Shift Finding in Sub-linear Time

Alexandr Andoni
MSR SVC

Haitham Hassanieh
MIT

Piotr Indyk
MIT

Dina Katabi
MIT

andoni@microsoft.com, {haithamh,indyk,dk}@mit.edu

Abstract

We study the following basic pattern matching problem. Consider a "code" sequence $\mathbf{c}$ consisting of n bits chosen uniformly at random, and a "signal" sequence $\mathbf{x}$ obtained by shifting $\mathbf{c}$ (modulo n) and adding noise. The goal is to efficiently recover the shift with high probability. The problem models tasks of interest in several applications, including GPS synchronization and motion estimation.

We present an algorithm that solves the problem in time $\tilde{O}(n^{(f/(1+f)})$, where $\tilde{O}(N^f)$ is the running time of the best algorithm for finding the closest pair among N "random" sequences of length $O(\log N)$. A trivial bound of $f = 2$ leads to a simple algorithm with a running time of $\tilde{O}(n^{2/3})$. The asymptotic running time can be further improved by plugging in recent more efficient algorithms for the closest pair problem.

Our results also yield a sub-linear time algorithm for approximate pattern matching algorithm for a random signal (text), even for the case when the error between the signal and the code (pattern) is asymptotically as large as the code size. This is the first sublinear time algorithm for such error rates.

1 Introduction

The *shift finding* problem is defined as follows. We are given a binary *code vector* $\mathbf{c}$, and a *signal vector* $\mathbf{x}$ obtained by rotating $\mathbf{c}$ by a shift τ and adding noise. The goal is to estimate τ by finding the shift that minimizes the distance between the signal and the shifted code. The code is assumed to be uncorrelated with any shift of itself, and hence minimizing the distance yield a good estimate of τ. The problem can be naturally extended to higher-dimensions, where the inputs, $\mathbf{c}$ and $\mathbf{x}$, are higher-dimensional matrices.

The importance of shift finding stems from two reasons: 1) it is a basic problem at the heart of several practical applications, including GPS synchronization [HAKI12, Kap96] and motion estimation [ITU], and 2) it has strong connections to a large body of work on string/pattern matching algorithms, and hence advances on this problem can shed new lights on those topics.

To see the practical use of shift finding, consider how a GPS receiver locks on the satellite signal [Kap96]. Each GPS satellite is assigned a CDMA code, which can be modeled as a random vector, $\mathbf{c}$, of length n, with each c_i chosen independently and uniformly at random from $\{-1, 1\}$. The satellite transmits its code repeatedly. To lock on the GPS signal, a receiver has to align the corresponding CDMA code, $\mathbf{c}$, with the received signal, $\mathbf{x}$. This allows the GPS receiver to estimate the delay in receiving the satellite code, which enables the receiver to locate itself with respect to the satellite. Since the GPS code is long and the receiver has to synchronize with the various satellites visible in its part of the globe, the shift finding process ends up being time consuming and a major contributor to GPS locking delay [HAKI12, Kap96]. Reducing the runtime complexity of shift finding can lead to faster GPS receivers, which also translates to reducing power consumption in these devices (see [HAKI12] and the references therein).

GPS is just one example of a class of applications that employ shift finding to align a code with an encoded signal in order to measure delay and/or motion. Other applications include motion estimation and compensation in video [ITU], packet synchronization in ultra wideband wireless transceivers [CLW+09], and the estimation of relative travel times of sound waves used for animal tracking or event localization [Spi00]. All these applications can benefit from faster algorithms for shift finding.

The best known algorithm for the general version of shift finding takes $O(n \log n)$-time, and works by convolving $\mathbf{c}$ and $\mathbf{x}$ using the Fast Fourier Transform (FFT). A recent paper [HAKI12] has proposed a linear-time algorithm for the specific case where both the code vector $\mathbf{c}$ and the noise are random. It is also known that one needs a lower bound of $\Omega(n^{1/2})$ queries to $\mathbf{c}$ and $\mathbf{x}$ in order to estimate the shift with constant probability [BEK+03, AN10]. This lower bound holds even for the case where the input is random. Perhaps

surprisingly, no sub-linear time algorithm is known for shift finding.

The lack of a sub-linear algorithm is particularly interesting since the problem is strongly related to the well-studied approximate string matching problem (also known as approximate pattern matching) [FP74, Nav01, ALP04, LV89]. In the string matching problem, we are given a string $\mathbf{c}$ of length m and a text vector $\mathbf{x}$ of length n, the goal is to find a shift t that minimizes the distance between $\mathbf{c}[0 \dots m-1]$ and $\mathbf{x}[t \dots t+m-1]$. Although there is a rich body of algorithms for approximate string matching, those algorithms are either based on the FFT (and hence run in at least $\Omega(n \log n)$ time), or improve over the FFT only when k, the number of mismatched coordinates between the string and the text, is small (i.e., $k \ll m$) [ALP04]. Investigating the case of potentially many mismatched coordinates (i.e., $k = o(m)$), as defined in the shift finding problem, can lead to new advances in this well-studied domain.

Our Results: In this paper, we consider the shift finding problem in a setting where $\mathbf{c}$ is random and $\mathbf{x}$ is equal to a shifted version of the code corrupted by noise. Our basic noise model assumes that $\mathbf{x} = \mathbf{c}^{(\tau)} + g$, where where $\mathbf{c}^{(\tau)}$ refers to the code shifted by τ, and the entries g_i are i.i.d. random variables taken from the normal distribution with 0 mean and variance σ. We also consider the Boolean error model where $\mathbf{x}$ is obtained by flipping each entry in $\mathbf{c}^{(\tau)}$ with probability $\eta < 1/2$.

Our first result is an algorithm that, for any constant σ, runs in time $O((n \log n)^{2/3})$. To the best of our knowledge, this is the first sub-linear-time algorithm for this problem. The algorithm is simple and easy to implement. This simplicity makes it a good candidate for adoption in practical applications. Perhaps surprisingly, the algorithm is purely combinatorial and does not use any algebraic techniques such as the FFT (even though it is inspired by the recent developments in sparse FFT algorithms [HIKP12b, HIKP12a]).

Our second result improves the algorithm further by utilizing a recent result of [Val12] on finding a correlated pair of vectors. Suppose that we are given N random d-dimensional vectors $\mathbf{x}^1 \dots \mathbf{x}^N$, where: each $\mathbf{x}^i$ is chosen uniformly at random from $\{0,1\}^d$, and the vectors are independent except for an unknown pair $\mathbf{x}^i, \mathbf{x}^j$, such that $\mathbf{x}^i$ is obtained from $\mathbf{x}^j$ by flipping each entry with probability $\eta < 1/2$. The algorithm of [Val12] identifies such a pair in $O(N^f)$ time for $f = 3/4 \cdot \omega$ for any constant $\eta < 1/2$, where ω is the square matrix multiplication runtime exponent. Since $\omega \leq 2.372$ [Wil12], we have $f \leq 1.779$. Using this algorithm as a subroutine, we give a shift finding algorithm with the running time of $O(n^{1-1/(1+f)}) \log^{O(1)} n = O(n^{0.641})$. The reduction is optimal in the following sense: if we were given an algorithm for finding a correlated pair of vectors with exponent $f = 1+\epsilon$, then the shift-finding algorithm would have the running time of $n^{(1+O(\epsilon))/2}$, which matches the lower bound on shift finding up to the ϵ term in the exponent.

Our third result generalizes both algorithms to incorporate the case where the code is of size m and the signal is of size n, for $m \ll n$. Hence, we obtain the first sub-linear approximate string matching algorithms that do not require the number of mismatched coordinates to be small (i.e., $k = o(m)$ in our algorithms).

Our Techniques: Building on [HAKI12], our algorithms use the technique of "folding". Specifically, the code $\mathbf{c}$ is partitioned into p blocks of size n/p that are added to each other; the same transformation is applied to $\mathbf{x}$. Folding has the property that if the signal $\mathbf{x}$ (approximately) matches the shifted code $\mathbf{c}^{(\tau)}$, then the folded signal matches the folded code shifted by $t' = \tau \mod (n/p)$. The shift t' can be then found in $O(n/p \log(n/p))$ time, which is sub-linear in n for p large enough.

However, folding itself takes time linear in n. To overcome this difficulty, we show that the optimal shift can be estimated using only a sub-linear number of samples from the folded sequences. This in turn requires accessing only a sub-linear number of samples from $\mathbf{c}$ and $\mathbf{x}$.

We present two algorithms for finding t' for the folded sequences. The simple variant of the algorithm enumerates all possible shifts in the folded sequence, and thus has a runtime roughly linear in n/p. For each shift, it computes the correlation between $O(\log n)$ pairs of samples from the folded code and the folded signal. This requires proving a sharp concentration bound for the estimated correlation value.

The more complex variant of the algorithm uses correlated-pair-finding procedure to perform this task in time sub-linear in n/p. Our reduction to the correlated-pair-finding problem is "black-box". However, at present, the only algorithm that achieves a sub-quadratic time for any constant $\eta < 1/2$ is the one due to [Val12], so we utilize that algorithm as the subroutine. To this end, however, we need to convert the i.i.d. samples of the folded sequences (which are real-valued) into a sequence of i.i.d. ± 1 values, which are needed as the input to the pair-finding algorithm. We show that simply taking the signs of the folded samples suffices for this purpose.

Note that it is plausible that we could use real-valued samples as the input to the pair-finding algo-

rithm, or alternatively use the ± 1 samples in our first algorithm. However (i) using ± 1 samples enables us to provide a black-box reduction to the pair-finding algorithm and (ii) using real-valued samples leads to an algorithm that is simpler and likely more practical.

2 Related Work

String/Pattern Matching: There is a rich body of work on approximate pattern matching (see [Nav01, ALP04] for an overview). Starting from the seminal work of [FP74], many of the approximate pattern matching algorithms use FFT, and hence have running times of at least $\Omega(n \log n)$. Faster algorithms exist if the number of mismatches k is bounded. In particular, the algorithm of [ALP04] (building on [LV89]) has the runtime of $O(n\sqrt{k \log k})$. Still, those algorithms improve over the FFT only for small values of k.

Little is known about pattern matching in sub-linear time. To the best of our knowledge, all such algorithms were developed for random inputs. In particular, the work in [KMP77] shows that the well-known Boyer-Moore algorithm for the exact pattern matching [BM77] has the expected running time of $O(n/m \cdot \log n)$, which is sub-linear in n for large m. This result has been generalized to the k-mismatch case [CM94], yielding an algorithm with a running time of $O(n/m \cdot k \log n)$.

Sketching: Recently, a number of papers have studied the *sketching complexity* of the shift-finding problem. The goal is to design a function F with output length of at most b bits, such that for any $\mathbf{c}$ and $\mathbf{x}$, one can estimate the distance between $\mathbf{c}^{(\tau)}$ and $\mathbf{x}$ under the optimal shift τ, given only $F(\mathbf{c})$ and $F(\mathbf{x})$, with probability of at least 2/3. The minimal value of b satisfying these conditions is referred to as the *sketching complexity* of the problem. It is known that for *average* inputs one can obtain sketch functions that use only $O(\log n)$ bits [KR06]. The worst-case sketching complexity of this problem is wide open ([IMNO11], Problem 13). The best bound achieves sketching complexity of $O(\sqrt{n \log n})$ for an approximation factor arbitrarily close to 1, and has large time complexity (at least $n \log n$) [CM11]. One can achieve lower sketching complexity, but the approximation factor becomes larger, of the order of $O(\log^2 n)$ [AIK08].

There is a connection between sketching and sub-linear time algorithms for the shift finding problem. Our algorithm implicitly uses a form of sketching, by summing all folded entries in $\mathbf{x}$ and $\mathbf{c}$. It is plausible that sub-linear bounds for the worst case sketching complexity would yield sub-linear-time algorithms for shift finding in the worst case.

3 Notation

We denote by $\mathbf{c} = c_0, \ldots, c_{n-1}$ a binary code where the coordinates $c_0, \ldots, c_{n-1}$ are independent and identically distributed random variables with values in $\{-1, 1\}$, such that $\Pr[c_i = 1] = \Pr[c_i = -1] = 1/2$. We use $\mathbf{c}^{(t)}$ to refer to the code $\mathbf{c}$ shifted by $t = 0, \ldots, n-1$, i.e., $c_i^{(t)} = c_{t+i \mod n}$.

Noise Model: We assume that the input $\mathbf{x}$ is obtained from $\mathbf{c}$ by adding some noise and a cyclic shift τ. We consider two models of noise: Gaussian noise, and Bernoulli noise. In the Gaussian noise model, the signal is $\mathbf{x} = \mathbf{c}^{(\tau)} + \mathbf{g}$, where $\mathbf{g} = (g_0, g_1, \ldots g_{n-1})$ denotes the noise vector. The coordinates of the noise vector $\mathbf{g}$ are independent and identically distributed random variables that follow a normal distribution with zero mean and standard deviation σ.

In the Bernoulli noise model, the signal is $\mathbf{x} = \mathbf{c}^{(\tau)} \odot \mathbf{b}$, where $\mathbf{b} \in \{-1, 1\}^n$ denotes the noise vector and $\odot$ denotes the pairwise product. The coordinates $b_0, \ldots, b_{n-1}$ of the noise vector $\mathbf{b}$ are independent and identically distributed random variables with bernoulli distribution from $\{-1, 1\}$, such that $\Pr[b_i = -1] = \eta$, i.e., coordinate $c_i^{(\tau)}$ is flipped with probability η.

Note that the problem in the Gaussian noise model can be reduced to the Bernoulli noise model:

FACT 3.1. *Consider vectors $\mathbf{c}$ and $\mathbf{x}$ such that $\mathbf{x} = \mathbf{c}^{(\tau)} + \mathbf{g}$ for some $0 \leq \tau < n$, where $\mathbf{g}$ is a vector of iid normal random variables with standard deviation $\sigma > 0$. Let $\mathbf{x}' = (x'_0, \ldots x'_{n-1})$ be the vector defined as $x'_i = \text{sign}(x_i)$. Then $\mathbf{x}'$ has distribution $\mathbf{x}' = \mathbf{c}^{(\tau)} \odot \mathbf{b}$, where $\mathbf{b}$ is a vector of iid Bernoulli random variables with bias $\Pr[b_i = -1] = \eta$ for $\eta = \frac{1}{2}\text{erfc}(1/\sigma)$ where* erfc(.) *is the complimentary error function. Note that $\eta = 1/2 - \Theta(1/\sigma)$ whenever $\sigma = \Omega(1)$.*

Folding: For a vector $\mathbf{c}$ and integer p, we let $\mathbf{c}(p)$ be the vector $\mathbf{c}$ folded p times into blocks of length n/p (assuming p divides n). Specifically, for $i = 0 \ldots n/p - 1$, we define $c(p)_i = \sum_{j=0}^{p-1} c_{i+jn/p}$. We define $\mathbf{x}(p)$ similarly. For example, note that $\mathbf{c}(1) = \mathbf{c}$.

Projection on a Set: Finally, for vector $\mathbf{c} \in \{-1, 1\}^n$ and set $X \subset \{0, 1, \ldots n-1\}$, we let $\mathbf{c}|_X$ be the vector $\mathbf{c}$ projected onto the coordinates of the set X.

4 Basic Algorithm

We first present the basic algorithm, which achieves a time complexity of $O((n \log n)^{2/3})$. In §6, we show how to combine this algorithm with an algorithm of Valiant to obtain a faster runtime.

Let p denote the number of folds, and $l = n/p$ the length of the folded vectors. (We later set $p = (n \log n)^{1/3}$.) To find the shift τ such that the input

is $\mathbf{x} = \mathbf{c}^{(\tau)}$, our algorithm takes two steps, as follows:

(1) Estimate the shift $\tau \mod l$:
The goal of this step is to find the shift $t' = \tau \mod l$, via the following (ideal) computation:

$$t' = \arg\max_{0 \le t < n/p} \mathbf{c}(p)^{(t)} \cdot \mathbf{x}(p), \tag{4.1}$$

where $\mathbf{c}(p)$ and $\mathbf{x}(p)$ are the code and signal folded into blocks of size n/p as defined in §3, and $\cdot$ is the scalar product.

However, we do not estimate t' directly using Eq. 4.1 because computing $\mathbf{x}(p)$ and $\mathbf{c}(p)$ takes a linear time in n. Instead, we estimate it using the following algorithm:

- Compute $\mathbf{x}(p)|_{\mathbf{X}}$ directly, where the set $\mathbf{X} = \{0, \ldots, \sqrt{l \log n} - 1\}$.
- Compute $\mathbf{c}(p)|_{\mathbf{Y}}$ directly, where the set $\mathbf{Y} = Y_0 \cup \ldots \cup Y_{\sqrt{l}-1}$ and the subsets Y_i are segments of length $s = O(\sqrt{\log n})$ equally spaced over $[0 \ldots l-1]$, i.e., $Y_i = \{i \times \sqrt{l}, \ldots, i \times \sqrt{l} + s - 1\}$.
- For each candidate shift $t \in [0 \ldots l-1]$, we compute the correlation on the intersecting coordinates i using the quantities computed above:

$$\begin{aligned} d_t &= \sum_{i \in \mathbf{X} \cap \{\mathbf{Y} - t\}} c(p)_i^{(t)} \cdot x(p)_i \\ &= \sum_{i \in \mathbf{X} \cap \{\mathbf{Y} - t\}} c(p)_{i+t} \cdot x(p)_i \end{aligned}$$

- Estimate $t' = \arg\max_t d_t$.

(2) Enumerate p possible values of τ :
To find the best match among the p shifts, t, that satify $t = t' \mod n/p$, we calculate the cross correlation between $\mathbf{x}|_S$ and $\mathbf{c}^{(t)}|_S$ where the set $S = \{0, \ldots, O(\log n)\}$. For $t = t' + j \cdot n/p$ and $j \in \{0, \ldots, p-1\}$, we estimate τ by computing

$$\tau = \arg\max_{t = t' + j \cdot n/p} \sum_{i \in S} c_i^{(t)} \cdot x_i.$$

5 Analysis of the Basic Algorithm

5.1 Correctness

We prove the following theorem.

THEOREM 5.1. (CORRECTNESS) *Fix the length $n \ge 1$, the error rate $0 < \eta < 1/2$, and the folding parameter $1 \le p \le n$. Given vectors $\mathbf{c}$ and $\mathbf{x} = \mathbf{c}^{(\tau)} \odot \mathbf{b}$ in the Bernoulli noise model with bias η, the algorithm from §4 recovers the shift τ with high probability.*

To prove this theorem, we need to introduce the following concentration bound, inspired by a dimensionality reduction result of [Ach03].

LEMMA 5.2. *Fix $n > 1$, $\epsilon, \delta > 0$, and sparsity $0 < \gamma \le 1$. We set C to be a sufficiently large constant. Let A be a matrix of size $r = C/\gamma \log 1/\delta$ by n, where each entry A_{ij} is independent identically distributed as: A_{ij} is 0 with probability $1-\gamma$, and otherwise A_{ij} is random from $\{-1, +1\}$.*

Then, for any vector $\mathbf{x} \in \mathbb{R}^n$, we have that $\|A\mathbf{x}\|_2^2$ is a $1 \pm \epsilon$ approximation to $\gamma r \|\mathbf{x}\|_2$ with probability at least $1 - \delta$. In particular, $\sum_{i=1}^{r} (\sum_{j=1}^{n} A_{i,j})^2 \in [(1-\epsilon)\gamma n r, (1+\epsilon)\gamma n r]$ with probability at least $1 - \delta$.

To prove lemma 5.2, we need a concentration bound for random projections using matrices with subgaussian entries. Specifically, we use the following lemma, adapted from [IN07]:

LEMMA 5.3. *Suppose that a real random variable Y is symmetric and has unit variance. Moreover, assume that Y is sub-gaussian, i.e. $E[e^{tY}] \le e^{\nu t^2}$ for all $t < 1/(200\nu^2)$, for some $\nu \ge 1$. Let $Y_1 \ldots Y_n$ be i.i.d. copies of Y, let $\mathbf{x}$ be an n-dimensional unit vector, and denote $U = \sum_{j=1}^{n} x_j Y_j$. If $U_1 \ldots U_r$ are i.i.d. copies of U then for all $0 < \epsilon < 25\nu$ we have*

$$\Pr[|1/r \sum_{j=1}^{r} U_i^2 - 1| \ge \epsilon] \le 2e^{-r\epsilon^2/(400\nu^2)}$$

Lemma 5.2 follows by applying Lemma 5.3 to the random variable $Y = A_{ij}/\sqrt{\gamma}$ using $\nu = 1/\sqrt{\gamma}$. To this end we need:

CLAIM 5.4. *For $\nu = 1/\sqrt{\gamma}$ and $t < 1/(200\nu^2) = \gamma/200$, we have*

$$E[e^{tY}] \le e^{t^2} \le e^{\nu t^2}$$

Proof. First note that $t/\sqrt{\gamma} \le (\gamma/200)/\gamma \le 1/2$. Observe that

$$\begin{aligned} E[e^{tY}] &= 1 - \gamma + \gamma/2[e^{-t/\gamma} + e^{t/\gamma}] \\ &\le 1 - \gamma + \gamma/2(1 - t/\sqrt{\gamma} + (t/\sqrt{\gamma})^2/2) \\ &\qquad + \gamma/2(1 + t/\sqrt{\gamma} + 2(t/\sqrt{\gamma})^2/2) \\ &\le 1 - \gamma + \gamma/2[2 + 2(t/\sqrt{\gamma})^2] \\ &= 1 - \gamma + \gamma + \gamma t^2/\gamma \\ &= 1 + t^2 \\ &\le e^{t^2} \end{aligned}$$

□

Now we can proceed with the proof of Theorem 5.1.

Proof. The main idea is to show that $t' = \tau \mod n/p$ can be computed using the formula:

$$t' = \arg\max_t \mathbf{c}(p)^{(t)} \cdot \mathbf{x}(p)$$

Instead of proving this statement directly, we show that if we sample coordinates from $\mathbf{c}(p)$ and $\mathbf{x}(p)$ and estimate the above sum from the colliding coordinates only (as we actually do in the algorithm), then the argmax still holds.

We first develop some notation for analyzing the above quantity. For $0 \le i < n/p$ and a potential shift $0 \le t < p$, define $a_{i,t} = (\sum_{j=0}^{p-1} c_{t+i+jn/p})(\sum_{j=0}^{p-1} x_{i+jn/p})$. Note that the quantity to estimate is equal to:

$$\begin{aligned}\mathbf{c}(p)^{(t)} \cdot \mathbf{x}(p) &= \sum_{i=0}^{n/p-1} \left(\sum_{j=0}^{p-1} c_{t+i+jn/p}\right) \cdot \left(\sum_{j=0}^{p-1} x_{i+jn/p}\right) \\ &= \sum_{i=0}^{n/p} a_{i,t}.\end{aligned}$$

Furthermore, we note that

$$\begin{aligned}a_{i,t} &= \left(\sum_{j=0}^{p-1} c_{t+i+jn/p}\right)\left(\sum_{j=0}^{p-1} c_{\tau+i+jn/p} b_{\tau+i+jn/p}\right) \\ &= \frac{1}{2}\left(\sum_j c_{t+i+jn/p}\right)^2 + \frac{1}{2}\left(\sum_j c_{\tau+i+jn/p} b_{\tau+i+jn/p}\right)^2 \\ &\quad - \frac{1}{2}\left(\sum_j c_{t+i+jn/p} - c_{\tau+i+jn/p} b_{\tau+i+jn/p}\right)^2\end{aligned}$$

Thus $a_{i,t}$ is a sum of three terms, where each term is a square of a sum of i.i.d. random variables.

The algorithm estimates t' by estimating the scalar product $\mathbf{c}(p)^{(t)} \cdot \mathbf{x}(p) = \sum a_{i,t}$ from $O(\log n)$ terms $a_{i,t}$. First we show that for $t' = \tau \mod n/p$, the value of the estimator of the scalar product is large. Later we will show that for $t \neq \tau \mod n/p$, the value of the estimator of the sum is small.

Lemma 5.5. (Correct shift) *Let $t' = \tau \mod n/p$. Let $S \subset \{0, 1, \ldots n/p-1\}$ be any set of size $s = O(\log n)$. Then, with high probability, we have that*

$$d_{t'} = \sum_{i \in S} c(p)_i^{(t')} \cdot x(p)_i \ge \tfrac{1}{2}(1-2\eta)p|S|.$$

Proof. We have that

$$\begin{aligned}d_{t'} &= \frac{1}{2}\sum_i \left(\sum_j c_{\tau+i+jn/p}\right)^2 \\ &\quad + \frac{1}{2}\sum_i \left(\sum_j c_{\tau+i+jn/p} b_{\tau+i+jn/p}\right)^2 \\ &\quad - 2\sum_i \left(\sum_j c_{\tau+i+jn/p}\left(\tfrac{1-b_{\tau+i+jn/p}}{2}\right)\right)^2.\end{aligned}$$

We apply Lemma 5.2 to each of the three terms separately. The first two terms have the exact same probability distribution, and we estimate them using the lemma with sparsity $\gamma = 1$. For failure probability $\delta = 1/n^2$, we obtain that

$$\begin{aligned}&\sum_i \left(\sum_j c_{\tau+i+jn/p}\right)^2, \sum_i \left(\sum_j c_{\tau+i+jn/p} b_{\tau+i+jn/p}\right)^2 \\ &\quad \in [(1-\epsilon)p|S|, (1+\epsilon)p|S|].\end{aligned}$$

For the last term, we note that we can apply Lemma 5.2 with sparsity $\gamma = \eta$ since $\Pr[b_i = -1] = \eta$. We obtain that the third term satisfies the following with high probability:

$$\begin{aligned}&\sum_i \left(\sum_j c_{\tau+i+jn/p}\left(\tfrac{1-b_{\tau+i+jn/p}}{2}\right)\right)^2 \\ &\quad \in [(1-\epsilon)\eta p|S|, (1+\epsilon)\eta p|S|].\end{aligned}$$

Putting all the bounds together, we have that

$$d_{t'} \ge \tfrac{p|S|}{2}(2(1-\epsilon) - 4(1+\epsilon)\eta) \ge p|S|(1-2\eta-\epsilon-2\epsilon\eta),$$

with high probability. Setting $\epsilon = \frac{1-2\eta}{2(1+2\eta)} \ge (1-2\eta)/4$, we obtain the desired result. □

Lemma 5.6. (Wrong shift) *Let $t \neq \tau \mod n/p$. Let $S \subset \{0, 1, \ldots n/p-1\}$ be any set of size $s = O(\log n)$. Then, with high probability, we have that*

$$d_t = \sum_{i \in S} c(p)_i^{(t)} \cdot x(p)_i < \tfrac{1}{2}(1-2\eta)p|S|.$$

Proof. As in the previous lemma, we apply Lemma 5.2

to the three terms of d_t. Now d_t is as follows:

$$d_t = \frac{1}{2}\sum_i \left(\sum_j c_{t+i+jn/p}\right)^2 + \frac{1}{2}\sum_i \left(\sum_j c_{\tau+i+jn/p} b_{\tau+i+jn/p}\right)^2 - 2\sum_i \left(\sum_j \frac{c_{t+i+jn/p} - c_{\tau+i+jn/p} b_{\tau+i+jn/p}}{2}\right)^2.$$

Again, the first two terms can be estimated using Lemma 5.2 with sparsity $\gamma = 1$. For failure probability $\delta = 1/n^2$, we obtain

$$\sum_i \left(\sum_j c_{t+i+jn/p}\right)^2, \sum_i \left(\sum_j c_{\tau+i+jn/p} b_{\tau+i+jn/p}\right)^2 \in [(1-\epsilon)p|S|, (1+\epsilon)p|S|].$$

To estimate the last term, we use the lemma with sparsity $\gamma = 1/2$. There is one complication that some random variables may appear twice in this term — in particular, when there exist some $i, i' \in S$ such that $t + i = \tau + i'$ — which breaks the assumption of independence. In this case, it is not hard to see that one can split the set S into two equal-size sets S_1, S_2, such that $\sum_{i\in S_k}\left(\sum_j \frac{c_{t+i+jn/p} - c_{\tau+i+jn/p} b_{\tau+i+jn/p}}{2}\right)^2$, for $k \in \{1, 2\}$, is each composed entirely of independent variables. We apply Lemma §5.2 to each of the two sums separately. We conclude that

$$2\sum_i \left(\sum_j \frac{c_{t+i+jn/p} - c_{\tau+i+jn/p} b_{\tau+i+jn/p}}{2}\right)^2 \in [2(1-\epsilon)\tfrac{1}{2}p|S|, 2(1+\epsilon)\tfrac{1}{2}p|S|].$$

Overall, we conclude that

$$d_t \le \tfrac{p|S|}{2}(2(1+\epsilon) - 2(1-\epsilon)) \le 2\epsilon p|S|,$$

with high probability. Setting $\epsilon = (1-2\eta)/5$, we arrive at the desired conclusion. □

Lemmas 5.5 and 5.6 thus show that the algorithm computes $t' = \arg\max_t d_t$ such that $t' = \tau \mod n/p$. Indeed, for each candidate $t \in \{0, \dots n/p - 1\}$, it computes d_t as a sum of $a_{i,t}$ for $i \in S = \mathbf{X} \cap \{\mathbf{Y} - t\}$. By construction of $\mathbf{X}$ and $\mathbf{Y}$, we have that $|S| = \Omega(\log n)$ to satisfy the conditions of the two lemmas.

It remains to only prove that, once we have $t' = \tau \mod n/p$, the second step of the algorithm recovers the entire shift τ. Indeed, the second step tests all possible $t \in \{t', t' + n/p, t' + 2n/p \dots\}$, including $t = \tau$. We need to show that, with high probability,

$$\tau = \arg\max_{t=t'+j\cdot n/p} \sum_{i\in S} c_i^{(t)} \cdot x_i.$$

Indeed, first consider the case that $t = \tau$. Then, we claim that $\sum_{i\in S} c_i^{(t)} \cdot x_i$ is at least $\frac{1}{2}(1-2\eta)|S|$. We use the following standard concentration bound.

LEMMA 5.7. (CHERNOFF) *Let* $\mathbf{b} = (b_0, \dots b_{n-1}) \in \{-1, 1\}^n$ *be a vector of iid Bernoulli random variable with bias* $\Pr[b_i = -1] = \eta$ *where* $0 < \eta < 1/2$. *Then, for a fixed* $\epsilon > 0$, *the sum* $\sum_i b_i$ *is inside the interval* $[(1 - 2\eta - \epsilon)n, (1 - 2\eta + \epsilon)n]$ *with probability at least* $1 - e^{-\Omega(\epsilon^2 n)}$.

Applying this lemma with $\epsilon = (1-2\eta)/2$, we have that $\sum_{i\in S} c_i^{(t)} \cdot x_i = \sum_{i\in S} b_{t+i} \ge \frac{1}{2}(1-2\eta)|S|$.

We now consider the case that $t \neq \tau \mod n$. Then $\sum_{i\in S} c_i^{(t)} \cdot x_i = \sum_{i\in S} c_{i+t} c_{i+\tau} b_{i+\tau}$. All the involved random variables are independent since $|t - \tau| \ge n/p \gg |S|$. Applying Lemma 5.7, we have that $\sum_{i\in S} c_i^{(t)} \cdot x_i \le \epsilon|S|$ with high probability. For $\epsilon = \frac{1}{3}(1-2\eta)$, we have that $\sum_{i\in S} c_i^{(t)} \cdot x_i \le \frac{1}{3}(1-2\eta)|S|$.

We conclude that $\tau = \arg\max_{t=t'+j\cdot n/p} \sum_{i\in S} c_i^{(t)} \cdot x_i$. This completes the proof of correctness. □

5.2 Runtime We now analyze the runtime of the algorithm.

THEOREM 5.8. (RUNTIME) *For any* $p < \sqrt{n}$ *and fixed* $0 < \eta < 1/2$, *the algorithm from §4 runs in time* $O(\sqrt{n/p \cdot \log n} \cdot p + n/p \cdot \log n)$. *In particular, for* $p = (n \log n)^{1/3}$, *the runtime becomes* $O((n \log n)^{2/3})$.

Proof. The first step of the algorithm estimates $c(p)_i$ and $x(p)_i$ for $O((n/p \log n)^{1/2})$ positions i. Each $c(p)_i$ is a sum of p values. Hence, this process takes $O((n/p \log n)^{1/2} \cdot p)$ time. Then the algorithm computes t' by trying n/p shifts t, each requiring $O(\log n)$ time. This adds another $O(n/p \cdot \log n)$ time. Finally, in the second step, the algorithm performs $O(p \log n)$ futher computation, which is a low order term. □

6 Faster Algorithm

In this section, we show how to improve our basic algorithm, using the following closest pair algorithm, which follows from an algorithm of Valiant [Val12].

THEOREM 6.1. ([VAL12]) *Consider* n *random vectors in* $\{\pm 1\}^d$, *which are fully independent, except for at most one pair of vectors. For any constant* c, *if this pair is* ρ*-correlated coordinate-wise and* $d \ge c\frac{\log n}{\rho^2}$, *then one can find this pair in time* $dn^{1.779}\rho^{-O(1)}$.

6.1 Algorithm Description Now we describe the new algorithm. As before, p is the number of folds, $l = n/p$, and $s = O(\log n)$.

(1) Estimate the shift $\tau \mod l$:

- Compute $\mathbf{x}(p)|_{\mathbf{X}}$ directly, where the set $\mathbf{X} = \{0, \ldots, \sqrt{l} + s - 1\}$.
- Compute $\mathbf{c}(p)|_{\mathbf{Y}}$ directly, where the set $\mathbf{Y} = Y_0 \cup \ldots \cup Y_{\sqrt{l}-1}$ and the subsets Y_i are segments of length s equally spaced over $[0 \ldots l-1]$, i.e., $Y_i = \{i \times \sqrt{l}, \ldots, i \times \sqrt{l} + s - 1\}$.
- For each $i \in \mathbf{X}$, let $x'(p)_i = \text{sign}(x(p)_i)$, and for each $i \in \mathbf{Y}$, let $c'(p)_i = \text{sign}(c(p)_i)$.
- Consider $2\sqrt{l}$ points in an s-dimensional space: $\mathbf{x}'(p) \mid_{X_i}$ where $X_i = \{i, \ldots i + s - 1\}$, for all $0 \le i < \sqrt{l}$, as well as $\mathbf{c}'(p) \mid_{Y_j}$ for $0 \le j < \sqrt{l}$. Call them $\{v_i\}_i \subset \{\pm 1\}^s$ and $\{u_j\}_j \subset \{\pm 1\}^s$.
- For $0 \le k < 2s$, let $I_k = \{k, k + 2s, k + 4s, k + 6s, \ldots\} \cap \mathbf{X}$. Use the closest pair algorithm from Theorem 6.1 on the vectors $\{v_i\}_{I_k}$ and u_j, for each $0 \le k < 2s$ separately. Out of all (at most) $2s$ outputs, choose the pair (v_i, u_j) that maximizes the dot product $v_i \cdot u_j \ge \Omega(s)$.
- Set $t' = j \times \sqrt{l} - i \mod n/p$.

(2) Enumerate p possible values of τ :
To find the best match among the p shifts t that satify $t = t' \mod n/p$, we calculate the cross correlation between $\mathbf{x}|_S$ and $\mathbf{c}^{(t)}|_S$ where the set $S = \{0, \ldots, O(\log n)\}$. For $t = t' + j \cdot n/p$ and $j \in \{0 \ldots p-1\}$, we estimate τ by computing

$$\tau = \arg\max_{t = t' + j \cdot n/p} \sum_{i \in S} c_i^{(t)} \cdot x_i.$$

6.2 Analysis of the Faster Algorithm We start from the following lemma.

LEMMA 6.2. *Consider two sums $S = \sum_{i=1}^{N} r_i$ and $S' = \sum_{i=1}^{N} r_i \cdot b_i$, where r_i's are i.i.d. random variables chosen uniformly at random from $\{-1, 1\}$, and $b_i \in \{\pm 1\}$ are any binary variables such that $\sum_{i=1}^{N} b_i \ge \alpha N$ for some $\alpha > 0$. Consider the probability*

$$P(\alpha, N) = \Pr[\text{sign}(S) \ne \text{sign}(S')]$$

For any $\alpha > 0$ there exists $\beta = \Theta(\alpha)$ such that for all $N \ge 1$ we have $P(\alpha, N) \le 1/2 - \beta$.

Proof. Let F be the set of indices i such that $b_i = -1$, and let $U = \{1 \ldots N\} - F$. Note that $|F| \le (1/2 - \alpha/2)N$. For any set $A \subset \{1 \ldots N\}$ let $S_A = \sum_{i \in A} r_i$. Clearly $S = S_U + S_F$ and $S' = S_U - S_F$. Thus, the event $\text{sign}(S) \ne \text{sign}(S')$ holds if and only if $|S_U| \le |S_F|$.

It suffices to estimate the probability that $|S_U| \le |S_F|$, where $|U| = N/2 + \gamma N/2$ and $|F| = N/2 - \gamma N/2$ with $\gamma \ge \alpha$.

Let $V \subset U$ be a set of $N/2 - \gamma N/2$ random variables. Note that the distribution of S_V is the same as S_F. Also, let $z = S_U - S_V$. Without loss of generality, we can assume $z \ge 0$.

We view the process of generating S_V and S_F as follows. First, we generate two random variables P_1 and P_2 that are each a sum of $|V| = N/2 - \gamma N/2$ iid random variables, conditioned on $P_1 \ge P_2 \ge 0$. Then we assign $\{P_1, P_2\}$ to $\{S_V, S_F\}$ randomly, with random ± 1 signs. It is easy to check that if $P_1 > P_2 + z$, then $\Pr[|z + S_V| > |S_F| \mid z, P_1, P_2] = 1/2$. On the other hand, if $P_1 \le P_2 + z$, then $\Pr[|z + S_V| > |S_F| \mid z, P_1, P_2] \ge 1/2$, since the event holds if $S_V > 0$, which happens with probability at least $1/2$. However, the probability could be higher than $1/2$ for specific values of z, P_1, P_2.

One case where such a bias occurs is when $z - P_1 > z/2$ and $P_2 < z/2$. Denote this event by E. In this case, $\Pr[|z + S_V| \ge |S_F| \mid z, P_1, P_2] \ge 3/4$. We estimate the probability of E using the following claim.

CLAIM 6.3. *Let $\sum_i x_i$ be a sum of k i.i.d. random variables distributed uniformly over $\{-1, 1\}$. Then, for each $s \in [-\sqrt{k}, \sqrt{k}]$, $\Pr[\sum x_i = s] \in [C_1/\sqrt{k}, C_2/\sqrt{k}]$ for some constants $C_2 > C_1 > 0$.*

Since z is a sum of γN random variable, using the claim, we have $z \ge \frac{1}{2}\sqrt{\gamma N/2}$ with probability at least $C_1/2$. Further, we have $P_1, P_2 \in [0, \frac{1}{5}\sqrt{\gamma N/2}]$ with probability at least $C_1\sqrt{\gamma}/5$.

Thus, we have that with probability at least $C_1/2 \cdot (C_1\sqrt{\gamma}/5)^2 = C_1^3\gamma/50$ over the choice of z and P_1, P_2, we have that $\Pr[|z + S_v| > |S_F| \mid z, P_1, P_2] \ge 3/4$.

All in all, we conclude that $\Pr[|S_U| > |S_F|] \ge \frac{1}{2}(1 - C_1^3\gamma/50) + \frac{3}{4} \cdot C_1^3\gamma/50 = \frac{1}{2} + C_1^3\gamma/200$. Hence, $P(\alpha, N) \le 1/2 - C_1^3\alpha/200$. □

We now proceed to show the correctness of the algorithm.

THEOREM 6.4. (CORRECTNESS) *Fix length $n \ge 1$, error rate $0 < \eta < 1/2$, and folding parameter $1 \le p \le n$. Given vectors $\mathbf{c}$ and $\mathbf{x} = \mathbf{c}^{(\tau)} \odot \mathbf{b}$ in the Bernoulli noise model with bias η, the algorithm from §6.1 recovers the shift τ with high probability.*

Proof. The proof is similar to that in §5. Let t' be such that $t' = \tau \mod n/p$. First, we have the following claim regarding the strings $\mathbf{x}'(p)$ and $\mathbf{c}'(p)$ obtained by taking signs of the folded strings.

CLAIM 6.5. *Fix any integers i, j. If $i + t' = j \mod n/p$, then $Pr[x'(p)_i = c'(p)_j] \ge 1/2 + \epsilon$, for some $\epsilon =$*

$\epsilon(1/2-\eta) > 0$. *On the other hand, if* $i + t' \neq j \mod n/p$, *then* $Pr[x'(p)_i = c'(p)_j] = 1/2$.

The claim follows from the Lemma 6.2 and that $\sum_{j=0}^{p-1} b_{i+jn/p} \geq \frac{1/2-\eta}{2} p$ with high probability.

Hence $\mathbf{x}'(p)$ and $\mathbf{c}'(p)$ are just a n/p-length instance of the original problem with new error $\eta' = 1/2 - \epsilon$. At this moment, the algorithm takes $s = O(\log n)$ length substrings of $\mathbf{x}'(p)$ and $\mathbf{c}'(p)$.

We can now apply the closest pair algorithm on the resulting vectors. The hope is that the algorithm finds the pair $v_i = \mathbf{x}'(p)|_{X_i}$ and $u_j = \mathbf{c}'(p)|_{Y_j}$ such that $i + t' = j\sqrt{n/p} \mod n/p$ for $0 \leq i, j < \sqrt{n/p}$ (note that such i, j exist by construction of sets $\mathbf{X}, \mathbf{Y}$). A technical detail is that we have to deal with some limited independence. Specifically, substrings of $\mathbf{x}'(p)$ may be overlapping, and substrings of $\mathbf{c}'(p)$ may be dependent on some substrings of $\mathbf{x}'(p)$ (for the wrong shift).

We prove the following claims. Remember that $v_i = \mathbf{x}'(p)|_{X_i}$ and $u_i = \mathbf{c}'(p)|_{Y_i}$.

Claim 6.6. *There exists some small constant* $C > 0$ *such that, for all* $0 \leq i, k < \sqrt{n/p}$ *and* $i < k$, $|v_i v_k| \leq Cs$, *with high probability.*

Proof. Fix some $i < k$. If $k \geq i + s$, then the statement follows immediately by Chernoff bound: $|v_i \cdot v_k| \leq O(\sqrt{s \log n}) \leq Cs$ with high probability.

The tricky part occurs when $k < i + s$, in which case v_i and v_k overlap. But one can partition the set of coordinates into two equal-sized sets S_1, S_2 such that $v_i' = v_i|_{S_1}, v_k' = v_k|_{S_1}$ are fully independent and $v_i'' = v_i|_{S_2}, v_k'' = v_k|_{S_2}$ are also fully independent. Since $|S_1|, |S_2| = s/2$, we apply Chernoff bound to each to conclude that $|v_i' v_k'| \leq Cs/2$ and $|v_i'' v_k''| \leq Cs/2$, with high probability. Hence $|v_i v_k| \leq Cs$ as well. □

We prove a similar claim for intersecting pairs of vectors v_i and u_j.

Claim 6.7. *There exists a small constant* $C > 0$ *such the following holds. Fix any* $0 \leq i, j < \sqrt{n/p}$, *and suppose* $i + t' \neq j\sqrt{n/p} \mod n/p$. *Then* $|v_i u_j| \leq Cs$ *with high probability.*

Proof. Again, if $s \leq |i + t' - j\sqrt{n/p}| \mod n$, then v_i and u_j are completely independent. So assume the opposite. In such a case, we can partition the set of coordinates into equal sized sets S_1 and S_2 so that $v_i|_{S_1}, u_j|_{S_1}$ are completely independent, and $v_i|_{S_2}, u_j|_{S_2}$ are completely independent too. Applying Chernoff bound to each of them completes the proof. □

Note that, for the right shift, when $i + t' = j\sqrt{n/p} \mod n/p$, we have, by Chernoff bound, that $|v_i u_j| \geq C'\epsilon s$ for some constant C' satisfying $C'\epsilon > C$.

At this moment, we can just plug-in the result for the closest pair problem 6.1. Note that the way we choose the sets of vectors to run, all the vectors, except at most one pair, are independent. After running all the $O(s)$ closest pair instances, by the above claims, we find the pair i, j that maximizes $\arg\max_{i,j} |v_i u_j|$, and, by the above claims, we have that $i + t' = j\sqrt{n/p} \mod n/p$.

The rest of the proof of the algorithm follows precisely the same way as in §5. □

We now switch to runtime analysis. For any $n \geq 1$ and $p \leq \sqrt{n}$, the algorithm from §6.1 runs in time $O(\sqrt{n/p} \log n \cdot p) + O(\log n \cdot T(2\sqrt{n/p}, O(\log n))$, where $T(n, d)$ is the running time of the random closest pair on n vectors in d-dimensional Hamming space (as in the statement of Theorem 6.1. For $T(n, O(\log n)) = n^f (\log n)^{O(1)}$, we set $p = n^{(f-1)/(f+1)}$ to obtain total time $n^{f/(f+1)} \log^{O(1)} n$. For $f = 1.779$ from Theorem 6.1, we obtain a total runtime of $O(n^{0.641})$.

7 Pattern Matching

We address the pattern matching problem in the random setting. The problem is defined as follows. Fix a text/signal $\mathbf{x}$ of length n. The pattern/code is a vector $\mathbf{c}$ of length $m \ll n$ obtained as follows. For a random $\tau \in \{0, 1, \ldots n - m\}$, pick a substring $\mathbf{x}_{[\tau, \tau+m-1]}$ of $\mathbf{x}$ starting at position τ of length m, and let $\mathbf{c} = \mathbf{x}_{[\tau:\tau+m-1]} \odot \mathbf{b}$, where $\mathbf{b}$ is Bernoulli noise of bias $\eta < 1/2$. Given $\mathbf{c}$ and $\mathbf{x}$, the goal is to recover τ.

Theorem 7.1. *Fix integers* $n \geq m \geq 1$, *a bias* $0 < \eta < 1/2$, *and assume* $m = \Omega((1/2-\eta)^{-1} \log n)$. *Suppose we are given a vector* $\mathbf{x}$ *of length* n, *and a vector* $\mathbf{c}$ *of length* m, *where* $\mathbf{c} = \mathbf{x}_{[\tau:\tau+m-1]} \odot \mathbf{b}$, *where* $\tau \in [0, n-m]$ *and* $\mathbf{b}$ *is a vector of i.i.d.* $\pm$ *random variables of bias* η. *There is an algorithm to reconstruct* τ *with high probability, in time* $O(n/m^{0.359})$.

Proof. The algorithm proceeds as follows. Let $I = \{0, \Delta, 2\Delta, 3\Delta, \ldots (n/\Delta - 1)\Delta\}$ for $\Delta = m/2$. For each $i \in I$, we take $\mathbf{y} = \mathbf{x}|_{[i:i+m-1]}$, and apply the algorithm from §6.1 to the strings $\mathbf{c}$ and $\mathbf{y}$. Once the algorithm returns a shift τ_i for the current i, check whether $\mathbf{x}|_{[i-\tau_i, i-\tau+m-1]} \cdot \mathbf{c} \geq \Omega((1/2-\eta)m)$. If this is the case, then report $\tau = i - \tau_i$.

We analyze the runtime of the algorithm. Since each application of the algorithm from §6.1 takes $O(m^{0.641})$ time, the entire algorithm runs in time $O(|I| \cdot m^{0.641}) = O(n/m \cdot m^{0.641}) = O(n/m^{0.359})$.

We continue with analyzing the correctness of the algorithm. For this, we just reuse the analysis of Theorem 6.4 from §6.2. First of all, note that the algorithm will never report a wrong τ. Thus we

just need to show the algorithm will find the right τ eventually.

Fix $i \in I$ to be such that $\tau \leq i \leq \tau + m/2$. In this case, for $\delta = i - \tau$, we notice that $\mathbf{x}|_{[i,i+m-\delta-1]} = \mathbf{c}|_{[\delta:m-1]} \odot \mathbf{b}|_{[\delta:m-1]}$. Hence the hope is that the algorithm recovers $\tau_i = \delta$, which would lead to recovery of the correct $\tau = i - \tau_i$. We are almost in the setting of the algorithm from §6.1, except that $\mathbf{y}|_{[m-\delta,m-1]}$, and $\mathbf{c}|_{[0,\delta-1]}$ are random independent bits. We can think of this as having $\mathbf{y} = \mathbf{c}^{(\delta)} \odot \bar{\mathbf{b}}$, where $\bar{b}_j = b_j$ for $j \in [0, m-\delta-1]$ and $\bar{b}_j$ is random ± 1 for $j \in [m-\delta, m-1]$.

Now take the p-folds of $\mathbf{y}$ and $\mathbf{c}$. This can be seen as a p-folds of the "common" part, $\mathbf{y}|_{[0,m-\delta-1]}$ and $\mathbf{c}|_{[\delta,m-1]}$, plus the p-folds of the random parts, $\mathbf{y}|_{[m-\delta,m-1]}$ and $\mathbf{c}|_{[0,\delta-1]}$. We now use Lemma 6.2. Note that for all integers l, we have $\sum_j \bar{b}_{l+jn/p} = \Omega((1/2-\eta)p)$ since at least half of the $\bar{b}(p)_j$'s have bias towards 1 (while the others are random ± 1). Also, for $k + \delta \neq l \mod n/p$, $y(p)_k$ is completely uncorrelated with $c(p)_l$. Thus we have the following claim:

Claim 7.2. *Fix any integers k, l, and let $c'(p)_k = \text{sign}(c(p)_k)$, $y'(p)_l = \text{sign}(y(p)_l)$.*

If $k + \delta = l \mod n/p$, then $\Pr[y'(p)_k = c'(p)_l] \geq 1/2 + \epsilon$, for some $\epsilon = \epsilon(1/2-\eta) > 0$. On the other hand, if $k + \delta \neq l \mod n/p$, then $\Pr[y'(p)_k = c'(p)_l] = 1/2$.

Given the above claim, the rest of the proof follows precisely the proof of Theorem 6.4. □

Acknowledgments: This research is supported by David and Lucille Packard Fellowship, and NSF grants CCF 1065125, CCF 1012042, and CNS 0831664.

References

[Ach03] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *J. Comput. System Sci.*, 66(4):671–687, 2003. Special issue on PODS 2001 (Santa Barbara, CA).

[AIK08] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *SODA*, pages 343–352, 2008.

[ALP04] A. Amir, M. Lewenstein, and E. Porat. Faster algorithms for string matching with k mismatches. *J. Algorithms*, 50 (2):257–275, 2004.

[AN10] A. Andoni and H. L. Nguyen. Near-optimal sublinear time algorithms for ulam distance. *SODA*, pages 76–86, 2010.

[BEK+03] T. Batu, F. Ergun, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, and R. Sami. A sublinear algorithm for weakly approximating edit distance. pages 316–324, 2003.

[BM77] R. S. Boyer and J.S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20:762–772, 1977.

[CLW+09] A. Chandrakasan, F. Lee, D. Wentzloff, V. Sze, G. Ginsburg, P. Mercier, D. Daly, and R. Blazquez. Low-power impulse uwb architectures and circuits. *Proceedings of the IEEE*, 97:332–352, 2009.

[CM94] W. Chang and T. Marr. Approximate string matching and local similarity. *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, page 259273, 1994.

[CM11] Michael S. Crouch and Andrew McGregor. Periodicity and cyclic shifts via linear sketches. In *APPROX-RANDOM*, pages 158–170, 2011.

[FP74] M. J. Fischer and M. S. Paterson. String matching and other products. *Complexity of Computation, SIAM-AMS Proceedings*, 7:113–125, 1974.

[HAKI12] H. Hassanieh, F. Adib, D. Katabi, and I. Indyk. Faster GPS via the sparse Fourier transform. *MOBICOM*, 2012.

[HIKP12a] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly optimal sparse Fourier transform. *STOC*, 2012.

[HIKP12b] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Simple and practical algorithm for sparse Fourier transform. *SODA*, 2012.

[IMNO11] P. Indyk, A. McGregor, I. Newman, and Krzysztof Onak. Open problems in data streams, property testing, and related topics. 2011. http://www.cs.umass.edu/m̃cgregor/papers/11-openproblems.pdf.

[IN07] P. Indyk and A. Naor. Nearest-neighbor-preserving embeddings. 3(3), 2007.

[ITU] ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG4-AVC). Advanced video coding for generic audiovisual services. May, 2003; v2, Jan. 2004; v3 (with FREext),Sept. 2004; v4, July 2005.

[Kap96] Elliott D. Kaplan. *Understanding GPS Principles and Applications*. Artech House Publishers, 1996.

[KMP77] D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching. *SIAM J. on Computing*, 6:323–350, 1977.

[KR06] Robert Krauthgamer and Yuval Rabani. Improved lower bounds for embeddings into L_1. In *SODA*, pages 1010–1017, 2006.

[LV89] G.M. Landau and U. Vishkin. Fast parallel and serial approximate string matching. *J. Algorithms*, 10 (2):157169, 1989.

[Nav01] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys (CSUR) Surveys*, 33(1):31 – 88, 2001.

[Spi00] John L. Spiesberger. Finding the right cross-correlation peak for locating sounds in multipath environments with a fourth moment function. *Journal of the Acoustical Society of America*, 108:1349–1352, 2000.

[Val12] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas with noise. In *FOCS. Preliminary version as ECCC TR12-006*, 2012.

[Wil12] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. pages 887–898, 2012.