On the Parikh-de-Bruijn grid

Péter Burcsi    Zsuzsanna Lipták    W. F. Smyth

ELTE Budapest (Hungary), U of Verona (Italy),
McMaster U (Canada) & Murdoch U (Australia)

LSD/LAW 2018
London, 8-9 Feb. 2018
**Def.** Given a string $s = s_1 \cdots s_n$ over a finite ordered alphabet $\Sigma$ of size $\sigma$, the Parikh-vector $\text{pv}(s)$ is the vector $(p_1, \ldots, p_\sigma)$ whose $i$'th entry is the multiplicity of character $a_i$.

**Ex.** $s = \text{aabaccba}$ over $\Sigma = \{a, b, c\}$, then $\text{pv}(s) = (4, 2, 2)$. 

Abelian stringology
Abelian stringology

Def. Given a string $s = s_1 \cdots s_n$ over a finite ordered alphabet $\Sigma$ of size $\sigma$, the Parikh-vector $p\mathbf{v}(s)$ is the vector $(p_1, \ldots, p_\sigma)$ whose $i$'th entry is the multiplicity of character $a_i$.

Ex. $s = \text{aabaccba}$ over $\Sigma = \{a, b, c\}$, then $p\mathbf{v}(s) = (4, 2, 2)$.

Def. Two strings over the same alphabet are Parikh equivalent (a.k.a. abelian equivalent) if they have the same Parikh vector. (i.e. if they are permutations of one another)

Ex. $\text{aaaabbcc}$ and $\text{aabcaabc}$ are both Parikh equivalent to $s$. 
Abelian stringology

**Def.** Given a string $s = s_1 \cdots s_n$ over a finite ordered alphabet $\Sigma$ of size $\sigma$, the Parikh-vector $\text{pv}(s)$ is the vector $(p_1, \ldots, p_\sigma)$ whose $i$'th entry is the multiplicity of character $a_i$.

**Ex.** $s = \text{aabaccba}$ over $\Sigma = \{a, b, c\}$, then $\text{pv}(s) = (4, 2, 2)$.

**Def.** Two strings over the same alphabet are Parikh equivalent (a.k.a. abelian equivalent) if they have the same Parikh vector. (i.e. if they are permutations of one another)

**Ex.** $\text{aaaabbcc}$ and $\text{aabcaabc}$ are both Parikh equivalent to $s$.

In Abelian stringology, equality is replaced by Parikh equivalence.
Abelian stringology

In Abelian stringology, equality is replaced by Parikh equivalence.

- Jumbled Pattern Matching
- abelian borders
- abelian periods
- abelian squares, repetitions, runs
- abelian pattern avoidance
- abelian reconstruction
- abelian problems on run-length encoded strings
- ...

Zs. Lipták, P. Burcsi, W.F. Smyth
On the Parikh-de-Bruijn grid
LSD/LAW 2018
Abelian stringology

In this talk, we introduce a new tool for attacking abelian problems.
Abelian stringology

In this talk, we introduce a new tool for attacking abelian problems.

But first: in what way are abelian problems different from their classical counterparts?

N.B.: Recall $\Sigma$ is finite and ordered, and $\sigma = |\Sigma|$. 
Example 1: Parikh-de-Bruijn strings

- **Recall:** A [de Bruijn sequence](https://en.wikipedia.org/wiki/De_Bruijn_sequence) of order $k$ over alphabet $\Sigma$ is a string over $\Sigma$ which contains every $u \in \Sigma^k$ exactly once as a substring.
- de Bruijn sequences exist for every $\Sigma$ and $k$
- correspond to Hamiltonian paths in the [de Bruijn graph](https://en.wikipedia.org/wiki/De_Bruijn_graph)
- can be constructed efficiently via Euler-paths in the de Bruijn graph of order $k - 1$


Zs. Lipták, P. Burcsi, W.F. Smyth On the Parikh-de-Bruijn grid
Example 1: Parikh-de-Bruijn strings

Def.

- the order of a Parikh vector (Pv) is the sum of its entries
  (\(\equiv\) length of a string with this Pv)
- a Parikh-de-Bruijn string of order \(k\) (a \((k, \sigma)\)-PdB-string) is a string \(s\) over an alphabet of size \(\sigma\) s.t.

\[
\forall \ p \ \text{Parikh vector of order} \ k \ \exists!(i, j) \ \text{s.t.} \ \mathbf{pv}(s_i \cdots s_j) = p
\]

(There is exactly one occurrence of a substring in \(s\) which has \(\text{Pv} \ p\).)
Example 1: Parikh-de-Bruijn strings

Def.

- the **order** of a Parikh vector (Pv) is the sum of its entries (= length of a string with this Pv)
- a **Parikh-de-Bruijn string** of order \( k \) (a \((k, \sigma)\)-PdB-string) is a string \( s \) over an alphabet of size \( \sigma \) s.t.

\[
\forall \ p \ \text{Parikh vector of order } k \ \exists! (i, j) \ \text{s.t.} \ \text{pv}(s_i \cdots s_j) = p
\]

(There is exactly one occurrence of a substring in \( s \) which has Pv \( p \).)

Ex.

- **aabbcca** is a \((2, 3)\)-PdB-string
Example 1: Parikh-de-Bruijn strings

Def.

• the order of a Parikh vector (Pv) is the sum of its entries
  (= length of a string with this Pv)
• a Parikh-de-Bruijn string of order $k$ (a \((k, \sigma)\)-PdB-string) is a string $s$ over an alphabet of size $\sigma$ s.t.

\[
\forall \ p \ \text{Parikh vector of order } k \ \exists! (i, j) \ \text{s.t.} \ \text{pv}(s_i \cdots s_j) = p
\]

(There is exactly one occurrence of a substring in $s$ which has Pv $p$.)

Ex.

• $aabbcca$ is a \((2, 3)\)-PdB-string
• $abbbcccaaacbc$ is a \((3, 3)\)-PdB-string
Example 1: Parikh-de-Bruijn strings

**Def.**

- the *order* of a Parikh vector (Pv) is the sum of its entries (= length of a string with this Pv)
- a **Parikh-de-Bruijn string** of order \( k \) (a \((k, \sigma)\)-PdB-string) is a string \( s \) over an alphabet of size \( \sigma \) s.t.

\[
\forall \ p \ Parikh \ vector \ of \ order \ k \ \exists ! (i, j) \ s.t. \ pv(s_i \cdots s_j) = p
\]

(There is exactly one occurrence of a substring in \( s \) which has Pv \( p \).)

**Ex.**

- \( aabbcca \) is a \((2, 3)\)-PdB-string
- \( abbbccccaaabc \) is a \((3, 3)\)-PdB-string
- but no \((4, 3)\)-PdB-string exists
Example 1: Parikh-de-Bruijn strings

Def.

- the order of a Parikh vector (Pv) is the sum of its entries (\(\equiv\) length of a string with this Pv)
- a Parikh-de-Bruijn string of order \(k\) (a \((k, \sigma)\)-PdB-string) is a string \(s\) over an alphabet of size \(\sigma\) s.t.

\[ \forall p \text{ Parikh vector of order } k \exists!(i, j) \text{ s.t. } pv(s_i \cdots s_j) = p \]

(There is exactly one occurrence of a substring in \(s\) which has Pv \(p\).)

Ex.

- \(aabbcca\) is a \((2, 3)\)-PdB-string
- \(abbbccccaaabc\) is a \((3, 3)\)-PdB-string
- but no \((4, 3)\)-PdB-string exists
- and no \((2, 4)\)-PdB-string exists
Example 2: Covering strings

Next best thing: covering strings.

Def.

- We call a string \( s \) \((k, \sigma)\)-covering if

\[ \forall p \text{ Parikh vector of order } k \ \exists (i, j) \text{ s.t. } pv(s_i \cdots s_j) = p \]

(There is at least one substring in \( s \) which has \( Pv \) \( p \).)
- The excess of \( s \) is:

\[ |s| - \left( \frac{\sigma + k - 1}{k} \right) + k - 1 \]

length of a PdB-string

Ex.

- \textcolor{red}{aaaabbbbccttccaaacabcb} is a shortest \((4, 3)\)-covering string, with excess 1.
- \textcolor{red}{aabcadbdccdd} is a shortest \((2, 4)\)-covering string, with excess 1.
Example 2: Covering strings

**Classical case:** If $s$ is a (classical) de Bruijn sequence of order $k$, then it also contains all $(k - 1)$-length strings as substrings.
Example 2: Covering strings

**Classical case:** If $s$ is a (classical) de Bruijn sequence of order $k$, then it also contains all $(k - 1)$-length strings as substrings.

**For PdB-strings,** this is not always true, e.g.

aaaaabbbbbcaaaaadbbbcccccccccdddddadaaccdbcbaccaccdddbddbdadacdddbbb

is a (5, 4)-PdB-string but is not (4, 4)-covering: no substring with $Pv(1, 1, 1, 1)$. 
The Parikh-de-Bruijn grid
Recall: de Bruijn graphs $B_k = (V, E)$, where $V = \Sigma^k$ and $(xu, uy) \in E$ for all $x, y \in \Sigma$ and $u \in \Sigma^{k-1}$.

Note that $E = \Sigma^{k+1}$. 
Recall: de Bruijn graphs $B_k = (V, E)$, where $V = \Sigma^k$ and $(xu, uy) \in E$ for all $x, y \in \Sigma$ and $u \in \Sigma^{k-1}$.

Note that $E = \Sigma^{k+1}$.

A straightforward generalization to Pv’s does not work, because edges do not uniquely correspond to $(k + 1)$-order Pv’s:
Let’s look at another example: Here, $\sigma = 3$, $k = 2$.

Again, in the abelian version, we have that several edges have the same label (i.e. here: the same 3-order $Pv$).
Turns out the right way to generalize de Bruijn graphs is the Parikh-de-Bruijn grid:
Turns out the right way to generalize de Bruijn graphs is the Parikh-de-Bruijn grid:
The Parikh-de-Bruijn grid

The (4, 3)-PdB-grid

green: $k$-order $Pv$’s (vertices), yellow: $(k + 1)$-order $Pv$’s (downward triangles/tetrahedra), blue: $(k - 1)$-order $Pv$’s (upward triangles/tetrahedra).
The Parikh-de-Bruijn grid

PdB-grid:
- \( V = k \)-order \( P \nu \)'s
- \( pq \in E \) iff exist \( x, y \in \Sigma \) s.t. \( p = q - x + y \)
- undirected edges (or: bidirectional edges)
- \((k - 1)\)- and \((k + 1)\)-order \( P \nu \)'s correspond to sub-simplices (triangles for \( \sigma = 3 \), tetrahedra for \( \sigma = 4 \) etc.)
- every string corresponds to a walk in the PdB-grid, but not every walk corresponds to a string
The Parikh-de-Bruijn grid

Every string corresponds to a walk in the PdB-grid, but not every walk corresponds to a string:
The Parikh-de-Bruijn grid

Every string corresponds to a walk in the PdB-grid, but not every walk corresponds to a string:

But with loops it’s possible!
The Parikh-de-Bruijn grid

Lemma
A set of $k$-order Parikh vectors is realizable if and only if the induced subgraph in the $k$-PdB-grid is connected.

realizable $=$ exists string with exactly these $k$-order sub-Pv’s.

Proof sketch
Use loops until undesired character $x$ exits, replace by new character $y$. 
The Parikh-de-Bruijn grid

Lemma
A set of $k$-order Parikh vectors is realizable if and only if the induced subgraph in the $k$-PdB-grid is connected.

realizable = exists string with exactly these $k$-order sub-Pv’s.

Proof sketch
Use loops until undesired character $x$ exits, replace by new character $y$.

Actually, better name: loops $\rightarrow$ bows (see next slide); one for each character.
The Parikh-de-Bruijn grid

$k = 4, \sigma = 3$

Walk corresponding to $aabacabb$. $(k + 1)$- and $(k - 1)$-order Pv’s: triangles incident to the edges traversed by the walk. The $(k + 1)$ and $(k - 1)$-order Pv’s for loops (same $k$-order Pv twice) lie in opposite direction, hence the name bow.
Back to Parikh-de-Bruijn and covering strings

**Theorem 1**
No $(k, 3)$-PdB strings exist for $k \geq 4$.

**Theorem 2**
A $(2, \sigma)$-PdB string exists if and only if $\sigma$ is odd.

**Theorem 3**
For every $\sigma \geq 3$ and $k \geq 4$, there exist $(k, \sigma)$-covering strings which are not $(k - 1, \sigma)$-covering.
Theorem 1 No \((k, 3)\)-PdB strings exists for \(k \geq 4\).
Parikh-de-Bruijn and covering strings

Theorem
A \((2, \sigma)\)-PdB string exists if and only if \(\sigma\) is odd.

Proof
Pv’s of order 2 have either the form \((0...0, 2, 0..0)\) or \((0...0, 1, 0...0, 1, 0..0)\). So \(s\) has to have exactly one substring of the form \(aa\) for all \(a \in \Sigma\), and either \(ab\) or \(ba\) for all \(a, b \in \Sigma\). Consider the undirected complete graph \(G = (V, E)\) with loops where \(V = \Sigma\) (N.B.: not the PdB-grid!): an Euler path exists iff \(\sigma\) is odd.
Theorem 3
For every $\sigma \geq 3$ and $k \geq 4$, there exist $(k, \sigma)$-covering strings which are not $(k-1, \sigma)$-covering.

Proof
$w = \text{aaaaabbbbbcabbaaaacacbbcbccacaccccbccccc}$

General construction:
- remove $(k - 1)$-order $P_v$ $p = (k - 3, 1, 1, 0, \ldots, 0)$ with incident edges and vertices
- the rest is connected, hence a string exists (Lemma)
- add vertices of $p$ without traversing edges incident to $p$
- can be done by detours from corners of PdB-grid
# Experimental results

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\sigma$</th>
<th>string</th>
<th>length (excess)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>aabbcca</td>
<td>7 (0)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>aabbbccaaabc</td>
<td>12 (0)</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>aaaaabbbbcbbbbbaacaacacb</td>
<td>19 (1)</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>aaaaaabbbacccccbbbbbbaacaaccb</td>
<td>27 (2)</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>aaaaabccccccaaaaabbbbbbbcccbbcabbaca</td>
<td>35 (2)</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>aabbbccbbcccabacaabcbbbbbbbaaaaaaacccccccccbba</td>
<td>44 (2)</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>aabbcadbcddd</td>
<td>12 (1)</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>aabbbcaadadbccaddddccc</td>
<td>22 (0)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>aabbbcaacadbdbbcacdddddaaaabdbbcccccd</td>
<td>38 (0)</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>aaaaabbbbbcaaaaadbbccccccdddaaaaccdbcbaccaccdddbbbaadacddbbbb</td>
<td>60 (0)</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>aabbcadbeccddeea</td>
<td>16 (0)</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>aaabbbcaadbbeaccbdddcccebededadceeeaa</td>
<td>37 (0)</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>aaaaabbbbbbcaadbbbbeaaccbddaeeabcccadbeeeaddccccceeeedddd...</td>
<td>73 (0)</td>
</tr>
</tbody>
</table>
Conclusion and open problems

• new tool for modeling and solving abelian problems
• find good characterization for walks which correspond to strings
• several open problems on PdB- and covering strings (see paper on Arxiv)
• apply PdB-grid to other abelian problems
THANK YOU!