The inapproximability of lattice and coding problems with preprocessing*

Uriel Feige†
Weizmann Institute of Science. Rehovot, Israel.

Daniele Micciancio‡
University of California, San Diego. La Jolla, California.

July 17, 2002

Abstract

We prove that the closest vector problem with preprocessing (CVPP) is NP-hard to approximate within any factor less than $\sqrt{5/3}$. More specifically, we show that there exists a reduction from an NP-hard problem to the approximate closest vector problem such that the lattice depends only on the size of the original problem, and the specific instance is encoded solely in the target vector. It follows that there are lattices for which the closest vector problem cannot be approximated within factors $\gamma < \sqrt{5/3}$ in polynomial time, no matter how the lattice is represented, unless NP is equal to P (or NP is contained in P/poly, in case of nonuniform sequences of lattices). The result easily extends to any ℓ_p norm, for $p \geq 1$, showing that CVPP in the ℓ_p norm is hard to approximate within any factor $\gamma < \sqrt[p]{5/3}$. As an intermediate step, we establish analogous results for the nearest codeword problem with preprocessing (NCPP), proving that for any finite field $GF(q)$, NCPP over $GF(q)$ is NP-hard to approximate within any factor less than $5/3$.

1 Introduction

Lattices are mathematical objects with a wide range of applications in mathematics and computer science. Lattices have received the attention of mathematicians for more than a century. For example, Gauss, Hermite and Dirichlet studied lattices (in the equivalent language of quadratic forms) because of their applications to Diophantine approximation and number theory. In computer science, lattices have attracted considerable attention after the discovery of the LLL basis reduction algorithm of Lenstra, Lenstra and Lovász [16]. This algorithm had a deep impact in many areas of computer science: using the LLL reduction algorithm it was possible to solve integer programming in a fixed number of variables [17], factor polynomials over the rationals [16], disprove century old conjectures in mathematics [25], break the Merkle-Hellman cryptosystem [24], check the solvability by radicals [15], solve low density subset-sum problems [7], heuristically factor integers [27] and solve many other Diophantine and cryptanalysis problems.

*A preliminary version of this paper appears in the Proceedings of the IEEE Conference on Computational Complexity - CoCo 2002. May 21–23, Montreal Canada, pp. 44-52. This is the full version of the paper.

†The author is the Incumbent of the Joseph and Celia Reskin Career Development Chair.

‡Research supported in part by NSF Career Award CCR-0093029.

Recently one more reason to study lattices specifically from a computational complexity point of view has emerged: the design of provably secure cryptographic functions (e.g., collision resistant hash functions or encryption schemes.) It has long been realized that in cryptography one needs problems that are hard to solve on the average: it is not enough to know that some keys are hard to break; when a user chooses his key at random, he wants a reasonable guarantee that *his* key is secure. In [1], Ajtai demonstrated that some lattice problems exhibit a surprising average-case/worst-case connection: if some lattice problem is hard to solve in the worst case, then some other lattice problem is hard when instances are chosen according to a certain probability distribution. This leads to the construction of cryptographic functions that are probably hard to break (on the average), based solely on a worst-case complexity assumption.

An integer lattice is an additive subgroup of $\mathbb{Z}^n$, and it can be represented by a basis, i.e., a set of linearly independent vectors such that lattice points can be written as integer linear combinations of the basis vectors. Throughout the paper, we use row notation for vectors, so that a lattice basis can be represented as a matrix $\mathbf{B}$, and the lattice generated by $\mathbf{B}$ is the set of all vectors $\mathcal{L}(\mathbf{B}) = \{\mathbf{xB} | \mathbf{x} \in \mathbb{Z}^n\}$, where $\mathbf{xB}$ is the standard vector matrix multiplication operation. The two most famous and widely studied lattice problems are the shortest vector problem and the closest vector problem. The shortest vector problem (SVP) is, given an integer lattice (represented by a basis $\mathbf{B}$) find the shortest nonzero vector in the lattice. The closest vector problem (CVP) is, given an integer lattice (represented by a basis $\mathbf{B}$) and a target vector $\mathbf{t}$, find the lattice point $\mathbf{xB}$ closest to the target $\mathbf{t}$. Both problems can be defined with respect to any norm, but the Euclidean norm is the most common. Approximation versions of these problems have also been considered. For example, in the γ approximate version of CVP, one is only required to find a lattice vector within a factor γ from the optimal, i.e., find an integer vector $\mathbf{x}$ such that $\|\mathbf{xB} - \mathbf{t}\| \leq \gamma \|\mathbf{yB} - \mathbf{t}\|$ for any (other) integer vector $\mathbf{y}$.

The computational difficulty of lattice problems often depends on the representation of the lattice (e.g., the choice of the basis). There are representations of lattices for which the best algorithms known for CVP has almost exponential approximation ratio. As an example of "bad" lattice representations, Micciancio [20] showed that the Hermite normal form (a normal form for lattices in which the basis matrix has triangular structure) can be used to obtain lattice bases for which the closest vector problem is as hard as possible in some precise technical sense, and suggested using this representation to improve the security and efficiency of lattice based cryptosystems. On the other hand, every lattice has a basis under which SVP is trivial (e.g., a basis that contains the shortest lattice vector as one of its columns.) Interestingly, [14, 13] show that every lattice has a representation (consisting of a sequence of n different bases) under which CVP can be efficiently approximated within a polynomial factor $O(n^{1.5})$. (This sequence of bases is perhaps hard to find, but it always exists.)

Is there a more clever choice of representation under which CVP can be solved exactly? (Recall that for SVP the answer is yes!) Is there a representation under which CVP can be approximated arbitrarily well? These are both natural questions. The first of these questions was negatively answered by Micciancio in [19], extending the results of Bruck, Naor and Lobstein [5, 18]. The current paper negatively answers the second of these questions. More precisely, we consider the following variant of CVP, called CVP with preprocessing (CVPP for short). CVPP asks for a function P (the preprocessing function) and an algorithm D (the decoding algorithm) with the following properties:

- On input a lattice basis $\mathbf{B}$, $P(\mathbf{B})$ returns a new description L of the lattice $\mathcal{L}(\mathbf{B})$ whose size is polynomially related to the size of $\mathbf{B}$, i.e. there exists a constant c such that $\text{size}(P(\mathbf{B})) < \text{size}(\mathbf{B})^c$ for all bases $\mathbf{B}$.

- Given $L = P(\mathbf{B})$ and a target vector $\mathbf{t}$, $D(L, \mathbf{t})$ computes a lattice point $\mathbf{Bx}$ closest to $\mathbf{t}$.

This formulation is inherently nonuniform. A uniform formulation of CVPP is also possible: given a (uniform) sequence of lattices $\mathcal{L}(\mathbf{B}_i)$ of increasingly big rank i, function P returns a uniform sequence of descriptions L_i, i.e., a polynomial time algorithm that on input i outputs a description L_i for lattice $\mathcal{L}(\mathbf{B}_i)$ in the sequence. For notational convenience, in this paper we concentrate on the nonuniform formulation, but all results hold for the uniform case as well. In either case, no complexity assumption is made on the preprocessing function P (other than the restriction on the size of the output). One may think of P as a preprocessing algorithm with unlimited computational resources. Only the running of D is used to measure the complexity of the decoding process, i.e., we say that CVPP is solvable in polynomial time if there exists a function P and a polynomial time algorithm D such that $D(P(\mathbf{B}), \mathbf{t})$ solves the CVP instance $(\mathbf{B}, \mathbf{t})$. The motivation behind this terminology is from coding theory: in these applications the lattice $\mathcal{L}(\mathbf{B})$ represents an encoding or encryption function, while the target $\mathbf{t}$ is the received message. In this context the closest vector problem corresponds to the decoding or decryption process. Notice that the lattice $\mathcal{L}(\mathbf{B})$ is usually fixed, and it is known long before transmission occurs. Therefore it makes sense to consider a variant of the closest vector problem in which the lattice is known in advance and can be arbitrarily preprocessed, and only the target vector $\mathbf{t}$ is specified as part of the input.

Recently, the construction of cryptographic functions with average-case/worst-case connection has offered one more reason to study CVP with preprocessing. In [1], Ajtai showed that if the shortest vector problem is hard to approximate (in the worst case) within some polynomial factor n^c (for $c \geq 8$), then one way functions exist. In [1] no substantial effort is made in order to get the smallest possible polynomial n^c. Subsequently, Cai and Nerurkar [6] showed that with a refined analysis and a careful choice of the parameters, the connection factor can be reduced to $n^{4+\epsilon}$. Decreasing the inapproximability factor necessary to prove the security of cryptographic functions with average-case/worst-case connection is important because SVP gets easier and easier as the factor increases. So, while it is reasonable to assume that SVP is hard to approximate within *small* (say n^2) factors, approximating SVP within large polynomial factors (say n^{100} or even n^{10}) might not be so hard, even if no polynomial time solution is currently known. Recently Micciancio [22] (see also [23]), showed that the inapproximability factor necessary to construct one way functions (in fact, collision resistant hash functions) can be reduced (using a construction possibly different from Ajtai's) to $O(n^{3.5} \log n)$, or even to $O(n^3 \log n)$ if an efficient CVP algorithm exists for a specific sequence of "almost perfect" lattices. It should be noted that the efficient CVP algorithm for almost perfect lattices is not used in the evaluation of the one way (hash) function of [22]. The algorithm is only used in the proof of security, so, it is enough to know that a polynomial time solution exists, and knowledge of the actual algorithm is not strictly necessary. Although such CVP algorithm for almost perfect lattices might exists, this paper shows that this algorithm cannot be obtained from general principles: even if these lattices are fixed in advance, no representation is guaranteed to exist that allows to (approximately) solve CVP in polynomial time. In order to further reduce the connection factor from $O(n^{3.5} \log n)$ to $O(n^3 \log n)$, an ad-hoc solution to CVP for the almost perfect lattices is needed.

In this paper we establish that CVPP is hard to approximate within any constant factor less than $\sqrt{5/3}$ (or, more generally, $\sqrt[p]{5/3}$ for the ℓ_p norm). The result is proved using the analogous problem for linear codes as an intermediate step. For any finite field $\mathbb{F}$, a linear code over $\mathbb{F}$ is a linear subspace of $\mathbb{F}^n$. The nearest codeword problem is, given a matrix $\mathbf{W}$ over $\mathbb{F}$ (representing linear code $\mathcal{C}(\mathbf{W}) = \{\mathbf{xW} | \mathbf{x} \in \mathbb{F}^n\}$) and a target vector $\mathbf{t}$, find the codeword $\mathbf{xW}$ closest (in the

Hamming distance[1]) to the target $\mathbf{t}$. Approximation and preprocessing versions of NCP are defined analogously. The existence of codes for which no polynomial time decoding procedure exists (under standard complexity assumptions), was first demonstrated by Bruck and Naor in [5]. In particular, [5] gives a reduction from an NP-hard problem to NCP such that the code depends only on the size of the original problem, and the specific instance is encoded solely in the target vector. It follows that there are codes for which the NCP cannot be solved in polynomial time, no matter how the code is represented, unless NP is contained in P/poly. Analogous results for lattices and CVP are presented in [19]. Unfortunately, [5, 19] only prove the hardness of NCPP and CVPP for the exact version, and [19] points out that the reductions in [5, 19] do not directly yield inapproximability results. Specifically, [5] and [19] give reductions from Simple Max Cut (SMC) and Exact 3-Cover (X3C) to NCPP and CVPP. (See [5, 19] or [11] for the definition of SMC and X3C.) The reductions hold for the exact versions of the problems, but they are not in general gap-preserving. So, even if SMC and X3C are hard to approximate, it is not clear how to obtain inapproximability results for NCPP and CVPP. In this paper we present a different reduction from an NP-hard problem to NCPP and CVPP that yields a gap for the coding and lattice problems.

The rest of the paper is organized as follows. We first prove, in Section 2, that NCPP over $GF(q)$ is hard to approximate within any factors less than $1+(2/3)(1-1/q)$. Notice that for large alphabet sizes, this factor approaches $5/3$, however for small alphabets, in particular for $q=2$, the inapproximability factor is close to $4/3$. Then, in Section 3, we use standard techniques of concatenating codes [10] to improve the inapproximability factor to $5/3$ for any finite field (in particular, for the binary and ternary alphabets). Finally, in Section 4, we reduce the problem of approximating NCP over $GF(2)$ (or $GF(3)$) within factor γ to the problem of approximating CVP in the ℓ_p norm within factor $\sqrt[p]{\gamma}$. The lattice produced by the reduction depends only on the code of the source problem (i.e., it does not depend on the target vector), therefore establishing the hardness of CVPP. Section 5 concludes with some remarks about the latest developments about NCPP and CVPP, the possibility of extending our results to any constant approximation factor, and other open problems.

2 Inapproximability of NCPP within some constant

In this section we prove that for any prime power q, the nearest codeword problem with preprocessing over $GF(q)$ is hard to approximate within any factor less than $1+(2/3)(1-1/q)$. In particular, the nearest codeword problem with preprocessing over the binary alphabet $GF(2)$ is hard to approximate within any factor less than $4/3$. We start with the slightly simpler proof for the case of binary fields $GF(2)$ in Subsection 2.1. Then, in Subsection 2.2 we present a more general reduction that applies to any finite field. In this and the next section, $\|\mathbf{x}\|$ denotes the Hamming norm of $\mathbf{x}$, i.e., the number of nonzero entries in $\mathbf{x}$, and $\text{dist}(\mathbf{x},\mathcal{W})$ denotes the Hamming distance between $\mathbf{x}$ and the nearest codeword in $\mathcal{W}$.

2.1 Binary fields

In this section we prove that NCPP over $GF(2)$ is NP-hard to approximate within any factor less than $4/3$. This is a special case of a more general result about arbitrary finite fields proved in the next subsection, and it is presented here mostly to explain the ideas of the reduction in a slightly simpler setting. Since the result described in this section can be immediately obtained as a corollary

[1]The Hamming distance between two vectors is the number of positions in which the two vectors differ.

to Theorem 3 proved in Subsection 2.2, here we will be rather informal and dispense with the most technical details. The reader is referred to Subsection 2.2 for a formal proof of the result.

We establish the hardness of approximating the nearest codeword problem with preprocessing over $\mathbb{F} = GF(2)$ by reduction from the problem of approximating the number of simultaneously satisfiable equations in a linear system $\mathbf{xA} = \mathbf{b}$. In [12] Håstad shows that this problem is NP-hard for any approximation factor less than 2, and the result holds even if each equation is restricted to contain at most three variables. An even more restricted version of this problem is considered by Berman and Karpinski [4], who show that the problem is NP-hard even if each equation contains exactly 3 variables, and each variable appears in exactly 3 equations.[2] This is the starting point of our reduction. More precisely, we start from the following theorem which can be deduced from [12, 4].

Theorem 1 *Let $\mathbf{xA} = \mathbf{b}$ be a system of n linear equations over $GF(2)$ where each equation contains exactly 3 variables and each variable appears in exactly 3 equations. For any constant $\epsilon > 0$, it is NP-hard to distinguish instances for which the maximum number of simultaneously satisfiable equations is at least $(1-\epsilon)n$, from instances for which this number is at most $((1/2)+\epsilon)n$.*

We want to reduce systems of n linear equations in n variables to instances of the nearest codeword problem $(\mathbf{W}, \mathbf{t})$ where the code $\mathbf{W}$ depends only on the number of equations n. Consider any system of equations $\mathbf{xA} = \mathbf{b}$ satisfying the conditions in the theorem, and let $x_1, \ldots, x_n$ be the n variable names. Since each variable appears exactly 3 times, we can annotate the variables with superscripts $k = 1, 2, 3$ such that each x_i^k appears exactly once in the system. Let

$$X = \{x_i^k : i = 1, \ldots, n, k = 1, 2, 3\}$$

be the set of annotated variables. Notice that there are at most $|X|^3 = (3n)^3$ possible annotated (homogeneous) equations containing exactly 3 variables. Moreover, no annotated equation can appear in the system more than once because different occurrences of the same variable are annotated with different superscripts. Therefore the set of (homogeneous) equations in an annotated system can be represented as a subset of X^3. Let

$$M = X^3$$

be the set of triples representing all possible equations. We define three $|M| \times |X|$ matrices $\mathbf{Q}^1, \mathbf{Q}^2, \mathbf{Q}^3$ with rows indexed by equations $m \in M$ and columns indexed by variables $x_i^k \in X$, and entries

$$\mathbf{Q}^l_{m,x_i^k} = \begin{cases} 0 & \text{if } v_l(m) = x_i^k \\ 1 & \text{otherwise} \end{cases}$$

In other words, for each annotated variable x_i^k occurring at position l in equation m, matrix $\mathbf{Q}^l$ contains one of the blocks $[0,1,1]$, $[1,0,1]$ or $[1,1,0]$, selected according to the value of k. What is important about these three blocks is that the only way to get the all-zero vector is to either add up all of them, or none. These matrices will be used to make sure that if we assign value b to one occurrence of a variable, then we must assign b also to all other occurrences of the same variable.

[2] Clearly, the equations must be linearly dependent, otherwise the problem can be solved in polynomial time using Gauss elimination. This is perfectly fine, and does not affect our reduction.

We can now define the code. The generator matrix of the code is given by

$$\mathbf{W} = \left[\begin{array}{ccc|c|ccc|c} \mathbf{I} & & & & \mathbf{I} & & & \mathbf{Q}^1 \\ & \mathbf{I} & & & & \mathbf{I} & & \mathbf{Q}^2 \\ & & \mathbf{I} & & & & \mathbf{I} & \mathbf{Q}^3 \\ \hline & & & \mathbf{I} & \mathbf{I} & \mathbf{I} & & & \mathbf{Q}^1 \\ & & & & \mathbf{I} & & \mathbf{I} & & \mathbf{Q}^2 \\ & & & & \mathbf{I} & & & \mathbf{I} & \mathbf{Q}^3 \end{array}\right]$$

where $\mathbf{I}$ is the $|M| \times |M|$ identity matrix, and empty spaces denote zeros. Each row of $\mathbf{W}$ can be indexed with a triple (m, k, a) where $a \in \{0, 1\}$ selects the top or bottom half of the matrix, $k = 1, 2, 3$ selects one of the three blocks of rows in each half, and $m \in M$ select a row within the block. The columns are divided into four blocks:

- Columns of the first block are called *ordinary* coordinates. These are used to count how many base codewords we are using.
- Columns of the second block are called *equation* coordinates, as we have one column for each equation.
- Columns in the third block are called *position* coordinates, and each column corresponds to a specific position $k = 1, 2, 3$ within an equation m.
- Columns in the last block are called *variable* coordinates, with each column corresponding to an annotated variable x_i^k and a bit $b \in \{0, 1\}$. These are used to assign boolean values to variable occurrences.

By repeating the columns multiple times, it is possible to give to each column a different weight. Ordinary coordinates will have weight 1, equation coordinates weight 2, and all other coordinates a large weight B, (e.g. $B = 5n$). For simplicity of notation, we will not explicitly repeat the columns in the definitions and proofs, but we assume that coordinates are weighed as described above.

This code can be used to represent a system $\mathbf{xA} = \mathbf{b}$ as follows. Let $\mathbf{a}$ be a vector of dimension $|M|$ with coordinates indexed by equations $m \in M$, and $\mathbf{a}_m = 1$ if and only if m is an equation in the annotated input system. Similarly, we define a vector $\mathbf{b}$ with $\mathbf{b}_m$ equal to the constant term of equation m in the input system (if m is one of the input equations), and $\mathbf{b}_m = 0$ otherwise. The target vector corresponding to the given system is

$$\mathbf{t} = \left[\begin{array}{cccccc|c|ccc|cc} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{b} & \mathbf{a} & \mathbf{a} & \mathbf{a} & \mathbf{0} & \mathbf{0} \end{array}\right]$$

with the coordinates corresponding to the blocks in the obvious way, and each coordinate weighted as in the definition of generating matrix $\mathbf{W}$. Consider a position (m, k) in the system, i.e., an equation m in $\mathbf{A}$ and an index $k = 1, 2, 3$. The position coordinate in the target vector contains a 1, and there are only two codewords (namely $(m, k, 0)$ and $(m, k, 1)$) containing a 1 in that coordinate. (See column (m, k) in Figure 1.) Since the position coordinates are given a large weight B, the only way to get close to the target is to use exactly one of the two codewords $(m, k, 0)$ or $(m, k, 1)$ and get equality with the target in that column. Depending on which of the two codewords we use, we will be adding a 0 or 1 in the column corresponding to coordinate m in the equation block. (See column (m) in Figure 1.) So, choosing $(m, k, 0)$ or $(m, k, 1)$ corresponds to assigning value $b = 0$ or $b = 1$ to the kth variable occurrence in equation m. On the other hand, if m is not at equation in our system, then the corresponding position coordinate in the target is 0, and we have to use

either both or none of the codewords $(m,k,0),(m,k,1)$. Notice that there is no incentive to use both, because the potential advantage of matching the target vector in the equation coordinate $\mathbf{b}_m$ (remember that equation coordinates have weight 2), is compensated by two ordinary coordinates becoming nonzero. So, if equation coordinates weight twice as much as ordinary coordinates, we can assume without loss of generality that the codeword closest to the target does not use basic codewords (m,k,b) with m not in the input system. (Formally, one can prove that at least one of the codewords nearest to the target satisfy this condition.) This shows that codewords nearest to the target correspond to assigning boolean values to the annotated variables. Finally, the reader can easily verify that the only way to get zero in the variable coordinates is to assign the same value to each occurrence of the same variable. (We omit the details here as a more general statement will be formally proved in the next subsection.)

To sum up, there is a close correspondence between variable assignments and codewords nearest to the target $\mathbf{t}$. The codeword corresponding to assignment ϕ equals the target in all variable and position coordinates, it equals the target in equation coordinate m if and only if the equation is satisfied by assignment ϕ, and uses exactly $3n$ of the base codewords from $\mathbf{W}$. Since each base codeword contributes one to the distance from the target in the corresponding ordinary coordinate, the distance of the codeword from the target equals $3n + 2(1-\alpha)n$, where α is the fraction of equations satisfied by assignment ϕ, and 2 is the weight of the equation coordinates. So, our reduction transforms the gap $((1/2+\epsilon)n,(1-\epsilon)n)$ of the original satisfiability problem into a gap $((4-2\epsilon)n,(3+2\epsilon)n)$ for the nearest codeword problem. By choosing ϵ sufficiently close to 1, one can make this gap arbitrarily close to 4/3.

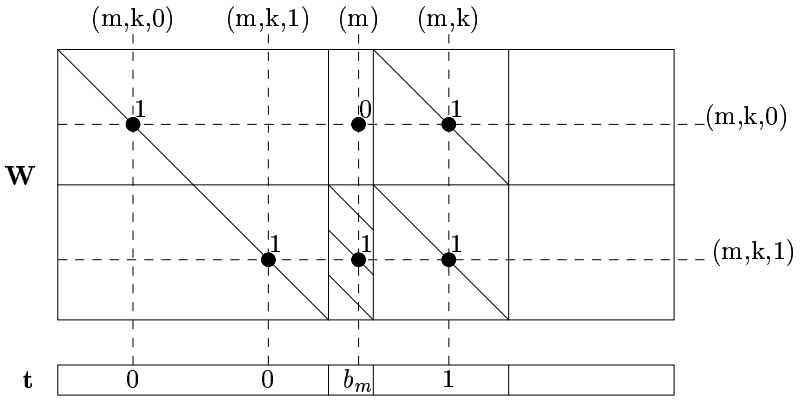

Figure 1: Nearest codewords and variable assignments. The diagonal lines represent 1's.

2.2 Arbitrary finite fields

In Section 3 we we show how the inapproximability factor for NCPP over $GF(2)$ (in fact, any finite field) can be improved to any constant less than 5/3. The improvement relies on the use of large extension fields, and requires, as an intermediate step, to prove inapproximability of NCPP for any finite field $GF(2^k)$.

Let q be an arbitrary, but fixed, prime power, and let $\mathbb{F} = GF(q)$ be the finite field with q elements. We prove the hardness of approximating the nearest codeword problem with preprocessing over $\mathbb{F}$ by reduction from the problem of approximating the number of simultaneously satisfiable equations in a linear system $\mathbf{xA} = \mathbf{b}$. Specifically, we use the following result from [12] as the

starting point of our reduction.

Theorem 2 *Let* $\mathbf{xA} = \mathbf{b}$ *be a system of* n *linear equations over* $GF(q)$ *where each equation contains at most 3 variables. For any constant* $\epsilon > 0$*, it is NP-hard to distinguish instances for which the maximum number of simultaneously satisfiable equations is at least* $(1-\epsilon)n$*, from instances for which this number is at most* $((1/q)+\epsilon)n$*.*

As a special case, we get the inapproximability of NCPP over $GF(2)$ for factors less than 4/3 already established in the previous subsection. However, this time the reduction is more complicated. The reason is that we cannot assume that each variable appears exactly 3 times, because [4] only establishes this result for the binary field $GF(2)$.[3] So, we show how to modify the reduction of Subsection 2.1 to work for the more general system of equations of Theorem 2.

We want to reduce systems of n linear equations where each equation contains at most 3 variables to instances of the nearest codeword problem $(\mathbf{W}, \mathbf{t})$ where the code $\mathbf{W}$ depends only on the number of equations n.

In order to describe the reduction we use the following notation. Variables in the system of linear equations have names of the form $x_{i,j}$ (with i and j positive integers), and that if variable $x_{i,j}$ occurs in the system, than it appears exactly i times. Notice that since each equation contains at most 3 variables, and each linear system under consideration consists of exactly n linear equations, we only need names $x_{i,j}$ with $i \leq 3n$ and $j \leq 3n/i$. We remark that not every variable $x_{i,j}$ is used in every system, so each variable $x_{i,j}$ can appear either 0 or i times. If variable $x_{i,j}$ is used (i.e., it appears i times), then the kth appearance (for $k = 1, \ldots, i$) of $x_{i,j}$ in a linear system is denoted $x^k_{i,j}$. Let X be the set of all possible annotated variables

$$X = \{x^k_{i,j} : i \cdot j \leq 3n,\ k \leq i\}$$

and let $\pi: X \to X$ be the permutation over X defined by

$$\pi(x^k_{i,j}) = x^{(k+1) \bmod i}_{i,j}. \tag{1}$$

A *variable assignment* is a function $\phi: X \to \mathbb{F}$ that associates to every variable symbol $x^k_{i,j}$ a field element $\phi(x^k_{i,j})$. We say that an assignment is *consistent* if and only if ϕ assigns the same value to all occurrences $x^k_{i,j}$ $(k = 1, \ldots, i)$ of the same variable $x_{i,j}$. Equivalently, ϕ is consistent if and only if $\phi(\pi(x)) = \phi(x)$ for all $x \in X$. The size of set X is easily bounded as follows:

$$|X| = \sum_{i=1}^{3n} \sum_{j=1}^{\lfloor 3n/i \rfloor} i = \sum_{i=1}^{3n} \lfloor 3n/i \rfloor \cdot i \leq \sum_{i=1}^{3n} 3n = (3n)^2. \tag{2}$$

In particular, $|X|$ is polynomial in the input size. Let

$$M = (\mathbb{F} \times X)^3$$

be the set of all triples $m = ((\alpha_1, x_1), (\alpha_2, x_2), (\alpha_3, x_3))$ with $\alpha_i \in \mathbb{F}$ and $x_i \in X$. We use pairs $(m, b) \in M \times \mathbb{F}$ to represent linear equations in three variables

$$\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 = b.$$

[3] It is probably the case that reduction in [4] can be generalized to work for any finite field. However, this is not know at the time of this writing.

Define projection functions $c_i : M \to \mathbb{F}$ and $v_i : M \to X$ (for $i = 1, 2, 3$) corresponding to the ith coefficient and the ith variable of m:

$$\begin{aligned} c_i((\alpha_1, x_1), (\alpha_2, x_2), (\alpha_3, x_3)) &= \alpha_i \\ v_i((\alpha_1, x_1), (\alpha_2, x_2), (\alpha_3, x_3)) &= x_i. \end{aligned}$$

Then, any system of linear equations with at most three variables per equation can be represented as a pair (A, b) where A is a subset of M and b is a function from A to $\mathbb{F}$. Notice that equations with less than three variables can be regarded as equations with exactly three variables in which some of the coefficients are 0. Notice also that even if the original system contains repeated equations, the annotated equations do not repeat because each annotated variable $x_{i,j}^k$ appears at most once. So, any linear system from Theorem 2 can be represented as a pair (A, b).

To sum up, the starting point of the reduction is a set $A \subset M$ of size $|A| = n$ and a function $b\colon A \to \mathbb{F}$ representing system of linear equations

$$\left\{ \sum_{i=1}^{3} c_i(m) v_i(m) = b(m) \quad : \quad m \in A \right\}. \tag{3}$$

Given a pair (A, b), the set of annotated variables $x_{i,j}^k$ that appear in A is denoted

$$X_A = \{v_i(m)\colon m \in A, i \in \{1, 2, 3\}\}. \tag{4}$$

We remark that not every pair (A, b) is a valid representation of linear systems. In particular, (A, b) is a valid representation if and only if $|A| = n$, $|X_A| = 3n$ and $\pi(X_A) = X_A$.

We use sets $X, M, \mathbb{F}$ to define a universal code $\mathbf{W}$ that allows to represent any linear system with n equations. (Notice that we cannot use set X_A in the definition of $\mathbf{W}$, because X_A depends on the specific system of linear equations to be reduced.) First we need to define the following vector spaces (over $\mathbb{F}$).

- V_M is the $|M|$-dimensional vector space (over $\mathbb{F}$) with canonical basis $\mathbf{e}_m$ (for $m \in M$). Here and below, for any set S and element $a \in S$, $\mathbf{e}_a$ represents the vector with $|S|$ coordinates which equals 1 at position a (assuming some standard ordering of the elements of S), and 0 everywhere else.

- V_{M3} is the $3|M|$-dimensional vector space (over $\mathbb{F}$) with canonical basis $\mathbf{e}_{m,i}$ (for $m \in M$ and $i \in \{1, 2, 3\}$).

- V_{M3F} is the $3q|M|$-dimensional vector space (over $\mathbb{F}$) with canonical basis $\mathbf{e}_{m,i,\alpha}$ (for $m \in M$, $i \in \{1, 2, 3\}$ and $\alpha \in \mathbb{F}$).

- V_{XF} is the $q|X|$-dimensional vector space (over $\mathbb{F}$) with canonical basis $\mathbf{e}_{x,\alpha}$ (for $x \in X$ and $\alpha \in \mathbb{F}$). We also define auxiliary vectors

$$\mathbf{u}_{x,\alpha} = \mathbf{e}_{x,\alpha} - \mathbf{e}_{\pi(x),\alpha} \in V_{XF}. \tag{5}$$

  Notice that these vectors are linearly dependent because they add up to $\mathbf{0}$.

Codewords are obtained concatenating elements from V_{M3F}, V_M, V_{M3} and V_{XF}. Vectors from V_M, V_{M3} and V_{XF} are repeated multiple times so to increase the weight of the corresponding

coordinates. In particular, vectors from V_M are repeated twice, while vectors from V_{M3} and V_{XF} are repeated $B = 5n$ times. In other words, we consider vectors of the form

$$\mathbf{w} = [\mathbf{w}^{(o)}, \mathbf{1}_2 \otimes \mathbf{w}^{(e)}, \mathbf{1}_B \otimes \mathbf{w}^{(p)}, \mathbf{1}_B \otimes \mathbf{w}^{(v)}] \tag{6}$$

where $\mathbf{w}^{(o)} \in V_{M3F}$, $\mathbf{w}^{(e)} \in V_M$, $\mathbf{w}^{(p)} \in V_{M3}$, $\mathbf{w}^{(v)} \in V_{XF}$ and $\mathbf{1}_k \otimes \mathbf{x}$ denotes the concatenation of k identical copies of vector $\mathbf{x}$. For any such vector, we call $\mathbf{w}^{(o)}$ the block of *ordinary* coordinates (which have unit weight), $\mathbf{w}^{(e)}$ the block of *equation* coordinates (with one coordinate per equation $m \in M$), $\mathbf{w}^{(p)}$ the block of *position* coordinates (with one coordinate per position within an equation), and $\mathbf{w}^{(v)}$ the block of *variable* coordinates (with one coordinate per variable assignment $x \leftarrow \alpha$).

We can now define the code. For every $m \in M$, $i \in \{1, 2, 3\}$ and $\alpha \in \mathbb{F}$ define codeword

$$\mathbf{w}_{m,i,\alpha} = [\mathbf{e}_{m,i,\alpha}, \mathbf{1}_2 \otimes (c_i(m)\alpha \cdot \mathbf{e}_m), \mathbf{1}_B \otimes \mathbf{e}_{m,i}, \mathbf{1}_B \otimes \mathbf{u}_{v_i(m),\alpha}] \tag{7}$$

having

- value 1 in the (m, i, α)th ordinary coordinate,
- value $\alpha c_i(m)$ in the mth coordinate of each equation block,
- value 1 in the (m, i)th coordinate of each position block,
- value 1 in the $(v_i(m), \alpha)$th coordinate and value -1 in the $(\pi(v_i(m)), \alpha)$th coordinate of each variable block,
- value 0 in all other positions.

We define $\mathcal{W}$ as the linear code over $\mathbb{F}$ with block-length $3q|M| + 2|M| + 3|M|B + q|X|B$ and rate $3q|M|$ generated by vectors (6):

$$\mathcal{W} = \left\{ \sum y_{m,i,\alpha} \cdot \mathbf{w}_{m,i,\alpha} : \ m \in M, i \in \{1, 2, 3\}, \alpha \in \mathbb{F}, y_{m,i,\alpha} \in \mathbb{F} \right\}. \tag{8}$$

Notice that each basis codeword has Hamming weight $3(B+1) = 15n+3$ and the block of ordinary coordinates is a systematic information set for the code.

Given a pair (A, b) encoding a linear system, define equation vector

$$\mathbf{b} = \sum_{m \in A} b(m)\mathbf{e}_m \tag{9}$$

and position vector

$$\mathbf{a} = \sum_{m \in A} (\mathbf{e}_{m,1} + \mathbf{e}_{m,2} + \mathbf{e}_{m,3}). \tag{10}$$

The target vector associates to (A, b) is

$$\mathbf{t} = [\mathbf{0}, \mathbf{1}_2 \otimes \mathbf{b}, \mathbf{1}_B \otimes \mathbf{a}, \mathbf{0}]. \tag{11}$$

Notice that for any vector $\mathbf{w}$ of the form (6) the Hamming distance of $\mathbf{w}$ from the target equals

$$\|\mathbf{w} - \mathbf{t}\| = \|\mathbf{w}^{(o)}\| + 2\|\mathbf{w}^{(e)} - \mathbf{b}\| + B\|\mathbf{w}^{(p)} - \mathbf{a}\| + B\|\mathbf{w}^{(v)}\|. \tag{12}$$

In the following lemma we prove a lower bound to the distance of target $\mathbf{t}$ from the code $\mathcal{W}$ that holds for any linear system (A, b).

Lemma 1 *Let code $\mathcal{W}$ and target $\mathbf{t}$ be as defined in (8) and (11). Then, the distance of target from code satisfies dist$(\mathbf{t}, \mathcal{W}) \geq 3n$.*

Proof: Consider a generic codeword

$$\mathbf{w} = \sum_{m \in M} \sum_{i=1}^{3} \sum_{\alpha \in \mathbb{F}} y_{m,i,\alpha} \cdot \mathbf{w}_{m,i,\alpha}$$

where $y_{m,i,\alpha}$ are arbitrary field elements. We prove that $\|\mathbf{w} - \mathbf{t}\| \geq 3n$. There are two cases. If $\mathbf{w}^{(p)} \neq \mathbf{a}$, then from (12) we get

$$\|\mathbf{w} - \mathbf{t}\| \geq B\|\mathbf{w}^{(p)} - \mathbf{a}\| \geq B > 3n.$$

On the other hand, if $\mathbf{w}^{(p)} = \mathbf{a}$, then the number of non-zero coefficients $y_{m,i,\alpha} \neq 0$ is at least $3n$ because $\mathbf{a}$ has $3n$ non-zero entries and for each base codeword $\mathbf{w}_{m,i,\alpha}$, the position block $\mathbf{w}^{(p)}_{m,i,\alpha} = \mathbf{e}_{m,i}$ has only 1 non-zero entry. So, in order to get $\mathbf{w}^{(p)} = \sum y_{m,i,\alpha}\mathbf{e}_{m,i} = \mathbf{a}$ we need to use at least $3n$ vectors. Finally, we notice that $\|\mathbf{w}^{(o)}\|$ equals exactly the number of non-zero coefficients, and therefore, again using (12),

$$\|\mathbf{w} - \mathbf{t}\| \geq \|\mathbf{w}^{(o)}\| \geq 3n.$$

□

If the linear system is satisfiable, then the lower bound is tight and dist$(\mathbf{t}, \mathcal{W}) = 3n$ as shown below.

Lemma 2 *Let $\phi\colon X \to \mathbb{F}$ be a consistent variable assignment and define an associated codeword*

$$\mathbf{w}_\phi = \sum_{m \in A} \sum_{i=1}^{3} \mathbf{w}_{m,i,\phi(v_i(m))}. \tag{13}$$

For any (A, b), if ϕ satisfies a $(1-\beta)$ fraction of equations (3), then $\|\mathbf{w}_\phi - \mathbf{t}\| = (2\beta + 3)n$. In particular, if the linear system associated to (A, b) is satisfiable, then dist$(\mathbf{t}, \mathcal{W}) = 3n$.

Proof: First, extend the assignment ϕ to M in the obvious way:

$$\phi(m) \quad = \quad \sum_{i=1}^{3} c_i(m) \cdot \phi(v_i(m)). \tag{14}$$

We evaluate the four components of $\|\mathbf{w}_\phi - \mathbf{t}\|$ as given in (12) separately.

1. For the ordinary coordinates we have

$$\mathbf{w}^{(o)}_\phi = \sum_{m \in A} \sum_{i=1}^{3} \mathbf{e}_{m,i,\phi(v_i(m))}$$

and therefore $\|\mathbf{w}^{(o)}_\phi\| = 3|A| = 3n$.

2. For the equation coordinates we have

$$\begin{aligned}\mathbf{w}_\phi^{(e)} &= \sum_{m\in A}\left(\sum_{i=1}^{3} c_i(m)\cdot\phi(v_i(m))\right)\cdot\mathbf{e}_m \\ &= \sum_{m\in A}\phi(m)\cdot\mathbf{e}_m\end{aligned}$$

and therefore $\|\mathbf{w}_\phi^{(e)}-\mathbf{b}\|=\beta n$.

3. For the position coordinates we have

$$\mathbf{w}_\phi^{(p)}=\sum_{m\in A}\sum_{i=1}^{3}\mathbf{e}_{m,i}=\mathbf{a}$$

and therefore $\|\mathbf{w}_\phi^{(p)}-\mathbf{a}\|=0$.

4. Finally, for the variable coordinates,

$$\mathbf{w}_\phi^{(v)}=\sum_{m\in A}\sum_{i=1}^{3}\mathbf{u}_{v_i(m),\phi(v_i(m))}=\sum_{x\in X_A}\mathbf{u}_{x,\phi(x)}$$

From the consistency requirement $\phi(\pi(x))=\phi(x)$ and $\pi(X_A)=X_A$ we get

$$\begin{aligned}\sum_{x\in X_A}\mathbf{e}_{x,\phi(x)} &= \sum_{x\in\pi(X_A)}\mathbf{e}_{x,\phi(x)} \\ &= \sum_{x\in X_A}\mathbf{e}_{\pi(x),\phi(\pi(x))} \\ &= \sum_{x\in X_A}\mathbf{e}_{\pi(x),\phi(x)}\end{aligned}$$

and using the definition of $\mathbf{u}_{x,\alpha}$ (5) we get

$$\sum_{x\in X_A}\mathbf{u}_{x,\phi(x)}=\sum_{x\in X_A}\mathbf{e}_{x,\phi(x)}-\sum_{x\in X_A}\mathbf{e}_{\pi(x),\phi(x)}=\mathbf{0}. \tag{15}$$

This proves that $\|\mathbf{w}_\phi^{(v)}\|=0$.

Substituting the four terms in the right hand side of (12) we get

$$\|\mathbf{w}_\phi-\mathbf{t}\|=3n+2\beta n+0+0=(2\beta+3)n.$$

□

As a corollary, we immediately get an upper bound on the distance of $\mathbf{t}$ from the code.

Corollary 1 *For any (A,b), the distance of target (11) from code (8) is at most dist$(\mathbf{t},\mathcal{W})\le 5n-2\lceil n/q\rceil<B$.*

Proof: The expected number of equations satisfied by a random variable assignment is $1/q$. It follows from an averaging argument that there exists an assignment ϕ that satisfies at least $\lceil n/q \rceil$ equations. Applying Lemma 2 with $\beta = (n - \lceil n/q \rceil)/n$, we get that there exists a codeword $\mathbf{w}_\phi$ within distance

$$(2\beta + 3)n = 5n - 2\lceil n/q \rceil < 5n = B$$

from the code. □

From Lemma 1 and Corollary 1, it follows that the distance $\mathrm{dist}(\mathbf{t}, \mathcal{W})$ is always in the range $[3n, 5n)$. So, we can always write $\mathrm{dist}(\mathbf{t}, \mathcal{W}) = (2\beta + 3)n$ for some $\beta \in [0, 1)$. Now we prove a converse of Lemma 2, showing that if the nearest codeword is within distance $(2\beta + 3)n$ from the target, then there is a consistent variable assignment that satisfies (at least) a $(1 - \beta)$ fraction of the equations.

Lemma 3 *For any linear system (A, b), let $\mathcal{W}$ and $\mathbf{t}$ be the code and target vector as defined in (8) and (11). If dist$(\mathbf{t}, \mathcal{W}) = (2\beta + 3)n$, then there exists a consistent variable assignment ϕ that satisfies at least a $(1 - \beta)$ fraction of equations (3).*

Proof: Let $\mathbf{w} = \sum_{m,i,\alpha} y_{m,i,\alpha} \mathbf{w}_{m,i,\alpha}$ be a codeword such that $\|\mathbf{w} - \mathbf{t}\| = \mathrm{dist}(\mathbf{t}, \mathcal{W})$. Notice that it must be $\mathbf{w}^{(p)} = \mathbf{a}$ and $\mathbf{w}^{(v)} = \mathbf{0}$, because otherwise (12) would give $\|\mathbf{w} - \mathbf{t}\| \geq B\|\mathbf{w}^{(p)} - \mathbf{a}\| + B\|\mathbf{w}^{(v)}\| \geq B$, contradicting the upper bound $\mathrm{dist}(\mathbf{t}, \mathcal{W}) < B$ from Corollary 1.

Define variable assignment ϕ as follows. For every $x \in X_A$, let

$$\phi(x) = \sum_{\alpha \in \mathbb{F}} \alpha \sum_{v_i(m) = x} y_{m,i,\alpha}$$

and let $\phi(x) = 0$ for all $x \in X \setminus X_A$. First of all, we prove that ϕ is consistent, i.e., $\phi(x) = \phi(\pi(x))$ for all $x \in X_A$. Consider the $(\pi(x), \alpha)$ coordinate of $\mathbf{w}^{(v)}$:

$$\begin{aligned}
\left\langle \mathbf{w}^{(v)}, \mathbf{e}_{\pi(x),\alpha} \right\rangle &= \left\langle \sum_{m,i} y_{m,i,\alpha} \cdot \mathbf{u}_{v_i(m),\alpha}, \mathbf{e}_{\pi(x),\alpha} \right\rangle \\
&= \sum_{m,i} y_{m,i,\alpha} \cdot \langle \mathbf{e}_{v_i(m),\alpha}, \mathbf{e}_{\pi(x),\alpha} \rangle - \sum_{m,i} y_{m,i,\alpha} \cdot \langle \mathbf{e}_{\pi(v_i(m)),\alpha}, \mathbf{e}_{\pi(x),\alpha} \rangle \\
&= \sum_{m,i} \{ y_{m,i,\alpha} : v_i(m) = \pi(x) \} - \sum_{m,i} \{ y_{m,i,\alpha} : \pi(v_i(m)) = \pi(x) \}.
\end{aligned}$$

Since $\mathbf{w}^{(v)} = \mathbf{0}$, the last term must be 0, giving equation

$$\sum_{m,i} \{ y_{m,i,\alpha} : v_i(m) = \pi(x) \} = \sum_{m,i} \{ y_{m,i,\alpha} : v_i(m) = x \}. \tag{16}$$

Multiplying both sides of the equation by α and adding up for all $\alpha \in \mathbb{F}$ we get

$$\begin{aligned}
\phi(x) &= \sum_{\alpha} \alpha \sum_{m,i} \{ y_{m,i,\alpha} : v_i(m) = \pi(x) \} \\
&= \sum_{\alpha} \alpha \sum_{m,i} \{ y_{m,i,\alpha} : v_i(m) = x \} \\
&= \phi(\pi(x)).
\end{aligned}$$

This proves that assignment ϕ is consistent. We claim that ϕ satisfies a $(1-\beta)$ fraction of the equations. We know that

$$\|\mathbf{w}-\mathbf{t}\| = \|\mathbf{w}^{(o)}\| + 2\|\mathbf{w}^{(e)}-\mathbf{a}\|.$$

The term corresponding to the ordinary coordinates equals

$$\|\mathbf{w}^{(o)}\| = |\{(m,i,\alpha) : y_{m,i,\alpha} \neq 0\}|.$$

Divide $\|\mathbf{w}^{(o)}\| = w_A + w'_A$ into two components

$$\begin{aligned} w_A &= |\{(m,i,\alpha) : m \in A, y_{m,i,\alpha} \neq 0\}| \\ w'_A &= |\{(m,i,\alpha) : m \notin A, y_{m,i,\alpha} \neq 0\}|. \end{aligned}$$

Since $\mathbf{w}^{(p)} = \sum y_{m,i,\alpha}\mathbf{e}_{m,i} = \mathbf{a}$, and $\mathbf{a}$ is 1 at all $3n$ coordinates $(m,i) \in A \times \{1,2,3\}$, there must be at least $3n$ non-zero coefficients $y_{m,i,\alpha}$ with $m \in A$. This proves that $w_A \geq 3n$. Now consider the second term w'_A. We claim that

$$w'_A \geq 2|\{m : m \in A, \mathbf{w}^{(e)}_m \neq \phi(m)\}|. \tag{17}$$

In particular, we prove that for any $m \in A$ such that $\mathbf{w}^{(e)}_m \neq \phi(m)$, there exist $i, i' \in \{1,2,3\}$ and $m' \notin A$ such that $v_i(m) = v_{i'}(m')$ and $|\{\alpha : y_{m',i',\alpha} \neq 0\}| \geq 2$. (Notice that the resulting pairs (m',i') are all distinct, because each variable $v_i(m) \in X_A$ cannot appear in more than one equation $m \in A$.) So, fix any $m \in A$. From the definition of ϕ we have

$$\phi(m) = \sum_{i\leq 3} c_i(m) \sum_{\alpha} \alpha \sum_{m',i'} \{y_{m',i',\alpha} : v_{i'}(m') = v_i(m)\}. \tag{18}$$

The corresponding equation coordinate in $\mathbf{w}^{(e)}$ equals

$$\mathbf{w}^{(e)}_m = \sum_{i\leq 3} c_i(m) \sum_{\alpha} \alpha y_{m,i,\alpha}. \tag{19}$$

Therefore, for all $m \in A$ such that (18) and (19) differ, there are $i \leq 3$ and $\alpha \in \mathbb{F}$ such that

$$\sum_{m',i'} \{y_{m',i',\alpha} : v_{i'}(m') = v_i(m)\} \neq y_{m,i,\alpha},$$

or, subtracting $y_{m,i,\alpha}$ from both sides,

$$\sum_{m',i'} \{y_{m',i',\alpha} : v_{i'}(m') = v_i(m), (m,i) \neq (m',i')\} \neq 0.$$

Since variable $v_{i'}(m') = v_i(m)$ cannot appear in A more than once, and $m \in A$, it must be $m' \notin A$. This proves that there exist $m' \notin A$, $i, i' \leq 3$, and α such that $v_{i'}(m') = v_i(m)$ and $y_{m',i',\alpha} \neq 0$. We still need to show that $y_{m',i',\alpha'} \neq 0$ for some other $\alpha' \neq \alpha$. Look at the position coordinates. We know that $\mathbf{w}^{(p)} = 0$. In particular the (m',i') coordinate is $\mathbf{w}^{(p)}_{m',i'} = \sum_{\alpha'} y_{m',i',\alpha'} = 0$, which rearranging the terms gives

$$\sum_{\alpha' \neq \alpha} y_{m',i',\alpha'} = -y_{m',i',\alpha} \neq 0.$$

So, $y_{m',i',\alpha'} \neq 0$ for some $\alpha' \neq \alpha$ and $|\{\alpha : y_{m',i',\alpha} \neq 0\}| \geq 2$. This proves (17).

We use bound (17) to evaluate the distance $\|\mathbf{w}-\mathbf{t}\|$:

$$\begin{aligned}\|\mathbf{w}-\mathbf{t}\| &\geq w_A + w'_A + 2\|\mathbf{w}^{(e)}-\mathbf{a}\| \\ &\geq 3n + 2|\{m : m \in A, \mathbf{w}_m^{(e)} \neq \phi(m)\}| + 2|\{m : \mathbf{w}_m^{(e)} \neq b(m)\}| \\ &\geq 3n + 2|\{m : m \in A, \mathbf{w}_m^{(e)} \neq \phi(m) \neq b(m)\}| + 2|\{m : m \in A, \mathbf{w}_m^{(e)} = \phi(m) \neq b(m)\}| \\ &= 3n + 2|\{m : m \in A, \phi(m) \neq b(m)\}|.\end{aligned}$$

Since $\|\mathbf{w}-\mathbf{t}\| = 3n+2\beta n$, we get $|\{m : m \in A, \phi(m) \neq b(m)\}| \leq \beta n$, i.e., the fraction of unsatisfied equations is at most β. $\square$

We are ready to prove the first inapproximability result for NCPP.

Theorem 3 *For any prime power q, and any $\gamma \leq 1 + (2/3)(1-1/q)$, approximating the nearest codeword problem with preprocessing over $GF(q)$ within γ is NP-hard. In particular, NCPP over $GF(2)$ is hard to approximate within any $\gamma < 4/3$, and NCPP over $GF(3)$ is hard to approximate within any $\gamma < 13/9$.*

Proof: The proof is by reduction from approximating the maximum number of simultaneously satisfiable equations in a linear system, when each equation contains at most three variables. Let (A, b) be such a system. We want to determine whether we can satisfy at least $(1-\epsilon)n$ equations, or at most $(1/q)+\epsilon$ equations. We build NCP instance $(\mathcal{W}, \mathbf{t})$ as in (8) and (11). Notice that the code $\mathcal{W}$ depends only on the number n of equations in the linear system. If the system is $1-\epsilon$ satisfiable, then by Lemma 2 there exists a codeword within distance $3n + 2\epsilon n$ from the target and $\text{dist}(\mathbf{t}, \mathcal{W}) \leq 3n + 2\epsilon$. On the other hand, if at most a $(1/q)+\epsilon$ fraction of the equations is satisfiable, then by Lemma 3 all codewords must be at least at distance $3n + 2(1-(1/q)-\epsilon)n$ from the target, and $\text{dist}(\mathbf{t}, \mathcal{W}) \geq 5n - 2((1/q)+\epsilon)n$. Taking ϵ small enough, one can make the gap

$$\gamma = \frac{5-(2/q)-2\epsilon}{3+2\epsilon} = 1 + \frac{2\,(1-(1/q)-2\epsilon)}{3+2\epsilon}$$

arbitrarily close to $1 + (2/3)(1-1/q)$. $\square$

3 Improving the inapproximability factor

In the previous section we showed that for any finite field $GF(q)$, the nearest codeword problem with preprocessing over $GF(q)$ is NP-hard to approximate within any factor less than $1+(2/3)(1-1/q)$. Notice that the inapproximability factor is always less than $5/3 = 1.666\ldots$. However, the inapproximability factor is much smaller for small fields. In particular, for binary and ternary alphabets $GF(2)$ and $GF(3)$, Theorem 3 only proves that NCPP is hard to approximate within factors up to $4/3 = 1.333\ldots$ and $13/9 = 1.444\ldots$ respectively. The cases $GF(2)$ and $GF(3)$ are especially important because they can be used to prove the inapproximability of the analogous problem for integer lattices, so we would like to establish the inapproximability of NCPP over these fields for the highest possible approximation factor.

In this section we prove the inapproximability of NCPP within any factor $\gamma < 5/3$, and for any (fixed) alphabet size. In particular, NCPP over $GF(2)$ is hard to approximate to within any factor less than $5/3 = 1.666\ldots$. The improvement is obtained using a general reduction from the nearest codeword problem over $GF(q^d)$ to the same problem over the smaller alphabet $GF(q)$. The reduction is pretty standard, and it has been used (more or less explicitly) several times in the literature. (See, for example, [9].) However, we were not able to find a suitable reference where the

result is conveniently stated in the form needed in this paper. For ease of reference, we state the result below, and briefly sketch the proof which is based on the standard technique of concatenating codes [10].

Theorem 4 *For any prime power q, and any positive integer d, there is a polynomial time reduction that on input a NCP instance $(\mathbf{W}, \mathbf{t})$ over $GF(q^d)$, output a NCP instance $(\mathbf{W}', \mathbf{t}')$ over $GF(q)$ such that*

$$dist(\mathbf{t}', \mathbf{W}') = (q^d - q^{d-1})\, dist(\mathbf{t}, \mathbf{W}).$$

Moreover, the new code $\mathbf{W}'$ depends only on $\mathbf{W}$, and the target $\mathbf{t}'$ depends only on $\mathbf{t}$.

Proof: Let $\mathcal{H}$ be the $[q^d, d, q^d - q^{d-1}]$ Hadamard code over $GF(q)$, defined as follows. Each codeword of $\mathcal{H}$ corresponds to a vector $\mathbf{x} \in GF(q)^d$, so the rate of the code is d. The codeword associated to $\mathbf{x}$ is obtained by evaluating all non-zero linear functions $f : GF(q)^d \to GF(q)$ at $\mathbf{x}$. Since there are q^d linear functions in d variables (including the zero function), the code has block length $q^d - 1$. Notice that any non-zero vector $\mathbf{x} \in GF(q)^d$ is mapped to 0 by exactly a $1/q$ fraction of the linear functions $f : GF(q)^d \to GF(q)$ (including the identically zero function). So, the minimum distance of the code is $q^d - q^{d-1}$, and all non-zero codewords have weight exactly $q^d - q^{d-1}$. The standard basis for the Hadamard code $\mathcal{H}$ is given by vectors $\mathbf{h}_i = f(\mathbf{e}_i)$, for $i = 1, \ldots, d$.

We use Hadamard code to map NCP instance $(\mathcal{W}, \mathbf{t})$ over $GF(q^d)$ to a corresponding instance $(\mathcal{W}', \mathbf{t}')$ over the base field $GF(q)$. Let $\alpha_1, \ldots, \alpha_d \in GF(q^d)$ be a basis for $GF(q^d)$ as a vector space over $GF(q)$ and let $\phi : GF(q^d) \to \mathcal{H}$ be the (unique) linear function such that $\phi(\alpha_i) = \mathbf{h}_i$ for all $i = 1, \ldots, d$. The concatenation function $\Diamond\mathcal{H}\colon GF(q^d)^l \to GF(q)^{l(q^d-1)}$ is given by

$$[x_1, \ldots, x_l]\Diamond\mathcal{H} = [\phi(x_1), \ldots, \phi(x_l)].$$

Function $\Diamond\mathcal{H}$ is extended to sets of vectors in the usual way $X\Diamond\mathcal{H} = \{\mathbf{x}\Diamond\mathcal{H}\colon \mathbf{x} \in X\}$, and concatenated code $\mathcal{W}\Diamond\mathcal{H}$ is just the result of applying the concatenation function $\Diamond\mathcal{H}$ to set $\mathcal{W}$. It is easy to see that a generating matrix for $\mathcal{W}\Diamond\mathcal{H}$ can be obtained applying the concatenation function to the vectors of any generating matrix for $\mathcal{W}$.

We use concatenation to define $\mathcal{W}' = \mathcal{W}\Diamond\mathcal{H}$ and $\mathbf{t}' = \mathbf{t}\Diamond\mathcal{H}$. Clearly, code $\mathcal{W}'$ depends only on the original code $\mathcal{W}$, and target $\mathbf{t}'$ depends only on $\mathbf{t}$. Moreover, since every non-zero codeword in $\mathcal{H}$ has weight exactly $q^d - q^{d-1}$, we have $\mathrm{dist}(\mathbf{t}', \mathcal{W}') = (q^d - q^{d-1})\mathrm{dist}(\mathbf{t}, \mathcal{W})$. □

The NP-hardness of NCPP over any finite field for any factor less than 5/3 easily follows from Theorem 3 and Theorem 4.

Theorem 5 *For any prime power q, and any constant $\gamma < 5/3$, approximating NCPP over $GF(q)$ within factor γ is NP-hard.*

Proof: Fix some prime power q and approximation factor $\gamma < 5/3$, and let $GF(q^d)$ be a sufficiently large extension field such that $1 + (2/3)(1 - 1/q^d) > \gamma$. By Theorem 3, NCPP over $GF(q^d)$ is hard to approximate within factor γ. We reduce NCPP over $GF(q^d)$ to NCPP over $GF(q)$, preserving the inapproximability factor. Let $(\mathbf{W}, \mathbf{t})$ be an instance of NCPP over $GF(q^d)$. Apply the transformation from Theorem 4 to NCP instance $(\mathcal{W}, \mathbf{t})$ over $GF(q^d)$ to obtain a new instance $(\mathcal{W}', \mathbf{t}')$ over the base field $GF(q)$ such that

$$\mathrm{dist}(\mathbf{t}', \mathcal{W}') = (q^d - q^{d-1})\mathrm{dist}(\mathbf{t}, \mathcal{W}).$$

Notice that the gap between yes and no instances for the new problem $(\mathcal{W}', \mathbf{t}')$ is also γ. Moreover, code $\mathcal{W}'$ depends only on the original code $\mathcal{W}$, so an efficient (approximate) decoding procedure for $\mathcal{W}$ can be easily obtained by suitably preprocessing the new code $\mathcal{W}'$. □

4 Inapproximability of CVPP

In this section we use the inapproximability of NCPP within factors $\gamma < 5/3$ to prove that the closest vector problem for integer lattices is also hard to approximate in any ℓ_p norm. The result easily follows from the hardness of NCPP, and a general transformation from codes to lattices. This transformation is also standard, and it has been used several times in the literature before. For example, it is implicitly used in [8] to establish the inapproximability of NCP within factors $O(n^{\log^{1-\epsilon} n})$. For future reference, we state the result below.

Theorem 6 *There is a polynomial time computable transformation that on input a generator matrix* $\mathbf{W}$ *over* $GF(q)$ *(for* $q = 2$ *or* $q = 3$*) outputs an integer lattice basis* $\mathbf{B}$ *such that for any integer target vector* $\mathbf{t} \in \mathbb{Z}^n$*,*

$$dist_p(\mathbf{t}, \mathcal{L}(\mathbf{B})) = \sqrt[p]{dist_H(\mathbf{t} \bmod q, \mathcal{C}(\mathbf{W}))}$$

where $dist_p$ *is the* ℓ_p *distance, and* $dist_H$ *is the Hamming distance. Moreover, if* $\mathbf{v}$ *is a lattice point within* ℓ_p*-distance* $\sqrt[p]{d}$ *from* $\mathbf{t}$*, then* $\mathbf{v} \bmod 2$ *is a codeword within Hamming distance* d *from* $\mathbf{t} \bmod 2$*.*

Proof: Give a code generator matrix $\mathbf{W}$, define lattice basis

$$\mathbf{B} = \begin{bmatrix} \mathbf{W} \\ q\mathbf{I} \end{bmatrix}.$$

It is easy to see that for any integer target vector $\mathbf{t}$, the lattice vector $\mathbf{v}$ closest to $\mathbf{t}$ satisfies $v_i \in \{t_i, t_i + 1, t_i - 1\}$ for all coordinates i. (Otherwise, one can find an even closer lattice vector adding a multiple of q.) So, $\mathbf{v}$ and $\mathbf{t}$ differ in exactly $\|\mathbf{v} - \mathbf{t}\|_p^p$ positions. Since $\mathbf{v} \bmod q$ is a codeword, the Hamming distance of $\mathbf{t}$ from code $\mathcal{C}(\mathbf{W})$ is at most $\text{dist}_p(\mathbf{v}, \mathcal{L}(\mathbf{B}))^p$. On the other hand, for any codeword $\mathbf{c}$ there is a lattice vector $\mathbf{c} + q\mathbf{r}$ (with $\mathbf{r} \in \mathbb{Z}^n$) such that $v_i \in \{t_i, t_i + 1, t_i - 1\}$ for all coordinates, and the number of positions in which $\mathbf{v}$ and $\mathbf{t}$ differ equals the Hamming distance between $(\mathbf{t} \bmod q)$ and $\mathcal{C}(\mathbf{W})$. It follows that $\text{dist}_p(\mathbf{v}, \mathcal{L}(\mathbf{B}))^p \leq \text{dist}_H(\mathbf{t}, \mathcal{C}(\mathbf{W}))$. □

Theorem 6 can be used to transform NCP instances into CVP instances. Notice that the transformation is gap-preserving for the ℓ_1 norm, but for any other ℓ_p norm it reduces the gap from γ to $\sqrt[p]{\gamma}$. In particular, it reduces approximating NCP within factors $\gamma < 5/3$ to approximating CVP in the ℓ_2 norm within factors $\gamma' < \sqrt{5/3}$. Notice also that for fields other than $GF(2)$ and $GF(3)$, the above proof does not goes through, and even in the ℓ_1 it is not clear how reduce NCP over large fields to CVP. Finally, the transformation from $\mathbf{W}$ to $\mathbf{B}$ is not a reduction from the minimum distance problem to the shortest vector problem because lattice $\mathbf{B}$ always contains short vectors of length q. Reducing the minimum distance problem to the shortest vector problem (even in the exact version) is an important open problem, as it would imply (using [28]) the NP-hardness of the shortest vector problem, which is currently known to be NP-hard (in the exact [2] or approximation version [21]) only under randomized reductions.

Theorem 7 *For any* $p \geq 1$*, approximating CVPP in the* ℓ_p *norm within any factor* $\gamma < \sqrt[p]{5/3}$ *is NP-hard.*

Proof: The proof is by reduction from NCPP over $GF(2)$. The reduction is very simple. Assume there is a preprocessing function P and an efficient decoding algorithm D that solve the CVPP. We build a preprocessing function P' and a decoding algorithm D' for NCPP over $GF(2)$. The preprocessing function P', on input a code $\mathbf{W}$, invokes Theorem 4 to transform $\mathbf{W}$ into a lattice basis

$\mathbf{B}$. (Notice that if we start from a uniform sequence of codes, then we obtain a uniform sequence of lattices, because the transformation described in Theorem 6 is computable in polynomial time.) Then apply preprocessing function P to $\mathbf{B}$. The output $P(\mathbf{B})$ will be used as the preprocessed description of the original code $\mathbf{W}$. The decoding algorithm D', on input a 0-1 vector $\mathbf{t}$, uses D and $P(\mathbf{B})$ to compute a lattice vector $\mathbf{v}$ closest to $\mathbf{t}$, and output codeword $\mathbf{t}$ mod 2. It follows from Theorem 6 that if $\mathbf{v}$ is a $\sqrt[p]{\gamma}$-approximate solution to CVP problem $(\mathbf{B}, \mathbf{t})$, then $\mathbf{t}$ mod 2 is a γ-approximate solution to NCP problem $(\mathbf{W}, \mathbf{t})$. Since, by Theorem 5, NCPP over $GF(2)$ is NP-hard to approximate within any factor less than $5/3$, this proves the NP-hardness of approximating CVPP in the ℓ_p norm within any factor less than $\sqrt[p]{5/3}$. $\square$

As a remark, a similar reduction (with $3\mathbf{I}$ instead of $2\mathbf{I}$) would also hold if we started from NCP over $GF(3)$. However, the reduction does not extend to even larger alphabets without decreasing the approximation factor. So, it is not clear how to prove the inapproximability of CVPP for factors close to $\sqrt[p]{5/3}$ directly from Theorem 3, without going through the extension field and concatenating code construction of Theorem 5.

5 Conclusion

We proved that NCPP and CVPP are hard to approximate within some constant factor, reducing the gap between known hardness results for lattice and coding problems with or without preprocessing. The inapproximability factors achieved are any $\gamma < 5/3$ for NCPP over any finite field $GF(q)$, and any $\gamma < \sqrt[q]{5/3}$ for CVPP in the ℓ_p norm. Very recently, the results for NCPP over $GF(2)$ have been improved by Regev [26] (using techniques from Probabilistically Checkable Proofs) to any $\gamma < 3$, leading also to stronger inapproximability results for CVPP (in the ℓ_p norm) within any $\gamma < \sqrt[p]{3}$. An interesting question is whether NCPP and CVPP are hard to approximate within any constant factor, or subpolynomial functions $n^{O(1/\log\log n)}$ [8], matching known results for NCP and CVP. A common technique to increase the inapproximability factor of lattice and coding problems in that of using tensor product constructions. For example, [3] shows that approximating NCP instance $(\mathbf{C}, \mathbf{t})$ within factor γ can be reduced to approximating NCP instance $(\mathbf{C} \otimes \mathbf{t}, \mathbf{t} \otimes \mathbf{t})$ within factor γ^2, where $\otimes$ denotes the tensor product operation. (The reader is referred to [3, 9] for the definition of tensor product.) As a different example, [9] reduces the problem of approximating the minimum distance of code $\mathbf{C}$ within factor γ to approximating the minimum distance of $\mathbf{C} \otimes \mathbf{C}$ within factor γ^2. Using this construction repeatedly, one can boost a constant inapproximability factor $\gamma > 1$ to any constant in polynomial time, or subpolynomial factors $n^{O(1/\log^\epsilon n)}$ in quasi polynomial time. Unfortunately, this technique does not apply to NCPP or CVPP because code $\mathbf{C} \otimes \mathbf{t}$ cannot be preprocessed without first knowing the target vector $\mathbf{t}$. A similar amplification technique where the new code (or lattice) $\mathbf{C}'$ depends only on the original code $\mathbf{C}$, would immediately give a hardness result for approximating NCPP and CVPP within *any* constant approximation factor, but no such technique is currently known.

Another interesting problem is proving the hardness of the nearest codeword problem when the distance between the target and the code is small relative to the minimum distance d of the code. In [9], this problem, called the *Relatively Near Codeword problem* (RNC), is shown NP-hard (under randomized reductions) when the target is within distance $(1/2 + \epsilon)d$ from the code. The proof of Dumer, Micciancio and Sudan [9] is by reduction from approximating NCP, and has the property that the RNC code depends only on the NCP code. Therefore, techniques in [9] could be used to prove the hardness of RNC with preprocessing. However, the reduction in [9] requires the inapproximability of NCPP within factors bigger than 2 in order to give interesting results about RNC with preprocessing (i.e., hardness results in which the distance of the target from the

code is not bigger than the minimum distance of the code). The recent improved inapproximability results of [26], imply that RNC with preprocessing is NP-hard to approximate within some constant factor even if the distance of the target from the code is less than the minimum distance of the code. Further improving the inapproximability of NCPP to any constant factor would imply that RNC with preprocessing is also NP-hard (again, under randomized reductions [9]) to approximate within any constant even when the distance of the target from the code is arbitrarily close to half the minimum distance.

References

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 99–108, Philadelphia, Pennsylvania, 22–24 May 1996. ACM.

[2] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 10–19, Dallas, Texas, 23–26May 1998.

[3] S. Arora, L. Babai, J. Stern, and E. Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, Apr. 1997. Preliminary version in FOCS'93.

[4] P. Berman and M. Karpinski. Approximation hardness of bounded degree min-csp and min-bisection. In P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *29th International Colloquium on Automata, Languages and Programming,, ICALP 2002*, volume 2380 of *Lecture Notes in Computer Science*, pages 623–632, Malaga, Spain, 8–13 July 2002. Springer.

[5] J. Bruck and M. Naor. The hardness of decoding linear codes with preprocessing. *IEEE Transactions on Information Theory*, 36(2):381–385, Mar. 1990.

[6] J.-Y. Cai and A. P. Nerurkar. An improved worst-case to average-case connection for lattice problems (extended abstract). In *38th Annual Symposium on Foundations of Computer Science*, pages 468–477, Miami Beach, Florida, 20–22 Oct. 1997. IEEE.

[7] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C.-P. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2(2):111–128, 1992.

[8] I. Dinur, G. Kindler, and S. Safra. Approximating CVP to within almost-polynomial factors is NP-hard. In *39th Annual Symposium on Foundations of Computer Science*, Palo Alto, California, 7–10 Nov. 1998. IEEE.

[9] I. Dumer, D. Micciancio, and M. Sudan. Hardness of approximating the minimum distance of a linear code. In *40th Annual Symposium on Foundations of Computer Science*, pages 475–484, New York, New York, 17–19 Oct. 1999. IEEE.

[10] G. D. Forney Jr. *Concatenated Codes.* Number 37 in Research Monograph. MIT Press, Cambridge, Massachusetts, 1966.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability, a guide to the theory of NP-completeness*. A Series of books in the mathematical sciences. W. H. Freeman, San Francisco, 1979.

[12] J. Håstad. Some optimal inapproximability results. Technical Report 97-37, Electronic Colloquium on Computational Complexity, 1997. Preliminary version in Proceedings of the 29th ACM Symposium on Theory of Computing.

[13] R. Kannan. *Annual Reviews of Computer Science*, volume 2, chapter Algorithmic Geometry of numbers, pages 231–267. Annual Review Inc., Palo Alto, California, 1987.

[14] J. C. Lagarias, H. W. Lenstra, Jr., and C.-P. Schnorr. Korkine-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–348, 1990.

[15] S. Landau and G. L. Miller. Solvability by radicals is in polynomial time. *J. Comput. Syst. Sci.*, 30(2):179–208, Apr. 1985. Preliminary version in STOC'83.

[16] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:513–534, 1982.

[17] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, Nov. 1983.

[18] A. Lobstein. The hardness of solving subset sum with preprocessing. *IEEE Transactions on Information Theory*, 36(4):943–946, July 1990.

[19] D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Transactions on Information Theory*, 47(3):1212–1215, 2001.

[20] D. Micciancio. Improving lattice based cryptosystems using the hermite normal form. In J. Silverman, editor, *Cryptography and Lattices Conference — CaLC'2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 126–145, Providence, Rhode Island, 29–30Mar. 2001. Springer-Verlag.

[21] D. Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, Mar. 2001. Preliminary version in FOCS 1998.

[22] D. Micciancio. Improved cryptographic hash functions with worst-case/average-case connection. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing – STOC'02*, Montreal, Canada, May 2002. ACM, ACM. Abstract also in Computational Complexity – CoCo'02.

[23] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002.

[24] A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In C. Pomerance, editor, *Cryptology and Computational Number Theory*, volume 42 of *Proc. Symp. Appl. Math.*, pages 75–88. AMS, 1990.

[25] A. M. Odlyzko and H. te Riele. Disproof of the mertens conjecture. *J. reine angew*, 357:138–160, 1985.

[26] O. Regev. Improved inapproximability of lattice and coding problems with preprocessing. Personal communication, June 2002.

[27] C.-P. Schnorr. Factoring integers and computing discrete logarithms via Diophantine approximation. In D. W. Davies, editor, *Advances in Cryptology—EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 281–293. Springer-Verlag, 8–11 Apr. 1991.

[28] A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, STOC'97*, pages 92–109, El Paso, Texas, 4–6 May 1997. ACM, ACM.