Least Random Suffix/Prefix Matches in Output-Sensitive Time

Niko Välimäki

Department of Computer Science
University of Helsinki
nvalimak@cs.helsinki.fi

23rd Annual Symposium on Combinatorial Pattern Matching

Suffix/Prefix Matching Problem

Input: A set of r strings of total length n.

Output: Longest non-zero length suffix/prefix match for each string-pair.

A suffix/prefix match (*overlap*):

```
VÄLIMÄKI
    ||||
   MÄKINEN
```

Motivation

Approximating the shortest common superstring.

Suffix/Prefix Matching Problem

Input: A set of r strings of total length n.

Output: Longest non-zero length suffix/prefix match for each string-pair.

A suffix/prefix match (*overlap*):

```
VÄLIMÄKI
    ||||
   MÄKINEN
```

Motivation

Approximating the shortest common superstring.

Longest Exact Overlaps

Optimal-time by [Gusfield & Landau & Schieber, 1992]

- $O(n + \text{output})$ time, $O(n)$ words,
- where output $\leq r^2$.

Space-efficient variant by [Ohlebusch & Gog, 2010]

- $O(n + \text{output})$ time, $8n$ bytes.

Finding *irreducible* overlaps [Simpson & Durbin, 2010]

- $O(n + \text{output})$ time, $2nH_k + o(n) + r \log r$ bits.

Approximate Overlaps

Output the "best overlap" (of length $\geq t$) s.t.

k-errors: suffix/prefix edit distance is $\leq k$,

ϵ-errors: suffix/prefix edit distance is $\leq \lceil \epsilon \ell \rceil$,
where ℓ is the length of the suffix.

Overlaps for $k = 1$:

```
VÄLIMÄKI      VÄLIMÄKI      VÄLIMÄKI-
    ||||        |||||          |||||
   MÄKINEN     -MÄKINEN       MÄKINEN
```

How to define the *best overlap* when indels are allowed?

Least Random Overlaps

Let $A[1\,..\,a]$ and $B[1\,..\,b]$ denote two random strings from Bernoulli source.

[Kececioglu & Myers, 1995] precomputed table $\Pr_\sigma(l, d)$,

- i.e. the probability that A and B align with d indels and $l = (a + b - d)/2$ matching symbols.
- The best overlap minimizes $\Pr_\sigma(l, d)$.
- $O(\epsilon n^2)$ time, where $\epsilon > 0$ denotes error-rate.

[Landau & Myers & Schmidt, 1998] generalized the likelihood:

- k-errors in $O(k\,|T_j|)$ time for a string-pair T_i and T_j.
- Over all string-pairs in $O(knr)$ time.

Least Random Overlaps

Let $A[1\,..\,a]$ and $B[1\,..\,b]$ denote two random strings from Bernoulli source.

[Kececioglu & Myers, 1995] precomputed table $\Pr_\sigma(l, d)$,

- i.e. the probability that A and B align with d indels and $l = (a + b - d)/2$ matching symbols.
- The best overlap minimizes $\Pr_\sigma(l, d)$.
- $O(\epsilon n^2)$ time, where $\epsilon > 0$ denotes error-rate.

[Landau & Myers & Schmidt, 1998] generalized the likelihood:

- k-errors in $O(k\,|T_j|)$ time for a string-pair T_i and T_j.
- Over all string-pairs in $O(knr)$ time.

In Practice: Sequence Assembly

Biological sequences have sequencing errors, SNPs...

Heuristical methods for *overlap-layout-consensus* assemby:

- ARACHNE [Batzoglou et al. 2002],
- Atlas [Havlak et al. 2004],
- Celera [Myers et al. 2000],
- Phrap [Green, 1994],
- UMD Overlapper [Roberts et al. 2004].

Filter based methods with $\Omega(n^2)$ worst-case:

- q-gram filters [Rasmussen & Stoye & Myers, 2006]
- suffix filters [Välimäki & Ladra & Mäkinen, 2010 & 2012]

Outline of Our Contributions

Method for short strings

- Adapt [Gusfield & Landau & Schieber, 1992] for least random overlaps.

Method for long strings

- Utilizes approximate dictionary matching [Cole et al. 2004],

$$\text{Query time: } O\Bigg(\underbrace{m}_{\text{Prepr.}} + \underbrace{\frac{(c_2 \log r)^k}{k!} \log\log r}_{\text{Time per suffix}} + \text{output} \Bigg)$$

Mixed length strings

- $O((n + \text{output})\,\text{polylog}(n))$ time, $O(n)$ space (for constant k)

Outline of Our Contributions

Method for short strings

- Adapt [Gusfield & Landau & Schieber, 1992] for least random overlaps.

Method for long strings

- Utilizes approximate dictionary matching [Cole et al. 2004],

Query time: $O\Big(\underbrace{m}_{\text{Prepr.}} + \underbrace{\frac{(c_2 \log r)^k}{k!} \log\log r}_{\text{Time per suffix}} + \text{output} \Big)$

Mixed length strings

- $O((n + \text{output})\,\text{polylog}(n))$ time, $O(n)$ space (for constant k)

Outline of Our Contributions

Method for short strings

- Adapt [Gusfield & Landau & Schieber, 1992] for least random overlaps.

Method for long strings

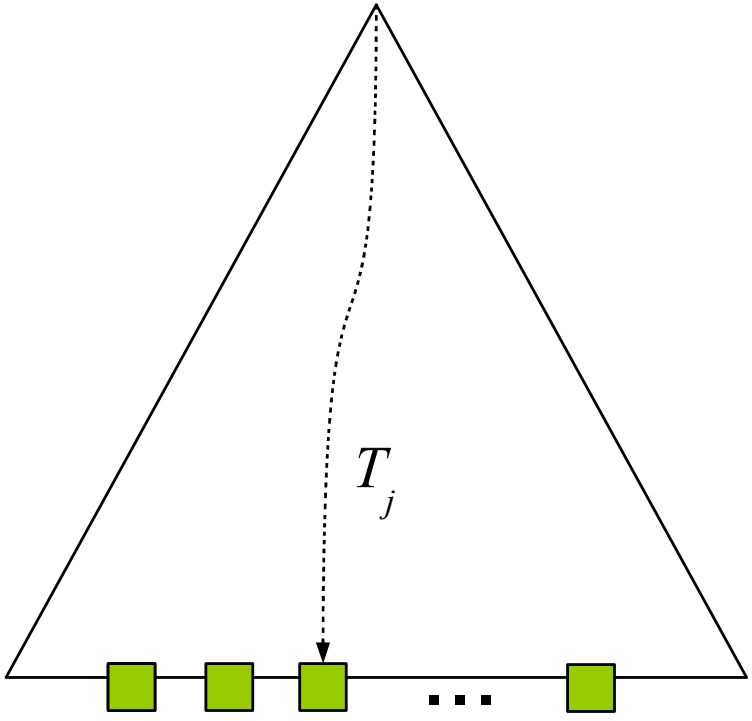
- Utilizes approximate dictionary matching [Cole et al. 2004],

Query time: $O\Big(\underbrace{m}_{\text{Prepr.}} + \underbrace{\frac{(c_2 \log r)^k}{k!} \log\log r}_{\text{Time per suffix}} + \text{output} \Big)$

Mixed length strings

- $O((n + \text{output})\,\text{polylog}(n))$ time, $O(n)$ space (for constant k)

Short Strings: Preprocessing Step



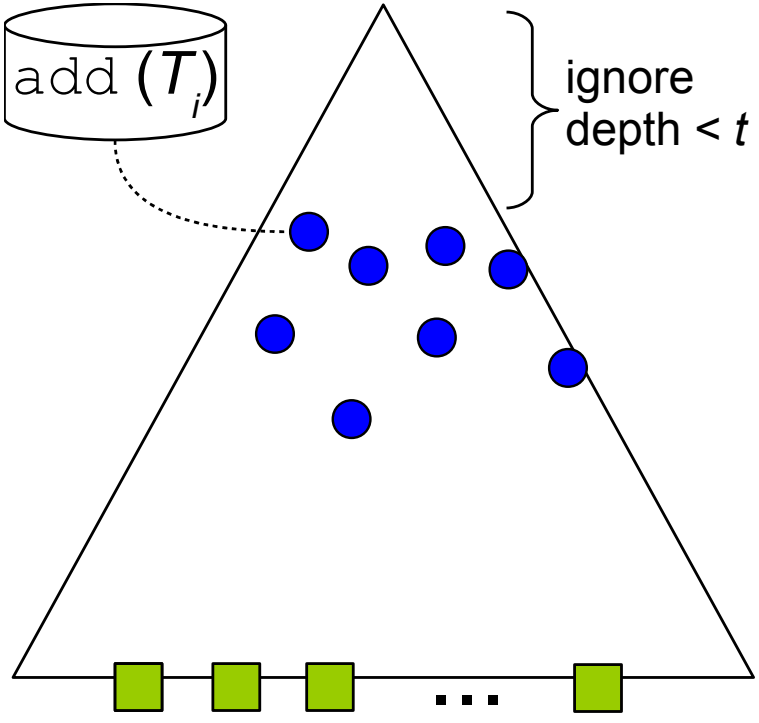
Assume strings of length $\leq \beta$.

1. Build a generalized suffix tree for $T_1, T_2, \ldots, T_r$.

Green leaf nodes:

r leafs, each spelling out whole T_j for each j.

Short Strings: Search Step



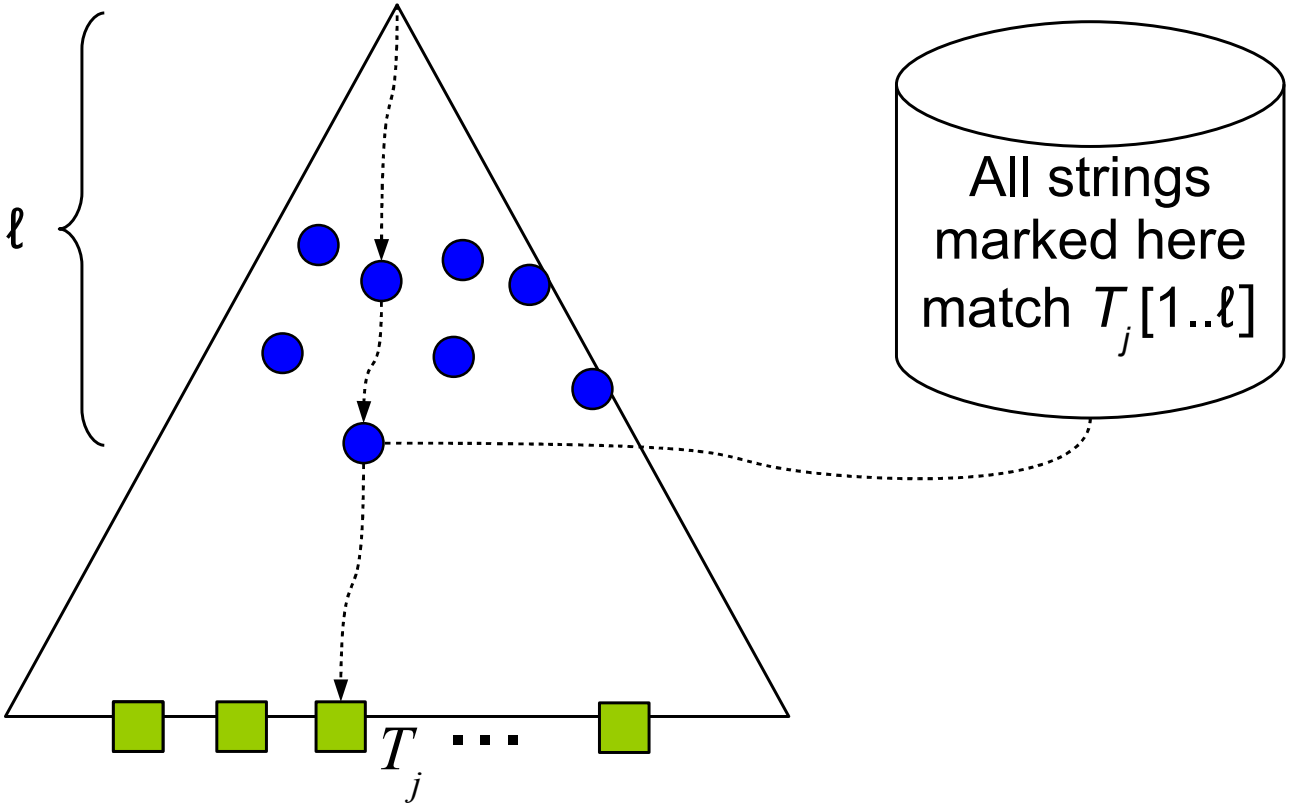
2. Approx. search for each T_i.

   Search in backward manner to cover all suffixes of T_i.

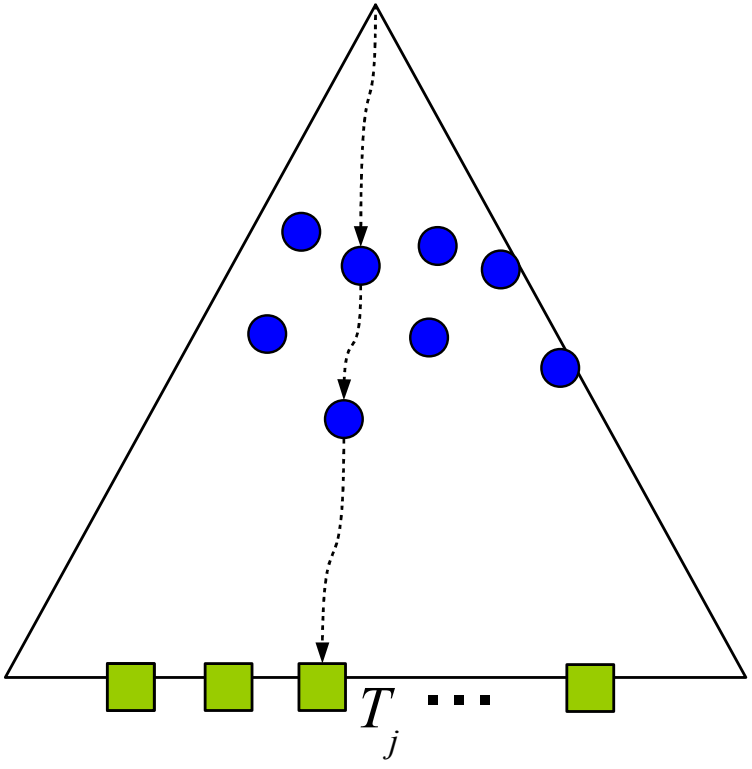
Blue nodes: $O(|T_i|^{k+1}\sigma^k)$ nodes whose upward path is within k-errors of one or more suffixes of T_i.

Searching all strings yields $O(n\beta^k\sigma^k)$ marks.

Short Strings: Search Step



Short Strings: Traversal Step



3. Depth-first traversal.

Use r stacks to collect marks

[Gusfield & Landau & Schieber, 1992]

Blue nodes

Push list items to corresponding stacks.

Green leafs

Output top-most stack-values.

Short Strings: Linear Space

Linear space for marks (in blue nodes):

- Step 2: Search $\lceil n/\beta^{k+1}\sigma^k \rceil$ strings at a time.
- Step 3: Need to repeat the traversal over disjoint sets of marks.

$O(n)$ words, time complexity is retained.

$nH_k(T) + \Theta(n)$ bits, time increases with $(\log n)$-factor.

Summary

"Open problem: longest approximate overlaps"

Summary

Earlier methods:

- $\Omega(r^2)$ time regardless of the output size.
- $O(knr)$ time [Landau & Myers & Schmidt, 1998]

We propose:

- First output-sensitive algorithms for least random overlaps:

$\beta \leq \frac{\log n}{\sigma \sqrt[k]{k}}$	$O(n \log^k n + \text{output})$	$k < \frac{\log n}{\log \log n}$
$\beta \geq \epsilon \log^k r$	$O(\frac{c^k}{k!} nr)$	$k < \frac{\log r}{\log \log r}$
Any β.	$O((n + \text{output}) \operatorname{polylog}(n))$	$k = O(1)$

Kiitos!