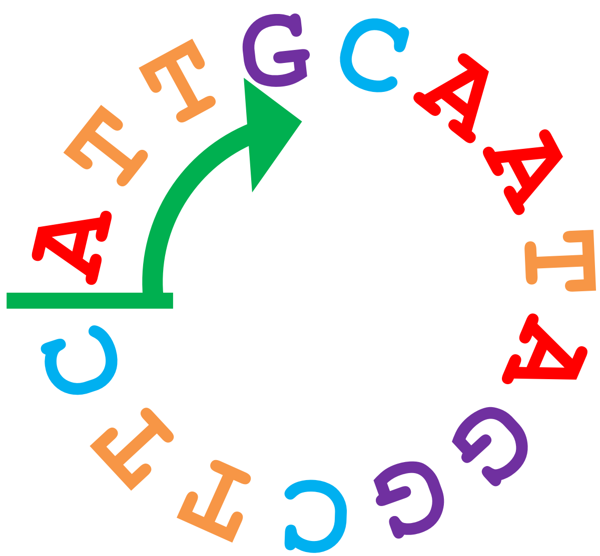
Finding Optimal Alignment and Consensus of Circular Strings

T. Lee*, J.C. Na, H. Park, K. Park, and J.S. Sim
*Seoul National University
June 23, 2010

Circular String

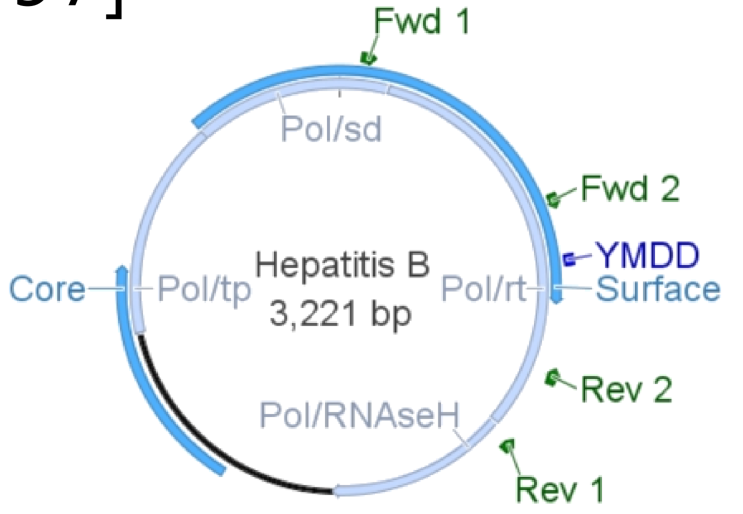
- The first (leftmost) symbol is wrapped around next to the last (rightmost) symbol
- A circular string of length n can be read as n different *linear* strings (called *instances*)



ATTGCAATAGGCTTC
TTGCAATAGGCTTCA
TGCAATAGGCTTCAT
GCAATAGGCTTCATT
CAATAGGCTTCATTG
• • • • • •

Circular Strings in Nature

- "*Bacterial* *and* *mitochondrial DNA* *is typically circular, … Consequently, tools for handling circular strings may someday be of use in those organisms."* [Gusfield 1997]



HBV's genome sequence

Consensus String Problem

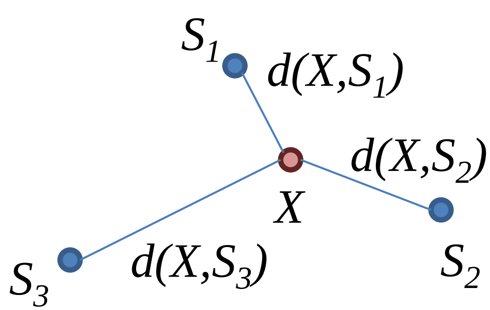
- Given a set $\mathbb{S}$ of strings, find a representative string (consensus) of $\mathbb{S}$
- Application: Motif recognition, …

$$
\mathbb{S}\left\{
\begin{array}{lcl}
s_1 & : & \texttt{a b c b a a c c a b e d a d a} \\
s_2 & : & \texttt{a a a b a b c c a b f d a c a} \\
s_3 & : & \texttt{a b a b d a c c a b e d a d a} \\
 & & \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet\ \bullet \\
s_m & : & \texttt{b b a b a a c a a b e d a d a}
\end{array}
\right.
$$

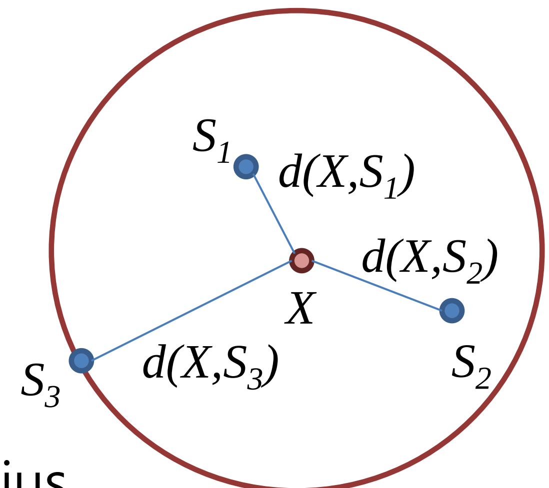
Consensus : a b a b a a c c a b e d a d a

Evaluation criteria

- Two major criteria of evaluating how well a string X represents a given set $\mathbb{S}$ are
  - Distance sum : The sum of distances from string X to the strings in $\mathbb{S}$
  - Radius : The longest distance from string X to the strings in $\mathbb{S}$



distance sum
$= d(X,S_1) + d(X,S_2) + d(X,S_3)$



radius
$= \max \{d(X,S_1), d(X,S_2), d(X,S_3)\}$

Evaluation criteria

- A good representative string should minimize distance sum and/or radius
  - Consensus minimizing distance sum (CS)
  - Consensus minimizing radius (CR)
  - Consensus minimizing both distance sum & radius (CSR)

Distance measures

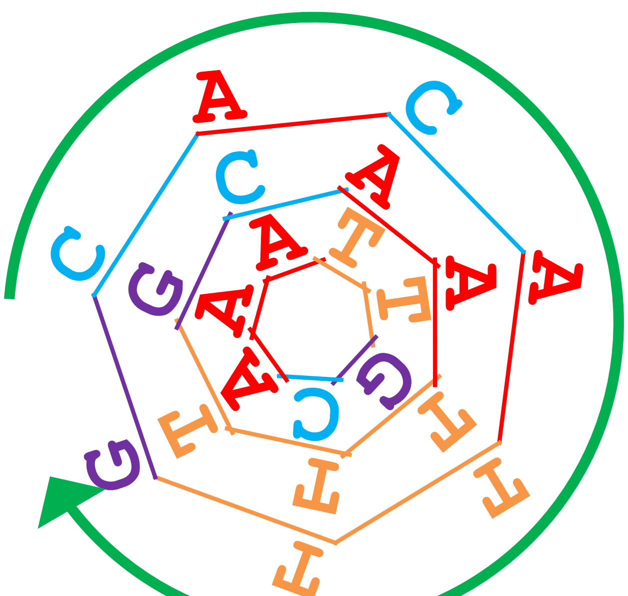
- Two well-known distances:
  - Hamming distance
    - Allowing only substitution
  - Edit distance
    - Allowing insertion and deletion as well as substitution

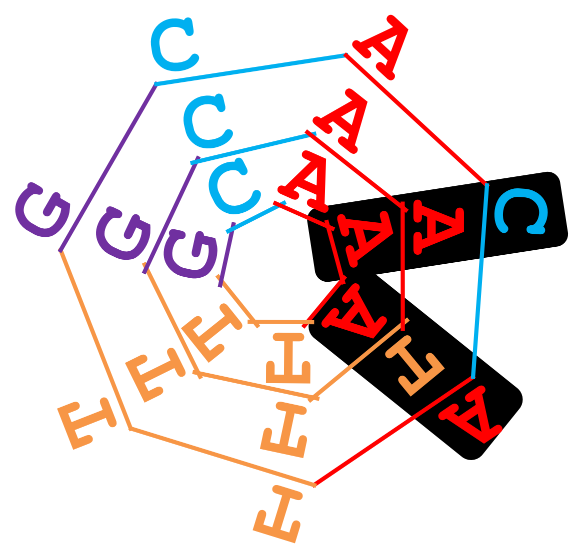
In this talk, we only consider the Hamming distance

Consensus of Circular Strings

- Find not only consensus but also the optimal alignment of given set of circular strings

$\mathbb{S}$={CACATTG, GCAATTT, AATTGCA}





Opt. alignment w/ ds=2 & r=1

Previous Works

- Consensus of linear strings
  - CS: easy to find (in linear time)
    - Take majority symbol in each aligned position
  - CR: hard in general
    - NP-Complete for arbitrary number of strings (even for strings drawn from binary alphabet)
    - Approximation / Fixed parameter solutions

Previous Works

- Fixed parameter solutions for CR and CSR
  - Algorithms for small constant m ($=|\mathbb{S}|$)
  - Gramm et al. proposed a direct combinatorial algorithm for 3 strings
  - Sze et al. showed a condition for the existence of string whose radius is less than or equal to r
  - Boucher et al. proposed an algorithm to find a string whose radius is $\leq$ r for 4 binary strings.
  - Amir et al. proposed a linear time algorithm for CSR of 3 strings

Previous Works

- Multiple alignment of circular strings [Mosig et al., Fernandes et al.]
  - The sum of pairs score
  - General purpose multiple sequence alignment techniques (e.g. clustalW)

Our Contribution

- Goal
  - Find the optimal consensus and alignment of given set of circular strings

- We present efficient algorithms for 3 or 4 circular strings of length n
  - $O(n^2 \log n)$ algorithm for CS of 3 strings
  - $O(n^2 \log n)$ algorithm for CR, CSR of 3 strings
  - $O(n^3 \log n)$ algorithm for CS of 4 strings

Problem Definitions

- Circular string S of length n
  - drawn from a constant-sized alphabet
  - The rth instance (linear string) $S(r)$
    - $S(r)=S[r]S[(r+1) \bmod n]\ldots S[(r-1) \bmod n]$
    - For circular string S=`abcde`,
      $S(0)$=`abcde`, $S(1)$=`bcdea`, $S(2)$=`cdeab`, …

Problem Definitions

- $\mathbb{S}=\{S_1,S_2,\ldots,S_m\}$ of m circular strings of length n
- Alignment $\rho=(\rho_1, \rho_2, \ldots ,\rho_m)$ (ρ_k : integer in $[0,n-1]$)
  - Juxtaposition of $S_1(\rho_1), S_2(\rho_2), \ldots , S_m(\rho_m)$
  - At most n^{m-1} (not n^m) distinct alignments
    - All $(\rho_1+k \bmod n, \rho_2+k \bmod n, \ldots ,\rho_m+k \bmod n)$ are equivalent for $k=0,\ldots,n-1$ under Hamming dist.

Problem Definitions

- Given alignment $\rho=(\rho_1, \rho_2, \ldots ,\rho_m)$ and a string X,
  - Distance sum of X for ρ : $E(\rho, X) = \Sigma_i\, d(X, S_i(\rho_i))$
  - Radius of X for ρ : $R(\rho, X) = \max_i d(X, S_i(\rho_i))$

- For each alignment ρ,
  - Min. distance sum for ρ : $E_{\min}(\rho) = \min_{X'} E(\rho, X')$
  - Min. radius for ρ : $R_{\min}(\rho) = \min_{X'} R(\rho, X')$

- For any alignment ρ' and string X'
  - Optimal distance sum : $E_{\mathrm{opt}} = \min_{\rho', X'} E(\rho', X')$
  - Optimal radius : $R_{\mathrm{opt}} = \min_{\rho', X'} R(\rho', X')$

※ $d(x,y)$: Hamming distance

Problem Definitions

- Optimal consensus problems

  Given a set $\mathbb{S}$ of m circular strings of length n, find an optimal alignment ρ and a string X (*if any*) that satisfy:

  - Problem CS $E(\rho, X) = E_{\mathrm{opt}}$
  - Problem CR $R(\rho, X) = R_{\mathrm{opt}}$
  - Problem CSR $E(\rho, X) = E_{\mathrm{opt}}$ and $R(\rho, X) = R_{\mathrm{opt}}$

In this talk, we will use notations CSk, CRk, and CSRk to represent Problems CS, CR, and CSR of k circular strings

Problem Definitions

- Bounded consensus problems

  Given a set $\mathbb{S}$ of m circular strings of length n,
  and two integer $s > 0$ and $r > 0$,
  find an ~~optimal~~ alignment ρ and a string X
  (*if any*) that satisfy:

  - Problem BS $E(\rho, X) \leq s$
  - Problem BR $R(\rho, X) \leq r$
  - Problem BSR $E(\rho, X) \leq s$ and $R(\rho, X) \leq r$

In this talk, we only present algorithms for CS and CSR, since they can be easily applied to CR, BS, BR, and BSR

Tools

- Algorithm for CSR of 3 linear strings [Amir et al. 09]

- Convolution method for counting matches/mismatches

CSR of 3 Linear Strings

- For three linear strings of equal length n, CSR can be found in O(n) time

s_1	a	a	b	b	b	b	a	a	a	a	a	a	b	c
s_2	a	a	b	a	a	a	b	b	b	a	a	b	c	b
s_3	a	a	b	a	a	a	a	a	a	b	b	c	a	a

CSR of 3 Linear Strings

- Every aligned position i is divided into five types and algorithm counts each Types k as c_k

	Type 0			Type 1			Type 2			Type 3		Type 4		
s_1	a	a	b	b	b	b	a	a	a	a	a	a	b	c
s_2	a	a	b	a	a	a	b	b	b	a	a	b	c	b
s_3	a	a	b	a	a	a	a	a	a	b	b	c	a	a

- Type 0: $s_1[i]=s_2[i]=s_3[i]$ (all matches)
- Type 1: $s_1[i]\neq s_2[i]=s_3[i]$ ($s_1[i]$ is the minority)
- Type 2: $s_2[i]\neq s_1[i]=s_3[i]$ ($s_2[i]$ is the minority)
- Type 3: $s_3[i]\neq s_1[i]=s_2[i]$ ($s_3[i]$ is the minority)
- Type 4: $s_1[i]\neq s_2[i]\neq s_3[i]$ (all mismatches)

$c_0=3$

$c_1=3$

$c_2=3$

$c_3=2$

$c_4=3$

CSR of 3 Linear Strings

- From $c_1 \sim c_4$, the minimum possible distance sum E_{min} and radius R_{min} can be determined:
  - $E_{min} = c_1 + c_2 + c_3 + 2c_4$
  - $R_{min} \geq \max(L_1, L_2)$ $(L_1 = \lceil \max_{i \neq j} d(s_i, s_j)/2 \rceil,\ L_2 = \lceil E_{min}/3 \rceil)$

	Type 0			Type 1			Type 2			Type 3		Type 4		
s_1	a	a	b	b	b	b	a	a	a	a	a	a	b	c
s_2	a	a	b	a	a	a	b	b	b	a	a	b	c	b
s_3	a	a	b	a	a	a	a	a	a	b	b	c	a	a
s	a	a	b	a	a	a	a	a	a	a	a	?	?	?

$L_1 = 5$, $L_2 = 5$

$E_{min} = 14$, $R_{min} = 5$

CSR of 3 Linear Strings

- Algorithm then finds (constructs) a consensus string s with E_{min} and radius R_{min} if it exists
  – By selecting Type 4 symbols wisely to balance the distances of strings from the consensus.

	Type 0			Type 1			Type 2			Type 3		Type 4		
s_1	a	a	b	b	b	b	a	a	a	a	a	a	b	c
s_2	a	a	b	a	a	a	b	b	b	a	a	b	c	b
s_3	a	a	b	a	a	a	a	a	a	b	b	c	a	a
s	a	a	b	a	a	a	a	a	a	a	a	b	c	c

$E_{min} = 14$, $R_{min} = 5$

Tools

- Lemma 1. For three linear strings of length n, if we are given counters c_1 to c_4,
  - E_{min}, R_{min} and the existence of a consensus w/ both E_{min} and R_{min} are determined in $O(1)$ time
  - We can construct such a consensus in $O(n)$ time
- Lemma 2. Given two circular strings X and Y of equal length n ($|\Sigma|$ is constant), the numbers of matches or mismatches in all n alignments can be computed in $O(n \log n)$ time, even with the presence of don't care or mismatch symbols.

Discrete Convolution

Given integer arrays t and p ($|\, t\,| = n$ and $|\, p\,| = m$),

$t \otimes p$ is the array of all inner products, where:

$$(t \otimes p)[i] = \sum_{j=0..m-1} t[i+j]\, p[j] \quad (i=0,\ \ldots\ ,\ n-m)\ .$$

i	*0*	*1*	*2*	***3***	*4*	*5*	*6*	*7*	*8*
t	1	0	0	1	0	1	0	0	0
				×	×	×	×		
p				1	1	1	0		
				$\sum$					
$\boldsymbol{t \otimes p}$	1	1	1	**2**					

Discrete Convolution

Given integer arrays t and p ($|t| = n$ and $|p| = m$),

$t \otimes p$ is the array of all inner products, where:

$$(t \otimes p)[i] = \sum_{j=0..m-1} t[i+j]\, p[j] \quad (i=0, \ldots, n-m).$$

i	0	1	2	3	4	5	6	7	8
t									
p									
$t \otimes p$	1	1	1	2	1				

Can be computed in $O(n \log m)$ time by using Fast Fourier Transform (FFT)

Counting Matches/Mismatches

- Using discrete convolution, we can count matches/mismatches in every alignments of two circular strings efficiently
  - in $O(n \log n)$ time ($|\Sigma|$ is constant)

Alignment	$(0, 0)$	$(0, 1)$	$(0, 2)$	$(0, 3)$
X	a a b a	a a b a	a a b a	a a b a
Y	a b b a	b b a a	b a a b	a a b b
Matches	3	1	1	3
Mismatches	1	3	3	1

Counting Matches/Mismatches

- Bit mask $B_{x,\sigma}$ and inverse bit mask $B'_{x,\sigma}$
  - For given string x and each symbol $\sigma \in \Sigma$,

$$B_{x,\sigma}[i] = \begin{cases} 1, & \text{if } x[i] = \sigma \\ 0, & \text{if } x[i] \neq \sigma \end{cases} \quad \text{and} \quad B'_{x,\sigma}[i] = \begin{cases} 0, & \text{if } x[i] = \sigma \\ 1, & \text{if } x[i] \neq \sigma \end{cases}$$

x	a	a	b	a
$B_{x,\mathtt{a}}$	1	1	0	1
$B_{x,\mathtt{b}}$	0	0	1	0

Counting Matches/Mismatches

i	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
$X^2(0)$	a	a	b	a	a	a	b	a
$Y(0)$	a	b	b	a				
$B_{X^2(0),\mathtt{a}}$	**1**	1	0	**1**	1	1	0	1
$B_{Y(0),\mathtt{a}}$	**1**	0	0	**1**				
$B_{X^2(0),\mathtt{a}} \otimes B_{Y(0),\mathtt{a}}$	**2**							
$B_{X^2(0),\mathtt{b}}$	0	0	**1**	0	0	0	1	0
$B_{Y(0),\mathtt{b}}$	0	1	**1**	0				
$B_{X^2(0),\mathtt{b}} \otimes B_{Y(0),\mathtt{b}}$	**1**							
$\sum_\sigma B_{X^2(0),\sigma} \otimes B_{Y(0),\sigma}$	**3**							

$X(0)X(0)$

#. matches in alignment (0,0)

Counting Matches/Mismatches

i	0	1	2	3	4	5	6	7
$X^2(0)$	`a`	`a`	`b`	`a`	`a`	`a`	`b`	`a`
$Y(0)$		`a`	`b`	`b`	`a`			
$B_{X^2(0),\mathtt{a}}$	1	$\mathbf{1}$	0	1	$\mathbf{1}$	1	0	1
$B_{Y(0),\mathtt{a}}$		$\mathbf{1}$	0	0	$\mathbf{1}$			
$B_{X^2(0),\mathtt{a}} \otimes B_{Y(0),\mathtt{a}}$	$\mathbf{2}$	$\mathbf{2}$						
$B_{X^2(0),\mathtt{b}}$	0	0	$\mathbf{1}$	0	0	0	1	0
$B_{Y(0),\mathtt{b}}$		0	$\mathbf{1}$	1	0			
$B_{X^2(0),\mathtt{b}} \otimes B_{Y(0),\mathtt{b}}$	$\mathbf{1}$	$\mathbf{1}$						
$\sum_\sigma B_{X^2(0),\sigma} \otimes B_{Y(0),\sigma}$	$\mathbf{3}$	$\mathbf{3}$						

#. matches in alignment (0,3)

Counting Matches/Mismatches

i	0	1	2	3	4	5	6	7
$X^2(0)$	a	a	b	a	a	a	b	a
$Y(0)$			a	b	b	a		
$B_{X^2(0),\mathtt{a}}$	1	1	0	1	1	**1**	0	1
$B_{Y(0),\mathtt{a}}$			1	0	0	**1**		
$B_{X^2(0),\mathtt{a}} \otimes B_{Y(0),\mathtt{a}}$	**2**	**2**	**1**					
$B_{X^2(0),\mathtt{b}}$	0	0	1	0	0	0	1	0
$B_{Y(0),\mathtt{b}}$			0	1	1	0		
$B_{X^2(0),\mathtt{b}} \otimes B_{Y(0),\mathtt{b}}$	**1**	**1**	**0**					
$\sum_\sigma B_{X^2(0),\sigma} \otimes B_{Y(0),\sigma}$	**3**	**3**	**1**					

#. matches in alignment (0,2)

Counting Matches/Mismatches

i	0	1	2	3	4	5	6	7
$X^2(0)$	a	a	b	a	a	a	b	a
$Y(0)$				a	b	b	a	
$B_{X^2(0),\mathtt{a}}$	1	1	0	1	1	1	0	1
$B_{Y(0),\mathtt{a}}$				1	0	0	1	
$B_{X^2(0),\mathtt{a}} \otimes B_{Y(0),\mathtt{a}}$	**2**	**2**	1	**1**				
$B_{X^2(0),\mathtt{b}}$	0	0	1	0	0	0	1	0
$B_{Y(0),\mathtt{b}}$				0	1	1	0	
$B_{X^2(0),\mathtt{b}} \otimes B_{Y(0),\mathtt{b}}$	**1**	**1**	0	**0**				
$\sum_\sigma B_{X^2(0),\sigma} \otimes B_{Y(0),\sigma}$	**3**	**3**	1	**1**				

#. matches in alignment (0,1)

Counting Matches/Mismatches

- Counting matches with mismatch '#'
  - $B_{x,\sigma}[i]=0$ if $x[i]=\#$ and $B_{y,\sigma}[i]=0$ if $y[i]=\#$ for $\forall\sigma$
- Counting mismatches with don't care '$'
  - $B_{x,\sigma}[i]=0$ if $x[i]=\$$ and $B'_{y,\sigma}[i]=0$ if $y[i]=\$$ for $\forall\sigma$

Tools

- Lemma 1. For three linear strings of length n, if we are given counters c_1 to c_4,
  - E_{min}, R_{min} and the existence of a consensus w/ both E_{min} and R_{min} are determined in $O(1)$ time
  - We can construct such a consensus in $O(n)$ time
- Lemma 2. Given two circular strings X and Y of equal length n ($|\Sigma|$ is constant), the numbers of matches or mismatches in all n alignments can be computed in $O(n \log n)$ time, even with the presence of don't care or mismatch symbols.

Algorithms

- A naïve algorithm
  - For CS3 or CSR3, we apply $O(n)$-time algorithm for linear strings in each of all $O(n^2)$ alignments → $O(n^3)$ time
  - Similarly, CS4 can be found in $O(n^4)$ time

- Our algorithms achieve $(n/\log n)$-speedup
  - $O(n^2 \log n)$ time for CS3 and CSR3
  - $O(n^3 \log n)$ time for CS4

Algorithm for Problem CS3

- Computation of $E_{\mathrm{min}}(0, \delta, \gamma)$
  - We first superpose $S_1(0)$ and $S_2(\delta)$ into Z_δ

    $$Z_\delta[i] = \langle S_1(0)[i], S_2(\delta)[i] \rangle$$

  - Generate bit masks of Z_δ and $S_3(\gamma)$

    $$B_{Z_\delta,\sigma}[i] = \begin{cases} 1, & \text{if } Z_\delta[i] = \langle \sigma, * \rangle \text{ or } \langle *, \sigma \rangle; \\ 0, & \text{otherwise} \end{cases}$$

  - Compute inner products of $B_{Z_\delta,\sigma}$ and $B'_{S_3(\gamma),\sigma}$
  - Key observation: sum of all inner products over σ equals to $E_{\mathrm{min}}(0, \delta, \gamma)$

Algorithm for Problem CS3

- For each σ, $B_{(Z_\delta)^2,\sigma} \otimes B'_{S_3(0),\sigma}$ produces the array of n inner products $B_{(Z_\delta)^2,\sigma}$ and $B'_{S_3(\gamma),\sigma}$ $(\gamma=0,..,n-1)$
- Thus, we can compute n distance sums $E_{\min}(0, \delta, 0) \sim E_{\min}(0, \delta, n-1)$ in $O(n \log n)$ time by FFT

Row 0	$E_{\min}(0, 0, 0)$	$E_{\min}(0, 0, 1)$	* * *	$E_{\min}(0, 0, n-1)$
***			* * *	
Row δ	$E_{\min}(0, \delta, 0)$	$E_{\min}(0, \delta, 1)$	* * *	$E_{\min}(0, \delta, n-1)$
***			* * *	
Row n-1	$E_{\min}(0, n-1, 0)$	$E_{\min}(0, n-1, 1)$	* * *	$E_{\min}(0, n-1, n-1)$

$E_{\min}$'s for all n^2 alignment ➔ $O(n^2 \log n)$ time

Algorithm for Problem CS3

- Workflow
  - Compute $E_{\min}$ for all alignments in $O(n^2 \log n)$ time by using FFT
  - Among all n^2 alignments, find the best alignment ρ with the smallest $E_{\min}$ (=E_{opt}) and construct a consensus in ρ

- Theorem 1. Problem CS3 can be solved in $O(n^2 \log n)$ time and $O(n)$ space

Algorithm for Problem CSR3

- Algorithm for CS3 directly computes $E_{\min}$ for each alignment, but it does not give any information of counters c_1 to c_4 which are essential to apply Lemma 1

- How can we efficiently count c_1 to c_4 for each alignment?
  - Convolution and system of linear equations

Algorithm for Problem CSR3

- For three linear strings, we obtain :

$$\begin{cases} c_1 + c_2 + c_3 + c_4 = n - c_0 \, , \\ c_1 + c_2 + c_4 = d(s_1, s_2) \, , \\ c_1 + c_3 + c_4 = d(s_1, s_3) \, , \\ c_2 + c_3 + c_4 = d(s_2, s_3) \; . \end{cases}$$

	Type 0			Type 1			Type 2			Type 3		Type 4		
s_1	a	a	b	b	b	b	a	a	a	a	a	a	b	c
s_2	a	a	b	a	a	a	b	b	b	a	a	b	c	b
s_3	a	a	b	a	a	a	a	a	a	b	b	c	a	a
$d(s_1, s_2)$	=			3	+		3	+				3		

Algorithm for Problem CSR3

- Once c_0 and the pairwise distances have been computed, we can compute c_1 to c_4

$$c_1 = n - d(s_2, s_3) - c_0 ,$$

$$c_2 = n - d(s_1, s_3) - c_0 ,$$

$$c_3 = n - d(s_1, s_2) - c_0 ,$$

$$c_4 = d(s_1, s_2) + d(s_1, s_3) + d(s_2, s_3) - 2c_0 - 2n$$

Algorithm for Problem CSR3

- Algorithm efficiently computes
  - $d(S_i(0), S_j(\delta))$'s for all i,j,δ in $O(n \log n)$ time using FFT (by Lemma 2)
  - Counters c_0's for all alignments in $O(n^2 \log n)$ time using FFT and superposition (similar to Algorithm CS3)

- Theorem 2. Problem CSR3 can be solved in $O(n^2 \log n)$ time and $O(n)$ space

Algorithm for Problem CS4

- Given four linear strings (instances),
  - We define 5 types and 15 counters wrt. combinations of distinct symbols
  - $E_{\min} = \Sigma_i b_i + 2(\Sigma_i c_i + \Sigma_i d_i) + 3e$

Type	A	B				C			D						E
s_1	♣	♣	♦	♦	♦	♣	♦	♦	♣	♣	♣	♦	♦	♦	♣
s_2	♣	♦	♣	♦	♦	♦	♣	♦	♣	♦	♦	♣	♣	♥	♦
s_3	♣	♦	♦	♣	♦	♦	♦	♣	♦	♣	♥	♣	♥	♣	♥
s_4	♣	♦	♦	♦	♣	♣	♣	♣	♥	♥	♣	♥	♣	♣	♠
Counter	a	b_1	b_2	b_3	b_4	c_1	c_2	c_3	d_1	d_2	d_3	d_4	d_5	d_6	e

Algorithm for Problem CS4

- System of linear equations

$$
\left\{
\begin{array}{lll}
a + b_3 + b_4 + c_3 + d_1 = M_{12} \,, & a + b_4 = M_{123} \,, & a = M_{1234} \,, \\
a + b_2 + b_4 + c_2 + d_2 = M_{13} \,, & a + b_3 = M_{124} \,, & \\
a + b_2 + b_3 + c_1 + d_3 = M_{14} \,, & a + b_2 = M_{134} \,, & \\
a + b_1 + b_4 + c_1 + d_4 = M_{23} \,, & a + b_1 = M_{234} \,, & \\
a + b_1 + b_3 + c_2 + d_5 = M_{24} \,, & & \\
a + b_1 + b_2 + c_3 + d_6 = M_{34} \,, & & \\
c_3 = d(s_{12}, s_{34}) \,, & c_2 = d(s_{13}, s_{24}) \,, & c_1 = d(s_{14}, s_{23}) \,, \\
a + \Sigma b_i + \Sigma c_i + \Sigma d_i + e = n \;. & &
\end{array}
\right.
$$

Algorithm for Problem CS4

- M_{ij}, M_{ijk}, M_{ijkl}: Numbers of pair/triple/quadruple-wise matches ($\{i,j,k,l\}=\{1,2,3,4\}$)
- Pair string s_{ij} of s_i and s_j
  - $s_{ij}[p] = s_i[p]$ if $s_i[p] = s_j[p]$; otherwise, $s_{ij}[p] = \$$
- Distance between s_{ij} and s_{kl}, $d(s_{ij}, s_{kl})$

s_1	a	b	a	a	a	a	a	b	b
s_2	a	c	a	a	b	b	b	b	b
s_3	b	a	a	a	a	c	b	a	c
s_4	b	a	b	a	b	d	a	b	c
s_{12}	a	$	a	a	$	$	$	b	b
s_{34}	b	a	$	a	$	$	$	$	c

$M_{12} = 5$

Algorithm for Problem CS4

- M_{ij}, M_{ijk}, M_{ijkl}: Numbers of pair/triple/quadruple-wise matches ($\{i,j,k,l\}=\{1,2,3,4\}$)
- Pair string s_{ij} of s_i and s_j
  - $s_{ij}[p] = s_i[p]$ if $s_i[p] = s_j[p]$; otherwise, $s_{ij}[p] = \$$
- Distance between s_{ij} and s_{kl}, $d(s_{ij}, s_{kl})$

s_1	a	b	a	a	a	a	a	b	b
s_2	a	c	a	a	b	b	b	b	b
s_3	b	a	a	a	a	c	b	a	c
s_4	b	a	b	a	b	d	a	b	c
s_{12}	a	$	a	a	$	$	$	b	b
s_{34}	b	a	$	a	$	$	$	$	c

$M_{12} = 5$

$M_{124} = 2$

Algorithm for Problem CS4

- M_{ij}, M_{ijk}, M_{ijkl}: Numbers of pair/triple/quadruple-wise matches ($\{i,j,k,l\}=\{1,2,3,4\}$)
- Pair string s_{ij} of s_i and s_j
  - $s_{ij}[p] = s_i[p]$ if $s_i[p] = s_j[p]$; otherwise, $s_{ij}[p] = \$$
- Distance between s_{ij} and s_{kl}, $d(s_{ij}, s_{kl})$

s_1	a	b	a	a	a	a	a	b	b
s_2	a	c	a	a	b	b	b	b	b
s_3	b	a	a	a	a	c	b	a	c
s_4	b	a	b	a	b	d	a	b	c
s_{12}	a	$	a	a	$	$	$	b	b
s_{34}	b	a	$	a	$	$	$	$	c

$M_{12} = 5$

$M_{124} = 2$

$M_{1234} = 1$

Algorithm for Problem CS4

- M_{ij}, M_{ijk}, M_{ijkl}: Numbers of pair/triple/quadruple-wise matches ($\{i,j,k,l\}=\{1,2,3,4\}$)
- Pair string s_{ij} of s_i and s_j
  - $s_{ij}[p] = s_i[p]$ if $s_i[p] = s_j[p]$; otherwise, $s_{ij}[p] = \$$
- Distance between s_{ij} and s_{kl}, $d(s_{ij}, s_{kl})$

s_1	a	b	a	a	a	a	a	b	b
s_2	a	c	a	a	b	b	b	b	b
s_3	b	a	a	a	a	c	b	a	c
s_4	b	a	b	a	b	d	a	b	c
s_{12}	a	$	a	a	$	$	$	b	b
s_{34}	b	a	$	a	$	$	$	$	c

$M_{12} = 5$

$M_{124} = 2$

$M_{1234} = 1$

$d(s_{12}, s_{34}) = 2$

Algorithm for Problem CS4

- By using FFT, we can efficiently compute, for all n^3 alignments,
  - all M_{ij}, M_{ijk}, M_{ijkl} in $O(n^2)$, $O(n^3)$, and $O(n^3 \log n)$ time
  - all $d(s_{ij}, s_{kl})$ in $O(n^3 \log n)$ time

- Theorem 3. Problem CS4 can be solved in $O(n^3 \log n)$ time and $O(n)$ space

Conclusion

- We proposed efficient algorithms to solve the optimal/bounded consensus problems for 3 or 4 circular strings

- Future works
  - Algorithms for CSR4 or more strings
  - Algorithms with edit distance measure

Thank you!