Indexable Elastic Founder Graphs of Minimum Height

with Suffix Tree Maneuvers

Nicola Rizzo Veli Mäkinen

Algorithmic Bioinformatics Research Group
Department of Computer Science, University of Helsinki, Finland
{nicola.rizzo,veli.makinen}@helsinki.fi

CPM 2022

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie ITN ALPACA (grant agreement No 956229).

Outline

The search for a pangenome data structure

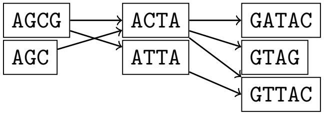
- computational pangenomics: find a data structure for a coherent collection of genomes supporting fast pattern matching
- graph-based candidates like Variation Graphs and Elastic Degenerate Strings usually represent a multiple sequence alignment (MSA)
- cannot support string matching in subquadratic time under the Orthogonal Vectors Hypothesis for simple classes of graphs

The Elastic Founder Graph

Definition (Mäkinen *et al* (2020))

A segmentation S of MSA$[1..m, 1..n]$ induces an elastic block graph $G(S) = (V, E, \ell)$ that we call elastic founder graph (EFG). EFGs respecting the semi-repeat-free property admit a poly-time index for linear-time pattern matching.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	A	G	C	G	A	C	T	A	G	A	T	A	C
2	A	G	C	–	A	C	T	A	G	–	T	A	G
3	A	G	C	G	A	T	T	A	G	T	T	A	C
4	A	G	C	–	A	C	T	A	G	T	T	A	C



Example

Segmentation $S = [1..4], [5..8], [9..13]$ induces this EFG $G(S)$.

- strings become node labels;
- edges are based on local occurrences $\Rightarrow$ recombination;

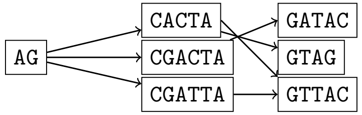
- linear-time construction algorithms for the gapless setting, non-trivial to extend to the general setting

The Elastic Founder Graph

Definition (Mäkinen *et al* (2020))

A segmentation S of MSA$[1..m, 1..n]$ induces an elastic block graph $G(S) = (V, E, \ell)$ that we call elastic founder graph (EFG). EFGs respecting the semi-repeat-free property admit a poly-time index for linear-time pattern matching.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	A	G	C	G	A	C	T	A	G	A	T	A	C
2	A	G	C	–	A	C	T	A	G	–	T	A	G
3	A	G	C	G	A	T	T	A	G	T	T	A	C
4	A	G	C	–	A	C	T	A	G	T	T	A	C


$$\begin{tikzcd}
 & \boxed{\mathtt{CACTA}} \arrow[rd] \arrow[rdd] & \boxed{\mathtt{GATAC}} \\
\boxed{\mathtt{AG}} \arrow[ru] \arrow[r] \arrow[rd] & \boxed{\mathtt{CGACTA}} \arrow[ru] & \boxed{\mathtt{GTAG}} \\
 & \boxed{\mathtt{CGATTA}} \arrow[r] & \boxed{\mathtt{GTTAC}}
\end{tikzcd}$$

Example

Segmentation $S = [1..4], [5..8], [9..13]$ induces this EFG $G(S)$.

- strings become node labels;
- edges are based on local occurrences $\Rightarrow$ recombination;

- linear-time construction algorithms for the gapless setting, non-trivial to extend to the general setting

EFG construction algorithms

- we concentrate on constructing a semi-repeat-free EFG minimizing the maximum block height
- an optimal segmentation is found via dynamic programming (details in the paper) after two important preprocessing steps:
    - computing all valid semi-repeat-free segments
    - computing the height information of all valid segments

Our contributions:

1. preprocessing in time $O(mn\alpha \log|\Sigma|)$, where α is the length of longest aligned common substring
2. we studied a refined height definition resulting in an $O(mn)$-time preprocessing and construction algorithm

Representing the valid segments

The first step is computing the valid segments.

Observation

If $[x..y]$ is valid, then $[x..y']$ is valid for all $y' > y$.

Definition

Given x, the minimal right extension $f(x)$ marks the first column so that $[x..f(x)]$ is valid.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	A	G	C	G	A	C	T	A	G	A	T	A	C
2	A	G	C	—	A	C	T	A	G	—	T	A	G
3	A	G	C	G	A	T	T	A	G	T	T	A	C
4	A	G	C	—	A	C	T	A	G	T	T	A	C

Example

$[3..4]$ is not semi-repeat-free but $[3..5]$ is, so $f(3) = 5$.

Representing the valid segments

The first step is computing the valid segments.

Observation

If $[x..y]$ is valid, then $[x..y']$ is valid for all $y' > y$.

Definition

Given x, the minimal right extension $f(x)$ marks the first column so that $[x..f(x)]$ is valid.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	A	G	C	G	A	C	T	A	G	A	T	A	C
2	A	G	C	–	A	C	T	A	G	–	T	A	G
3	A	G	C	G	A	T	T	A	G	T	T	A	C
4	A	G	C	–	A	C	T	A	G	T	T	A	C

Example

$[3..4]$ is not semi-repeat-free but $[3..5]$ is, so $f(3) = 5$.

Representing the valid segments

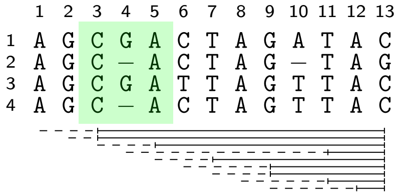
The first step is computing the valid segments.

Observation

If $[x..y]$ is valid, then $[x..y']$ is valid for all $y' > y$.

Definition

Given x, the minimal right extension $f(x)$ marks the first column so that $[x..f(x)]$ is valid.



	1	2	3	4	5	6	7	8	9	10	11	12	13
1	A	G	C	G	A	C	T	A	G	A	T	A	C
2	A	G	C	–	A	C	T	A	G	–	T	A	G
3	A	G	C	G	A	T	T	A	G	T	T	A	C
4	A	G	C	–	A	C	T	A	G	T	T	A	C

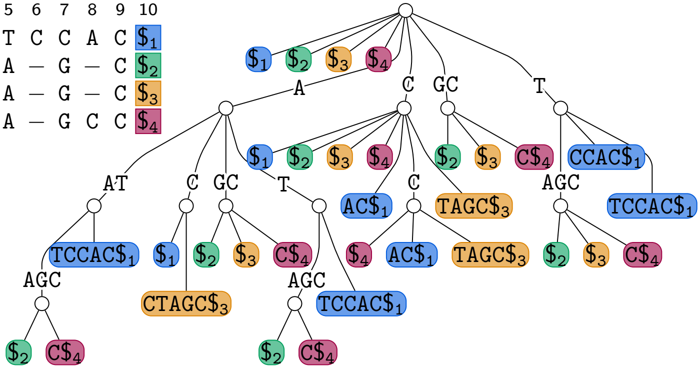
Example

$[3..4]$ is not semi-repeat-free but $[3..5]$ is, so $f(3) = 5$.

The generalized suffix tree

The main tool we use to compute $[x..f(x)]$ is GST_{MSA}, the generalized suffix tree of strings $\text{spell}(\text{MSA}[i, 1..n]) \cdot \$_i$.

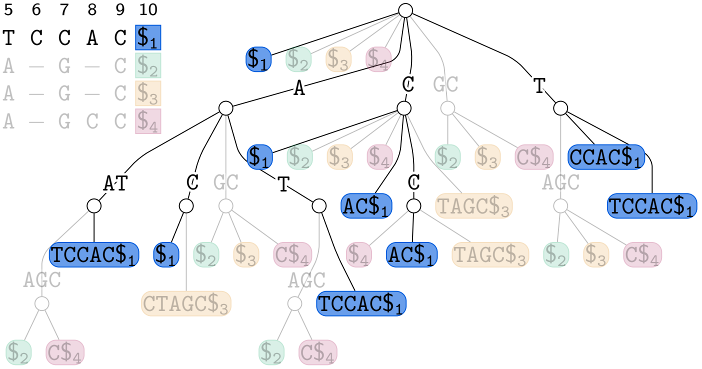
1	2	3	4	5	6	7	8	9	10
A	A	–	T	T	C	C	A	C	$\$_1$
A	A	–	T	A	–	G	–	C	$\$_2$
A	C	C	T	A	–	G	–	C	$\$_3$
A	A	–	T	A	–	G	C	C	$\$_4$



The generalized suffix tree

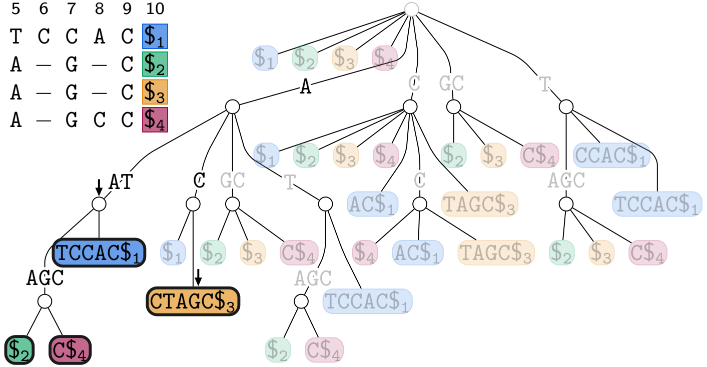
The main tool we use to compute $[x..f(x)]$ is $\mathrm{GST}_{\mathrm{MSA}}$, the generalized suffix tree of strings $\mathrm{spell}(\mathrm{MSA}[i, 1..n]) \cdot \$_i$.

1	2	3	4	5	6	7	8	9	10
A	A	—	T	T	C	C	A	C	$\$_1$
A	A	—	T	A	—	G	—	C	$\$_2$
A	C	C	T	A	—	G	—	C	$\$_3$
A	A	—	T	A	—	G	C	C	$\$_4$



From the MSA to the suffix tree

1	2	3	4	5	6	7	8	9	10
A	A	–	T	T	C	C	A	C	$\$_1$
A	A	–	T	A	–	G	–	C	$\$_2$
A	C	C	T	A	–	G	–	C	$\$_3$
A	A	–	T	A	–	G	C	C	$\$_4$



- we can break down $f(x)$ to single rows
- suffixes $\Rightarrow$ leaves of GST_{MSA} $\Rightarrow$ exclusive ancestors
- we navigate back to the MSA with rank and select queries
- $f(x)$ can be computed in time $O(m)$ $\Rightarrow$ global $O(mn)$ time

Min max height in the gapless setting

Now that we have the valid segments, we need to compute their height information.

Observation

If MSA$[1..m, 1..n]$ has no gaps, height $H([x..y])$ is increasing with respect to y.

Definition

We define the meaningful right extensions R_x as $r_{x,1}, \ldots, r_{x,c_x}$, the positions y where height $H([x..y])$ changes (increases).

	1	2	3	4	5	6	7	8
	T	T	C	C	C	G	G	$\$_1$
	T	T	A	C	C	G	A	$\$_2$
	T	T	A	C	A	C	A	$\$_3$
	T	T	A	C	A	C	G	$\$_4$
	T	T	A	C	A	A	G	$\$_5$
	G	T	C	A	A	G	G	$\$_6$
$H([1..y])$	2	2	3	3	4	5	6	6

Min max height in the gapless setting

Now that we have the valid segments, we need to compute their height information.

Observation

If MSA$[1..m, 1..n]$ has no gaps, height $H([x..y])$ is increasing with respect to y.

Definition

We define the meaningful right extensions R_x as $r_{x,1}, \ldots, r_{x,c_x}$, the positions y where height $H([x..y])$ changes (increases).

In the gapless setting, $|R_x| \leq m$ so $\sum_{x=1}^{n} |R_x| \in O(mn)$:

- **Norri et al (2019)** $O(mn)$-time computation of all R_x w/ height values (in a different context from the semi-repeat-free one)
- **Mäkinen et al (2020)** $O(mn)$-time computation of $f(x)$

Result: $O(mn)$-time segmentation algorithm for gapless MSAs.

The more difficult setting with gaps

In the setting with gaps, $R_x \in O(n)$:

$$\begin{array}{rccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & & n-2 & & n \\
1 & \mathtt{T} & \mathtt{A} & - & \mathtt{A} & - & \mathtt{A} & - & \dots & \mathtt{A} & - & \mathtt{C} \\
2 & \mathtt{T} & - & \mathtt{A} & - & \mathtt{A} & - & \mathtt{A} & & - & \mathtt{A} & \mathtt{C} \\
H([1..y]) & 1 & 2 & 1 & 2 & 1 & 2 & 1 & \cdots & 2 & 1 & 1
\end{array}$$

Thus $\sum_{x=1}^{n} |R_x| \in O(n^2)$:

- computing R_x + height info naïvely (keyword tries) yields a $O(mn\alpha \log|\Sigma|)$-time algorithm, where α is the length of the longest aligned substring between any two rows
- construction algorithm processes all R_x + height info as before

Solution is $O(mn^2 \log|\Sigma|)$: can we do better?

Prefix-aware height

	1	2	3	4	5	6	7	8	9	10
	T	C	—	C	—	—	C	G	—	$\$_1$
	T	—	A	C	—	—	C	—	—	$\$_2$
	T	—	A	C	—	A	C	—	A	$\$_3$
	T	—	A	C	—	A	C	G	—	$\$_4$
	T	—	A	C	A	A	—	G	—	$\$_5$
	G	C	A	—	—	A	—	G	—	$\$_6$
$\overline{H}([1..y])$	-	2	3	3	3	3	5	5	6	6

Definition

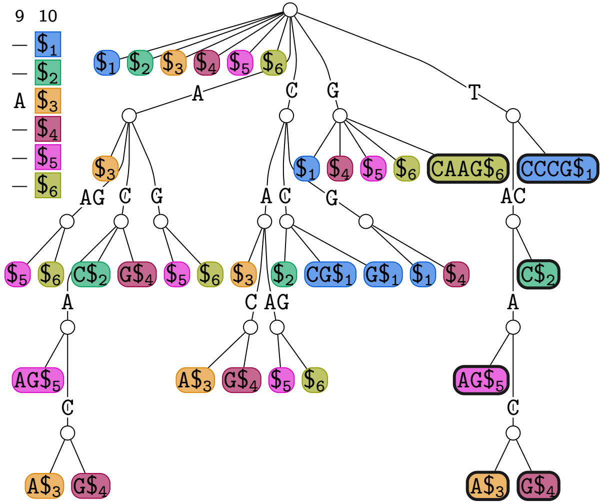
We define $\overline{H}([x..y])$ as the number of distinct strings in $[x..y]$ that are not proper prefixes of other strings in $[x..y]$.

- $\overline{H}([x..y]) \leq H([x..y])$ so it is a lower bound
- we can define the meaningful prefix-aware extensions $\overline{R}_x$ $\Rightarrow \sum_{x=1}^{n} |\overline{R}_x| \in O(mn)$
- plug-and-play with the construction algorithm

The suffix tree uncovers the prefix-aware height

Let's concentrate on the forest of $\mathrm{GST}_{\mathrm{MSA}}$ for $[x..n]$, with $x = 1$

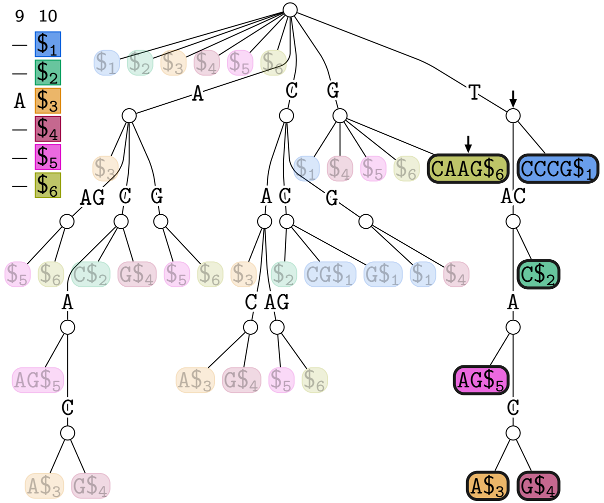
1	2	3	4	5	6	7	8	9	10
T	C	—	C	—	—	C	G	—	$\$_1$
T	—	A	C	—	—	C	—	—	$\$_2$
T	—	A	C	—	A	C	—	A	$\$_3$
T	—	A	C	—	A	C	G	—	$\$_4$
T	—	A	C	A	A	—	G	—	$\$_5$
G	C	A	—	—	A	—	G	—	$\$_6$



The suffix tree uncovers the prefix-aware height

Let's concentrate on the forest of $\mathrm{GST}_{\mathrm{MSA}}$ for $[x..n]$, with $x = 1$

1	2	3	4	5	6	7	8	9	10
T	C	—	C	—	—	C	G	—	$\$_1$
T	—	A	C	—	—	C	—	—	$\$_2$
T	—	A	C	—	A	C	—	A	$\$_3$
T	—	A	C	—	A	C	G	—	$\$_4$
T	—	A	C	A	A	—	G	—	$\$_5$
G	C	A	—	—	A	—	G	—	$\$_6$

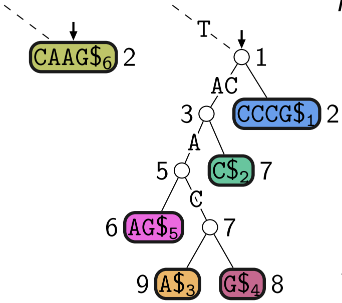


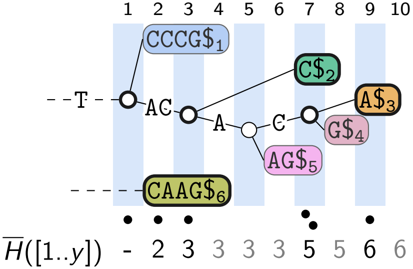
The suffix tree uncovers the prefix-aware height

- for each node we find the first ending column of the relative MSA occurrence

1	2	3	4	5	6	7	8	9	10
T	C	–	C	–	–	C	G	–	$\$_1$
T	–	A	C	–	–	C	–	–	$\$_2$
T	–	A	C	–	A	C	–	A	$\$_3$
T	–	A	C	–	A	C	G	–	$\$_4$
T	–	A	C	A	A	–	G	–	$\$_5$
G	C	A	–	–	A	–	G	–	$\$_6$
-	2	3	3	3	3	5	5	6	6

$\overline{H}([1..y])$





- most of these ending columns mark a +1 increase in $\overline{H}$

Final suffix tree maneuvers

Computing these values using just GST_{MSA} takes $O(m^2 n)$ time in total.

We obtain a linear-time solution with suffix tree maneuvers:

- we can compute the pos values in $O(mn)$ time with GPT_{MSA}, the generalized prefix tree of the MSA rows
- $O(1)$-time GST_{MSA}-to-GPT_{MSA} navigation thanks to weighted ancestor queries/affix trees

Conclusions

Summary of results for optimal EFG construction:

- $O(mn)$-time solution for min max height in the gapless setting
- $O(mn\alpha \log|\Sigma|)$-time solution for the setting with gaps
- $O(mn)$-time solution for min max prefix-aware height

Future work:

- extending EFGs to allow segments containing empty strings

Bibliography

- Tuukka Norri, Bastien Cazaux, Dmitry Kosolobov and Veli Mäkinen. Linear time minimum segmentation enables scalable founder reconstruction. Algorithms for Molecular Biology 14 (2019)
- Veli Mäkinen, Bastien Cazaux, Massimo Equi, Tuukka Norri, and Alexandru I. Tomescu. Linear time construction of indexable founder block graphs. WABI 2020.
- Massimo Equi, Tuukka Norri, Jarno Alanko, Bastien Cazaux, Alexandru I. Tomescu, and Veli Mäkinen. Algorithms and complexity on indexing elastic founder graphs. ISAAC 2021.