Efficient Algorithms

for Finding Submasses in Weighted Strings

Nikhil Bansal — *IBM Watson*

Mark Cieliebak — *ETH Zurich*

Zsuzsanna Lipták — *Bielefeld University*

The Problem

s string over weighted alphabet $\qquad \mu : \Sigma \to \mathbb{N}$, $\mu(s) := \sum_i \mu(s_i)$

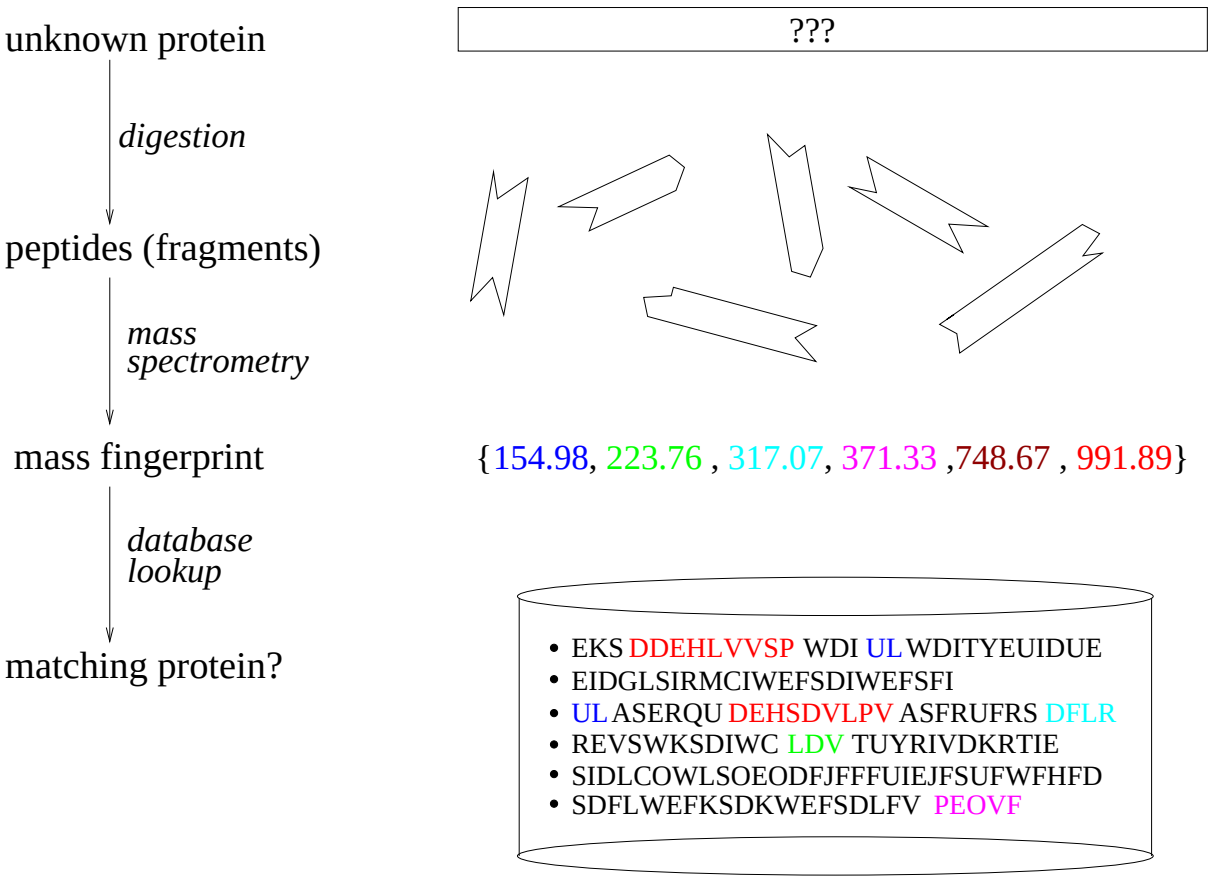
MMSARGDFLNYALSLMRSHN DEHSDVLPV LDVCSLKHVAYVFQALIYWIKAMNQQ...

$\mu = 992$

input: $M_1(=992), M_2, M_3, \ldots, M_k$

- Which M_i are submasses of s? (occurrence)
- For each submass M_i, return a witness. (witness)
- For each submass M_i, return all witnesses. (all-witnesses)

Motivation: Mass Spectrometry (MS)



Two simple algorithms

LINSEARCH: sweep-through (two pointers), no preprocessing, query: $\mathcal{O}(kn)$ time, $\mathcal{O}(1)$ space
(for all three problems)

BINSEARCH: preprocess string, compute and sort all submasses.

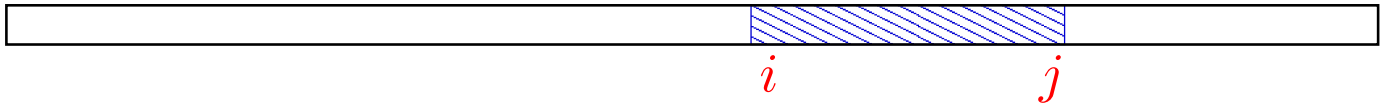
- preprocessing: $\mathcal{O}(n^2 \log n)$ time
- occurrence and witness: $\mathcal{O}(\sigma(s))$ space, $\mathcal{O}(k \log n)$ query time
- all-witnesses: $\mathcal{O}(n^2)$ space, $\mathcal{O}(k \log n + K)$ query time

$\sigma(s) =$ no. submasses of s
$K =$ size of output $(\leq kn)$

Algorithm Polly: Main idea

Define $p_i := \sum_{j=1}^{i} \mu(s_j)$ (i'th prefix mass).

Idea: one submass = one prefix mass - another prefix mass



$$\mu(s_i \ldots s_j) = p_j - p_{i-1}$$

Encoding Submasses with Polynomials

w.l.o.g.: $\mu(s_i) = s_i$

$$P(x) := \sum_{i=1}^{n} x^{p_i} = x^{s_1} + x^{s_1+s_2} + x^{s_1+s_2+s_3} + \ldots$$

$$Q(x) := \sum_{i=1}^{n} x^{-p_{i-1}} = x^0 + x^{-s_1} + x^{-s_1-s_2} + \ldots$$

$$C(x) := P(x) \cdot Q(x) = \sum_m c_m x^m$$

$\mu_s = \sum_i \mu(s_i)$

Note: $\mu_s \leq n \cdot \mu_{\max}$

An Example

$\mu(a) = 2$, $\mu(b) = 3$, $\mu(c) = 5$.

$s = baac$

$P(x) = x^3 + x^5 + x^7 + x^{12}$,
$Q(x) = x^0 + x^{-3} + x^{-5} + x^{-7}$,
$C(x) = x^{-4} + 2x^{-2} + 3x^0 + 2x^2 + x^3 + x^4 + 2x^5 + 2x^7 + x^9 + x^{12}$.

submasses	$2, 3, 4, 5, 7, 9, 12$
2 witnesses	$2, 5$, and 7
1 witness	$3, 4, 9, 12$.

Properties of $C(x)$

for positive exponents m:

$$
\begin{aligned}
c_m \neq 0 \quad &\Leftrightarrow \quad m \text{ is submass of } s, \\
c_m \quad &= \quad \text{no. of witnesses of } m, \\
\sum_{m>0} c_m \quad &= \quad \binom{n+1}{2} = \text{ overall no. of witnesses}, \\
\#\{c_m \neq 0 \,:\, i > 0\} \quad &= \quad \sigma(s) = \text{ no. of submasses of } s.
\end{aligned}
$$

Computing $C(x)$

- naively: $\mathcal{O}(n^2)$
- with Fast Fourier Transform (FFT):

$$\begin{aligned} & \mathcal{O}(\mu_s \log \mu_s), && \text{since } \deg C(x) = 2\mu_s \\ = \mathcal{O}(n\,\mu_{\max} \log(n\,\mu_{\max})), && \mu_{\max} = \text{largest char. mass} \\ = \mathcal{O}(n \log n), && \text{if } \mu_{\max} \text{ constant.} \end{aligned}$$

- with new randomized compression method for sparse convolutions (Cole & Hariharan, 2002):

$$\mathcal{O}(\sigma(s) \log^2 \sigma(s)) \quad \text{expected time.}$$

Performance Comparison

For occurrence problem:

LINSEARCH	$\mathcal{O}(kn)$
BINSEARCH	$\mathcal{O}((n^2 + k) \log n)$
POLLY	$\mathcal{O}(\mu_s \log \mu_s + k \log n)$
If $\mu_{\max}$ constant:	$\mathcal{O}((n + k) \log n)$
POLLY with compression	$\mathcal{O}(\sigma(s) \log^2 \sigma(s) + k \log n)$ expected

Good if $\sigma(s) << n^2$.

Finding a witness

Idea: If we know where a witness ends, then we can find its beginning in $\mathcal{O}(\log n)$ time (binary search amongst prefix masses)

$$R(x) := \sum_{i=1}^{n} i \cdot x^{p_i} = 1x^{s_1} + 2x^{s_1+s_2} + 3x^{s_1+s_2+s_3} + \ldots$$

$$Q(x) := \sum_{i=1}^{n} x^{-p_{i-1}} = x^0 + x^{-s_1} + x^{-s_1-s_2} + \ldots$$

$$F(x) := R(x) \cdot Q(x) = \sum_{m} f_m x^m.$$

For $m > 0$: $c_m = 1 \quad \Rightarrow \quad f_m =$ end of the (sole) witness of m.

Example cont.

$\mu(a) = 2$, $\mu(b) = 3$, $\mu(c) = 5$.

$s = baac$

$R(x) = x^3 + 2x^5 + 3x^7 + 4x^{12}$

$F(x) = x^{-4} + 3x^{-2} + 6x^0 + 5x^2 + x^3 + 3x^4 + 6x^5 + 7x^7 + 4x^9 + 4x^{12}$.

Compare with $C(x)$:

$C(x) = x^{-4} + 2x^{-2} + 3x^0 + 2x^2 + x^3 + x^4 + 2x^5 + 2x^7 + x^9 + x^{12}$.

$\Rightarrow$ the only witnesses of the submasses $3, 4, 9$, and 12 end at positions $1, 3, 4$, and 4, respectively.

Algorithm Polly-Las Vegas

1. Enforce singletons: repeat sufficently often independently
   (a) choose random subsets $I_1, I_2 \subseteq \{1, \ldots, n\}$
   (b) compute P and R using I_1, and Q using I_2
   (c) compute and compare new C and new F
   (d) if $c_m = 1$ then store f_m (= end of witness);
   mark m as successful
2. for each successful query, find beginning of witness
3. for each unsuccessful **submass** query, run LINSEARCH.

Polly-Las Vegas: Analysis

1. Enforce singletons:
   repeat sufficently often independently $\mathcal{O}(\log^2 n)$
   (a) choose random subsets $I_1, I_2 \subseteq \{1, \ldots, n\}$
   (b) compute P and R using I_1, and Q using I_2 $\mathcal{O}(\mu_s \log \mu_s)$
   (c) compute and compare new C and new F $\mathcal{O}(\mu_s \log \mu_s)$
   (d) if $c_m = 1$ then store f_m (= end of witness);
   mark m as successful
2. for each successful query, find beginning of witness $\mathcal{O}(\log n)$
3. for each unsuccessful submass query, run LINSEARCH. $\mathcal{O}(n)$

Polly-Las Vegas: Analysis (2)

Preprocessing: $\mathcal{O}(\mu_s \log^3 \mu_s)$.

Query phase: For appropriately chosen parameters,

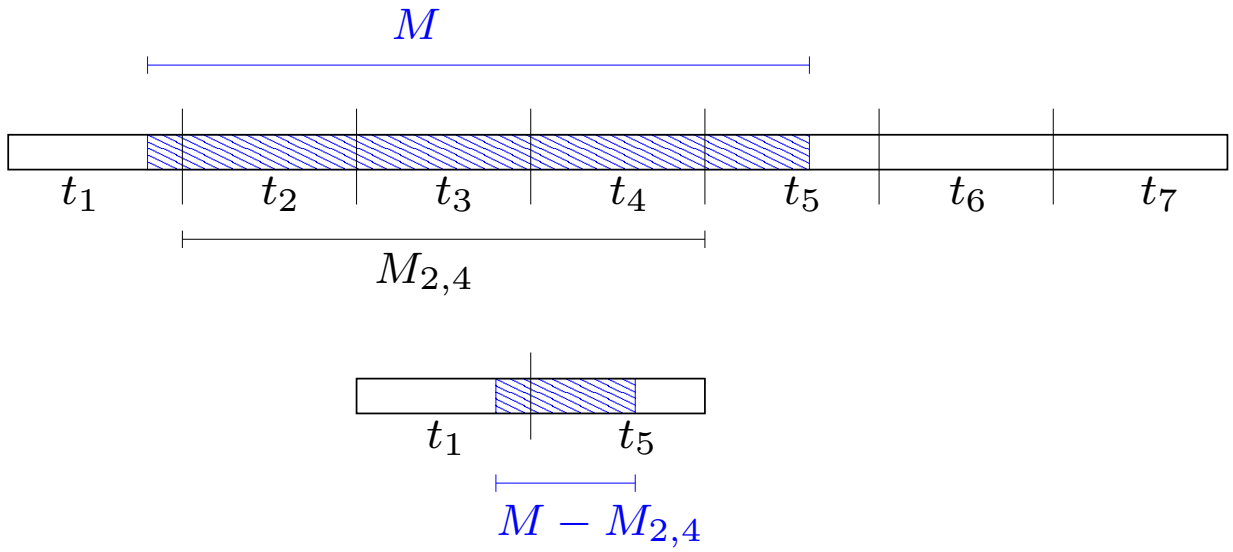
- success probability for given query $= 1 - 1/n^3$,
- success prob. for all queries $= 1 - 1/n$,

$\Rightarrow$ total expected running time $\mathcal{O}(\mu_s \log^3 \mu_s + k \log n)$.

Compare to $\min(kn, n^2 \log n + k \log n)$ with simple algorithms.

A deterministic algorithm for finding all witnesses

Idea: divide s in g parts; find where witnesses begin and end; combine POLLY and LINSEARCH



(border-spanning witness)

Polly-divide: Algorithm and Analysis

1. Compute all C_i and $D_{i,j}$ $\mathcal{O}(g\mu_s \log \mu_s)$
2. For each query mass M_ℓ, make list of i's and (i,j)'s where witnesses begin and end $\mathcal{O}(g\mu_s \log \mu_s)$
3. For each entry in list, run LINSEARCH $\mathcal{O}(K\frac{n}{g})$

$\Rightarrow$ for appropriate choice of g: $\mathcal{O}((Kn\mu_s \log \mu_s)^{1/2})$

Compare to $\min(kn, n^2 \log n + k \log n)$ for simple algorithms.

Conclusion

- Main idea: Encoding submasses with polynomials for occurrence and witnesses. FFT for fast polynomial multipliclation.
- Three algorithms:
  - simple (Polly) for occurrence and submass-counting
  - randomized (Polly-Las Vegas) for witness
  - deterministic (Polly-Divide) for all-witnesses
- Better than simple algorithms when $\sigma(s) << n^2$
- Not yet applicable to mass spec. experiments, but hopefully in the future (esp. DNA mass spec).

Thank you.