Electronic Colloquium on Computational Complexity, Report No. 65 (2008)

Deterministic Approximation Algorithms for the Nearest Codeword Problem

Noga Alon
Tel Aviv University and
Institute for Advanced Study
nogaa@tau.ac.il

Rina Panigrahy
Microsoft Research
rina@microsoft.com

Sergey Yekhanin
Microsoft Research
yekhanin@microsoft.com

Abstract

The Nearest Codeword Problem (NCP) is a basic algorithmic question in the theory of error-correcting codes. Given a point $v \in \mathbb{F}_2^n$ and a linear space $L \subseteq \mathbb{F}_2^n$ of dimension k NCP asks to find a point $l \in L$ that minimizes the (Hamming) distance from v. It is well-known that the nearest codeword problem is NP-hard. Therefore approximation algorithms are of interest. The best efficient approximation algorithms for the NCP to date are due to Berman and Karpinski. They are a deterministic algorithm that achieves an approximation ratio of $O(k/c)$ for an arbitrary constant c, and a randomized algorithm that achieves an approximation ratio of $O(k/\log n)$.

In this paper we present new deterministic algorithms for approximating the NCP that improve substantially upon the earlier work. Specifically, we obtain:

- *A polynomial time $O(n/\log n)$-approximation algorithm;*
- *An $n^{O(s)}$ time $O(k \log^{(s)} n/\log n)$-approximation algorithm, where $\log^{(s)} n$ stands for s iterations of $\log$, e.g., $\log^{(2)} n = \log\log n$;*
- *An $n^{O(\log^* n)}$ time $O(k/\log n)$-approximation algorithm.*

We also initiate a study of the following Remote Point Problem (RPP). Given a linear space $L \subseteq \mathbb{F}_2^n$ of dimension k RPP asks to find a point $v \in \mathbb{F}_2^n$ that is far from L. We say that an algorithm achieves a remoteness of r for the RPP if it always outputs a point v that is at least r-far from L. In this paper we present a deterministic polynomial time algorithm that achieves a remoteness of $\Omega(n \log k/k)$ for all $k \leq n/2$. We motivate the remote point problem by relating it to both the nearest codeword problem and the matrix rigidity approach to circuit lower bounds in computational complexity theory.

1 Introduction

The Nearest Codeword Problem (NCP) is a basic algorithmic question in the theory of error-correcting codes. Given a point $v \in \mathbb{F}_2^n$ and a linear space $L \subseteq \mathbb{F}_2^n$ of dimension k NCP asks to find a point $l \in L$ that minimizes the (Hamming) distance from v. The nearest codeword problem is equivalent to the problem of finding a vector $x \in \mathbb{F}_2^k$ that minimizes the number of unsatisfied linear equations in the system $xG = v$, given a matrix $G \in \mathbb{F}_2^{k \times n}$ and a vector $v \in \mathbb{F}_2^n$. It is well-known that the NCP is NP-hard. Therefore approximation algorithms are of interest.

The best efficient approximation algorithms for the NCP to date are due to Berman and Karpinski [3]. They are a deterministic algorithm that achieves an approximation ratio of $O(k/c)$ for an arbitrary constant c, and a

ISSN 1433-8092

randomized algorithm that achieves an approximation ratio of $O(k/\log n)$.[1] There has been a substantial amount of work on hardness of approximation for the NCP [2, 5, 4, 1]. The best result to date is due to Dinur et. al. [4]. It shows NP-hardness of approximating the NCP to within $n^{O(1/\log\log n)}$. Alekhnovich [1] has made a conjecture that implies inapproximability of the NCP to within $n^{1-\epsilon}$, for every $\epsilon > 0$.

In this paper we develop new *deterministic* algorithms for approximating the NCP. Specifically, we obtain:

1. A polynomial time $O(n/\log n)$-approximation algorithm;
2. An $n^{O(s)}$ time $O(k\log^{(s)} n/\log n)$-approximation algorithm, where $\log^{(s)} n$ stands for s iterations of $\log$, e.g., $\log^{(2)} n = \log\log n$;
3. An $n^{O(\log^* n)}$ time $O(k/\log n)$-approximation algorithm.

Our first algorithm matches the performance of the randomized algorithm of [3] for $k = \Omega(n)$. This is the regime that is of primary importance for the coding theory applications. Our second algorithm improves substantially upon the deterministic algorithm of [3], and nearly matches the randomized algorithm of [3] in terms of the approximation ratio. Finally, our third algorithm has the same approximation ratio as the randomized algorithm of [3] and a slightly super-polynomial running time. All our algorithms (as well as other known algorithms for the NCP in the literature) can be easily generalized to fields other than $\mathbb{F}_2$.

Remote point problem. In this work we also initiate a study of the following Remote Point Problem (RPP). Given a linear space $L \subseteq \mathbb{F}_2^n$ of dimension k RPP asks to find a point $v \in \mathbb{F}_2^n$ that is far from L. We say that an algorithm achieves a remoteness of r for the RPP if it always outputs a point v that is at least r-far from L. We present a deterministic polynomial time algorithm that achieves a remoteness of $\Omega(n\log k/k)$ for all $k \leq n/2$. Our algorithm for the remote point problem is closely related to our first approximation algorithm for the nearest codeword problem.

We motivate the remote point problem by relating it to the matrix rigidity approach to circuit lower bounds in computational complexity theory. The notion of matrix rigidity was introduced by Leslie Valiant in 1977 [10]. In what follows we say that a set $A \subseteq \mathbb{F}_2^n$ is r-far from a linear space $L \subseteq \mathbb{F}_2^n$ if A contains a point that is r-far from L. (Observe, that this is quite different from the usual notion of distance between sets.) Valiant called a set $A \subseteq \mathbb{F}_2^n$ rigid if for some fixed $\epsilon > 0$, A is n^ϵ-far from every linear space $L \subseteq \mathbb{F}_2^n$, $\dim L = n/2$. Valiant showed that if a set $A \subseteq \mathbb{F}_2^n$ is rigid and $|A| = O(n)$; then the linear transformation from n bits to $|A|$ bits induced by a matrix whose rows are all elements of A can not be computed by a circuit of XOR gates that simultaneously has size $O(n)$ and depth $O(\log n)$.[2]

Valiant's work naturally led to the challenge of constructing a small explicit rigid set A, (since such a set yields an explicit linear map, for that we have a circuit lower bound). This challenge has triggered a long line of work. For references see [6, 7, 8, 9]. Unfortunately, after more than three decades of efforts, we are still nowhere close to constructing an explicit rigid set with the parameters needed to get implications in complexity theory. In particular there are no known constructions of sets $A \subseteq \mathbb{F}_2^n$ of size $O(n)$ that are $\omega(1)$-far from every linear space $L \subseteq \mathbb{F}_2^n$, $\dim L = n/2$. Moreover if we restrict ourselves to sets A of size n; then we do not know how to construct an explicit set that is just 3-far from every linear space of dimension $n/2$, despite the fact that a random set A of cardinality n is $\Omega(n)$-far from every such space with an overwhelming probability.

[1] In fact, Berman and Karpinski [3] only claim that their randomized algorithm achieves a $O(k/\log k)$ approximation. However it is immediate from their analysis that they also get a $O(k/\log n)$ approximation.

[2] The original paper of Valinat [10] and the follow-up papers use a somewhat different language. Specifically, they talk about matrices A whose rank remains no less than $n/2$ even after every row is modified in less than n^ϵ coordinates; rather than about sets A that for every linear space $L \subseteq \mathbb{F}_2^n$, $\dim L = n/2$ contain a point $a \in A$ that is n^ϵ-far from L. However, it is not hard to verify that the two concepts above are equivalent.

In this paper we propose the remote point problem as an intermediate challenge that is less daunting than the challenge of designing a small rigid set, and yet could help us develop some insight into the structure of rigid sets. Recall that a rigid set is a set that is simultaneously n^ϵ-far from *every* linear space L, $\dim L = n/2$. Given the state of art with constructions of explicit rigid sets we find it natural to consider an easier algorithmic Remote Set Problem (RSP) where we are given a single linear space L, and our goal is to design an $O(n)$-sized set $A_L \subseteq \mathbb{F}_2^n$ that is n^ϵ-far from L. Clearly, if we knew how to construct explicit rigid sets, we could solve the RSP without even looking at the input. The remote point problem is a natural special case of the remote set problem. Here we are given a linear space $L \subseteq \mathbb{F}_2^n$ and need to find a single point that is far from L.

In this paper we present an algorithm that for every linear space $L \subseteq \mathbb{F}_2^n$, $\dim L = n/2$ generates a point that is $\Omega(\log n)$-far from L. (For spaces L of dimension $k < n/2$, our algorithm generates a point of distance at least $\Omega(n \log k/k)$ from L.) We are not aware of efficient algorithms to generate points (or $O(n)$-sized collections of points) further away from a given arbitrary linear space of dimension $n/2$.

Organization. We present our first approximation algorithm for the NCP in section 2. We present our second and third algorithms in section 3. We present our algorithm for the remote point problem in section 4.

2 An $O(n/\log n)$-approximation algorithm

We start with the formal statements of the NCP and of our main result.

Nearest codeword problem.

- INSTANCE: A linear code $L = \{xG \mid x \in \mathbb{F}_2^k\}$ given by a generator matrix $G \in \mathbb{F}_2^{k \times n}$ and a vector $v \in \mathbb{F}_2^n$.
- SOLUTION: A codeword $l \in L$.
- OBJECTIVE FUNCTION (to be minimized): The Hamming distance $d(l, v)$.

Theorem 1 *Let $c \geq 1$ be an arbitrary constant. There exists a deterministic $n^{O(c)}$ time $\lceil n/c \log n \rceil$-approximation algorithm for the NCP.*

In order to proceed with the proof we need the following notation:

- For a positive integer d, let $B_d = \{x \in \mathbb{F}_2^n \mid d(0^n, x) \leq d\}$ denote a Hamming ball of radius d.
- For a collection of vectors $M \subseteq \mathbb{F}_2^n$, let $\mathrm{Span}(M)$ denote the smallest linear subspace of $\mathbb{F}_2^n$ containing M.
- For sets $A, B \subseteq \mathbb{F}_2^n$, we define $A + B = \{a + b \mid a \in A, b \in B\}$.

The next lemma is the core of our algorithm. It shows that a d-neighborhood of a linear space L can be covered by a (small) number of linear spaces M_S of larger dimension, in such a way that no linear space M_S contains points that are too far from L.

Lemma 2 *Let L be a linear space, and $d \leq t$ be positive integers. Let $B_1 \setminus \{0^n\} = \bigcup_{i=1}^{t} B_1^i$ be an arbitrary partition of the set of n unit vectors into t disjoint classes each of size $\lceil n/t \rceil$ or $\lfloor n/t \rfloor$. For every $S \subseteq [t]$ such that $|S| = d$ let $M_S = \mathrm{Span}\left(L \cup \left(\bigcup_{i \in S} B_1^i\right)\right)$. Then*

$$L + B_d \subseteq \bigcup_S M_S \subseteq L + B_{d\lceil n/t \rceil}, \tag{1}$$

where S runs over all subsets of $[t]$ of cardinality d.

Proof: We first show the left containment. Let v be an arbitrary vector in $L+B_d$. We have $v = l+e_{j_1}+\ldots+e_{j_{d'}}$, where $d' \leq d$, all e_{j_r} are unit vectors and $l \in L$. For every $r \in [d']$ let $i_r \in [t]$ be such that $j_r \in B_1^{i_r}$. Consider a set $S \subseteq [t]$ such that $|S| = d$ and $i_1, \ldots, i_{d'} \in S$. It is easy to see that $v \in M_S$.

We proceed to the right containment. Let $S = \{i_1, \ldots, i_d\}$ be an arbitrary subset of $[t]$ of cardinality d. Recall that the cardinality of every set $B_1^{i_r}$, $r \in [d]$ is at most $\lceil n/t \rceil$. Therefore every element $v \in M_S$ can be expressed as a sum $v = l + y$, where $l \in L$ and y is a sum of at most $d\lceil n/t \rceil$ unit vectors. Thus $v \in L + B_{d\lceil n/t \rceil}$. ■

We are now ready to proceed with the proof of the theorem.

Proof of theorem 1: Observe that if the point v is more than $c \log n$-far from L; then any vector in L (for instance, the origin) is an $\lceil n/c \log n \rceil$-approximation for v. Let us assume that $d(v, L) \leq c \log n$ and set $t = \lceil c \log n \rceil$. Our algorithm iterates over values $d \in [0, \lceil c \log n \rceil]$. For each d we generate all linear spaces $M_S, S \subseteq [t], |S| = d$ as defined in lemma 2. We check whether v is contained in one of those spaces. Lemma 2 implies that after at most $d(v, L)$ iterations we get $v \in M_S$, for some $S = \{i_1, \ldots, i_d\}$. We expand v as a sum $v = l + y$ where $l \in L$ and y is a sum of at most $d\lceil n/c \log n \rceil$ unit vectors from $\bigcup_{r=1}^{d} B_1^{i_r}$. Obviously, $d(v, l) \leq d(v, L)\lceil n/c \log n \rceil$. We report l as our $\lceil n/c \log n \rceil$-approximation for v. The pseudo-code is below.

Set $t = \lceil c \log n \rceil$;
For every $d \in [0, c \log n]$
  For every $S = \{i_1, \ldots, i_d\} \subseteq [t]$ such that $|S| = d$
    If $v \in M_S$ **Then**
      Begin
        Represent v as $v = l + y$, where $l \in L$ and y is a sum of unit vectors from $\bigcup_{r=1}^{d} B_1^{i_r}$;
        Output l;
        Terminate;
      End
Output 0^n;

It is easy to see that the algorithm above runs in time $n^{O(c)}$. The first loop makes $O(c \log n)$ iterations. The second loop makes at most $2^{\lceil c \log n \rceil} = n^{O(c)}$ iterations. Finally, the internal computation runs in $n^{O(1)}$ time. ■

3 A recursive $O(k \log^{(s)} n / \log n)$-approximation algorithm

The goal of this section is to prove the following

Theorem 3 *Let $s \geq 1$ be an integer and $c \geq 1$ be an arbitrary constant. There exists a deterministic $n^{O(cs)}$ time $\lceil k \log^{(s)} n / c \log n \rceil$-approximation algorithm for the NCP, where the constant inside the O-notation is absolute and $\log^{(s)} n$ denotes s iterations of the $\log$ function.*

Proof: Our proof goes by induction on s and combines ideas from our $O(n/\log n)$-approximation algorithm of section 2 with ideas from the deterministic approximation algorithm of Berman and Karpinski [3]. We start with some notation.

- Let $x^*G = l^* \in L$ denote some fixed optimal approximation of v by a vector in L.
- Let $E = \{i \in [n] \mid l_i^* \neq v_i\}$ be the set of coordinates where l^* differs from v.
- In what follows we slightly abuse the notation and use the letter G to denote the multi-set of columns of the generator matrix of L (as well as the generator matrix itself).

- We call a partition of the multi-set $G = \bigcup_{i}^{h} G_i$ into disjoint sets *regular* if for every $i \in [h]$, the vectors in G_i are linearly independent and:

$$\mathrm{Span}(G_i) = \mathrm{Span}\left(\bigcup_{j \geq i}^{h} G_j\right). \tag{2}$$

  Again, in what follows we slightly abuse the notation and use symbols G_i, $i \in [h]$ to denote the sets of columns of the generator matrix, the corresponding subsets of $[n]$, and the sub-matrices of the generator matrix of L.

- We denote the restriction of a vector $u \in \mathbb{F}_2^n$ to coordinates in a set $S \subseteq [n]$, by $u\,|_S \in \mathbb{F}_2^{|S|}$.

The following claim (due to Berman and Karpinski [3]) constitutes the base case of the induction. We include the proof for the sake of completeness.

Base case of the induction: Let $c \geq 1$ be an arbitrary constant. There exists a deterministic $n^{O(c)}$ time $\lceil k/c \rceil$-approximation algorithm for the NCP.

Proof of the base case: We start with an informal description of the algorithm. Our goal is to "approximately" recover x^* from v (which is a "noisy" version of l^*). Recall that l^* and v differ in coordinates that belong to E. We assume that $|E| < n/\lceil k/c \rceil$ since otherwise any vector in the space L is a valid $\lceil k/c \rceil$-approximation for v. The algorithm has two phases. During the first phase we compute a regular partition of the multi-set G. Note that such a partition necessarily has at least $h \geq n/k$ classes. Therefore there is a class $G_i, i \in [h]$ such that

$$|G_i \cap E| \leq (n/\lceil k/c \rceil)/(n/k) \leq c.$$

During the second phase we iterate over all classes $G_i, i \in [h]$ of the regular partition, trying to "fix" the differences between $v\,|_{G_i}$ and $l^*\,|_{G_i}$ and thus "approximately" recover x^*. More specifically, for every $i \in [h]$ we solve the system $xG_i = u$ for x, for every u that differs from $v\,|_{G_i}$ in up to c coordinates. (In cases when the system $xG_i = u$ happens to be under-determined we take an arbitrary single solution.) This way every class in the regular partition gives us a number of candidate vectors x. In the end we select a single vector that yields the best approximation for v.

To see that the algorithm indeed produces a valid $\lceil k/c \rceil$-approximation for v, consider the smallest index i such that $|G_i \cap E| \leq c$. Note that one of the linear systems that we are going to solve while processing the i-th class of the regular partition is $xG_i = l^*\,|_{G_i}$. Let x be an arbitrary solution of the above system. Clearly,

$$d(xG, v) = \sum_{j=1}^{i-1} d\left(xG_j, v\,|_{G_j}\right) + \sum_{j=i}^{h} d\left(xG_j, v\,|_{G_j}\right). \tag{3}$$

However for every $j \leq i-1$ we have

$$d\left(xG_j, v\,|_{G_j}\right) \leq k \leq c\lceil k/c \rceil \leq d\left(l^*\,|_{G_j}, v\,|_{G_j}\right)\lceil k/c \rceil, \tag{4}$$

by our choice of i. Also, $xG_i = l^*\,|_{G_i}$ and formula (2) yield

$$xG_j = l^*\,|_{G_j}, \tag{5}$$

for all $j \geq i$. Combining formulae (4), (5) and (3) we get $d(xG, v) \leq d(l^*, v)\lceil k/c \rceil$ and thus xG is a $\lceil k/c \rceil$-approximation for v. The pseudo-code of the algorithm is below:

Obtain a regular partition $G = \bigcup_{i \in h} G_i$;
Set $x_{\text{best}} = 0^k$;
For every $i \in [h]$
  For every vector y in $\mathbb{F}_2^{|G_i|}$ such that the Hamming weight of y is at most c
    Begin
      Find an $x \in \mathbb{F}_2^k$ such that $xG_i = v\,|_{G_i} + y$;
      If $d(xG, v) < d(x_{\text{best}}G, v)$ **Then Set** $x_{\text{best}} = x$;
    End
Output $x_{\text{best}}G$;

It is easy to see that the algorithm above runs in time $n^{O(c)}$. The first loop makes $O(n)$ iterations. The second loop makes at most n^c iterations. Finally, obtaining a regular partition and the internal computation both run in $n^{O(1)}$ time.

We now proceed to the induction step.

Induction step: Let $s \geq 1$ be an integer and $c \geq 1$ be an arbitrary constant. Suppose there exists a deterministic $n^{O(cs-c)}$ time $\lceil k \log^{(s-1)} n / c \log n \rceil$-approximation algorithm for the NCP; then there exists deterministic $n^{O(cs)}$ time $\lceil k \log^{(s)} n / c \log n \rceil$-approximation algorithm for the NCP.

Proof of the induction step: The high level idea behind our algorithm is to reduce the nearest codeword problem on an instance (G, v) to $n^{O(c)}$ (smaller) instances of the problem and to solve those instances using the algorithm from the induction hypothesis.

We start in a manner similar to the proof of the base case. Our goal is to "approximately" recover the vector x^* from v (which is a "noisy" version of l^*). Recall that l^* and v differ in coordinates that belong to E. We assume that $|E| < n/\lceil k \log^{(s)} n / c \log n \rceil$ since otherwise any vector in the space L is a valid $\lceil k \log^{(s)} / c \log n \rceil$-approximation for v. Our algorithm has two phases. During the first phase we compute a regular partition of the multi-set G. Note that such a partition necessarily has at least $h \geq n/k$ classes. Therefore there is a class $G_i, i \in [h]$ such that

$$|G_i \cap E| \leq (n/\lceil k \log^{(s)} n / c \log n \rceil)/(n/k) \leq c \log n / \log^{(s)} n.$$

During the second phase we iterate over all classes $G_i, i \in [h]$ of the regular partition, trying to locate a large subset $W \subseteq G_i$ such that $l^*\,|_W = v\,|_W$. We use such a subset to restrict our optimization problem to $x \in \mathbb{F}_2^k$ that satisfy $xG\,|_W = v\,|_W$ and thus obtain a smaller instance of the NCP. More formally, during the second phase we:

1. Set
$$b = \left\lfloor \frac{c \log n}{\log^{(s)} n} \right\rfloor, \qquad t = \left\lceil \frac{2c \log n \log^{(s-1)} n}{\log^{(s)} n} \right\rceil. \tag{6}$$
2. Set $x_{\text{best}} = 0^k$.
3. For every $i \in [h]$:
4. Set $G' = \bigcup_{j \geq i} G_j$.
   (a) If $k \geq t$ then
      i. Split the class G_i into a disjoint union of t sets $G_i = \bigcup_{r=1}^{t} G_i^r$, each of size $\lceil |G_i|/t \rceil$ or $\lfloor |G_i|/t \rfloor$.
      ii. For every $S \subseteq [t]$ such that $|S| = b$, set $W = \bigcup_{r \in [t] \setminus S} G_i^r$:

iii. Consider an affine optimization problem of finding an $x \in \mathbb{F}_2^k$ that minimizes $d\left(xG', v\,|_{G'}\right)$, subject to $xG\,|_W = v\,|_W$. Properties of the regular partition imply that here we are minimizing over an affine space L' of dimension $|G_i| - |W|$, in $\mathbb{F}_2^{|G'|}$.

iv. Turn the problem above into a form of an NCP (in $\mathbb{F}_2^n$, padding both the target vector v and the matrix G' with zeros) and solve it approximately for x using the algorithm from the induction hypothesis. (Note that every affine optimization problem of minimizing $d(xJ + z, v)$ over x for $J \in \mathbb{F}_2^{k\times n}$ and $z, v \in \mathbb{F}_2^n$, can be easily turned into a form of an NCP, i.e., the problem of minimizing $d(xJ, v + z)$ over $x \in \mathbb{F}_2^k$.

v. If $d(xG, v) < d(x_{\rm best}G, v)$ then set $x_{\rm best} = x$.

(b) Else

i. For every vector y in $\mathbb{F}_2^{|G_i|}$ such that the Hamming weight of y is at most b :

ii. Find an $x \in \mathbb{F}_2^k$ such that $xG_i = v\,|_{G_i} + y$;

iii. If $d(xG, v) < d(x_{\rm best}G, v)$ then set $x_{\rm best} = x$.

5. Output $x_{\rm best}G$.

We now argue that the algorithm above indeed obtains a valid $\lceil k \log^{(s)} n / c \log n\rceil$-approximation for the NCP. We first consider (the easier) case when $k < t$. Our analysis is similar to the analysis of the base case of the induction. Let $i \in [h]$ be the smallest index such that $|G_i \cap E| \le \lfloor c \log n / \log^{(s)} n \rfloor = b$. Note that one of the linear systems that we are going to solve while processing the i-th class of the regular partition is $xG_i = l^*\,|_{G_i}$. Let x be an arbitrary solution of the above system. We need to bound $d(xG, v)$ from above. Clearly,

$$d(xG, v) = \sum_{j=1}^{i-1} d\left(xG_j, v\,|_{G_j}\right) + d\left(xG', v\,|_{G'}\right). \tag{7}$$

However for every $j \le i - 1$ we have

$$d\left(xG_j, v\,|_{G_j}\right) \le k \le \frac{c\log n}{\log^{(s)} n}\left\lceil k \,/\, \left(\frac{c\log n}{\log^{(s)} n}\right)\right\rceil \le d\left(l^*\,|_{G_j}, v\,|_{G_j}\right)\left\lceil \frac{k\log^{(s)} n}{c\log n}\right\rceil, \tag{8}$$

by our choice of i. Also, $xG_i = l^*\,|_{G_i}$ and formula (2) yield

$$xG' = l^*\,|_{G'}, \tag{9}$$

Combining formulae (8), (9) and (7) we get $d(xG, v) \le d(l^*, v)\lceil k \log^{(s)} n / c \log n\rceil$.

We now proceed to the $k \ge t$ case. Again, let $i \in [h]$ be the smallest index such that $|G_i \cap E| \le b$. Note that one of the sets $W \subseteq G_i$ considered when processing the class G_i will necessarily have an empty intersection with the set E. Let $x \in \mathbb{F}_2^k$ be an approximate solution of the corresponding problem of minimizing $d\left(xG', v\,|_{G'}\right)$, subject to $xG\,|_W = v\,|_W$, produced by an algorithm from the induction hypothesis. We need to bound $d(xG, v)$ from above. Formulae (7) and (8) reduce our task to bounding $d\left(xG', v\,|_{G'}\right)$. Observe that when minimizing $d\left(xG', v\,|_{G'}\right)$, subject to $xG\,|_W = v\,|_W$, we are minimizing over an affine space of dimension k', where

$$k' \le \lceil k/t\rceil b \le \left\lceil \frac{k\log^{(s)} n}{2c\log n \log^{(s-1)} n}\right\rceil \frac{c\log n}{\log^{(s)} n}.$$

Note that $k \ge t$ implies

$$\left\lceil \frac{k\log^{(s)} n}{2c\log n \log^{(s-1)} n}\right\rceil \le \frac{k\log^{(s)} n}{c\log n \log^{(s-1)} n}.$$

Therefore $k' \leq k/\log^{(s-1)} n$ and the approximation algorithm from the induction hypothesis yields a $\lceil k/c\log n\rceil$-approximate solution, i.e.,

$$d\left(xG', v\,|_{G'}\right) \leq d\left(l^*\,|_{G'}, v\,|_{G'}\right)\lceil k/c\log n\rceil. \tag{10}$$

Combining formulae (8), (10) and (7) we get $d(xG, v) \leq d(l^*, v)\lceil k\log^{(s)} n/c\log n\rceil$.

To estimate the running time note that the external loop of our algorithm makes $O(n)$ iterations and the internal loop makes at most $\binom{t}{b}$ iterations where each iteration involves a recursive $n^{O(cs-c)}$ time call if $k \geq t$. It is easy to see that

$$\binom{t}{b} \leq (et/b)^b \leq \left(\frac{4ec\log n\log^{(s-1)} n}{\log^{(s)} n}\frac{c\log^{(s)} n}{\log n}\right)^{c\log n/\log^{(s)} n} = n^{O(c)},$$

where the second inequality follows from $b \leq t/2$ and $t \leq 4c\log n\log^{(s-1)} n/\log^{(s)} n$. Combining the estimates above we conclude that the total running time of our algorithm is $n^{O(cs)}$. ■

Choosing $s = \lceil\log^* n\rceil$ in theorem 3 we obtain

Theorem 4 *Let $c \geq 1$ be an arbitrary constant. There exists a deterministic $n^{O(c\log^* n)}$ time $\lceil k/c\log n\rceil$-approximation algorithm for the NCP.*

4 The remote point problem

We start with a formal statement of the remote point problem.

Remote point problem.

- INSTANCE: A linear code $L = \{xG \mid x \in \mathbb{F}_2^k\}$ given by a generator matrix $G \in \mathbb{F}_2^{k\times n}$.
- SOLUTION: A point $v \in \mathbb{F}_2^n$.
- OBJECTIVE FUNCTION (to be maximized): The Hamming distance $d(L, v)$ from the code L to a point v.

We start with an algorithm that generates $c\log n$-remote points for linear spaces of dimension $k \leq n/2$.

Theorem 5 *Let $c \geq 1$ be an arbitrary constant. There exists a deterministic $n^{O(c)}$ time algorithm that for a given linear space $L \subseteq \mathbb{F}_2^n, \dim L \leq n/2$ generates a point v such that $d(L, v) \geq c\log n$, provided n is large enough.*

Proof: At the first phase of our algorithm we set $d = \lceil c\log n\rceil$, $t = \lceil 4c\log n\rceil$ and use lemma 2 to obtain a family of $\binom{t}{d} = n^{O(c)}$ linear spaces M_S, $S \subseteq [t], |S| = d$ such that

$$L + B_{\lceil c\log n\rceil} \subseteq \bigcup_S M_S.$$

It is readily seen from the construction of lemma 2 that the dimension of every space M_S is at most $n/2 + n/3 = 5n/6$, provided n is large enough.

At the second phase of our algorithm we generate a point v that is not contained in the union $\bigcup_S M_S$, (and therefore is $\lceil c\log n\rceil$-remote from L.) We consider a potential function Φ that for every set $W \subseteq \mathbb{F}_2^n$ returns

$$\Phi(W) = \sum_S |W \cap M_S|,$$

where the sum is over all $S \subseteq [t], |S| = d$. We assume that n is large enough, so that

$$\Phi(\mathbb{F}_2^n) = \sum_S |M_S| = \binom{t}{d} |M_S| < 2^n.$$

We initially set $W = \mathbb{F}_2^n$ and iteratively reduce the size of W by a factor of two (cutting W with coordinate hyperplanes). At every iteration the value of $\Phi(W)$ gets reduced by a factor of two or more. Therefore after n iterations we arrive at a set W that contains a single point v such that $\Phi(\{v\}) = 0$. That point is $\lceil c \log n \rceil$-remote from L. For a set $W \subseteq \mathbb{F}_2^n$, $i \in [n]$, and $b \in \mathbb{F}_2$ let $W|_{x_i=b}$ denote the set $\{x \in W \mid x_i = b\}$. The pseudo-code of our algorithm is below:

Set $t = \lceil 4c \log n \rceil$ and $d = \lceil c \log n \rceil$;
Obtain $\binom{t}{d}$ linear spaces M_S as defined in lemma 2.
Set $W = \mathbb{F}_2^n$;
For every i in $[n]$
  If $\Phi(W|_{x_i=0}) \leq \Phi(W|_{x_i=1})$ **Set** $W = W|_{x_i=0}$; **Else Set** $W = W|_{x_i=1}$;
Output the single element of W;

Note that every evaluation of the potential function Φ in our algorithm takes $n^{O(c)}$ time, since all we need to do is compute the dimensions of $\binom{t}{d} = n^{O(c)}$ affine spaces $W \cap M_S$. The algorithm involves $2n$ such computations and therefore runs in $n^{O(c)}$ time. ■

Remark 6 It is easy to see that the algorithm of theorem 5 can be extended to generate points that are $c \log n$-far from a given linear space of dimension up to $(1 - \epsilon)n$ for any constant $\epsilon > 0$.

We now present our algorithm for the remote point problem in its full generality.

Theorem 7 *Let $c \geq 1$ be an arbitrary constant. There exists a deterministic $n^{O(c)}$ time algorithm that for a given linear space $L \subseteq \mathbb{F}_2^n, \dim L = k \leq n/2$ generates a point v such that $d(L, v) \geq \lfloor n/2k \rfloor \lceil 2c \log k \rceil$, provided n is large enough.*

Proof: We partition the multi-set of columns of the matrix G in $h = \lceil n/2k \rceil$ multi-sets G_i, $i \in [h]$ in such a way that every multi-set G_i, (with possibly a single exception) has size exactly $2k$. Next for all multi-sets G_i of size $2k$ we use the algorithm of theorem 5 to obtain a point v_i that is $2c \log k$-remote from the space $\{xG_i \mid x \in \mathbb{F}_2^k\} \subseteq \mathbb{F}_2^{2k}$. Finally, we concatenate all vectors v_i together (possibly padding the result with less than $2k$ zeros) to obtain a vector $v \in$ that is $\lfloor n/2k \rfloor \lceil 2c \log k \rceil$-remote from L. ■

5 Conclusion

In this paper we have given three new deterministic approximation algorithms for the nearest codeword problem. Our algorithms improve substantially upon the (previously best known) deterministic algorithm of [3]. Moreover, our algorithms approach (though do not match) the performance of the randomized algorithm of [3]. Obtaining a complete derandomization remains a challenging open problem.

We have also initiated a study of the remote point problem that asks to find a point far from a given linear space $L \subseteq \mathbb{F}_2^n$. We presented an algorithm that achieves a remoteness of $\Omega(n \log k/k)$ for linear spaces of dimension $k \leq n/2$. We consider further research on the remote point problem (and the related remote set problem) to be a promising approach to constructing explicit rigid matrices in the sense of Valiant [10].

Acknowledgement

Sergey Yekhanin would like to thank Venkat Guruswami for many helpful discussions regarding this work.

References

[1] M. Alekhnovich, "More on average case vs. approximation complexity," In *Proc. of the 44rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 298-307, 2003.

[2] S. Arora, L. Babai, J. Stern, and Z. Sweedy, "Hardness of approximate optima in lattices, codes, and linear systems," *Journal of Computer and System Sciences*, vol. 54, issue 2, pp. 317-331, 1997.

[3] P. Berman and M. Karpinski, "Approximating minimum unsatisfiability of linear equations," In *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 514-516, 2002.

[4] I. Dinur, G. Kindler, R. Raz, and S. Safra, "Approximating CVP to within almost-polynomial factors is NP-hard," *Combinatorica*, vol. 23, issue 2, pp. 205-243, 2003.

[5] I. Dumer, D. Miccancio, and M. Sudan, "Hardness of approximating the minimum distance of a linear code," *IEEE Transactions on Information Theory*, vol. 49, issue 1, pp. 22-37, 2003.

[6] J. Friedman, "A note on matrix rigidity," *Combinatorica*, vol. 13, issue 2, pp. 235-239, 1993.

[7] B. Kashin and A. Razborov, "Improved lower bounds on the rigidity of Hadamard matrices," *Mathematical Notes*, vol. 63, issue 4, pp. 471-475, 1998.

[8] S. Lokam, "Spectral methods for matrix rigidity with applications to size-depth trade-offs and communication complexity," *Journal of Computer and System Sciences*, vol. 63, issue 3, pp. 449-473, 2001.

[9] M. Shokrollahi, D. Speilman, and V. Stemann, "A remark on matrix rigidity," *Information Processing Letters*, vol. 64, issue 6, pp. 283-285, 1997.

[10] L. Valiant, "Graph-theoretic arguments in low level complexity," *Proc. of 6th Symposium on Mathematical Foundations of Computer Science (MFCS)*, pp. 162-176, 1977.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/