Extending the Parameterized Burrows-Wheeler Transform

Eric M. Osterkamp (University of Münster)
Dominik Köppl (University of Yamanashi)

based on the slides for the final presentation of the Bachelor thesis of the first author
Pattern Matching

• alphabet $\Sigma$
• text $T \in \Sigma^*$, pattern $P \in \Sigma^*$
• occurrence of $P$ in $T$: substring of $T$ that equals $P$
• PM: count all occurrences of $P$ in $T$
  write as $T.count(P)$
• goal: index $T$ for efficient PM

$\Sigma = \{a, b, c\}$
$T = \text{acabcaababba}$
$P = ab$
• occurrences of $P$ in $T$
  at positions 3, 7 and 9
$T.count(P) = 3$
Parameterized Strings

• alphabet $\Sigma_s$ of static symbols ($s$-symbols)
• alphabet $\Sigma_p$ of parameterized symbols ($p$-symbols)
• $\Sigma_s \cap \Sigma_p = \emptyset$
• string over $\Sigma := \Sigma_s \cup \Sigma_p$ is a parameterized string ($p$-string)
• character in $\Sigma$ called $\text{symbol}$, $\sigma := |\Sigma|$ size of alphabet
example
• $\Sigma_s = \{a, b\}, \Sigma_p = \{A, B, C\}$
• $T = \text{ACAbCAabABBA}$
Parameterized Matching (p-Matching)

- \( U, V \) p-strings
- \( U \) p-matches \( V \) if \(|U| = |V|\) and \( \exists \) a bijection \( \psi: \Sigma_p \to \Sigma_p \) with
  - \( U[i] = V[i] \) if \( V[i] \in \Sigma_s \)
  - \( U[i] = \psi(V[i]) \) otherwise
- write \( U =_p V \) iff \( U \) and \( V \) p-match

example
- \( U = bBCAaCB \)
- \( V = bCABaAC \)
- \( \psi(A) = C, \psi(B) = A, \psi(C) = B \)

\[
U = \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
b & B & C & A & a & C & B \\
\end{array}
\]

\[
V = \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
b & C & A & B & a & A & C \\
\end{array}
\]

\[\Rightarrow U =_p V\]
Parameterized Pattern Matching (PPM)

- $T$: text p-string
- $P$: pattern p-string
- occurrence of $P$ in $T$: substring of $T$ that p-matches $P$
- PPM: count all occurrences of $P$ in $T$, written as $T.\text{count}(P)$
- goal: index text $T$ for efficient PPM

[Diagram showing example text and pattern with occurrence highlighted]
# Indexes for PPM

<table>
<thead>
<tr>
<th>data structure</th>
<th>time for PPM</th>
<th>reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>suffix tree</td>
<td>$O(m \log \sigma)$</td>
<td>[Baker ‘93]</td>
</tr>
<tr>
<td>suffix array</td>
<td>$O(m + \log n)$</td>
<td>[Deguchi + ‘08]</td>
</tr>
<tr>
<td>position heap</td>
<td>$O(m \log \sigma + m \sigma_p)$</td>
<td>[Diptarama+ ‘17]</td>
</tr>
<tr>
<td>suffix tray</td>
<td>$O(m + \log \sigma)$</td>
<td>[Fujisato+ ‘21]</td>
</tr>
<tr>
<td>DAWG</td>
<td>$O(m \log \sigma)$</td>
<td>[Nakashima+ ‘22]</td>
</tr>
</tbody>
</table>

- $\sigma := |\Sigma|$ alphabet size
- $\sigma_p := |\Sigma_p|$  
- $n := |T|$, text size
- $m : |P|$, pattern length

All data structures need $O(n \log n)$ bits.
PPM in small memory

- parameterized Burrows-Wheeler transform (pBWT) [Ganguly+ ’17]
  - \( n \lg \sigma + O(n) \) bits
  - computes \( T.\text{count}(P) \) in \( O(m \log \sigma) \) time

- simplified pBWT [Kim, Cho ‘21]
  - \( 2n \lg \sigma + O(n) \) bits
  - same time complexities

- both approaches use space linear in the number of bits of the input!
Applications for PPM

many use cases
• software maintenance [Baker ‘97]
• plagiarism detection
• analyzing genetic data [Shibuya ’04]
RNA matching

- matching RNA base pair
  - $X = \text{AUGCAUC}$
  - $Y = \text{CGAUUCGU}$
  - $\psi(X) = Y$

- but some RNA structures are cyclic, so there is a need for cyclic pattern matching $\Rightarrow$ circular parameterized pattern matching (CPPM)

$$
\psi:
\begin{align*}
A & \leftrightarrow C, \\
U & \leftrightarrow G, \\
C & \leftrightarrow U, \\
G & \leftrightarrow A
\end{align*}

$$

$$
X = \text{AUGCAUC} =_{p} \text{CGAUUCGU} = Y
$$

[Shibuya’ 04]
Circular PPM (CPPM)

- text p-strings \( T = \{T_1, \ldots, T_d\} \)
- pattern p-string \( P \)
- occurrence of \( P \) in \( T \) refers to the starting position of a substring of \( T_1, \ldots, T_d \) that p-matches \( P \)
- all text p-strings are viewed circularly
- CPPM: count all occurrences of \( P \) in \( T \)
- goal: index texts \( T_1, \ldots, T_d \) for efficient CCPM
Example: CPPM

• $T = \{AC, AbC, Aab, ABBA\}$
• $P = BA$
• occurrences of $P$ in $T$ at positions 1, 2, 4, 9 and 11
• $T.\text{count}(P) = 5$
Simple idea for CPPM

- general naive approach for matching $P$ in $T$ circularly:
- perform classic matching of $P$ in $T \cdot T$
- may generate pseudo results in the second part
- discard pseudo results in postprocessing

Example:

$T = \text{ABBA}$

$P = \text{BAA}$

$p = \text{BAA}$
From pBWT to epBWT

define epBWT based on two encodings
- prev-encoding $\langle V \rangle$ [Baker ‘93]
- Hashimoto-encoding «V», [Hashimoto+ ‘22]
motivation is explained with a review of pBWT
BWT (Burrows-Wheeler transform):

- last character of all cyclic rotations sorted in lexicographic order
- \( T.\text{count}(P) \) via length of reported range of backward search
- how to use that technique with p-matching?

pBWT

review of the simplified pBWT [Kim, Cho ‘21] for PPM
Comparing p-Strings

- consider conjugates of $V = AABB$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>B</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>

sort lexicographically

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>B</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

- $AABB = p$ BBAA $\neq p$ ABBA $= p$ BAAB, but $AABB < ABBA < BAAB < BBAA$
- cannot use original p-string to sort or p-match!
prev-Encoding

given p-string $V$, compute prev-encoding $\langle V \rangle$ of $V$ as follows:

- replace leftmost occurrence of any p-symbol in $V$ by $\infty$
- replace each other by distance to its previous occurrence

for every p-string $U$: $\langle V \rangle = \langle U \rangle \iff V =_p U$ [Baker ‘93]

\[
T = \infty \infty 2 \ b \ 3 \ 3 \ a \ b \ 3 \ \infty \ 1 \ 3 \\
\langle T \rangle = \infty \infty 2 \ b \ 3 \ 3 \ a \ b \ 3 \ \infty \ 1 \ 3
\]

- but unstable under rotation

\[
\langle T \rangle[2..]\langle T \rangle[1] = \infty \ 2 \ b \ 3 \ 3 \ a \ b \ 3 \ \infty \ 1 \ 3 \ \infty \\
T[2..]T[1] = C \ A \ b \ C \ A \ a \ b \ A \ B \ B \ A \ A \\
\langle T[2..]T[1] \rangle = \infty \ \infty \ b \ 3 \ 3 \ a \ b \ 3 \ \infty \ 1 \ 3 \ 1
\]
Hashimoto-Encoding [Hashimoto+ ‘22]

• view p-string $V$ circularly and replace each occurrence of a p-symbol in $V$ by the number of distinct p-symbols until its next occurrence
• write <<$V$>> for the Hashimoto-encoding of $V$

$$T = \text{A C A b C A a b A B B A}$$

$$\text{«}T\text{»} = 2 2 2 \ b \ 3 \ 1 \ a \ b \ 2 \ 1 \ 3 \ 1$$

• for every p-string $U$: «$V$» = «$U$» $\Leftrightarrow$ $V =_p U$ [Hashimoto+ ‘22]
• encoding is commutative with rotation!

$$\text{«}T\text{»}[2..]\text{«}T\text{»}[1] = 2 2 \ b \ 3 \ 1 \ a \ b \ 2 \ 1 \ 3 \ 1 \ 2$$

$$T[2..]T[1] = \text{C A b C A a b A B B A A}$$

$$\text{«}T[2..]T[1]\text{»} = 2 2 \ b \ 3 \ 1 \ a \ b \ 2 \ 1 \ 3 \ 1 \ 2$$
Parameterized BWT (pBWT)

- text $T = \text{ACAbCAabABBA}\$ 
- \text{«}T\text{»} = 222b31ab2131\$
- pBWT($T$) = ($F_T$, $L_T$)

- first and last symbols of Hashimoto-encoded conjugates sorted by their prev-encodings

- similar entries of both strings are sorted by succeeding context! [Iseri+ ’23]
LF (Mapping) Property

- text $T = ACAbCAabABBA$.
- «$T$» = 2 1 2 2 3 b 1 3 1 1 a 1 b 2 2 4 1 3 2 3 1 3 $1$
- first column $F_T = $ $1_1 \ a_1 \ b_1 \ b_2 \ 1_3 \ 1_2 \ 3 \ 1_2 \ 3 \ 2 \ 3 \ 1 \ 2_2 \ 2_4 \ 2_1$
- last column $L_T = $ $1_3 \ 1_1 \ 2_2 \ a_3 \ 3 \ 3 \ 2 \ 2_1 \ 1_2 \ b_2 \ b_2 \ $1
- define permutation $LF_T$ by mapping from $i$th occurrence of a symbol $x \in \Sigma_S \cup [1..|\Sigma_p|]$ in $L_T$ to $i$th occurrence of $x$ in $F_T$.
- LF property: maps $x_k$ of $L_T$ to $x_k$ of $F_T$!
epBWT

from pBWT to epBWT
ω-Order

idea: use the infinite iteration of a conjugate as key for sorting

• $V^\omega$: infinite iteration of $V$
• $\text{root}(V) := \text{primitive root of } V$ ($V = ababab \Rightarrow \text{root}(V) = ab$)

• $V, U$: finite strings
• $V =_\omega U :\iff \text{root}(V) = \text{root}(U)$
• $V <_\omega U :\iff \exists i: V^\omega[\ldots i] = U^\omega[\ldots i] \land V^\omega[i + 1] < U^\omega[i + 1]$
Extending the $\omega$-Order to p-Strings

- $V, U :$ finite p-strings
- $V =_\omega U :\iff \text{root}('V') = \text{root}('U')$
- $V <_\omega U :\iff \exists i: \langle V^\omega \rangle[..i] = \langle U^\omega \rangle[..i] \land \langle V^\omega \rangle[i + 1] < \langle U^\omega \rangle[i + 1]$

- extended $\omega$-order already used when defining the pBWT!
- coincides with prev-order for p-strings of the same length
Example: \( \omega \)-Order on p-Strings

\[ T_1 = AB, \quad T_2 = ABA, \quad T_3 = ABAB \]
\[ \langle T_1 \rangle < \langle T_2 \rangle < \langle T_3 \rangle \]

\[ T_1^\omega[..8] = ABAABAABA \]
\[ T_2^\omega[..8] = ABAABAABA \]
\[ T_3^\omega[..8] = ABAABAABA \]
\[ \langle T_1 \rangle = \infty \infty \]
\[ \langle T_2 \rangle = \infty \infty 2 \]
\[ \langle T_3 \rangle = \infty \infty 2 \ 2 \]

\[ T_1 \],[..8] = \infty \infty 2 2 2 2 2 2 \]
\[ T_2 \],[..8] = \infty \infty 2 1 3 2 1 3 \]
\[ T_3 \],[..8] = \infty \infty 2 2 2 2 2 2 \]
\[ T_2 \prec_\omega T_1 =_\omega T_3 \]
Extended pBWT (epBWT)

- sort conjugates by $\omega$-order tie-break:
  - first by index of text string,
  - second by text position
- $T = \{\text{AC, AbC, Aab, ABBA}\}$
- epBWT($T$) = ($F_T$, $L_T$)
- first and last symbols of Hashimoto-encoded conjugates sorted by their prev-encodings in $\omega$-order
LF (Mapping) Property

- $T = \{\text{AC, AbC, Aab, ABBA}\}$
- $\{\text{«T}_1\», \text{«T}_2\», \text{«T}_3\», \text{«T}_4\}\} = \{2122, 23b124, 1a1b2, 25122613\}$
- first column $F_T = a_1b_2b_11_23_12_34_24_25262122$
- last column $L_T = 1_1a_2b_22_42_56_13_12221$
- define permutation $\text{LF}_T$ by mapping from $i$th occurrence of a symbol $x \in \Sigma_S \cup [1..|\Sigma_P|]$ in $L_T$ to $i$th occurrence of $x$ in $F_T$
- only maps $x_k$ of $L_T$ to $x_k$ of $F_T$ if Hashimoto-encoded texts are primitive! («AC» and «ABBA» are not primitive!)
- remedy: build epBWT on the Hashimoto-encoded roots!
Summary

epBWT is a CPPM index for a set of p-strings

- builds upon pBWT of [Kim, Cho ’21] and eBWT of [Mantaci+ ’07]
- uses $2n \lg \sigma + O(n)$ bits of space

partially in the paper (full version will follow):

- $T.\ count(P)$ in $O(m \lg \sigma)$ time for CPPM
- reconstruction of input up to p-matching equivalence
- construction of index in $O\left(n \frac{\lg^2 n}{\lg \lg n}\right)$ time with $O(n \lg n)$ bits of space
- applications to other matchings such as Cartesian-Tree matching