Shift Finding in Sub-linear Time

Alexandr Andoni

Haitham Hassanieh

Piotr Indyk

Dina Katabi

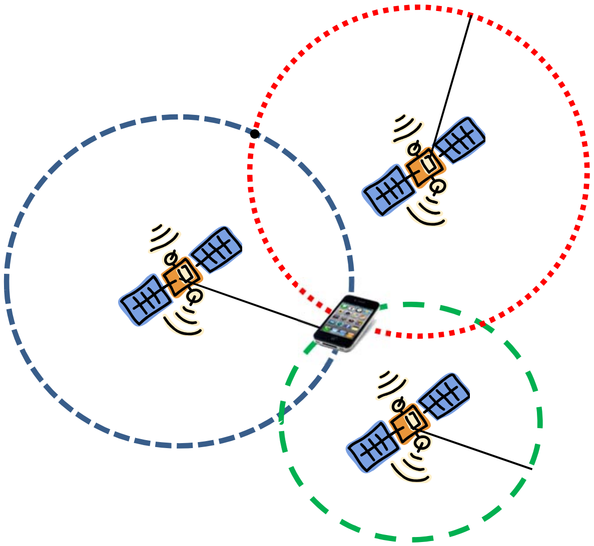
Massachusetts Institute of Technology

Shift Finding

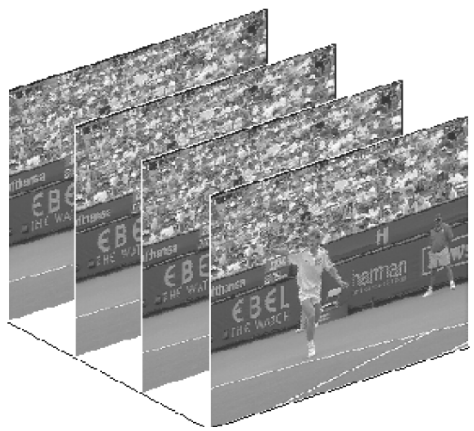
- Given a code vector $\mathbf{c} \in \{-1,+1\}^n$
- Signal vector $\mathbf{x} = \mathbf{c}^{(\boldsymbol{\tau})} + \mathbf{g}$
  - $\mathbf{c}^{(\tau)}$ is the code shifted by $\boldsymbol{\tau}$, $\mathbf{c}_i^{(\boldsymbol{\tau})} = \mathbf{c}_{i+\tau \bmod n}$
  - $\mathbf{g}$ is Gaussian noise vector with variance σ^2
  - Noise can also be Boolean $\mathbf{x} = \mathbf{c}^{(\tau)} \odot \mathbf{b}$ (bit flips: $\mathbf{b}_i = -1$ with probability η < 1/2)
- Estimate τ :

$$\tau = \underset{t}{\operatorname{argmax}}\ \mathbf{x} \cdot \mathbf{c}^{(t)}$$

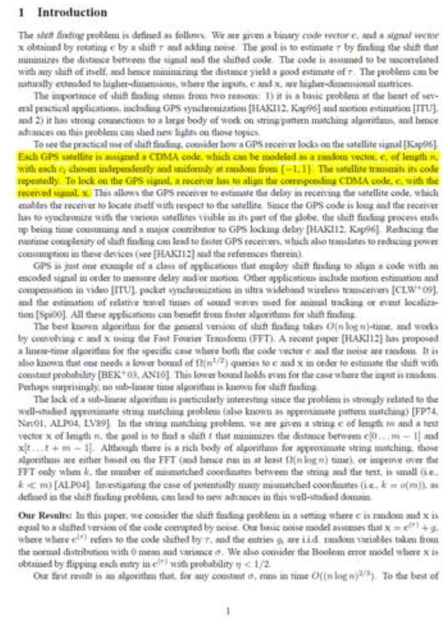
Shift Finding



GPS Synchronization



Motion Estimation (video encoding)



1 Introduction

The *shift finding* problem is defined as follows. We are given a binary *code vector* c, and a *signal vector* x obtained by rotating 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, c and 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, 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, c, with the received signal, 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 c and 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 c and the noise are random. It is also known that one needs a lower bound of $\Omega(n^{1/2})$ queries to c and 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 c of length m and a text vector x of length n, the goal is to find a shift t that minimizes the distance between $c[0 \ldots m-1]$ and $x[t \ldots t+m-1]$. 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.

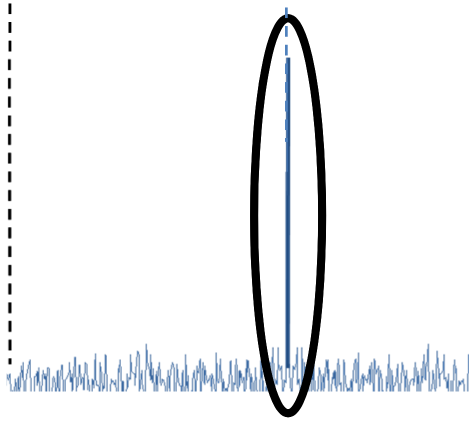
Our Results: In this paper, we consider the shift finding problem in a setting where c is random and x is equal to a shifted version of the code corrupted by noise. Our basic noise model assumes that $x = c^{(\tau)} + g$, where where $c^{(\tau)}$ refers to the code shifted by τ, and the entries g_i are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

1

String/ Pattern Matching

GPS Example



Correlation

Compute $\mathbf{c} * \mathbf{x}$: $O(n^2)$

Shift Finding using FFT

Convolution in time $\leftrightarrow$ Multiplication in frequency.

$$\begin{array}{l} \mathbf{c} \rightarrow \mathrm{FFT} \rightarrow \hat{\mathbf{c}} \searrow \\ \qquad\qquad\qquad\qquad \hat{\mathbf{c}} \odot \hat{\mathbf{x}} \rightarrow \mathrm{IFFT} \rightarrow \mathbf{c} * \mathbf{x} \\ \mathbf{x} \rightarrow \mathrm{FFT} \rightarrow \hat{\mathbf{x}} \nearrow \end{array}$$

Compute $\mathbf{c} * \mathbf{x}$ in frequency domain : $O(n \log n)$

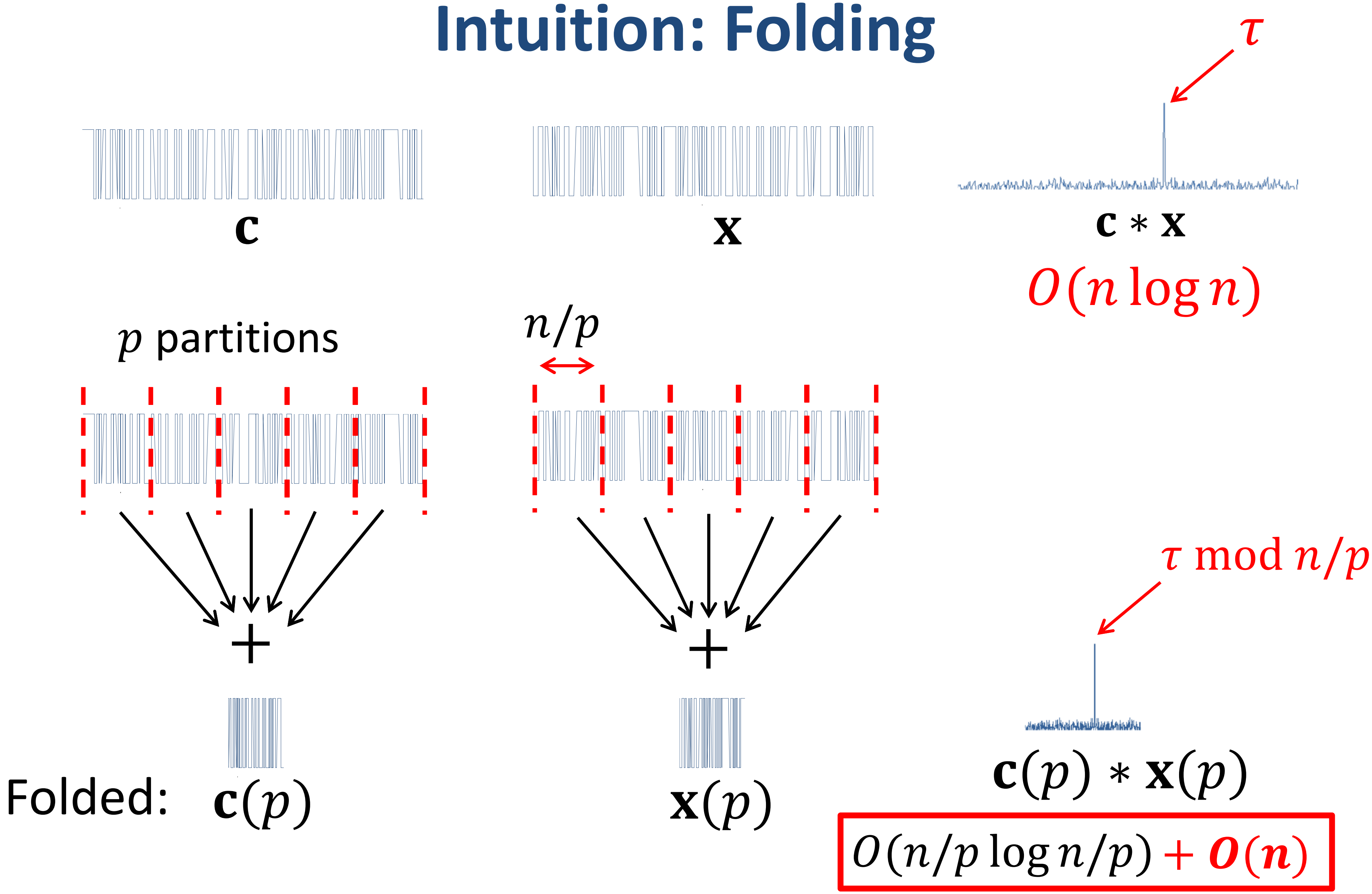
Previous Work

- Lower Bound $\Omega(\sqrt{n})$ [BEK+03], [AN10]
- Linear Time Algorithm: $O(n)$ [HAKI12]
  - Assuming the noise is Gaussian and bounded by $O(n/\log^2 n)$
- Approximate pattern matching algorithms:
  - Code $\mathbf{c}$ of length m and signal $\mathbf{x}$ of length $n \gg m$
  - Find shift τ that minimizes distance between $\mathbf{c}$ and $\mathbf{x}\,[\tau : \tau + m - 1]$
  - Known Results:
    - Use FFT and have run times: $\Omega(n \log n)$ [FP74]
    - Improves over FFT small number of mismatched coordinates k: $O\left(n\sqrt{k \log k}\right)$ [ALP04]
    - Sub-linear in n for large m : $O(n/m\; k \log n)$ [CM94]

Our Results

- First sub-linear time algorithms for the shift finding problem.
- Assumptions:
  - The code c is random ($\mathbf{c}_i = +1$ with prob. $1/2$)
  - Noise is Gaussian with variance $O(1)$
  - Noise is Boolean with flip probability $< 1/2$
- Results:
  - Simple algorithm that recovers the correct shift with large constant probability in $O\left((n \log n)^{2/3}\right)$
  - Faster algorithm with running time to $O(n^{0.641})$
    - Uses fast matrix multiplication
  - Generalize the algorithms to pattern matching
    - Sub-linear time : $O(n/m^{0.359})$
    - Do not require the number of mismatched coordinates k to be small (i.e. $k = O(m)$)

Intuition: Folding



Sub-linear Time Shift Finding Algorithms

At high level, both algorithms have 2 steps:

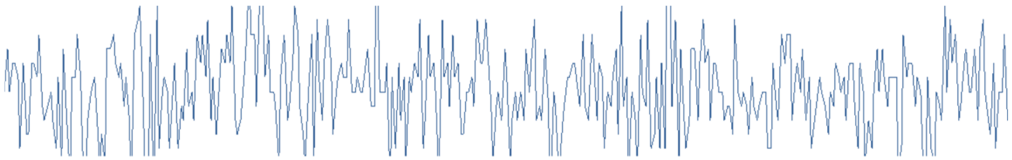
1. Estimate the shift $\tau \bmod n/p$

2. Out of p shifts that satisfy $\tau \bmod n/p$, find τ

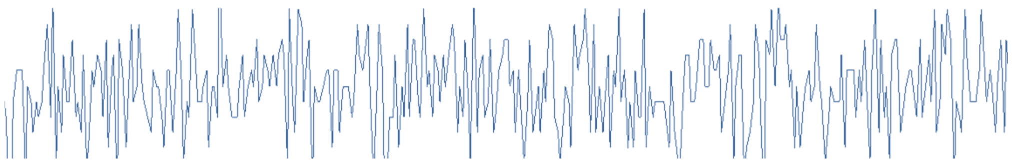
Rest of the talk:

- Simple Algorithm with run time $O\left((n \log n)^{2/3}\right)$
- Faster Algorithm with run time $O(n^{0.641})$

Step 1: Estimate the shift $\tau \bmod n/p$

$\mathbf{x}(p)$ 

$O(n)$

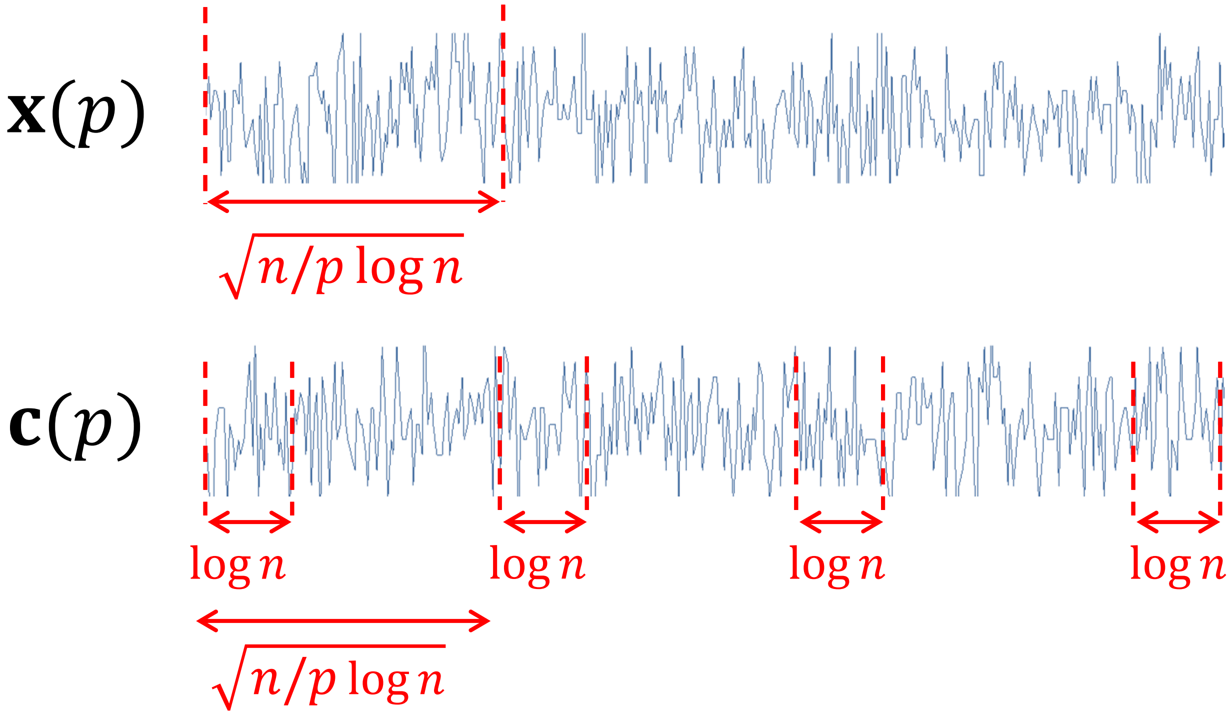
$\mathbf{c}(p)$ 

$$\tau \bmod n/p = \operatorname*{argmax}_{0 \le t < n/p} \mathbf{c}(p)^{(t)} \cdot \mathbf{x}(p)$$

- Goal: for $0 \le t < n/p$ estimate:

$$d_t = \mathbf{c}(p)^{(\mathbf{t})} \cdot \mathbf{x}(p) = \sum_{0 \le i < n/p} \mathrm{c}(p)_{\boldsymbol{i}}^{(\boldsymbol{t})} \cdot \mathrm{x}(p)_i$$

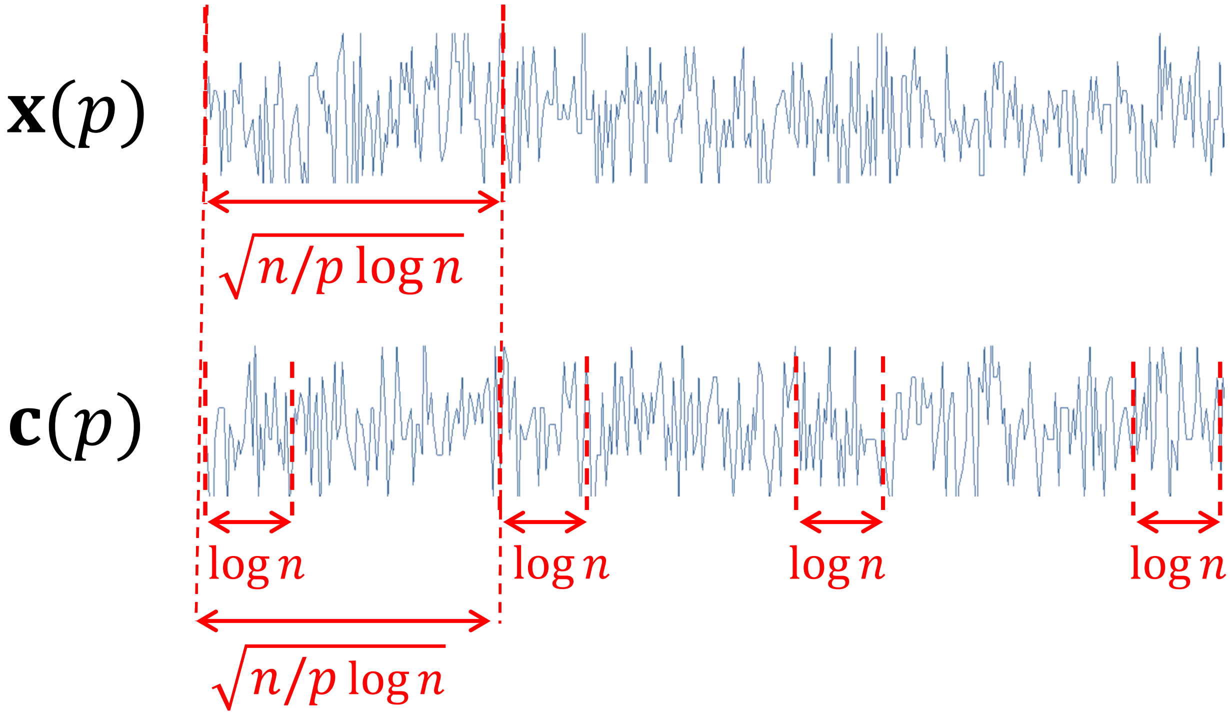
Step 1: Estimate the shift $\tau \bmod n/p$



$$d_t = \sum c(p)_i^{(t)} \cdot x(p)_i$$

- Compute the first $\sqrt{n/p \log n}$ samples of $\mathbf{x}(p)$
- Compute $\sqrt{n/(p \log n)}$ equally spaced segments of $\mathbf{c}(p)$ each of length $O(\log n)$

Step 1: Estimate the shift $\tau \bmod n/p$



$$d_t = \sum \mathrm{c}(p)_i^{(t)} \cdot \mathrm{x}(p)_i$$

- For any shift t , $\mathbf{c}(p)^{(t)}$ and $\mathbf{x}(p)$ have $O(\log n)$ common terms
- Approximate d_t using $O(\log n)$ terms of the summation
- Estimate: $\tau \bmod n/p = \underset{0 \le t < n/p}{\operatorname{argmax}}\ d_t$

Step 2: Out of p shifts satisfying $\tau \bmod n/p$, find τ

- We know: $\tau = \underset{0 \le t < n}{\operatorname{argmax}} \; \mathbf{x} \cdot \mathbf{c}^{(t)}$
- Only p shifts are equal to $\tau \bmod n/p$
- Enumerate p shifts: $\tau = \underset{t = \tau \bmod n/p}{\operatorname{argmax}} \; \mathbf{x} \cdot \mathbf{c}^{(t)}$
- Approximate distance using $O(\log n)$ samples:

$$\tau = \underset{t = \tau \bmod n/p}{\operatorname{argmax}} \sum_{0 \le i < \log n} \mathrm{c}_i^{(t)} \cdot x_i$$

Runtime of Simple Algorithm

- Step 1: Estimate $\tau \bmod n/p$
  - Compute $\sqrt{n/p \log n}$ samples of $\mathbf{c}(p)$ and $\mathbf{x}(p)$ → $O\left(p\sqrt{n/p \log n}\right)$
  - For $0 \leq t < n/p$, compute d_t using $O(\log n)$ samples → $O(n/p \log n)$
- Step 2: Enumerate p shifts → $O(p \log n)$

Total Runtime : $O\left(p\sqrt{n/p \log n} + n/p \log n\right)$

$= O\left((n \log n)^{2/3}\right)$ for $p = (n \log n)^{1/3}$

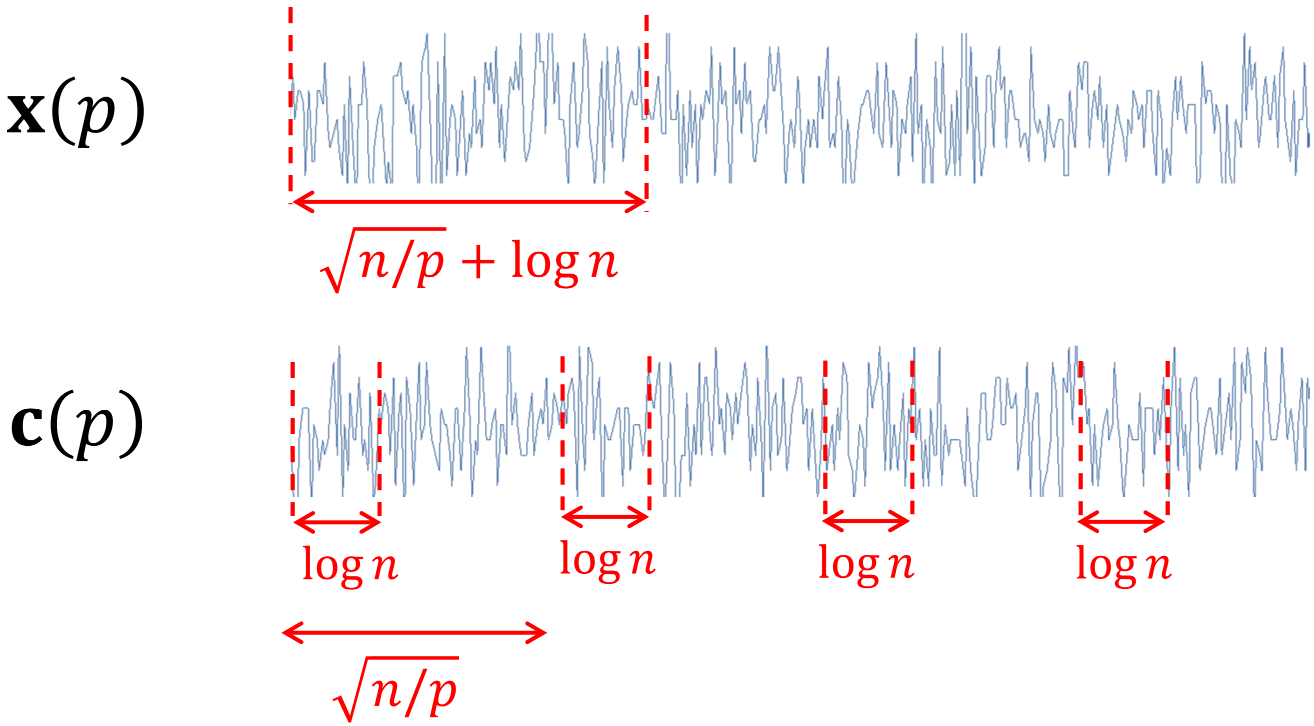
Faster Algorithm

- Step 1: Estimate $\tau \bmod n/p$
  – Use the closest pair algorithm which follows from Valiant [Val12]:

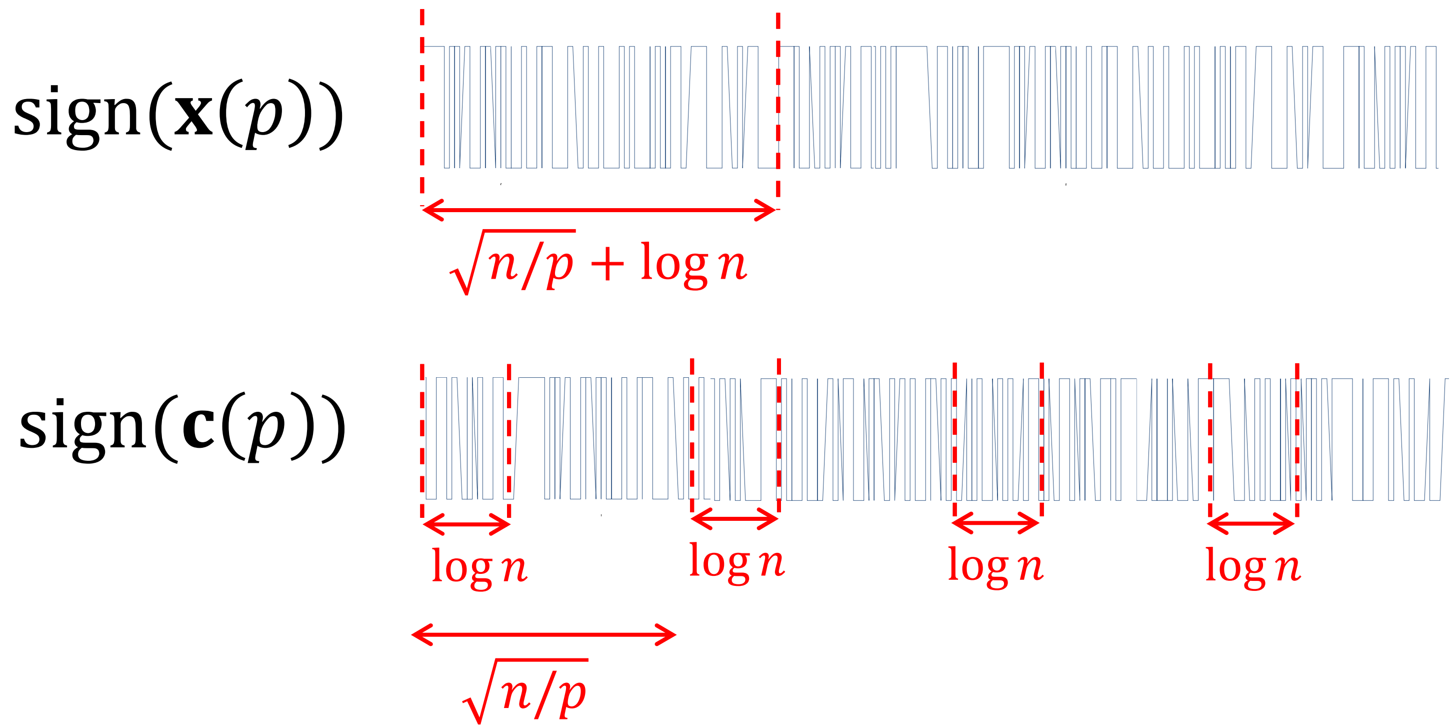
  Given n independent random vectors $\{\pm 1\}^d$, except for at most one pair that is ρ-correlated, we can find this pair in $dn^f \rho^{O(1)}$ time where $f \leq 1.779$

- Step 2: Enumerate p shifts

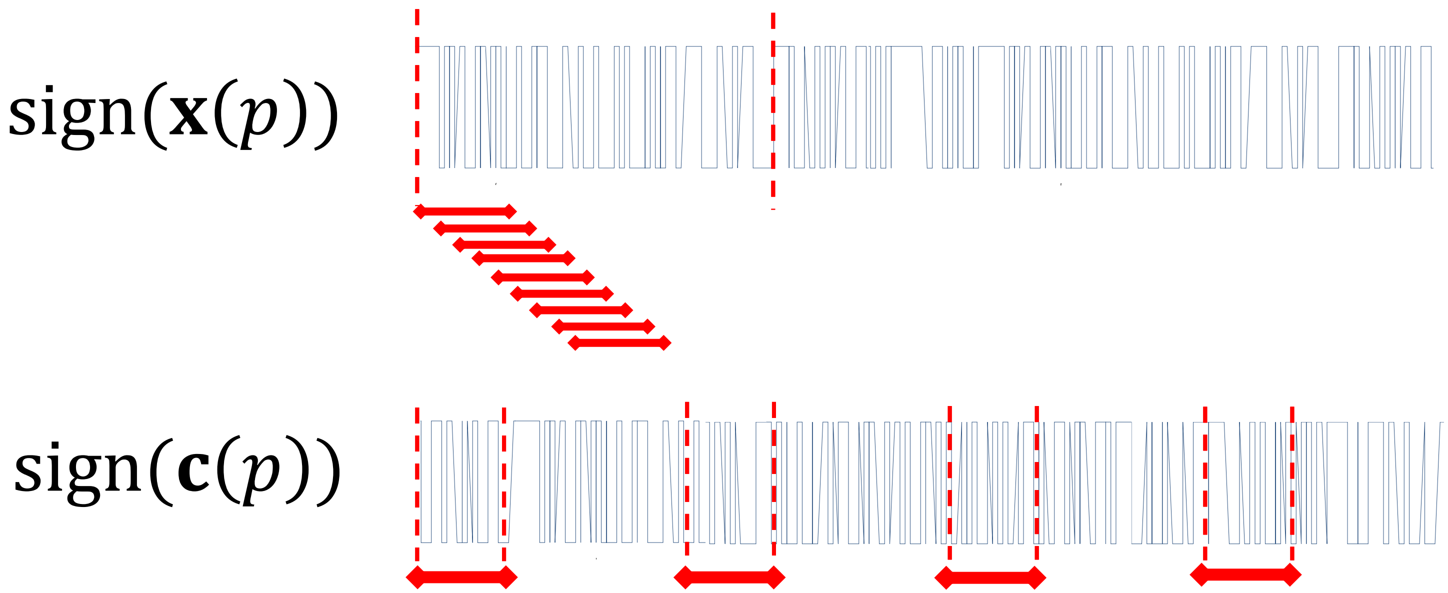
Faster Algorithm



Faster Algorithm



Faster Algorithm



- $2 \times \sqrt{n/p}$ vectors from $\{\pm 1\}^{\log n}$
- Find the two most correlated vectors

Runtime of Faster Algorithm

- Step 1: Estimate $\tau \bmod n/p$
  - Compute $\sqrt{n/p}\log n$ samples of $\mathbf{c}(p)$ and $\mathbf{x}(p)$ → $O\left(p\log n\sqrt{n/p}\right)$
  - Run the algorithm form [VAL12] $\log n$ times → $O\left(\left(\sqrt{n/p}\right)^{f}\log^{O(1)} n\right)$

- Step 2: Enumerate p shifts → $O(p\log n)$

Total Runtime for $p = n^{(f-1)/(f+1)}$: $O\left(n^{f/(f+1)}\log^{O(1)} n\right)$

for [VAL12], $f = 1.779$: $O\left(n^{0.641}\right)$

If , $f = 1+\epsilon$: $O\left(n^{(1+O(\epsilon))/2}\right)$ → Matches lower bound

Conclusion

- First sub-linear time algorithms for shift finding problem
- Simple algorithm with runtime in $O\left((n \log n)^{2/3}\right)$
- Faster algorithm with runtime to $O(n^{0.641})$
- Generalize to pattern matching with runtime $O(n/m^{0.359})$
- To achieve lower bound of $\Omega(\sqrt{n})$, it suffices to have a closest pair algorithm that is linear in number of vectors.