 1 - kc$. Note that a maximum matching in the resulting graph is computable in polynomial-time, and vertices appearing in it constitute a vertex cover. Thus if the maximum matching contains $< kc$ vertices, then the formula is clearly recognized in polynomial time as satisfiable. So Corollary 4 implies that if NP $\neq$ *few*P, then there is a formula in which superpolynomially many satisfying assignments map to independent sets, and in addition the maximum matching has $\geq kc$ vertices. Our claim implies that some pair of independent sets (corresponding to different satisfying assignments) have an intersection of size $> 1 - kc$. Hence the reduction is not code-like.

4 Code-like Reductions v/s Codes

This section contains results and observations that clarify the connection between code-like reductions and codes. On the one hand, Section 4.1 shows that code-like reductions for clique can be viewed as giving rise to graphs in which large cliques form an error-correcting code over a certain alphabet. (This explains our choice of the term "code-like.") On the other hand, Section 4.2 shows that current reductions for Clique (and other problems) are code-like precisely because the result NP = PCP($\log n$, 1) uses error-correcting codes. Finally, Section 4.3 formalizes *strongly Levin* reduction, and shows that if we restrict ourselves to such reductions, a proof of NP = PCP($\log n$, 1) necessarily leads to constructions of error-correcting codes. Since known techniques produce strongly Levin reductions, this result helps explain why current proofs of NP = PCP($\log n$, 1) use error-correcting codes.

4.1 How many free bits?

The *free bit* parameter is an important parameter associated with probabilistic verifiers that check membership proofs for SAT. This parameter was first defined in [FK94], and then refined in [BS94, BGS95]. Reducing the parameter to a real number b is equivalent to giving a sequence of reductions from SAT to the promise problem *RCLIQUE*(b, ϵ), one for every $\epsilon > 0$. The problem *RCLIQUE*(b, ϵ) is defined next. (Note that one side of the equivalence between free bits and reductions to *RCLIQUE* follows from the definition of free bits and a now-standard reduction in [FGL+91]. The other side is an easy subcase of a more general transformation in [BGS95].)

Definition 4 *For $b, \epsilon > 0$, the* RCLIQUE*(b, ϵ) problem is a subcase of the clique problem defined by the following conditions on the input: (i) the graph has $K^{1+b+\epsilon}$ vertices for some K. (ii) the vertices can be partitioned into $N = K^{1+\epsilon}$ disjoint independent sets of size K^b, and this partition is given with the input. (iii) The clique number is guaranteed to be either exactly N or $< N/K$.*

(Note: [BGS95] define a variant of *RCLIQUE* in which the clique number in the first case is $N(1 - 1/K)$ instead of N. This variant corresponds to their verifier with *imperfect completeness*, and modifying the proof below for their variant is easy.)

Note that if SAT has a verifier that uses b free bits, then *RCLIQUE*(b, ϵ) is NP-hard, which in turn implies the NP-hardness of approximating the clique number of n-vertex graphs within a factor $n^{\frac{1}{1+b+\epsilon}}$. Thus a corollary of our Theorem 6 is: *If NP $\neq$ few P reducing bit parameter below* 1 *necessitates a reduction that is not code-like Levin.* Next we outline a more direct coding-theoretic proof of this statement.

Consider an instance of RCLIQUE(b, ϵ) with $K^{b+1+\epsilon}$ vertices that are divided into $N = K^{1+\epsilon}$ independent sets. Since a clique and an independent set intersect in at most one vertex, a clique of size N contains exactly one vertex from each of the N independent sets. Thus a clique of size N can be viewed as a *word* of length N over the alphabet $[1, K^b]$. (The ith symbol in the word specifies the element from the ith set present in the clique.) Further, if two such cliques intersect in a set of size ρN, the words representing them have hamming distance $N(1 - \rho)$.

Now consider a code-like Levin reduction from SAT to RCLIQUE(b, ϵ). By definition of code-like, every two cliques of size N that represent different satisfying assignments have an intersection of size less than N/K. Thus the words representing them have hamming distance at least $N(1 - 1/K)$. (In other words, the set of words corresponding to different satisfying assignments forms an *an error-correcting code* over the alphabet $[1, K^b]$ with minimum distance $N(1 - 1/K)$.)

Now if NP $\neq$ *few* P, the number of words in this error-correcting code is superpolynomial in the worst case (Proposition 3). The next fact implies that such a large code must satisfy $K^b \geq N$, which implies $b \geq 1 + \epsilon$. This finishes the proof.

Fact:[vL91]*For any $b < 1$ there can be at most $O(K)$ words of length N over the alphabet $[1, K^b]$ whose pairwise distance is more than $N(1 - 1/K)$.*

Lastly, to justify our choice for the term "code-like reduction," we show — heavily using the main result of [BGS95] — that *every* code-like Levin reduction to Clique (and not just reductions that involve the free bit parameter) can be viewed as giving rise to graphs in which a large set of cliques form an error-correcting code.

Proposition 8 *Any code-like Levin reduction that proves the hardness of approximating Clique within a factor $n^{\frac{1}{1+b}}$ can be turned into a code-like Levin (probabilistic) reduction to RCLIQUE($b-\epsilon,\epsilon$) for every $\epsilon > 0$.*

The proof of the proposition involves using the [BGS95] transformation, and observing that it preserves the code-like property. In general, all transformations which involve graph products seem to preserve the code-like property; see Section 4.2.1.

4.2 Why are current reductions code-like?

Existing inapproximability results are code-like because they all follow[1] (see e.g., [AL95]) in a fairly simple way — sometimes using Raz's parallel repetition theorem [Raz94] — from the statement NP = PCP($\log n, 1$). And the only known proof of NP = PCP($\log n, 1$) has a "code-like" nature, as defined below. Then in Section 4.2.1 we illustrate, using Clique as an example, how this code-like nature then becomes part of other inapproximability results.

The statement NP = PCP($\log n, 1$) is equivalent [ALM+92] to the existence of a polynomial-time reduction f from SAT to MAX-3SAT that, for some fixed constant $\epsilon > 0$, behaves as follows.

$$\begin{aligned} \varphi \in \text{SAT} &\Rightarrow \text{MAX-3SAT}(f(\varphi)) = 1 \\ \varphi \notin \text{SAT} &\Rightarrow \text{MAX-3SAT}(f(\varphi)) < 1-\epsilon. \end{aligned} \quad (2)$$

(Here MAX-3SAT($f(\varphi)$) denotes the maximum fraction of clauses in φ that can be simultaneously satisfied.) Currently this reduction is Levin.

Definition 5 Let f be a Levin reduction satisfying the property in (2), φ be any boolean formula, and h be its forward witness transformation. Reduction f is *code-like* if for every two satisfying assignments s_1, s_2 of φ that have images under h, we have: $h(s_1)$ and $h(s_2)$ satisfy different literals in at least ϵ fraction of clauses in $f(\varphi)$.

Notes: (i) The reader can check (reasoning along the lines of Example 1) that such reductions are quite "natural." (ii) The "code-like" has a more intuitive restatement once we transform the 3CNF formula [PY91] such that every variable occurs exactly 5 times. Then there is some $\epsilon' > 0$ such that $h(s_1)$ and $h(s_2)$ differ in ϵ' fraction of the variables.

An examination of the proof of NP = PCP($\log n, 1$) shows that it gives rise to a code-like reduction to MAX-3SAT. In fact,

[1] Note that using NP = PCP($\log n, 1$) just as a "black box" does not lead to the best possible inapproximability result for problems such as Clique, only to a result in the "right ballpark." Proving the best result has involved looking at the proof of NP = PCP($\log n, 1$); see e.g. [BS94, BGS95].

4.2.1 The Clique Reduction

Now we explain how to prove the inapproximability result for Clique, focussing upon how the code-like nature transfers from the MAX-3SAT reduction into the code-like of the Clique reduction. For simplicity, we only describe how to prove the hardness of approximating clique number within a factor n^{δ} for some fixed $\delta > 0$.

Let $f(\varphi)$ be the instance of MAX-3SAT obtained from the reduction in Equation 2. Reduce it to clique using the idea in [GJ79]: Produce a graph $f'(\varphi)$ containing a vertex for each literal appearing in $f(\varphi)$. Connect every pair of vertices with an edge, unless if they "contradict" each other (i.e., they represent some variable x_i and its negation $\overline{x}_i$), or belong to the same clause.

It is easily seen that picking a clique of size k in $f'(\varphi)$ corresponds to picking an assignment that satisfies k clauses in $f(\varphi)$. Thus for some l, either $\omega(f'(\varphi)) = l$ or $\omega(f'(\varphi)) < l(1-\epsilon)$, depending upon whether or not φ is satisfiable. Further, if f was a code-like reduction to MAX-3SAT, then f' is a code-like reduction to Clique: cliques of size l that correspond to different satisfying assignments of φ intersect in fewer than $l(1-\epsilon)$ vertices.

Now use "randomized graph-products" [BS89]: Pick poly(n) random subsets $S_1, S_2, \ldots$ of vertices in $f'(\varphi)$, where each set is of size $\log n$. (For convenience we describe the construction as randomized; it can be derandomized [AFWZ93].) Construct a graph $f''(\varphi)$ with one vertex for each subset. Connect S_i, S_j with an edge iff $S_i \cup S_j$ is a clique in $f'(\varphi)$. Thus if $S_{i_1}, S_{i_2}, \ldots,$ form a clique in this new graph, their union is a clique in $f'(\varphi)$.

A simple calculation shows that there is a correspondence between cliques in $f'(\varphi)$ and $f''(\varphi)$. However, in going from one graph to the next, the clique sizes scale up on an exponential scale. Thus if the ratio between the sizes of two cliques was ρ in $f'(\varphi)$, it is $\rho^{\log n}$ in $f''(\varphi)$. In particular, suppose cliques of size l in $f'(\varphi)$ map to cliques of size K (for some K) in $f''(\varphi)$. Then cliques of size $l(1-\epsilon)$ map to cliques of size $K \cdot (1-\epsilon)^{\log n}$, which is $K \cdot n^{-\delta}$ for some $\delta > 0$.

Further, it should be clear from the above description why f" is a code-like reduction from SAT to Clique. Our proof of correctness *explicitly* argued that *every* clique of size $< (1-\epsilon)l$ in graph $f'(\varphi)$ stays small in $f''(\varphi)$. But then, as an unintended side-effect of doing this, we end up ensuring that the intersection of any two cliques of size l (that correspond to different satisfying assignments) stays small, since that intersection is itself a clique of size $< (1-\epsilon)l$.

4.3 PCP Theorem: Are codes necessary?

Nothing in our formalization thus far implies that Levin reductions used in inapproximability results must necessarily be code-like. This may well be because our definition of Levin reductions was too heavily influenced by classical reductions, which prove the hardness of exact optimization and not approximation. Specifically, our definition of "witness transformation" allowed a witness to be mapped to only one witness. But in inapproximability results, this mapping is better viewed as a multivalued function. For example, in the Levin reduction in Example 1, a satisfying assignment maps to a clique of size $K(n)$. But the true witnesses are subcliques of size $K(n)/\beta$, of which there are many in a clique of size $K(n)$.

Because of such observations we define *strongly Levin reductions,* which allow witness transformations to be many-valued functions. Among the many possible definitions, we give the one that is general enough to capture all current reductions that we know of.

Definition 6 A *strongly Levin reduction* (f, g, h) is defined similarly as in Definition 1, except for the following differences (i)h *is multivalued:* If s is a satisfying assignment, $h(s)$ is a (possibly empty) set of solutions (in instance $f(\varphi)$) of value at least K', and the elements of $h(s)$ can be enumerated at a cost of polynomial time per element (ii)*invertibility:* h is invertible in the following strong sense: Given a solution y of value at least K' such that $y \in \cap_{s \in S} h(s)$ for some set S of satisfying assignments (i.e., y is in the image set of one or more satisfying assignment), the inverting map enumerates the elements of S while using polynomial time per element of S.

The following simple observation shows that unless we come up with a reduction that is not Strongly Levin, the reduction to MAX-3SAT described in Equation 2 (in other words, the reduction arising out of NP = PCP($\log n$, 1)) must necessarily be code-like on worst-case instances (assuming collision-intractable hash functions exist).

For, we can apply the reduction to the boolean formula obtained in Lemma 2. If the reduction is not code-like on this formula, there exist satisfying assignments s_1, s_2 of φ such that $h(s_1)$ and $h(s_2)$ — which are assignments of $f(\varphi)$ — agree on more than $1 - \epsilon$ fraction of clauses.

Now modify the witness transformation h to become multivalued: a satisfying assignment s maps to the set of assignments $h'(s) = \mathrm{Ball}(h(s), \epsilon)$, consisting of all assignments that agree with $h(s)$ in $1 - \epsilon$ fraction of the clauses.

We assumed $h(s_1) \in \mathrm{Ball}(h(s_2), \epsilon)$. Hence, if h' satisfies the invertibility property of a strongly Levin reduction, we can generate s_2 given s_1, thus violating the cryptographic assumption.

5 How not to construct "counterexamples."

This paper aims to encourage researchers to discover reductions that are not code-like and Levin and then get around our impossibility results. But we include some caveats.

Note that our impossibility results for reductions hold under weaker assumptions than we discussed. For instance, in the case of clique, a reduction becomes useless if on even *one* boolean formula it produces a graph with superpolynomially-many "well-separated" cliques. Some "counterexample" approaches suggested to us did not take this fact into account (e.g., they were trying to modify known reductions on special instances of SAT, so that our definitions cease to apply).

Next, we mention an approach frequently suggested (by many readers of the first draft of this paper) to get around our impossibility results. We show that the approach, if it could be carried through, implies NP = coNP.

The idea – obviously motivated by our heavy use of the conjecture NP $\neq$ fewP – is the following: (i) Construct a reduction f_1 that maps unsatisfiable formulae to graphs with $\omega \leq n^{\epsilon}$, maps uniquely satisfiable formulae (i.e. those with a unique satisfying assignment) to graphs with $\omega \geq n^{1-\epsilon}$, and all other satisfiable formulae to graphs with $n^{\epsilon} < \omega < n^{1-\epsilon}$. (ii) Given any boolean formula, map it using the Valiant-Vazirani reduction [VV86] to another boolean formula, and then use reduction f_1 to map it to clique. By the known property of the [VV86] reduction, the composite reduction always maps unsatisfiable formulae to graphs with $\omega \leq n^{\epsilon}$. But given a satisfiable formula, with probability $1/n$ it produces a graph with $\omega \geq n^{1-\epsilon}$. Thus it proves the NP-hardness — under a randomized reduction — of approximating Clique within a factor $n^{1-2\epsilon}$, which is much bigger than $\sqrt{n}$ for small ϵ.

The problem lies in part (i): it implies coNP is reducible to NP. For, a trivial modification of any boolean formula gives a boolean formula that has exactly 1 more satisfying assignment. Thus an unsatisfiable formula becomes uniquely satisfiable, and a satisfiable formula changes into one with at least 2 satisfying assignments. Thus the reduction in (i) can be used to reduce coNP to NP.

(Note: a variation of (i), in which we replace the phrase "uniquely satisfiable formulae" with "boolean formulae with polynomially-many satisfying assignments," doesn't work for similar reasons.)

6 Conclusions

Recent inapproximability results depend upon constructions of verifiers that probabilistically check membership proofs for SAT. Current constructions of these verifiers automatically provide an explicit algorithm for *encoding* satisfying assignments, such that encodings of two different satisfying assignments have large hamming distance.

The fact that this encoding algorithm — as well as its associated decoding algorithm — should fall right out of the construction is surprising. We have explained this surprising fact by observing that the algorithm is merely the *witness transformation* of a (Levin) reduction. Furthermore, the minimum distance property of the encoding algorithm (what we called the *code-like* property earlier) merely serves to ensure that witnesses correspond to unique witnesses (as explained in Example 1).

We have also identified the limitations of such code-like Levin reductions. Thus two possibilities arise. First, that non code-like reductions exist, and are substantially more powerful. We speculate that such reductions will depend on nonconstructive arguments (current proofs of results such as NP = PCP($\log n, 1$) are constructive; see Section 7.1). The second possibility is that non code-like problems do not exist for the problems we have considered. In particular, this would mean (assuming NP $\neq$ *few* P) that achieving an approximation ratio $n^{0.5+\epsilon}$ for Clique is not NP-hard. Both possibilities need investigation.

Finally, we suspect that code-like reductions are inherently limited for minimization problems such as graph bisection and graph expansion. A proof of this would certainly explain why no inapproximability results have been obtained for these problems.

Acknowledgements

Thanks to Laci Babai for introducing me to the Packing Lemma at some point. Thanks also to Oded Goldreich, Uri Feige, Joe Kilian, Leonid Levin, Dick Lipton, Moni Naor, Toni Pitassi, Sasha Razborov, Ran Raz, Mike Sipser, Madhu Sudan, Umesh Vazirani, Avi Wigderson, and Andy Yao for useful conversations.

References

[AR88] E. Allender and R. Rubinstein. P-printable sets. *SIAM J. Computing* 17:1193–1202. 1988.

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

[AL95] S. Arora and C. Lund. Hardness of approximations. Survey chapter to appear in a book on Approximation Algorithms, D. Hochbaum, ed.. Available from the authors., 1995.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proc. 33rd IEEE FOCS*, pp 13–22, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proc. 33rd IEEE FOCS*, pp 2–13, 1992.

[BFLS] L. Babai and L. Fortnow and L. Levin and M. Szegedy. Checking Computations in Polylogarithmic Time. In *STOC 1991*, pp 21–31.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient multi-prover interactive proofs with applications to approximation problems. In *Proc. 25th ACM STOC*, pp 113–131, 1993.

[BGS95] M. Bellare, O. Goldreich, and M. Sudan. These proceedings.

[BH92] R. Boppana and M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32:180–196, 1992.

[Blu91] A. L. Blum. *Algorithms for approximate graph coloring.* PhD thesis, MIT, 1991.

[BS89] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. In *Proceedings of 7th STACS*, pp 256–267. LNCS 349, 1989.

[BS94] M. Bellare and M. Sudan. Improved non-approximability results. In *Proc. 26th ACM STOC*, pp 184–193, 1994.

[Bus86] S. Buss. *Bounded Arithmetic.* bibliopolis, Napoli, 1986.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM STOC*, pp 151–158, 1971.

[CRS93] S. Chari, P. Rohatgi, and A. Srinivasan. Randomness-optimal unique element isolation, with applications to perfect matching and related problems. In *Proc. 25th ACM STOC*, pp 458 –467, 1993.

[F9̈4] M. Fürer. Improved hardness results for approximating the chromatic number. Manuscript, August 1994, 1994.

[Fei95a] U. Feige. Randomized graph products and the Lovász theta function. In *Proc. 27th ACM STOC*, 1995. to appear.

[Fei95b] U. Feige. Set cover has a theshold at $\ln n$. Unpublished Manuscript, 1995.

[FGL+91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd IEEE FOCS*, pp 2–12, 1991.

[FK94] U. Feige and J. Kilian. Two prover protocols–low error at affordable rates. In *Proc. 26th ACM STOC*, pp 172–183, 1994.

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

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

[Joh74] D. S. Johnson. Approximation algorithms for combinatorial problems. *JCSS*, 9:256–278, 1974.

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

[KLS93] S. Khanna, N. Linial, and S. Safra. On the hardness of approximating the chromatic number. In *Proc. 2nd ISTCS*, pp 250–260. IEEE Computer Society Press, 1993.

[KMS94] D. Karger, R. Motwani, and M. Sudan. Graph coloring through semi-definite programming. In *Proc. 35th IEEE FOCS*, 1994.

[Lev73] L. Levin. Universal'nyĭe perebornyĭe zadachi (universal search problems : in Russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973. See also the appendix of "A survey of Russian approaches to Perebor (Brute force Search) Algorithms," by B. A. Trakhtenbrot, *Annals of the History of Computing:* 6:4 384–400, 1984.

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

[LY93a] C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In *Proc. ICALP*, pp 40–51, 1993.

[LY93b] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. In *Proc. 25th ACM STOC*, pp 286–293, 1993.

[MVV87] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[PY91] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *JCSS*, 43:425–440, 1991. Preliminary Version in Proc. ACM STOC, 1988.

[Raz] R. Raz. personal communication.

[Razb] A. A. Razborov. personal communication.

[Raz94] R. Raz. A parallel repetition theorem. *Proc. ACM STOC 1995*pp-

[R90] J. Rompel. One-way functions are Necessary and Sufficient for Secure Signatures. *Proc. ACM STOC 1990*, pp 387–394.

[Val76] L. G. Valiant. Relative complexity of checking and evaluating. *IPL*, 5:20–23, 1976.

[vL91] J. H. van Lint. *Introduction to Coding Theory, 2nd Ed., Chapter 5.* Springer Verlag, Heidelberg, 1991.

[VV86] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *TCS*, 47:85–93, 1986.

7 Appendix

We touch upon some points relevant to the discussion in the main body of the paper.

7.1 Why Levin reductions are prevalent

After seeing the preliminary version of this paper, Razborov [Razb] forwarded a hypothesis why all known reductions are Levin: their correctness is provable in Buss's constructive theory of number theory, S_2^1. This theory formalizes NP predicates as Σ_1^b formulae, that is, those of the form $(\exists_b y)\,\phi(y)$, where $\exists_b$ quantifies over strings of polynomial size. Notice that a many-one (or Karp) reduction is a polynomial-time map from one NP predicate to the other, such that one is true iff the other is. The well-known *witnessing theorem* about S_2^1 [Bus86] implies that if the correctness of the reduction is provable in S_2^1, then there exists another polynomial-time map from witnesses for one NP predicate to witnesses for the other. Thus a many-one reduction whose correctness is provable in S_2^1 is automatically Levin.

We observe (with some help from [Razb, Raz]) that current inapproximability results use reductions whose correctness is provable in S_2^1. Specifically, it is easily checked that NP = PCP($\log n, 1$) and Raz's parallel repetion theorem [Raz94] are provable in S_2^1, and all other inapproximability results follow from these two using constructive arguments (see [AL95]).

Thus Razborov's suggestion helps explain why current Karp reductions are Levin.

Non-Levin reductions. Coming up with non-Levin reductions is easy. Using a conjectured one-way permutation such as discrete log, *every* Levin reduction reduction can be changed so that there is no polynomial-time map from witnesses to witnesses. Beigel and Rudich have shown us different but similar constructions. (We note that the correctness of the cryptographic primitives used in such constructions is believed to be unprovable in S_2^1.) However, we know of no *useful* reduction that is not Levin.

7.2 Probabilistic Levin Reductionns

Now we indicate how to extend our impossibility results to probabilistic reductions. Probabilistic NP-completeness are rare: we know of only one that cannot be viewed as a (deterministic) nonuniform reduction: the Valiant-Vazirani reduction [VV86]. So we only sketch a formalization of such reductions.

A *probabilistic Levin reduction* is defined analogously as in Definition 1, except the map f is probabilistic. The witness transformations g, h are deterministic once f's random string has been fixed.

The reduction is *code-like* if h satisfies the usual code-like property with probability more than $1-1/n^c$ for every $c > 0$.

Definition 7 *USAT* is the set of boolean formulae with a unique satisfying assignment. *few SAT* is the set of boolean formulae with poly(n) satisfying assignments (n = number of variables).

Example 2 A probabilistic reduction from SAT to USAT appears in [VV86]. With probability $1/n$, it maps a satisfiable formula φ with n variables, to a formula ψ in USAT. The new formula ψ is satisfied by only those satisfying assignments of φ that have inner-product 0 (mod 2) with k vectors $u_1, \ldots u_k \in \{0,1\}^n$, where the number $k \in [1, n]$, and the vectors $u_1, \ldots, u_k$ have been picked randomly by the reduction.

Here the witness transformation is a partial function: a satisfying assignment has an image (namely, itself) iff its dot product with the randomly-picked vectors is 0. $\Box$

As pointed out in [VV86], the reduction is not a true reduction to USAT, since with probability $1/n$ or so, almost *all* satisfying assignments have images (i.e., the new formula is not even in *few SAT*). This problem has not been rectified by subsequent improvements to the reduction [MVV87, CRS93]. Therefore it has often been conjectured (informally) that this problem is inherent. In other words, not only is NP $\neq$ *few*P, but any probabilisitic reduction that tries to reduce SAT to a *few*P language must – on worst case inputs – have a probability at least $1/n^{c'}$ of producing an instance that is not in the *few*P language, where $c' > 0$ is some fixed constant.

Using this (hitherto informal) conjecture instead of NP $\neq$ *few*P, we can prove the impossibility results for probabilistic code-like Levin reductions similarly as our results for deterministic reductions.