Pattern Matching in Sub-Linear Time Using a Sparse Fourier Transform Approach

Nagaraj T. Janakiraman, Avinash Vem, Krishna R. Narayanan
Department of Electrical & Comp. Engg., Texas A&M University, College Station, TX, U.S.A
{tjnagaraj,vemvinash,krn}@tamu.edu

Abstract

We consider the problem of querying a string (or, a database) of length N bits to determine all the locations where a substring (query) of length M appears either exactly or is within a Hamming distance of K from the query. We assume that sketches of the original signal can be computed off line and stored. Using the sparse Fourier transform computation based approach introduced by Pawar and Ramchandran, we show that all such matches can be determined with high probability in sub-linear time. Specifically, if the number of matches is $L = N^{\lambda}$ and $M = N^{\mu}$, as $N \to \infty$, we show that the locations can be determined with a probability that approaches 1, with a computational complexity that is only $O\left(\max\left(\frac{N}{M}\log N, ML\log N\right)\right) = O\left(N^{\max(1-\mu,\mu+\lambda)}\log N\right)$ for $K \leq \frac{2}{3}M$. Further, the number of Fourier transform coefficients that need to be computed, stored and accessed, i.e., the sketching complexity of this algorithm is only $O\left(N^{\max(\mu,1-\mu)}\right)$. Several extensions of the main theme are also discussed.

I. Introduction and Problem Statement

We consider the pattern matching problem in the random setting which is a problem that has been studied extensively in computer science. In this problem, we have a signal $\underline{x} := (x[1], x[2], \ldots, x[N])$ of length N symbols representing a string, library or database. We first consider the random setting of the problem in which $x[i]$'s are a sequence of independent and identically distributed (i.i.d) random variables taking values in $\mathscr{A} := \{1, -1\}$, although extensions to other alphabets $\mathscr{A} \subseteq \mathbb{R}$ are possible. We assume that sketches of $\underline{x}$ can be computed offline and stored, and the one-time computational complexity of creating the sketch is ignored.

Exact Pattern Matching: In the exact pattern matching problem, a substring of length M of $\underline{x}$, namely $\underline{y} := \underline{x}[\tau : \tau + M - 1]$ is obtained by taking M consecutive symbols from $\underline{x}$ and is presented as a query. This pattern may appear within $\underline{x}$ in L different locations $\tau_1, \tau_2, \ldots, \tau_L$ and the objective is to determine all the locations $\tau_i, \forall i \in [1 : L]$. We consider the probabilistic version where our objective is to recover the matching locations with a probability that approaches 1 as K, M and $N \to \infty$.

Approximate Pattern Matching: In the approximate pattern matching version, $\underline{y}$ is a noisy version of a substring, i.e., $\underline{y} = \underline{x}[\tau : \tau + L - 1] \odot \underline{b}$ where $\underline{b}$ is a noise sequence with $b[i] \in \pm 1$ such that $d_H(\underline{y}, \underline{x}[\tau : \tau + M - 1]) \leq K$. Here d_H denotes the Hamming distance, and $\odot$ represents component-wise multiplication. The objective is to determine all locations τ_i such that $d_H(\underline{y}, \underline{x}[\tau_i : \tau_i + M - 1]) \leq K$ with a probability that approaches 1 as K, M and $N \to \infty$.

These problems are relevant to many applications including text matching, DNA sequencing, and is particularly relevant now due to the interest in applications involving huge volumes of data. The presented results are most useful in the following situations - (i) the string $\underline{x}$ is available before querying and one time computations such as computing a sketch of $\underline{x}$ can be performed off line and the sketch can be stored in the database and the complexity of computing the sketch of $\underline{x}$ can be ignored. Then, when queries in the form of $\underline{y}$ appears, one would like to decrease the computational complexity in searching for the string $\underline{y}$. (ii) The string $\underline{x}$ is not centrally available but parts of the string are sensed by different data collecting nodes distributively and communicated to a central server. A search query is presented to the server and the server needs to decide if the string appeared in the data sensed by one or more of the distributed nodes and the time instants where the query appeared. In this case, we would like to minimize the amount of data communicated by the nodes to the server and the computational complexity in

This work was supported by the National Science Foundation under grants

searching for the string. Finally, the proposed approach is most useful when the query $\underline{y}$ is not a pattern that can be predicted and, therefore, creating a look up table to quickly identify patterns is not efficient. For e.g., $\underline{x}$ could be an i.i.d sequence of ± 1 and $\underline{y}$ could be a substring of $\underline{x}$.

A naive approach to searching for a substring $\underline{y}$ in $\underline{x}$ would involve computing the cross correlation between them denoted by $\underline{r} = [r[1], \ldots, r[N]]$

$$r[m] = (\underline{x} * \underline{y})[m] \triangleq \sum_{i=1}^{M} x[m+i-1]y[i] \tag{1}$$

and choosing the index m that maximizes $r[m]$. This approach uses all the N samples of $\underline{x}$ and has a super-linear computational complexity of $MN = N^{1+\mu}$. A computationally more efficient approach uses the fact that $R_{XY}[m]$ can be computed by taking the inverse Fourier transform of the product of the Fourier transforms of $\underline{x}$ and $\underline{y}^*[-n]$, where $\underline{y}^*[-n]$ is the conjugate of time reversed $\underline{y}$. Such an approach still uses all the N samples of $\underline{x}$ but reduces the computational complexity to $O(N \log N)$. Note that even though the Fourier transform of $\underline{x}$ can be precomputed, the N-point Fourier transform of $\underline{y}$ still needs to be computed resulting in the $O(N \log N)$ computational complexity.

Both the exact pattern matching problem and the approximate pattern matching problem have been extensively studied in computer science and two recent papers [1] and [2] provide a brief summary of the results. For the exact matching problem, Boyer and Moore presented an algorithm in [3] for finding the first occurrence of the match (only τ_1) that has an expected complexity of $O(N/M \log N) = O(N^{1-\mu} \log N)$, whereas the worst case complexity (depending on τ_1's) can be $O(N \log N)$. For large M, the algorithm indeed has an average complexity that is sub-linear in N. This algorithm has been generalized to the approximate pattern matching problem in [4] with an average case complexity of $O(NK/M \log N)$ which provides a sub-linear time algorithm only when $K \ll M$. In the work by Andoni, Hassanieh, Indyk and Katabi [1], the authors gave the first sub-linear time algorithm with a complexity of $O(N/M^{0.359})$ even for $K = O(M)$.

II. Our Main Results and Relation to Prior Work

Assuming that a sketch of $\underline{x}$ of size $O(N/M)$ can be computed and stored,

1) For the exact pattern matching problem, we show an algorithm which uses a sketching function for $\underline{y}$ with $O(\max(M \log N, N/M \log N))$ samples and a computational complexity of $O(\max(ML \log N, N/M \log N))$. When $M = N^{\mu}, L = N^{\lambda}$, this gives a sub-linear time algorithm.
2) For the approximate pattern matching problem, for $K = O(M)$, our algorithm has a sketching complexity of $\underline{y}$ is $O(\max(M \log N, N/M \log N))$ samples and computational complexity of $O(\max(ML \log N, N/M \log N))$ for all $K = O(M)$.

Our paper is inspired by and builds on two recent works by Hassanieh et al in [5] and Pawar and Ramchandran [6]. In [5] Hassanieh et al considered the correlation function computation problem for a GPS receiver and exploited the fact that the cross-correlation vector $\underline{r}$ is a very sparse signal in the time domain and, hence, the Fourier transform of $\underline{r}$ need not be evaluated at all the N points. In the GPS application, which was the focus of [5], the query $\underline{y}$ corresponds to the received signal from the satellites and, hence, the length of the query was at least N. As a result, the computational complexity is still $O(N \log N)$ (still linear in N) and the only the constants were improved in relation to the approach of computing the entire Fourier transform. In a later paper by Andoni *et al.,* [1], a sub-linear time algorithm for shift finding in GPS is presented; however, this algorithm is completely combinatorial and eschews algebraic techniques such as FFT-based techniques.

There are important differences between our paper and [5], [1], [3], [2]. First and foremost, the algorithms used for pattern matching are entirely different. While their algorithms are combinatorial in nature, our algorithm is algebraic and uses signal processing and coding theoretic primitives. Secondly, the system model considered in our paper is different from the model in [5], [1], [3], [2] in that we allow for preprocessing or creating a sketch of the data $\underline{x}$. Our algorithm exploits this fact and results in a computational complexity $O(N/M)$ which is better than that in [1] for the approximate pattern matching problem. Finally, we also consider the problem of finding all matches of the pattern $\underline{y}$ instead of looking for only one match. In [6], Pawar and Ramchandran present an algorithm based

on aliasing and the peeling decoder for computing the Fourier transform of a signal with noisy observations for the case when the Fourier transform is known to be sparse. However, they do not consider the pattern matching problem. Our algorithm is very similar to that of Pawar and Ramchandran's algorithm and is optimized for the specific noise model that arises from the string matching problem. Due to some special constraints on the signal model of $\underline{\mathrm{r}}$ that is induced in this problem, we are able to compute the Sparse Inverse Discrete Fourier Transform (SIDFT) with $O(N^{1-\alpha}\log N)$, $0<\alpha\leq 1$, time complexity as compared to $O(N\log N)$ time complexity in FFAST, i.e. sub-linear time complexity as opposed to linear time complexity. The main contributions of this paper is to show that Pawar and Ramchandran's algorithm with some key modifications in the decoding process can be used to solve the pattern matching algorithm in sub-linear time.

III. Notations

This table below will introduce the notations we use in this paper.

Symbol	Notational Meaning
N	Size of the string or database in symbols
$M=N^{\mu}$	Length of the query in symbols
$L=N^{\lambda}$	Number of matches
K	Maximum Hamming distance allowed between $\underline{\mathrm{y}}$ and $\underline{\mathrm{x}}$
$G=N^{\gamma}$	Number of blocks
$\tilde{N}=N^{1-\gamma}$	Length of one block
$A=N^{\alpha}$	Sub-sampling parameter
B	Number of shifts
d	Number of stages in the FFAST algorithm

We denote vectors using the underbar, time domain signals using lowercase letters and the frequency domain signals using uppercase letter. For example $\underline{\mathrm{x}}=(x[1],x[2],\cdots x[N])$ denotes a time domain signal with i^{th} time component denoted by $x[i]$, and $\underline{\mathrm{X}}=\mathscr{F}\underline{\mathrm{x}}$ denotes the Fourier coefficients of $\underline{\mathrm{x}}$. Matrices are denoted using uppercases letters. We differentiate a signal from a matrix by having a underbar for a signal vector. We denote the set $0,1,2\cdots,N-1$ by $[N]$.

IV. Description Of The Algorithm

In this section we describe our algorithm, with sample and time complexities that are sub-linear in N, to find the locations $\underline{\tau}=(\tau_1,\tau_2,\cdots\tau_L)$ in the string $\underline{\mathrm{x}}$, where the string (query) $\underline{\mathrm{y}}$ matches. This is achieved by computing the cross-correlation $\underline{\mathrm{r}}$ of $\underline{\mathrm{x}}$ and $\underline{\mathrm{y}}$, defined in Eqn. (1), and finding the positions where there is a significant peak.

The main idea here is that $\underline{\mathrm{R}_{XY}}$ is sparse (upto some noise) with dominant peaks at L positions (τ) where the strings match, and noise components at $N-L$ positions where the strings do not match. Consider the case of exact matching,

$$r[m] \;=\begin{cases} M, & \text{if } m\in\mathscr{T} \\ n_m, & m\in[N]-\mathscr{T} \end{cases} \tag{2}$$

where, $\mathscr{T}:=\{\tau_1,\tau_2,\cdots,\tau_L\}$, n_m is the noise component that is induced due to correlation of two i.i.d. sequence of random variables each taking values from $\mathscr{A}:=\{+1,-1\}$. The $\underline{\mathrm{r}}$ can also be computed as shown below:

$$\underset{\text{I}}{\underline{\mathrm{r}}}=\underset{\text{I}}{\mathscr{F}_N^{-1}}\ \{\underset{\text{II}}{\mathscr{F}_N\{\underline{\mathrm{x}}\}}\odot\ \underset{\text{III}}{\mathscr{F}_N\{\underline{\mathrm{y}}'\}}\} \tag{3}$$

where $\mathscr{F}_N\{\cdot\}$ and $\mathscr{F}_N^{-1}\{\cdot\}$ refers to N-point discrete Fourier transform and its inverse respectively, $\odot$ is the point-wise multiplication operation and $y'[n]=y^*[-n]$. As evident from Equation 3, our algorithm for computing R_{XY} consists of three stages:

I *Computing sparse* $\mathscr{F}^{-1}$:

Since $\underline{r}$ is sparse, we use a Robust Sparse Inverse Discrete Fourier Transform(RSIDFT) framework to compute the L-sparse coefficients. The architecture of RSIDFT is similar to FFAST proposed in [6], but the decoding algorithm has some key modifications to handle the noise model induced in this problem.

A. RSIDFT Framework

Let $\underline{R} = \underline{X} \odot \underline{Y}'$ be the FFT of the cross-correlation of $\underline{x}$ and $\underline{y}$. We will see in Sec. IV-C how do we compute the required samples of $\underline{X}$ and $\underline{Y}$ but for this section we will assume that $\underline{X}$ and $\underline{Y}$ are available and the required samples of $\underline{R}$ is computed and input to RSIDFT framework. The RSIDFT framework computes the L dominant coefficients of $\underline{r}$ by using only a sub-sampled version of $\underline{R}$.

Consider the RSIDFT framework shown in Figure 1. The framework consists of *d-stages*, each with a different sub-sampling factor f_i computed as following. Let $N = P_1 \times P_2 \times \cdots P_d$ be the prime factorization of N, the length of the signal $\underline{x}$. For a given α, we choose distinct $f_i = \prod P_j$, $i \in [d]$ such that $f_i \approxeq N^{\alpha}$. In each stage, there are B *branches* with shifts from $\underline{s} = [s_1, s_2, \cdots s_B]$, where $s_1 = 0$ in the first branch and the rest chosen randomly from $[N^{\alpha}]$. We can also carefully choose the shifts to satisfy mutual coherence property and restricted isometric property described in the analysis section.

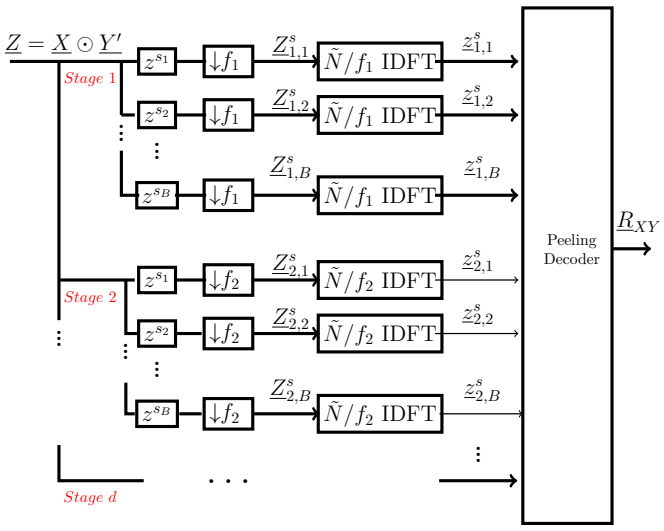

Fig. 1. RSIDFT Framework to compute inverse Fourier Transform of a signal $\underline{R}$ that is sparse in time domain.

Given the input $\underline{R}$, in branch j of i^{th} stage or referred to as *branch* (i, j) for convenience, RSIDFT sub-samples the signal $\underline{R}$ at

$$\mathscr{S}_{i,j} := \{s_j,\ s_j + f_i,\ s_j + 2f_i,\ \cdots s_j + n_i f_i\},$$

where $n_i := \lfloor \frac{N}{f_i} \rfloor$ to obtain $\underline{R}^s_{i,j} = \underline{R}[\mathscr{S}_{i,j}]$. The sub-sampling operation is followed by a n_i-point IDFT in each branch of stage i to obtain $\underline{r}^s_{i,j}$. Notice that $\underline{r}^s_{i,j}$ is an aliased version of $\underline{r}$ due to the property that sub-sampling in Fourier domain is equivalent to aliasing in time domain.

Let $\underline{r}^b_{i,p_i}$ be the vector of observations that is formed by combining all p_i^{th} coefficients of $\underline{r}^s_{i,j}$ (belonging to stage i) together as a vector, where $1 \le p_i \le n_i$, i.e.

$$\underline{r}^b_{i,p_i} = \begin{bmatrix} r^s_{i,1}[p_i] \\ r^s_{i,2}[p_i] \\ \vdots \\ r^s_{i,B}[p_i] \end{bmatrix}$$

In Fig. we represent the relation between $\underline{\mathrm{r}}$ and $\underline{\tilde{\mathrm{r}}}^b := [\underline{\mathrm{r}}^b_{1,1}, \underline{\mathrm{r}}^b_{1,2}, \ldots, \underline{\mathrm{r}}^b_{1,n_1}, \ldots, \underline{\mathrm{r}}^b_{2,1}, \ldots, \underline{\mathrm{r}}^b_{2,n_2}, \ldots, \underline{\mathrm{r}}^b_{B,1}, \ldots, \underline{\mathrm{r}}^b_{B,n_B}]$, aliased versions of the cross-correlation vector $\underline{\mathrm{r}}$ concatenated together, via a Tanner graph. We refer to $\underline{\tilde{\mathrm{r}}}^b$ as sensing signal. The nodes on the left, which we refer to as *variable nodes*, represent the N elements of vector $\underline{\mathrm{r}}$. And similarly the nodes on the right, which we refer to as *bin nodes*, represent the $\sum_{i \le B} n_i$ sub-sensing signals. We will now describe the decoding algorithm which takes the sensing signal $\underline{\tilde{\mathrm{r}}}^b$ as input and estimates the L-sparse $\underline{\mathrm{r}}$.

B. Decoder

Maybe remove Each coefficient of $\underline{\mathrm{r}}^b_{i,p_i}$ is a bin that has N^α coefficients from $\underline{\mathrm{R}_{XY}}$ hashed into it. This can be seen using a Tanner graph with $\underline{\mathrm{R}_{XY}}$ as the left nodes (bit nodes) and the aliased coefficients $\underline{\mathrm{r}}^b_{i,p_i}$ as the right nodes (check-nodes). Observe from the Tanner graph that the degree of each bit node is d and that of each bin node is approximately N^α. We refer to a bin node as *zero-ton* (or $\mathscr{H}_z$) if the number of variable nodes with significant amplitude connected to the bin node is zero. The *singleton ($\mathscr{H}_s$), double-ton ($\mathscr{H}_d$) and multi-ton ($\mathscr{H}_m$)* bin nodes are defined similarly where the number of significant variable nodes connected are one, two and greater than two, respectively. The peeling decoder has the following three steps in the decoding process.

- *Bin Identification*: In this step a bin node is classified as a zero-ton or a singleton or a multi-ton. At bin (i,j) the classification is done based on comparing the first observation $r^b_{i,j}[1]$, which corresponds to zero shift, with a predefined threshold. The threshold is set differently for Exact Matching and Approximate Matching cases. Let the value of $r^b_{i,j}[1]$ be z, then the classification rules can be written as follows:
  Exact Matching:
  $$\mathscr{H}(i,j) = \begin{cases} \mathscr{H}_z & z/M < 0.5 \\ \mathscr{H}_s & 0.5 \le z/M \le 1.5 \\ \mathscr{H}_d \text{ or } \mathscr{H}_m & z/M > 1.5 \end{cases}$$
  Approximate Matching:
  $$\mathscr{H}(i,j) = \begin{cases} \mathscr{H}_z & z/M < \gamma_1 \\ \mathscr{H}_s & \gamma_1 < z/M < \gamma_2 \\ \mathscr{H}_d & \gamma_2 < z/M < \gamma_3 \\ \mathscr{H}_m & z/M > \gamma_3 \end{cases}$$
  where $(\gamma_1, \gamma_2, \gamma_3) = (\frac{(1-3\eta/2)}{2}, \frac{(1-\eta/2)3}{2}, \frac{(5-3\eta)}{2})$.
- *Position Identification*: If a bin node (i,j) is classified as a singleton, we need to identify the position of the significant non-zero variable node connected to it. This is done by correlating the observation vector $\underline{\mathrm{r}}^b_{i,j}$ with each column of $S = [\underline{s_1}\ \underline{s_2} \cdots \underline{s_A}]$ and then picking the column index, p' that produced the maximum correlation value.
  $$p' = \arg\max_l \ \underline{s_l}^T\ \underline{\mathrm{r}}^b_{ij}$$
  Given that a check-node is classified as a Single-ton, we need to identify the position of the non-zero coefficient hashed into it. This is done by correlating the observation vector $\underline{z^b_{i,p_i}}$ with each column $\underline{g_l}$ of $G = [\underline{g_1}\ \underline{g_2} \cdots \underline{g_A}]$, where $\underline{g_l} = \left[w_1^l,\ w_2^l,\ \cdots\ w_B^l\right]^T$ with $w_k = e^{j\frac{2\pi s_k}{N}}$, and then picking the column index p' that produced the maximum value.
  $$p' = \arg\max_l \ \underline{s_l}^T\ \underline{z^b_{i,p_i}}$$
- *Peeling Process:* Although the idea behind the peeling process is same for Exact Matching and the Approximate Matching scenarios, there are some minor differences in their implementation. The main idea in this peeling based decoder is to find a singleton bin, identify the variable node connected to the bin and remove it's contribution from all the bin nodes connected to this variable node.

Algorithm 1 Peeling based recovery algorithm

while $\exists i \in [M_1] : \mathscr{H}_i = \mathscr{H}_Z$ or $\mathscr{H}_S$, **do**
  if $\mathscr{H}_i = \mathscr{H}_Z$ **then**
    Remove the bin i
  Assign 0 to all the variables connected
  else if $\mathscr{H}_i = \mathscr{H}_S(k, x[k])$ **then** Assign $x[k]$ to k^{th} variable in bin i
  Subtract $x[k]\mathbf{s}_k$ from connected $\mathbf{y}_i$
  Remove the bin and all variables connected

Exact Matching: Here we remove the identified Singleton's contribution from all the check-nodes it participates in.

Approximate Matching: Here we only remove the identified singleton's contribution from a double-ton and do not alter multi-tons whose *degree* > 2.

We exploit this property of sparsity to compute $\underline{\mathrm{R}_{XY}}$ by using only a subset of samples from $\underline{\mathrm{X}} = \mathscr{F}\{\underline{\mathrm{x}}\}$ and $\underline{Y}' = \mathscr{F}\{\underline{\mathrm{y}}'\}$, each sampled at positions $l \in \mathscr{S}$, where $\mathscr{S} = \mathscr{S}_{1,1} \cup \mathscr{S}_{1,2} \cup \cdots \cup \mathscr{S}_{i,j} \cup \mathscr{S}_{d,B} \subset \{0, 1, \cdots, N-1\}$ and $\mathscr{S}_{i,j}$, $1 \le i \le d$ and $1 \le j \le B$ are disjoint sets of size $|\mathscr{S}_{i,j}| \approxeq N^{1-\alpha}$ with periodic sample points from $[N]$ given by

$$\mathscr{S}_{i,j} = \{s_j,\ s_j + f_i,\ s_j + 2f_i,\ \cdots s_j + \lfloor \frac{N}{f_i} \rfloor f_i\} \tag{4}$$

where s_j's and f_i's are constants chosen based on the requirements from Robust Sparse Inverse Fourier Transform (RSIDFT) framework described in Section IV-A.

C. Sketches of $\underline{\mathrm{X}}$ and $\underline{Y}$

I *Computing the sketch of* $\underline{\mathrm{x}}$: We assume that the sketch of $\underline{\mathrm{x}}$, $\underline{\mathrm{X}}[l] = \mathscr{F}\{\underline{\mathrm{x}}\}$ is precomputed at positions $l \in \mathscr{S}$ and stored in a database.

II *Computing the sketch of* $\underline{\mathrm{y}}$: For every new query $\underline{\mathrm{y}}$, $\underline{Y}'[l]$ is computed at $l \in \mathscr{S}$. Naively, the FFT algorithm can be used to compute this with $O(N \log N)$ complexity. Since only a subset $\mathscr{S}$ of samples from $\underline{Y}'$ is needed, this can be done by using a folding technique described below. The idea behind this folding technique is to induce aliasing in $\underline{\mathrm{y}}'$ and then taking a smaller point Fourier Transform to compute the sub-sampled version $\underline{Y}'$. Aliasing is induced by folding the signal $\underline{\mathrm{y}}'$ into blocks of length $N^{1-\alpha}$ and adding them. The desired sub-sampling patterns in frequency domain are induced by multiplying $\underline{\mathrm{y}}'$ with suitable exponentials. Let us denote the aliased versions of $\underline{\mathrm{y}}'$ by $\underline{y^a_{i,j}}'$. Then, $\underline{y^a_{i,j}}'$ is given by

$$y^a_{i,j}{}'[p] = \sum_{m=0}^{\lfloor \frac{N}{f_i} \rfloor} y'[p + mf_i] e^{j\frac{2\pi s_j}{N}} \tag{5}$$

Taking $\frac{N}{f_i}$ point DFT of $\underline{y^a_{i,j}}'$ produces $\underline{Y}'[l]$ sub-sampled at $l \in \mathscr{S}_{i,j}$. To obtain all the samples in $\mathscr{S}$, the folding procedure needs to be carried out dB times, once for each (i, j) pair, where $1 \le i \le d$ and $1 \le j \le B$.

V. Performance Analysis

In this section, we will analyze the overall probability of error involved in finding the correct position of match. This can be done by analyzing the following three error events and then using a union bound to bound the total probability of error.

- $\mathscr{E}_1$: Event that a bin is wrongly classified (Bin Classification)
- $\mathscr{E}_2$: Event that a position is wrongly identified (Position Identification)
- $\mathscr{E}_3$: Event that the peeling process fails (Graph Analysis)

Let us analyze the three events independently. Let $\underline{Z}_j = (Z_j[1], Z_j[1], \cdots Z_j[B])$ denote the observation at bin j.

Lemma 1. *For a sufficiently large M, let* $\underline{X} = (X_1, X_2, \cdots, X_M)$ *and* $\underline{Y} = (Y_1, Y_2, \cdots, Y_M)$ *be two iid Bernoulli* $p = 0.5$ *random sequences. Then,the correlation between X and Y, defined by* $\rho = \sum_{i=1}^{M} X_i Y_i$ *follows a Gaussian distribution with mean zero and variance M.*

Proof.

$$\begin{aligned} E(\rho) &= E(\textstyle\sum_{i=1}^{M} X_i Y_i) = \sum_{i=1}^{M} E(X_i)(Y_i) = 0 \\ Var(\rho) &= E(\rho^2) - (E(\rho))^2 \\ E(\rho^2) &= E((\textstyle\sum_{i=1}^{M} X_i Y_i)^2) = E(\sum_{i=1}^{M} (X_i Y_i)^2) + 2E(\sum_{i \neq j} X_i Y_i X_j Y_j) \\ &= E(\textstyle\sum_{i=1}^{M} (X_i Y_i)^2) = \sum_{i=1}^{M} E((X_i Y_i)^2) = M \end{aligned}$$

□

A. Bin Classification

Approximate Pattern Matching:

The bin-*i* is classified as zero-ton, single-ton or a multi-ton if the first observation, $Z_i[1]$, i.e., the observation corresponding to the zero shift, satisfies a threshold constraint.

The value of first observation from each bin is a sum of N^α co-efficients from the correlation vector that are hashed into it. This can be modeled as a random variable given by

$$Z_i[1] = \begin{cases} n \sim \mathcal{N}(0, N^\alpha M), & \text{if it is zero-ton} \\ M - K + n \sim \mathcal{N}(M-K, N^\alpha M), & \text{if it is a single-ton} \\ 2(M-K) + n \sim \mathcal{N}(2(M-K), N^\alpha M), & \text{if it is a double-ton} \end{cases}$$

Here, $n = \sum_k n_k$ refers to the random variable that captures the sum of N^α random variables n_k, where $n_k \sim \mathcal{N}(0, M)$ points to the correlation of two iid Bernoulli random vectors of length M that do not match(Lemma 1). If the sequences match, the correlation value becomes M.

Also notice that, we approximate a multi-ton with a double-ton, since a double-ton has a smaller mean of all the multi-tons, and hence becomes the worst case scenario for the probability of error analysis.

If we set the threshold at $(M-K)/2$, we can detect a zero-ton in each bin with error probability Pe_z given by

$$Pe_z = Pr(Z_i[1] > \frac{M-K}{2}) = Q\left(\sqrt{\frac{(M-K)^2}{4MN^\alpha}}\right) = Q\left(\sqrt{\frac{M^2(1-\eta)^2}{4MN^\alpha}}\right) \leq e^{-\frac{(1-\eta)^2 N^\mu - \alpha}{8}}$$

Similarly, a singe-ton can be be classified as a zero-ton or a multi-ton(double-ton) with error probability, Pe_s, given by

$$\begin{aligned} Pe_s &= Pr(Z_i[1] < \tfrac{M-K}{2}) + Pr(Z_i[1] > \tfrac{3M-K}{2}) \\ &= Q\left(\sqrt{\tfrac{M^2(1-\eta)^2}{4MN^\alpha}}\right) + Q\left(\sqrt{\tfrac{M^2(1-2\eta)^2}{4MN^\alpha}}\right) \leq 2e^{-\frac{(1-2\eta)^2 N^\mu - \alpha}{8}} \end{aligned}$$

Also, a multi-ton (approximated to double-ton) can be classified as a singleton with error probability Pe_m, given by

$$Pe_m = Pr(Z_i[1] < \frac{3M-K}{2}) = Q\left(\sqrt{\frac{M^2(1-2\eta)^2}{4MN^\alpha}}\right) \leq e^{-\frac{(1-2\eta)^2 N^\mu - \alpha}{8}}$$

So the overall probability of error in bin-detection, Pe_{bin}, including the bins across all stages and blocks, is given by

$$Pe_{bin} \leq \underbrace{N^{\gamma}}_{\#\text{ of blocks}} \underbrace{N^{1-\gamma-\alpha}}_{\#\text{ of bins}} \underbrace{\frac{1-\gamma}{\alpha}}_{\#\text{ of stages}} \underbrace{Pe_z + Pe_s + Pe_m}_{\text{error per bin}} = \frac{1-\gamma}{\alpha}\, N^{1-\alpha}\, e^{-\frac{(1-2\eta)^2 N^{\mu-\alpha}}{8}}$$
$$= e^{\log\frac{1-\gamma}{\alpha} + \log N^{1-\alpha} - \frac{(1-2\eta)^2 N^{\mu-\alpha}}{8}}$$

To make the error probability arbitrarily small, we impose the following condition on the exponent of Pe_{total}.

$$\log\frac{1-\gamma}{\alpha} + \log N^{1-\alpha} - \frac{(1-2\eta)^2 N^{\mu-\alpha}}{8} < 0$$
$$\implies \frac{(1-2\eta)^2 N^{\mu-\alpha}}{8} > \log\frac{1-\gamma}{\alpha} + \log N^{1-\alpha}$$
$$\implies \mu - \alpha > 8\left(\frac{\log(\log(\frac{1-\gamma}{\alpha}))}{\log N} + \frac{\log(\log(N)) + \log(1-\alpha)}{\log N}\right)$$

As $N \to \infty$, the right hand side approaches 0. Hence the condition reduces to

$$\alpha < \mu$$

B. Position Identification

Exact Matching:

$$\mathbf{Z_i} = \begin{bmatrix} \mathbf{s_1} & \cdots & \mathbf{s_p} & \cdots & \mathbf{s_A} \end{bmatrix} \times \begin{bmatrix} n_1 \\ \vdots \\ R_{XY}[p] \\ \vdots \\ n_j \\ \vdots \\ n_A \end{bmatrix}$$

where $n_j \sim \mathcal{N}(0, M),\ j \neq p$.

$$Z_i[1] = R_{XY}[p] + \sum_{j\neq p} n_j = R_{XY}[p] + n$$

where $n \sim \mathcal{N}(0, (N^{\alpha}-1)M)$. We can find the value of $R_{XY}[p]$ with probability of error $Pe_s \leq 2e^{-\frac{(1-2\eta)^2 N^{\mu-\alpha}}{8}}$.

Given that we know $R_{XY}[p]$,

$$\mathbf{Z_i} = R_{XY}[p]\ \mathbf{s_p} + \sum_{j\neq p} n_j \mathbf{s_j}$$

The estimated position p' is given by

$$p' = \arg\max_{l}\ \mathbf{s_l^T}\ \mathbf{Z_i}$$

Let us consider two cases:

Case 1: $p' = p$

$$c_1 = \mathbf{s_{p'}^T}\ \mathbf{Z_i} = R_{XY}[p]\ \mathbf{s_{p'}^T}\ \mathbf{s_p} + \sum_{j\neq p} n_j\ \mathbf{s_{p'}^T}\ \mathbf{s_j}$$
$$\leq B\ M + \sum_{j\neq p} 2n_j\ \sqrt{B\log(5N^{\alpha})}$$

Case 2: $p' \neq p$

$$\begin{aligned} c_2 &= \mathbf{s_{p'}^T}\ \mathbf{Z_i} \ = \ R_{XY}[p]\ \mathbf{s_{p'}^T}\ \mathbf{s_p} + \textstyle\sum_{j\neq p',p} n_j\ \mathbf{s_{p'}^T}\ \mathbf{s_j} \ + \ n_{p'}\ \mathbf{s_{p'}^T}\ \mathbf{s_p} \\ &\leq 2M\ \sqrt{B\log(5N^\alpha)} + \textstyle\sum_{j\neq p',p}\ 2n_j\ \sqrt{B\log(5N^\alpha)}\ +\ n_{p'}B \end{aligned}$$

$$\begin{aligned} c_1 - c_2 &= R_{XY}[p](B - \ \mathbf{s_{p'}^T}\ \mathbf{s_p})\ -\ n_{p'}(B - \ \mathbf{s_{p'}^T}\ \mathbf{s'_p})\ +\ \textstyle\sum_{j\neq p',p} n_j(\ \mathbf{s_p^T}\ \mathbf{s_j} - \ \mathbf{s_{p'}^T}\ \mathbf{s_j}) \\ &\leq R_{XY}[p](B-\mu_{max})\ -\ n_{p'}(B-\mu_{max})\ +\ \textstyle\sum_{j\neq p',p} n_j\mu_{max} \end{aligned}$$

Notice that c_1 and c_2 are both Gaussian random variables given by

$$\begin{aligned} c_1 &\sim \mathscr{N}(BM, 4N^\alpha MB\log(5N)) \\ c_2 &\sim \mathscr{N}(2M\sqrt{B\log(5N)}, M\ B^2 + 4N^\alpha MB\log(5N)) \\ c_1 - c_2 &\sim \mathscr{N}(R_{XY}[p](B-\mu_{max}), M((B-\mu_{max})^2 + (N^\alpha - 2)\mu_{max}^2) \end{aligned}$$

The probability that event $\mathscr{E}_2$ happens, given that $\mathscr{E}_1$ doesn't happen is given by

$$P(\mathscr{E}_2/\bar{\mathscr{E}_1}) = Pr(c_2 > c_1)$$

$$P(\mathscr{E}_2/\bar{\mathscr{E}_1}) \leq Q\left(\sqrt{\frac{R_{XY}^2[p](B-\mu_{max})^2}{M(B-\mu_{max})^2 + M(N^\alpha-2)\mu_{max}^2)}}\right)$$

where $\mu_{max} = 2\sqrt{B\log 5N^\alpha}$. If $B = O(\log 5N^\alpha)$, then the above equation reduces to

$$P(\mathscr{E}_2/\bar{\mathscr{E}_1}) \leq Q\left(\sqrt{\frac{M}{1+4\ (N^\alpha-2)}}\right)$$

If $\mu > \alpha$, the error vanishes.

Approximate Matching:

$$\mathbf{Z_i} = \begin{bmatrix}\mathbf{s_1} & \cdots & \mathbf{s_p} & \cdots & \mathbf{s_A}\end{bmatrix} \times \begin{bmatrix} n_1 \\ \vdots \\ R_{XY}[p] \\ \vdots \\ n_j \\ \vdots \\ n_A \end{bmatrix}$$

where $n_j \sim \mathscr{N}(0, M),\ j \neq p$.

$$\begin{aligned} Z_i[1] &= R_{XY}[p] + \textstyle\sum_{j\neq p} n_j \\ &= R_{XY}[p] + n \end{aligned}$$

where $n \sim \mathscr{N}(0, (N^\alpha - 1)M)$. Once we identify a Singleton with probability of error $Pe_s \leq 2e^{-\frac{(1-2\eta)^2 N^{\mu-\alpha}}{8}}$, we can fix the value of $R_{XY}[p] = M(1-\eta/2)$ since we are only interested in finding the positions and not the exact value of correlations.

Now that we know $R_{XY}[p]$,

$$\mathbf{Z_i} \ = \ R_{XY}[p]\ \mathbf{s_p} + \sum_{j\neq p} n_j \mathbf{s_j}$$

The estimated position p' is given by

$$p' = \arg\max_{l} \mathbf{s_l^T}\ \mathbf{Z_i}$$

Also, let p_1 be the position of the previously peeled node from this check-node. There can only be at most one peeled edge as we restrict our peeling process to a doubleton and do not consider other multi-tons.

Let us consider two cases:

Case 1: $p' = p$

$$\begin{aligned} c_1 &= \mathbf{s_{p'}^T}\ \mathbf{Z_i} \;=\; R_{XY}[p]\ \mathbf{s_{p'}^T}\ \mathbf{s_p} \;+\; \textstyle\sum_{j\neq p,p_1} n_j\ \mathbf{s_{p'}^T}\ \mathbf{s_j} \;\pm\; \frac{\eta M}{2}\ \mathbf{s_{p'}^T}\ \mathbf{s_{p_1}} \\ &\leq B\ (M-\eta M/2) + \textstyle\sum_{j\neq p} 2n_j\ \mu_{max} \pm \frac{\eta M}{2}\mu_{max} \end{aligned}$$

Case 2: $p' \neq p$

$$\begin{aligned} c_2 &= \mathbf{s_{p'}^T}\ \mathbf{Z_i} \;=\; R_{XY}[p]\ \mathbf{s_{p'}^T}\ \mathbf{s_p} + \textstyle\sum_{j\neq p',p,p_1} n_j\ \mathbf{s_{p'}^T}\ \mathbf{s_j} \;+\; n_{p'}\ \mathbf{s_{p'}^T}\ \mathbf{s_p} \;\pm\; \frac{\eta M}{2}\ \mathbf{s_{p'}^T}\ \mathbf{s_{p_1}} \\ &\leq (M-\eta M/2)\ \mu_{max} + \textstyle\sum_{j\neq p',p}\ 2n_j\ \mu_{max} \;+\; n_{p'}B \pm \frac{\eta M}{2}\mu_{max} \end{aligned}$$

$$\begin{aligned} c_1 - c_2 &= R_{XY}[p](B - \mathbf{s_{p'}^T}\ \mathbf{s_p}) \;-\; n_{p'}(B - \mathbf{s_{p'}^T}\ \mathbf{s'_p}) \;+\; \textstyle\sum_{j\neq p',p,p_1} n_j(\ \mathbf{s_p^T}\ \mathbf{s_j} - \mathbf{s_{p'}^T}\ \mathbf{s_j}) \pm\ \eta M\ \mathbf{s_{p'}^T}\ \mathbf{s_{p_1}} \\ &\leq (M-\eta M/2)(B-\mu_{max}) \;-\; n_{p'}(B-\mu_{max}) \;+\; \textstyle\sum_{j\neq p',p} n_j\mu_{max} \pm \eta M\mu_{max} \end{aligned}$$

Notice that c_1 and c_2 are both Gaussian random variables given by

$$\begin{aligned} c_1 &\sim \mathcal{N}(B\ (M-\eta M/2) \pm \tfrac{\eta M}{2}\mu_{max}\ ,\ 4N^{\alpha}MB\log(5N)) \\ c_2 &\sim \mathcal{N}(2M\sqrt{B\log(5N)}\ ,\ M\ B^2 + 4N^{\alpha}MB\log(5N)) \\ c_1 - c_2 &\sim \mathcal{N}((M-\eta M/2)(B-\mu_{max}) \pm \eta M\mu_{max}\ ,\ M((B-\mu_{max})^2 + (N^{\alpha}-2)\mu_{max}^2) \end{aligned}$$

The probability that event $\mathscr{E}_2$ happens, given that $\bar{\mathscr{E}}_1$ doesn't happen is given by

$$P(\mathscr{E}_2/\bar{\mathscr{E}}_1) = Pr(c_2 > c_1)$$

$$P(\mathscr{E}_2/\bar{\mathscr{E}}_1) \leq Q\left(\sqrt{\frac{((M-\eta M/2)(B-\mu_{max}) \pm \eta M\mu_{max})^2}{M(B-\mu_{max})^2 + M(N^{\alpha}-2)\mu_{max}^2)}}\right)$$

where $\mu_{max} = 2\sqrt{B\log 5N^{\alpha}}$. If $B = O(\log 5N^{\alpha})$, then the above equation reduces to

$$P(\mathscr{E}_2/\bar{\mathscr{E}}_1) \leq Q\left(\sqrt{\frac{M(1-3\eta/2)^2}{1+4\ (N^{\alpha}-2)}}\right)$$

If $\mu > \alpha$, the error vanishes.

VI. Complexity Analysis

In this section, we will analyze the number of samples used from the actual database to run the algorithm and also the computational complexity as a function of the system parameters.

A. Sample Complexity

As stated in the problem statement, we exploit the sparseness in IDFT computations to compute the correlations using less number of samples from the database.

In each branch of the FFAST algorithm, executed over a block of samples, we down-sample the $\tilde{N}$ samples by a factor of N^{α} to get $N^{1-\gamma-\alpha}$ samples. We also do this for each chosen shift and we get a total of $N^{1-\gamma-\alpha+\beta}$ samples per block per stage. We repeat this for $d = \frac{1-\gamma}{\alpha}$ such stages to get a total complexity of $dN^{1-\gamma-\alpha+\beta}$ samples per block. So, the total number of samples is given by

$$\text{Total \# of samples required (S)} \;=\; \underbrace{\frac{1-\gamma}{\alpha}}_{d \text{ stages}} \; \underbrace{N^{1-\gamma-\alpha+\beta}}_{\text{Samples per block per stage}} \; \underbrace{N^{\gamma}}_{\text{\# of blocks}} \;=\; \frac{1-\gamma}{\alpha}\, N^{1-\alpha+\beta}$$

B. Computational Complexity

Consider the operations involved in the computation of correlation of X and Y, R_{XY}, as shown below.

$$R_{XY} = \underbrace{IDFT}_{\text{I}} \; \{ \underbrace{DFT\{X\}}_{\text{II}} \times \; \underbrace{DFT\{Y\}}_{\text{III}} \}$$

In each block of samples from the database, the process of computing the correlation involves three sets of operations: two $\tilde{N}$-point DFT operations (II, III) sampled at $N^{1-\gamma-\alpha+\beta}$ points(based on sampling requirements of FFAST architecture) and one sparse-IDFFT operation (I). We assume that the database already contains sampled version of DFT $\{\ X\ \}$ stored and hence we exclude those computations from the complexity calculations. So, it is enough that we analyze the computations involved in operations (I, III) on each block to get the overall computational complexity.

1) Operation - III:
   Notice that the operation $DFT\{Y\}$ is just a one time computation and can be reused in each block since the query is same for all blocks. Since we only need to compute the sampled version of the DFT$\{Y\}$ in each branch, we can do some clever implementations to reduce the complexity. We can create aliasing on Y and then take a shorter length ($N^{1-\gamma-\alpha}$-point) DFT to obtain the sampled version of $\tilde{N}$-point DFT of Y.
   This method of computing sampled DFT$\{Y\}$ involves two operations - folding and adding (aliasing), and then a $N^{1-\gamma-\alpha}$-point DFT. The complexity involved in computing down-sampled DFT$\{Y\}$, C_{III}, is given by

$$C_{III} \;=\; \underbrace{\frac{1-\gamma}{\alpha}}_{d \text{ stages}} \; \Big(\; \underbrace{N^{\delta+\beta}}_{\text{\# of additions (aliasing)}} \; + \; \underbrace{N^{1-\gamma-\alpha+\beta}\, \log N^{1-\gamma-\alpha}}_{\text{Shorter FFTs}} \; \Big)$$

   Folding and adding, for each shift, involves adding N^{α} vectors of length $N^{1-\gamma-\alpha}$. We know that the length of the query is $L = N^{\delta}$, i.e., the number of non-zero elements in Y (zero-padded version of the query) is L and hence we only need to compute L additions instead of the length of the vector.
2) Operation - I:
   Taking advantage of the fact that the correlations are sparse (with some noise), we use a FFAST based architecture to reduce the complexity involved in computing IDFT. The number of computations involved in this process,C_I, is given by

$$C_I \;=\; \underbrace{N^{\gamma}}_{\text{\# of blocks}} \underbrace{\frac{1-\gamma}{\alpha}}_{d \text{ stages}} \underbrace{N^{1-\gamma-\alpha+\beta}\, \log N^{1-\gamma-\alpha}}_{\text{Shorter IFFTs /block/stage}} \;=\; \frac{1-\gamma}{\alpha}\, N^{1-\alpha+\beta}\, \log N^{1-\gamma-\alpha}$$

   Total number of computations, $C = C_{III} + C_I$, is given by

$$C \;=\; \frac{1-\gamma}{\alpha}\, (N^{\delta+\beta} \;+\; N^{1-\gamma-\alpha+\beta}\, \log N^{1-\gamma-\alpha} \;+\; N^{1-\alpha+\beta}\, \log N^{1-\gamma-\alpha})$$

REFERENCES

[1] A. Andoni, H. Hassanieh, P. Indyk, and D. Katabi, "Shift finding in sub-linear time," in *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 457–465, Society for Industrial and Applied Mathematics, 2013.

[2] A. Amir, M. Lewenstein, and E. Porat, "Faster algorithms for string matching with k mismatches," *Journal of Algorithms*, vol. 50, no. 2, pp. 257–275, 2004.

[3] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, 1977.

[4] W. I. Chang and T. G. Marr, "Approximate string matching and local similarity," in *Annual Symposium on Combinatorial Pattern Matching*, pp. 259–273, Springer, 1994.

[5] H. Hassanieh, F. Adib, D. Katabi, and P. Indyk, "Faster gps via the sparse fourier transform," in *mobicom*, pp. 353–364, ACM, 2012.

[6] S. Pawar and K. Ramchandran, "A robust r-ffast framework for computing a k-sparse n-length dft in o (k log n) sample complexity using sparse-graph codes," in *Proc. IEEE Int. Symp. Information Theory*, pp. 1852–1856, 2014.