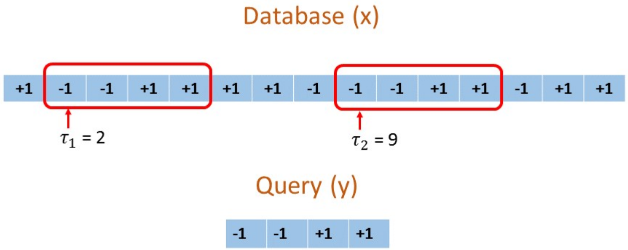
Sub-string/Pattern Matching in Sub-linear Time Using a Sparse Fourier Transform Approach

Krishna R. Narayanan

Department of Electrical and Computer Engineering
Texas A&M University

Problem Statement



- Database/String: $\underline{x} = [x[0], x[1], \cdots, x[N-1]]$ (length N)
- Query/Substring: $\underline{y} = [y[0], x[1], \cdots, x[M-1]]$ (length $M = N^{\mu}$)
- Determine all the L locations $\underline{\tau} = [\tau_1, \tau_2, \cdots \tau_L]$ with high probability where
  1. Exact Matching: $\underline{y}$ appears exactly in $\underline{x}$
     - $\underline{y} := \underline{x}[\tau : \tau + M - 1]$
  2. Approximate Matching: $\underline{y}$ is a noisy substring of $\underline{x}$
     - $\underline{y} := \underline{x}[\tau : \tau + M - 1] \odot \underline{b}$
     - $\underline{b}$ is a noise sequence with $d_H(\underline{y}, \underline{x}[\tau : \tau + M - 1]) \leq K$

Main Result

Theorem 1

Assume that a sketch of $\underline{x}$ of size $O(\frac{N}{M}\log N)$ can be precomputed and stored. Then for the exact pattern matching and approximate pattern matching (with $K = \eta M,\ 0 \leq \eta \leq 1/6$) problems, with the number of matches L scaling as $O(N^{\lambda})$, our algorithm has

- *a sketching function for $\underline{y}$ that computes $O(\frac{N}{M}\log N) = O(N^{1-\mu}\log N)$ samples*
- *a computational complexity of $O(\max(N^{1-\mu}\log^2 N, N^{\mu+\lambda}\log N))$*
- *a decoder that recovers all the L matching positions with a failure probability that approaches zero asymptotically*

Main Result

Theorem 1

Assume that a sketch of $\underline{x}$ of size $O(\frac{N}{M}\log N)$ can be precomputed and stored. Then for the exact pattern matching and approximate pattern matching (with $K = \eta M,\ 0 \leq \eta \leq 1/6$) problems, with the number of matches L scaling as $O(N^{\lambda})$, our algorithm has

- *a sketching function for $\underline{y}$ that computes $O(\frac{N}{M}\log N) = O(N^{1-\mu}\log N)$ samples*
- *a computational complexity of $O(\max(N^{1-\mu}\log^2 N, N^{\mu+\lambda}\log N))$*
- *a decoder that recovers all the L matching positions with a failure probability that approaches zero asymptotically*

Note

Particularly when $L = O(1)$ or $L < \frac{N}{M}$ (i.e. $\lambda < 1-\mu$) our algorithm has a sub-linear time complexity.

Some Prior Work

Exact Matching

- [?]: First occurrence of the match (only τ_1)
  - Average complexity - $O(N^{1-\mu} \log N)$ (sublinear)
  - Worst case complexity - $O(N \log N)$

Approximate Matching

- [?]: Generalization of [**?**]
  - Average complexity - $O(NK/M \log N)$ (sub-linear only when $K \ll M$)
- [?]: $O(N/M^{0.359})$ (sub-linear even when $K = O(M)$)
  - Combinatorial in nature

Sparse Fourier Transform Approach

- [?]: Faster GPS receiver
  - Exploited sparsity in Correlation function R_{XY}
- [?]: Robust Sparse Fourier Transform
  - Sparse Graph code Approach
  - Computational complexity : $O(N \log N)$

Motivation

- **Cross-correlation** ($\underline{r}$):

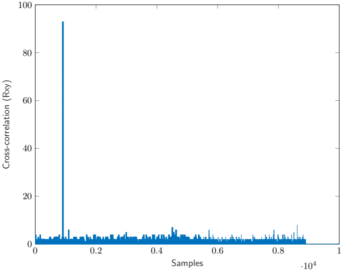
$$r[m] = (\underline{x} * \underline{y})[m] \triangleq \sum_{i=0}^{M-1} x[m+i]y[i], \quad 0 \leq m \leq N-1$$

- **Naive implementation**: $O(MN) = O(N^{1+\mu})$ (super-linear complexity)
- **Fourier Transform Approach**: $O(N \log N)$ complexity

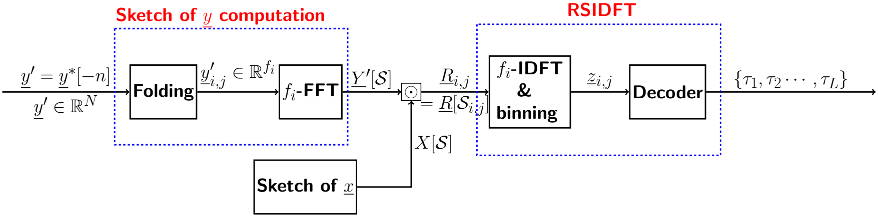
Key Observation

- $\underline{r}$ is Sparse with some noise.

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



Sparse Fourier Transform Approach

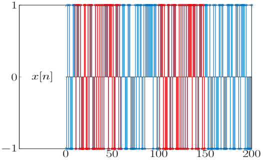


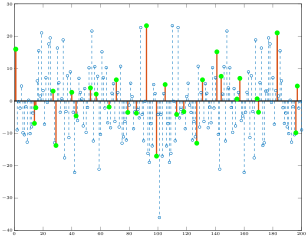
$$\boxed{\underline{r} = \underbrace{\mathcal{F}_N^{-1}}_{3} \; \{\underbrace{\mathcal{F}_N\{\underline{x}\}}_{1} \odot \; \underbrace{\mathcal{F}_N\{\underline{y}'\}}_{2}\}}$$

1. *Sketch of $\underline{x}$:* Assume $\underline{X}[l] = \mathcal{F}\{\underline{x}\}$ is precomputed at positions $l \in \mathcal{S}$.
2. *Sketch of $\underline{y}$:*
   - Compute $\underline{Y}'[l] = \mathcal{F}\{\underline{y}'\}$ for $l \in \mathcal{S}$.
   - Only M non-zero values in $\underline{y}'$ - Efficient computation (folding and adding)
3. *Sparse $\mathcal{F}^{-1}$*:
   - Robust Sparse Inverse Fourier Transform (RSIDFT)
   - Efficient Implementation- sublinear time and sampling complexity

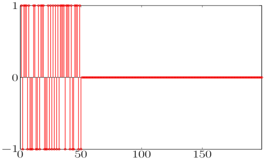
Example

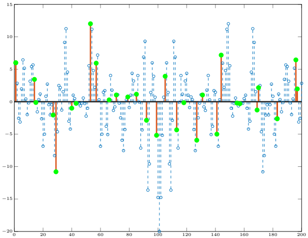
Fourier Domain (N-pt FFT)

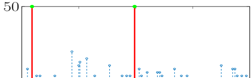




$y[n]$:

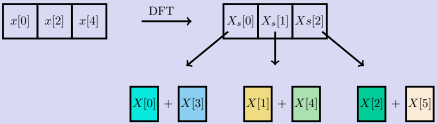




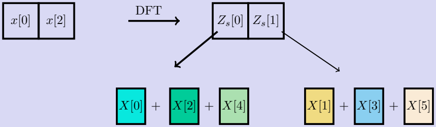


Aliasing and Sparse Graph Codes

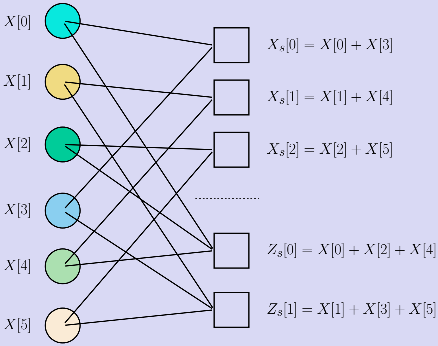
x_s: Sub-sampled by $f_1 = P_1 = 2$



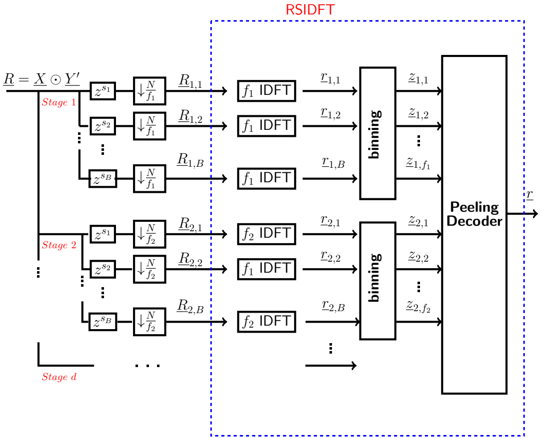
z_s: Sub-sampled by $f_2 = P_2 = 3$



Factor graph

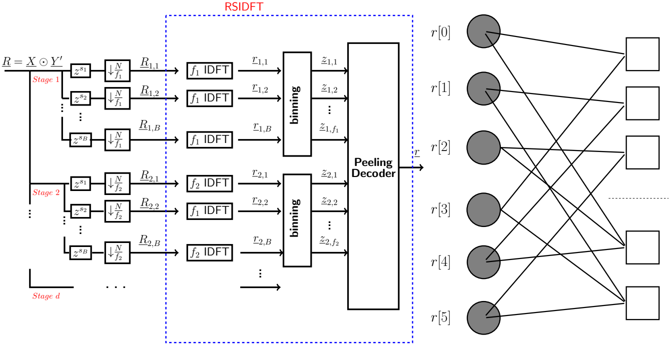


RSIDFT Framework



- Similar to FFAST
- Key modifications
  - Correlation peak is always positive
  - Take advantage in decoding algorithm - sub-linear time complexity

RSIDFT Framework



$$w_i^k := \left(e^{j\frac{2\pi s_i}{N}}\right)^k$$

$$\underline{z}_{1,1} = \begin{bmatrix} r_{1,1}[1] = r[0]w_1^0 + r[3]w_1^3 \\ r_{1,2}[1] = r[0]w_2^0 + r[3]w_2^3 \end{bmatrix}$$

$$\underline{z}_{1,2} = \begin{bmatrix} r_{1,1}[2] = r[1]w_1^1 + r[4]w_1^4 \\ r_{1,2}[2] = r[1]w_2^1 + r[4]w_2^4 \end{bmatrix}$$

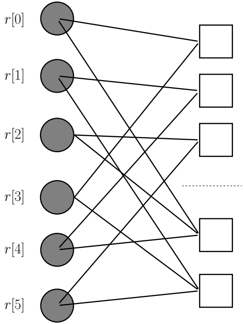
$$\underline{z}_{1,3} = \begin{bmatrix} r_{1,1}[3] = r[2]w_1^2 + r[5]w_1^5 \\ r_{1,2}[3] = r[2]w_2^2 + r[5]w_2^5 \end{bmatrix}$$

$$\underline{z}_{2,1} = \begin{bmatrix} r_{2,1}[1] = r[0]w_1^0 + r[2]w_1^2 + r[4]w_1^4 \\ r_{2,2}[1] = r[0]w_2^0 + r[2]w_2^2 + r[4]w_2^4 \end{bmatrix}$$

$$\underline{z}_{2,2} = \begin{bmatrix} r_{2,1}[2] = r[1]w_1^1 + r[3]w_1^3 + r[5]w_1^5 \\ r_{2,2}[2] = r[1]w_2^1 + r[3]w_2^3 + r[5]w_2^5 \end{bmatrix}$$

RSIDFT-Decoding (Peeling Decoder)

$$w_i^k := \left(e^{j\frac{2\pi s_i}{N}}\right)^k$$



$$\underline{z}_{1,1} = \begin{bmatrix} r_{1,1}[1] = r[0]w_1^0 + r[3]w_1^3 \\ r_{1,2}[1] = r[0]w_2^0 + r[3]w_2^3 \end{bmatrix}$$

$$\underline{z}_{1,2} = \begin{bmatrix} r_{1,1}[2] = r[1]w_1^1 + r[4]w_1^4 \\ r_{1,2}[2] = r[1]w_2^1 + r[4]w_2^4 \end{bmatrix}$$

$$\underline{z}_{1,3} = \begin{bmatrix} r_{1,1}[3] = r[2]w_1^2 + r[5]w_1^5 \\ r_{1,2}[3] = r[2]w_2^2 + r[5]w_2^5 \end{bmatrix}$$

$$\underline{z}_{2,1} = \begin{bmatrix} r_{2,1}[1] = r[0]w_1^0 + r[2]w_1^2 + r[4]w_1^4 \\ r_{2,2}[1] = r[0]w_2^0 + r[2]w_2^2 + r[4]w_2^4 \end{bmatrix}$$

$$\underline{z}_{2,2} = \begin{bmatrix} r_{2,1}[2] = r[1]w_1^1 + r[3]w_1^3 + r[5]w_1^5 \\ r_{2,2}[2] = r[1]w_2^1 + r[3]w_2^3 + r[5]w_2^5 \end{bmatrix}$$

- Observations: $\underline{z}_{i,k} = \left[r_{i,1}[k], r_{i,2}[k], \cdots, r_{i,B}[k]\right]^T$
- Decoding- 3 steps
  1. Bin Classification
  2. Position Identification
  3. Peeling Process

Decoder

Bin Classification

- Classify each check-node - Zero-ton / Single-ton / Multi-ton
- Threshold constraints on first observation $z_{i,k}[1] = z$
- Threshold varies with η
  - different for exact($\eta = 0$) and approximate matching

$$\widehat{\mathcal{H}}_{i,j} = \begin{cases} \mathcal{H}_z & z/M < \gamma_1 \\ \mathcal{H}_s & \gamma_1 < z/M < \gamma_2 \\ \mathcal{H}_d & \gamma_2 < z/M < \gamma_3 \\ \mathcal{H}_m & z/M > \gamma_3 \end{cases}$$

where $(\gamma_1, \gamma_2, \gamma_3) = (\frac{1-2\eta}{2}, \frac{3-4\eta}{2}, \frac{5-6\eta}{2})$

Decoder

Position Identification

- Observation:

$$\underline{z}_{i,k} = \mathbb{W}_{i,k} \times \begin{bmatrix} r[k+(0)f_i] \\ r[k+(1)f_i] \\ \vdots \\ r[k+(g_i-1)f_i] \end{bmatrix}$$

- Sensing Matrix:

$$\mathbb{W}_{i,k} = \left[\underline{w}^k, \underline{w}^{k+f_i}, \ldots, \underline{w}^{k+(g_i-1)f_i}\right] \text{ where } \underline{w}^k = \begin{bmatrix} e^{\frac{j2\pi k s_1}{N}} \\ e^{\frac{j2\pi k s_2}{N}} \\ \vdots \\ e^{\frac{j2\pi k s_B}{N}} \end{bmatrix}, \quad g_i = \frac{N}{f_i}$$

- Column that gives maximum correlation with the observation

$$\boxed{\hat{k} = \underset{k \in \{j + l g_i\}}{\arg\max} \quad \underline{z}_{i,j}^{\dagger} \underline{w}^k}$$

Decoder

Peeling Process:

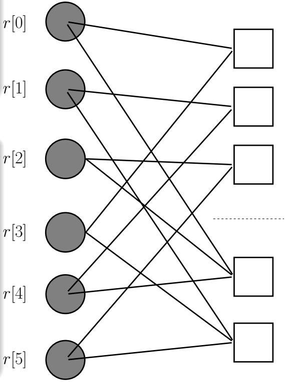
Exact Matching

- Remove a decoded variable node's contribution from all participating bin nodes

Approximate Matching

- Remove a decoded variable node's contribution only from neighboring single-tons and double-tons
- Avoid error propagation

$$w_i^k := \left(e^{j\frac{2\pi s_i}{N}}\right)^k$$



$$z_{1,1} = \begin{bmatrix} r_{1,1}[1] = r[0]w_1^0 + r[3]w_1^3 \\ r_{1,2}[1] = r[0]w_2^0 + r[3]w_2^3 \end{bmatrix}$$

$$z_{1,2} = \begin{bmatrix} r_{1,1}[2] = r[1]w_1^1 + r[4]w_1^4 \\ r_{1,2}[2] = r[1]w_2^1 + r[4]w_2^4 \end{bmatrix}$$

$$z_{1,3} = \begin{bmatrix} r_{1,1}[3] = r[2]w_1^2 + r[5]w_1^5 \\ r_{1,2}[3] = r[2]w_2^2 + r[5]w_2^5 \end{bmatrix}$$

$$z_{2,1} = \begin{bmatrix} r_{2,1}[1] = r[0]w_1^0 + r[2]w_1^2 + r[4]w_1^4 \\ r_{2,2}[1] = r[0]w_2^0 + r[2]w_2^2 + r[4]w_2^4 \end{bmatrix}$$

$$z_{2,2} = \begin{bmatrix} r_{2,1}[2] = r[1]w_1^1 + r[3]w_1^3 + r[5]w_1^5 \\ r_{2,2}[2] = r[1]w_2^1 + r[3]w_2^3 + r[5]w_2^5 \end{bmatrix}$$

Simulation Results

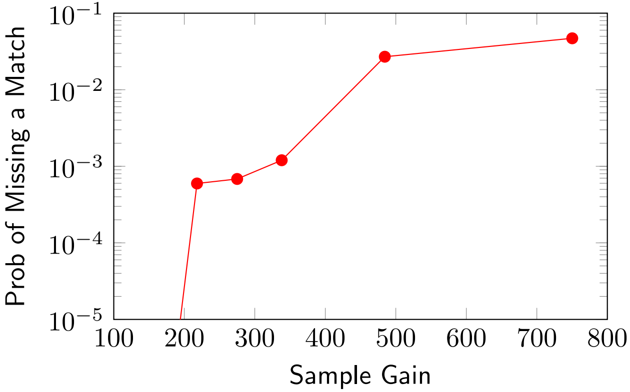


Figure: Plot of Probability of Missing a Match vs. Sample Gain for Exact Matching of a substring of length $M = 10^5$ from a equiprobable binary {+1,-1} sequence of length $N = 10^{12}$, divided into $G = 10^5$ blocks each of length $\tilde{N} = 10^7$. The substring was simulated to repeat in $L = 10^6$ locations uniformly at random.

Simulation Results

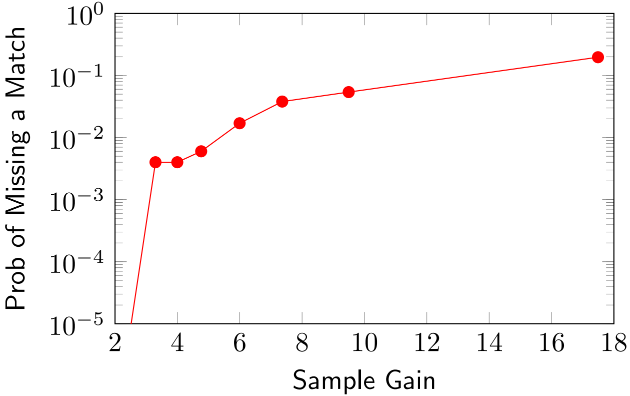


Figure: Plot of Probability of Missing a Match vs. Sample Gain for Exact Matching of a substring of length $M = 10^3$ from a equiprobable binary $\{+1,-1\}$ sequence of length $N = 10^{12}$, divided into $G = 10^6$ blocks each of length $\tilde{N} = 10^6$. The substring was simulated to repeat in $L = 10^6$ locations uniformly at random.

Questions?

Thank you!