Parallel and Distributed Compressed Indexes

Luís M. S. Russo  Gonzalo Navarro  Arlindo L. Oliveira

CITI, Departamento de Informática, Faculdade de Ciências e Tecnologia, FCT, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal.
lsr@di.fct.unl.pt

Dept. of Computer Science, University of Chile
gnavarro@dcc.uchile.cl

INESC-ID/IST
aml@kdbio.inesc-id.pt

21st Annual Symposium on Combinatorial Pattern Matching
Outline

1. Motivation
   - The Problem We Studied
   - Previous Work

2. Parallel Compressed Indexes
   - Generalized Branching
   - Pattern Matching

3. Distributed Compressed Indexes
   - Distributed Compressed Suffix Arrays
   - Distributed Fully-Compressed Suffix Trees

4. Conclusions
   - Summary
Suffix trees are important for several string problems:

- pattern matching
- longest common substring
- super maximal repeats
- bioinformatics applications
- etc
Example (Suffix Tree for *abbbab*)

- Suffix Tree representation of the string *abbbab*.
- The tree nodes represent suffixes of the string.
- The numbers at the bottom of the tree nodes correspond to the positions in the string.
- The tree structure shows the relationships between suffixes, with edges pointing to the next character in each suffix.

For more details, refer to the presentation slides.
Compressed Indexes

Problem (Indexes need too much space)

*Pointer based representations require $O(n \log n)$ bits.*

Hence we use compressed indexes that use only $n \log \sigma + o(n \log \sigma)$ bits.

- **Succinct structures**, based on **RANK** and **SELECT**.
- **Data compression**, that represent $T$ in $O(nH_k)$ bits.

Examples

FM-index, Compressed Suffix Arrays, LZ-index, etc.

We use a compressed index that supports $\psi$ and LF. For example the Alphabet-Friendly FM-Index.
Problem (Indexes are sequential)

*How to adapt compress indexes to shared-memory parallel machines? How to distribute compressed indexes across several machines?*
Various data layouts have been considered for distributing classical suffix trees and arrays [1, 2, 3], with optimal speedups.

Mäkinen et al. [2] achieved optimal speedups for a batch of queries with CSAs.

We proposed near-optimal speed ups for single queries over CSAs and FCSTs.
Intermediate search status is either a node/point in the suffix tree or an interval of leaves.

**Example**

Interval \([3, 6]\) represents node \(b\).
Observation

*We use* generalized branching.

Example

$O((\log \log n)^2 \log \sigma_n)$ time with the FCST.
Observation

We use generalized branching.

Example

\[ a \]
\[ b \]
\[ b \]
\[ a \]
\[ b \]
\[ b \]
\[ b \]
\[ a \]
\[ b \]
\[ a \]
\[ b \]
\[ b \]
\[ b \]
\[ a \]
\[ b \]
\[ a \]

\[ O((\log \log n)^2 \log_\sigma n) \]

time with the FCST.

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira
Observation

*We use* generalized branching.

Example

\[ \begin{array}{cccccc}
0 & a & b & b & b & a \\
1 & b & b & b & b & b \\
2 & a & a & a & a & a \\
3 & b & b & b & b & b \\
\end{array} \]

\[ O((\log \log n)^2 \log_\sigma n) \]

time with the FCST.
Observation

*We use* generalized branching.

Example

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\$ & $ & $ & $ & $ & $ & $ \\
a & b & b & a & a & a & b \\
b & b & a & b & b & b & b \\
\end{array}
\]

\[O((\log \log n)^2 \log_\sigma n)\]
time with the FCST.
Observation

We use generalized branching.

Example

$O((\log \log n)^2 \log_\sigma n)$ time with the FCST.
Problem

How to search for a pattern $P = abbbbabbb$ in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O\left(\frac{m}{p} + \log n \log \log n (\log p + \log \log n \log \log p)\right)$. 

![Pattern Matching Tree](image-url)
Problem

How to search for a pattern $P = abbbabb$ in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O\left(\frac{m}{p} + \log n \log \log n (\log p + \log \log n \log \log p)\right)$. 

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira
Problem

How to search for a pattern $P = abbbbabbb$ in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(m/p + \log n \log \log n (\log p + \log \log n \log \log p))$. 

```
abbb

ab
  a
  b
bb
  b
  b
bb
  b
  b
```
**Problem**

*How to search for a pattern $P = abbbabb$ in the index, using $p = 4$ processors in parallel?*

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(\frac{m}{p} + \log n \log \log n (\log p + \log \log n \log \log p))$. 

```
    NULL
   /   \
abbb  NULL
 /   \   /
ab    bb  ba
 /  \
 a   b  b  a
```

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira

Parallel and Distributed Compressed Indexes
Problem

How to search for a pattern \( P = abbbabb \) in the index, using \( p = 4 \) processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.

\[ O\left(\frac{m}{p} + \log n \log \log n (\log p + \log \log n \log \log p)\right) \]
Pattern Matching

Problem

How to search for a pattern \( P = abbbabbb \) in the index, using \( p = 4 \) processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- \( O(\frac{m}{p} + \log n \log \log n (\log p + \log \log n \log \log p)) \).

![Pattern Matching Diagram]

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira
Parallel and Distributed Compressed Indexes
Problem

*How to search for a pattern* $P = abbbabbb$ *in the index, using* $p = 4$ *processors in parallel?*

- Split the pattern and search in parallel.
- Merge the resulting points.

$O\left(\frac{m}{p} + \log n \log \log n (\log p + \log \log n \log \log p)\right)$. 

---

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira
**Problem**

*How to search for a pattern $P = abbbabb$ in the index, using $p = 4$ processors in parallel?*

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O\left(\frac{m}{p} + \log n \log \log n (\log p + \log \log n \log \log p)\right)$. 

---

**Diagram**

```
           NULL
          /   \        /   \    
        abbb   NULL  ab   ba  
       /   \   /   \   /   \   /   \ 
      ab    bb  ba   bb   a   b   b  
      / \   / \  / \  / \  / \  / \  
     a  b  b  b  b  a  b  b
```
Matching statistics

Problem

*How to compute the matching statistics of a pattern* $P$, *in parallel?*

Example

$T = abbbab$

$P = abbbbabbb$

4, 3, 5, 4, 3, 3, 2, 1
Problem

*How to compute the matching statistics of a pattern* $P$, *in parallel?*

Example

$T = abbbab$

$P = abbbbabb$

$4, 3, 5, 4, 3, 3, 2, 1$
Problem

*How to compute the matching statistics of a pattern* $P$, *in parallel?*

Example

$T = a b b b a b$
$P = a b b b b a b b$
$4, 3, 5, 4, 3, 3, 2, 1$
Problem

How to compute the matching statistics of a pattern $P$, in parallel?

Example

$T = abbbab$

$P = abbbabbb$

4, 3, 5, 4, 3, 3, 2, 1
Problem

How to compute the matching statistics of a pattern $P$, in parallel?

Example

$T = abbbab$
$P = abbbbab$
$4, 3, 5, 4, 3, 3, 4, 3, 2, 1$
Matching statistics

**Problem**

How to compute the matching statistics of a pattern $P$, in parallel?

**Example**

$T = abb$bab

$P = abbb$babb

4, 3, 5, 4, 3, 3, 2, 1
**Problem**

*How to compute the matching statistics of a pattern $P$, in parallel?*

**Example**

$T = abbbab$

$P = abbbbab$

4, 3, 5, 4, 3, 3, 2, 1
Problem

How to compute the matching statistics of a pattern $P$, in parallel?

Example

$T = abbbaab$
$P = abbbaba$ 
$4, 3, 5, 4, 3, 3, 2, 1$
Problem

How to compute the matching statistics of a pattern $P$, in parallel?

Example

$T = abbbab$

$P = abbbbabb$

4, 3, 5, 4, 3, 3, 2, 1$
Problem

How to compute $ms(2)$ for $P = abbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.
Matching statistics

Problem

How to compute \( ms(2) \) for \( P = abbbabb \) ?

- Build a generalized branch tree.
- Move on the tree merging points.
- \( O((m/p) \log m \log n (\log \log n)^2) \).

![Branch Tree Diagram]

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira

Parallel and Distributed Compressed Indexes
Problem

How to compute $ms(2)$ for $P = abbbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.
Problem

How to compute $ms(2)$ for $P = abbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.
Problem

*How to compute $ms(2)$ for $P = abbbabb$?*

- Build a generalized branch tree.
- Move on the tree merging points.

$O((m/p) \log m \log n (\log \log n)^2)$. 

![Generalized Branch Tree Diagram](diagram)
Matching statistics

Problem

How to compute $ms(2)$ for $P = abbbbabb$ ?

- Build a generalized branch tree.
- Move on the tree merging points.

$O((m/p) \log m \log n(\log \log n)^2)$. 

![Diagram of a generalized branch tree with nodes labeled 'Null', 'bb', 'bb', and 'b'.]
Problem

How to compute $ms(2)$ for $P = abbbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.

![Generalized Branch Tree Diagram]
Matching statistics

Problem

How to compute $ms(2)$ for $P = abbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.

![Image of generalized branch tree](null)
How to compute $ms(2)$ for $P = abbbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$. 

![Generalized Branch Tree Diagram]
**Problem**

*How to compute $ms(2)$ for $P = abbbabb$?*

- Build a generalized branch tree.
- Move on the tree merging points.

$$O((m/p) \log m \log n (\log \log n)^2)$$
Problem

How to compute $ms(2)$ for $P = abbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.
**Problem**

*How to compute the longest common substring between $P$ and $T$, in parallel?*

- It is a side effect of matching statistics.
Problem

*How to compute the longest common substring between P and T, in parallel?*

- It is a side effect of matching statistics.
Problem

How to determine the maximal repeated substrings of $T$, in parallel?

Example

$T = abbbab$.

- Classical solution is the left-diverse internal nodes.
- Notice that left-diverse is equivalent to $\text{COUNT}(\text{LF}(\text{LETTER}(v_i, -1), v)) \neq \text{COUNT}(v)$.
- Hence each node can be verified in parallel.
Problem

*How to determine the maximal repeated substrings of $T$, in parallel?*

Example

$T = abbbab$.

- Classical solution is the left-diverse internal nodes.
- Notice that left-diverse is equivalent to $\text{COUNT}(\text{LF}(\text{LETTER}(v_i, -1), v)) \neq \text{COUNT}(v)$.
- Hence each node can be verified in parallel.
Maximal Repeats

Example

```
<table>
<thead>
<tr>
<th>Value</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
</tr>
<tr>
<td>6</td>
<td>b</td>
</tr>
</tbody>
</table>
```

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira

Parallel and Distributed Compressed Indexes
Problem

How to determine the maximal repeated substrings of $T$, in parallel?

Example

$T = abbbab$.

- Classical solution is the left-diverse internal nodes.
- Notice that left-diverse is equivalent to $\text{COUNT}(\text{LF}(\text{LETTER}(v_l, -1), v)) \neq \text{COUNT}(v)$.
- Hence each node can be verified in parallel.
Problem

*How to determine the maximal repeated substrings of \( T \), in parallel?*

Example

\[ T = abbbab. \]

- Classical solution is the left-diverse internal nodes.
- Notice that left-diverse is equivalent to \( \text{COUNT}(\text{LF}(\text{LETTER}(v_l, -1), v)) \neq \text{COUNT}(v) \).
- Hence each node can be verified in parallel.
Problem

Simulate the access to a big CSA?

- Store a bitmap \( ld = 0100101010101 \).
- \( \text{LF}_c(X, [v_l, v_r]) = \) 
  \[ \min_{j=0}^{q-1} \{ \text{SELECT}_j(xv_j, l + 1) \}, \max_{j=0}^{q-1} \{ \text{SELECT}_j(xv_j, r + 1) \} \]
Problem

Simulate the access to a big CSA?

- Store a bitmap $ld = 0100101010101$.
- $LF_C(X, [v_l, v_r]) = \min_{j=0}^{q-1}\{\text{SELECT}_j(xv_j, l + 1)\}, \max_{j=0}^{q-1}\{\text{SELECT}_j(xv_j, r + 1)\}$

Example
**Problem**

*Simulate the access to a big CSA?*

- Store a bitmap $ld = 0100101010101$.
- $LF_C(X, [v_l, v_r]) = \left[ \min_{j=0}^{q-1} \{ \text{SELECT}_j(xv_j, l + 1) \}, \max_{j=0}^{q-1} \{ \text{SELECT}_j(xv_j, r + 1) \} \right]$
Problem

How to simulate the access to a big FCST by using several distributed FCSTs?

- Store sampled node identifying bitmaps.
- Merge different LSA as in CSAs.

((0)(1)((2)((3)(4))))((5)(6)(7)((8)(9)(10)(11)(12))))

((0 1 (2 (3 (4)))) (5)(6)(7)(8 9 10 11 12))

B: 1 0 0 1 0 1 0 10111 1011011011 0 0 0 0 1 1
B0: 1 0 1 0 1 0 1 111 1011 11011 0 0 1 1
B1: 1 0 1 1 1 01111 1 11011 11 0 0 0 1 1
How to compute operations over FCSTs:

- $\text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) = \max_{0 \leq i < d}\{i + \text{SDEP}(\text{LCSA}(\psi^i(v_l), \psi^i(v_r)))\}$.

- $\text{LCA}(v, v') = \text{LF}(v[0..i - 1], \text{LCSA}(\psi^i(\min\{v_l, v'_l\}), \psi^i(\max\{v_r, v'_r\})))$, for the $i$ in sdep.
How to compute operations over FCSTs:

1. \( \text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) = \max_{0 \leq i < d} \{ i + \text{SDEP}(\text{LCSA}(\psi^i(v_l), \psi^i(v_r))) \} \).

2. \( \text{LCA}(v, v') = \)

   \( \text{LF}(v[0..i - 1], \text{LCSA}(\psi^i(\min\{v_l, v'_l\}), \psi^i(\max\{v_r, v'_r\}))) \),

   for the \( i \) in sdep.
Classical FCST operations

How to compute operations over FCSTs:

- \( \text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) = \max_{0 \leq i < d} \{ i + \text{SDEP}(\text{LCSA}(\psi^i(v_l), \psi^i(v_r))) \} \).

- \( \text{LCA}(v, v') = \)
  \[
  \text{LF}(v[0..i - 1], \text{LCSA}(\psi^i(\min\{v_l, v'_l\}), \psi^i(\max\{v_r, v'_r\})), \text{for the } i \text{ in sdep}.)
  \]
Problem

How to simulate the access to a big FCST by using several distributed FCSTs?

- Store sampled node identifying bitmaps.
- Merge different LSA as in CSAs.

$$
((0)(1)((2)((3)(4)))((5)(6)(7)((8)(9)(10)(11)(12))))
$$

$$
((0)(1)(2(3(4)))((5)(6)(7)(8)(9)(10)(11)(12)))
$$

<table>
<thead>
<tr>
<th>B</th>
<th>1 0 0 1 0 1 0 10111 1011011011 0 0 0 0 0 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>1 0 1 0 1 0 1 111 1011 11011 0 0 1 1</td>
</tr>
<tr>
<td>B1</td>
<td>1 0 1 1 10111 1 11011 11 0 0 0 1 1</td>
</tr>
</tbody>
</table>
Problem

**How to simulate the access to a big FCST by using several distributed FCSTs?**

- Store sampled node identifying bitmaps.
- Merge different LSA as in CSAs.

\[( (0)(1)((2)((3)(4))))((5)(6)(7)((8)(9)(10)(11)(12))) \]

\[(0 \ 1 \ (2 \ (3 \ (4))) \ (5 \ (6 \ (7) \ (8 \ 9 \ 10 \ 11 \ 12)) \ ) \]

\[B : \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 10111 \ 1011011011 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \]

\[B0 : \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 111 \ 1011 \ 11011 \ 0 \ 0 \ 0 \ 1 \ 1 \]

\[B1 : \ 1 \ 0 \ 1 \ 1 \ 10111 \ 11011 \ 110 \ 0 \ 0 \ 0 \ 1 \ 1 \]
We presented parallel and distributed compressed indexes. Our solutions obtain:

- Fast operations in compressed space.
- Support for very large indexes.
Thanks for listening.

Questions?
Marín, M., Navarro, G.: 
Distributed query processing using suffix arrays. 

Mäkinen, V., Navarro, G., Sadakane, K.: 
Advantages of backward searching — efficient secondary memory and distributed implementation of compressed suffix arrays. 

Clifford, R.: 
Distributed suffix trees. 
J. Discrete Algorithms 3(2-4) (2005) 176–197