CSE 566 Spring 2023

Universal Hitting Set & Bloom Filters

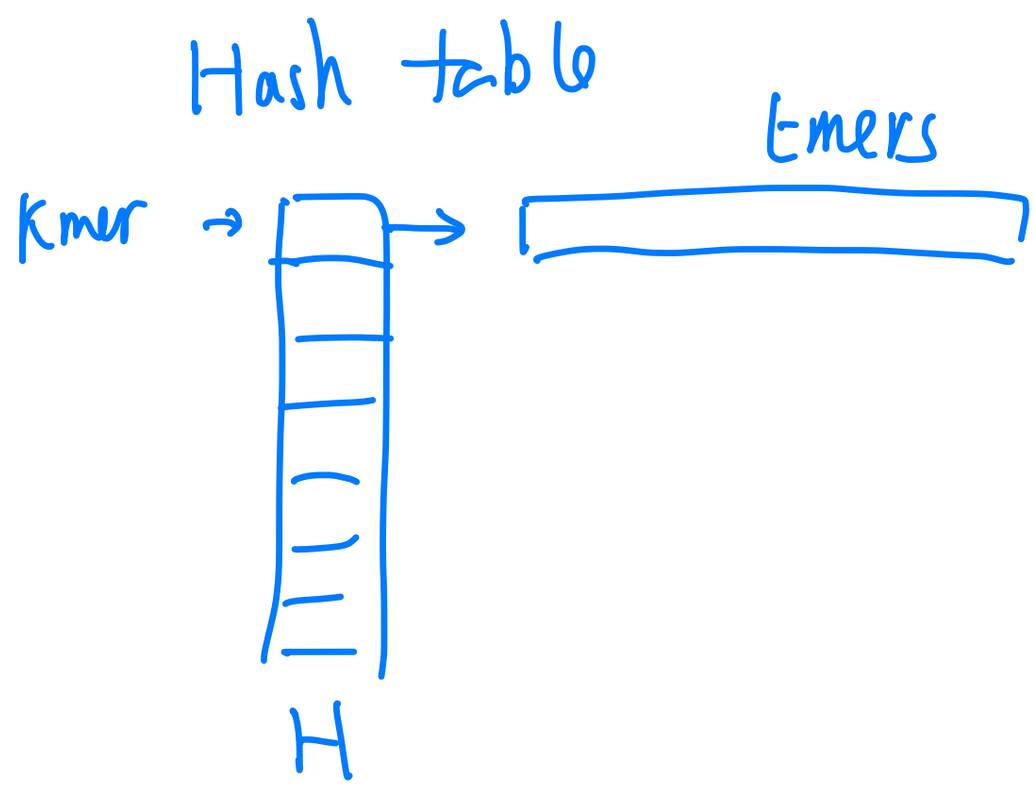
Instructor: Mingfu Shao

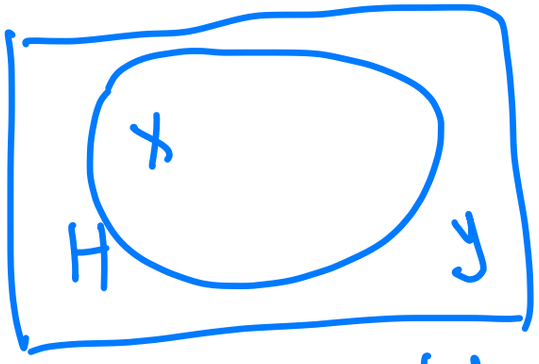
Seeking Orders with Lower Density

- It is desirable to have a low expected density.
- Open questions:
  - What is the order π with the lowest expected density?
  - Is the lower bound of $1.5/(1 + w)$ reachable?
- Current methods to construct orders with lower expected density are all based on universal hitting set.

Universal Hitting Set (UHS)

- A university hitting set is a set of kmers H such that every string of length $L = w + k - 1$ has at least one kmer in H.
- A UHS hits every string of length L.
- UHS has applications such as indexing sequences. of length L





UHS and Minimizers

U: all kmers

- Let H be a UHS. We define an order π is compatible with H if for $x \in H$ and $y \notin H$ we always have $\pi(x) < \pi(y)$.
- Let π be an order that is compatible with a UHS H. Then the Minimizer using π will pick a subset of H.
- Proof:

- Let π be an order that is compatible with a UHS H. Then the expected density of π have an upper bound: $|H|/|\Sigma|^k$.
- Proof:

density = Pr(next kmer gets picked)

$\leq$ Pr(next kmer $\in H$)

$= |H|/|\Sigma|^k$.

S:

Finding UHS of Smaller Size

- UHS is closely related to dBG with vertices being Σ^k
- A string of length $L = w + k - 1$ corresponds to a path of length w in the dBG.
- Find a UHS is equivalent to finding a subset of vertices of dBG that covers every path of length w.

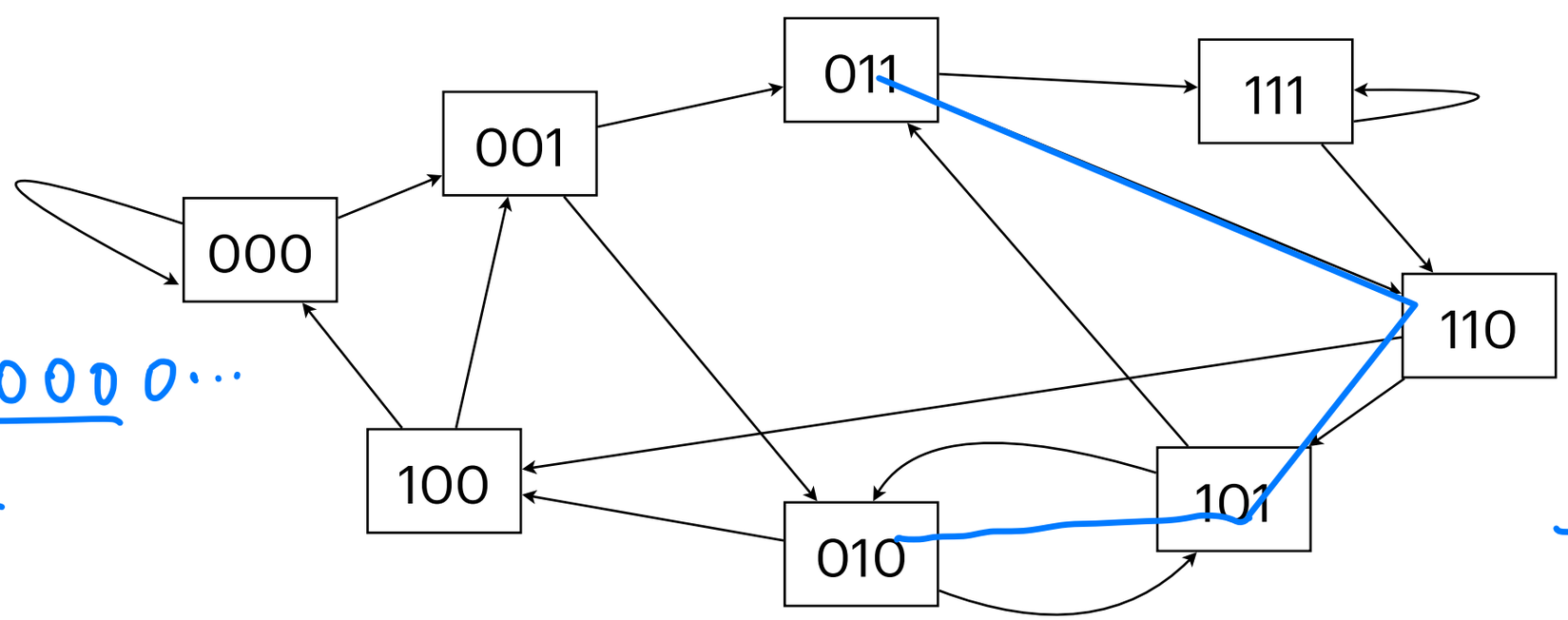
$\Sigma = \{0, 1\}$

$k = 3$

$w = 4$

$L = 6$

011010



00000000...

L

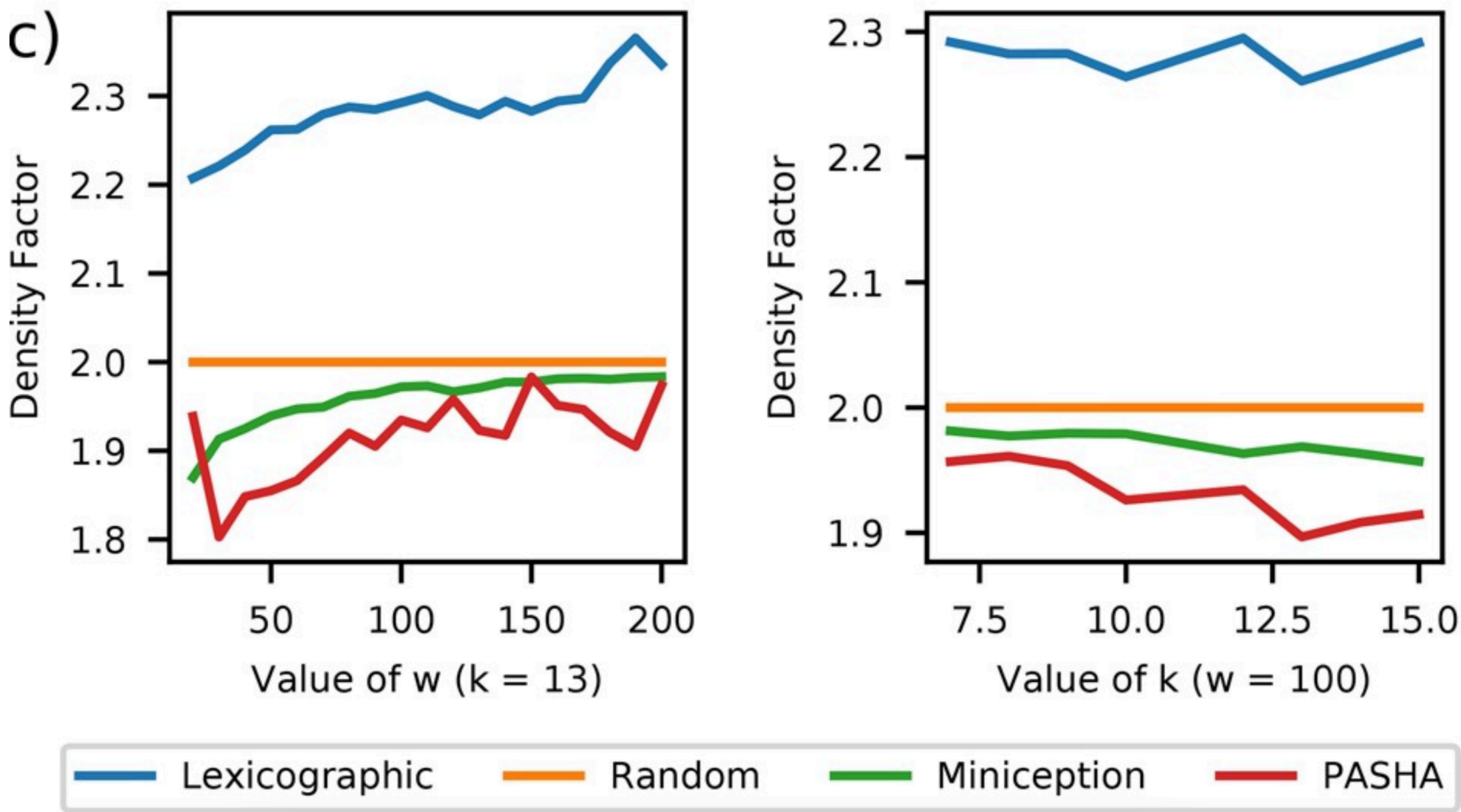
Existing Algorithms to Find UHS

- A greedy approach (DOCKS):
  - To compute vertices that breaks all cycles of dBG;
  - Then to compute vertices that cover all paths of length k. w.
- Existing work:
  - Compact Universal k-mer Hitting Sets (2016). Docks
  - A Randomized Parallel Algorithm for Efficiently Finding Near-Optimal Universal Hitting Sets (2020)
- Remains challenging (scaling only to k = 16).

Minimizers with Lower Density

- Improving the performance of minimizers and winnowing schemes (2017)
- Lower Density Selection Schemes via Small Universal Hitting Sets with Short Remaining Path Length (2020)
- Improved design and analysis of practical minimizers (2020)
- Sequence-specific minimizers via polar sets (2021)
- DeepMinimizer: A Differentiable Framework for Optimizing Sequence-Specific Minimizer Schemes (2022)

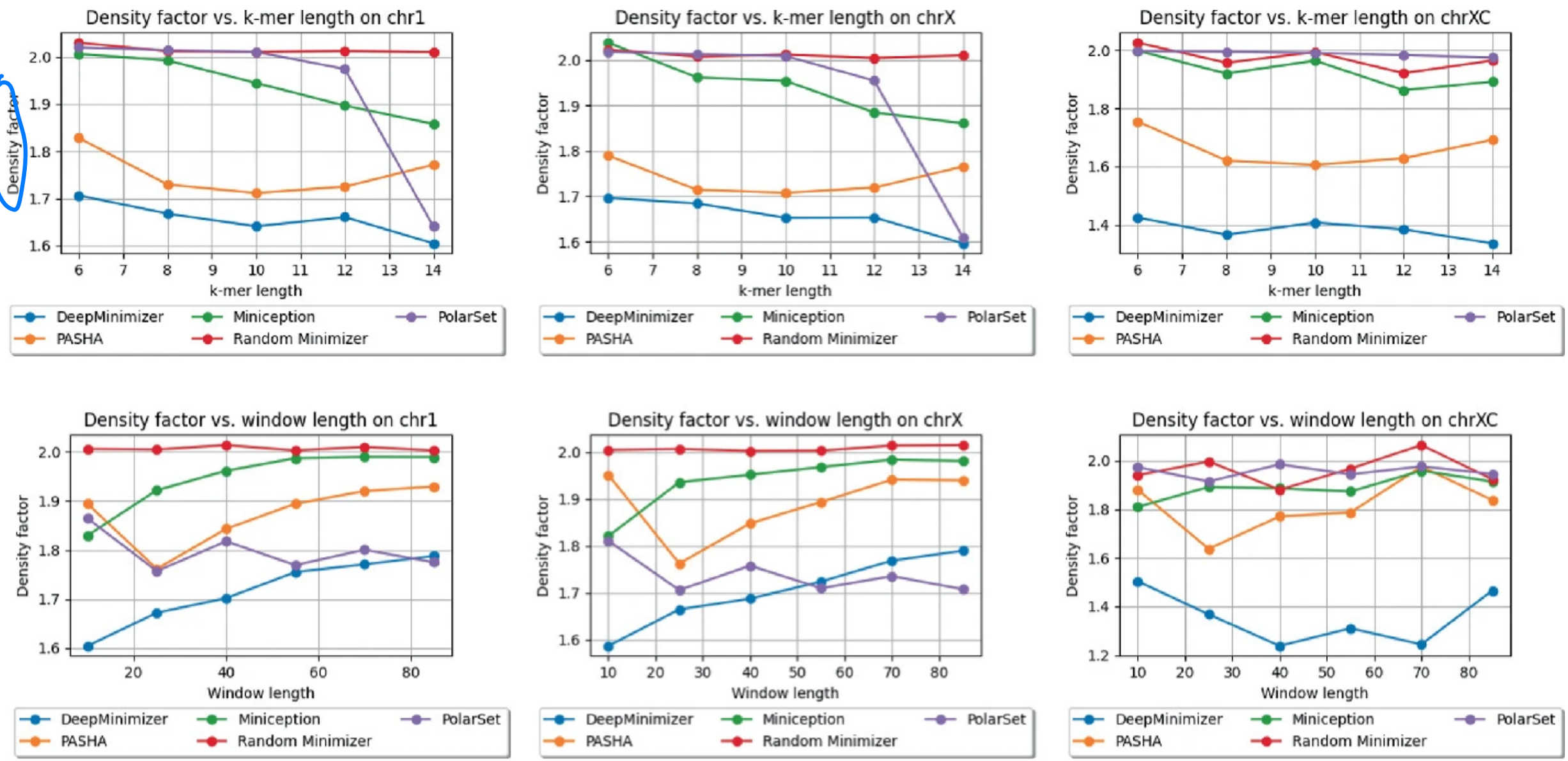
Some Results



Improved design and analysis of practical minimizers (2020)

Data-Dependent Minimizers

$\frac{\sigma}{1+w}$



DeepMinimizer: A Differentiable Framework for Optimizing Sequence-Specific Minimizer Schemes (2022)

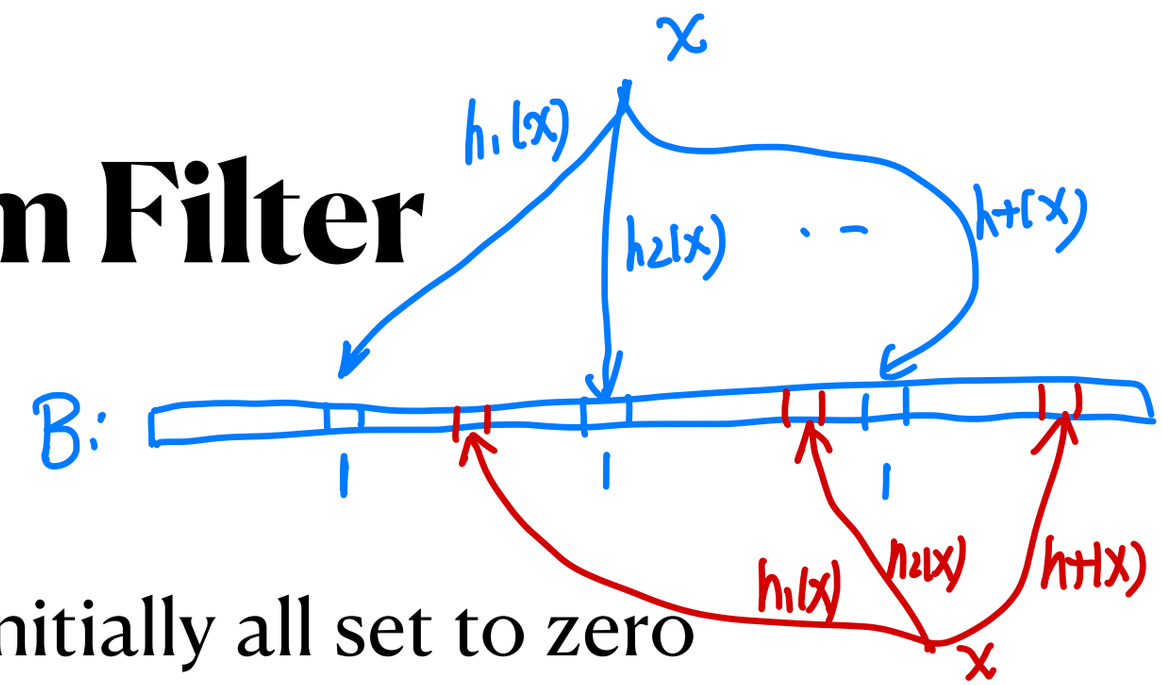
Membership Query

- Store a large set S where memory/space is at a premium
  - All kmers in a large collection of genomic datasets
  - All sentences from a large set of documents
  - All malicious websites/URLs
- A data structure for membership query supports:
  - Add(S, x): add item x to the set S
  - Query(S, x): return true if $x \in S$

Approximate Membership Query

- Two types of errors:
  - False-positive: Query(S, x) returns true when $x \notin S$
  - False-negative: Query(S, x) returns false when $x \in S$
- Both are bad but false-negatives are often more severe.
- Bloom filter is a data structure that avoids false-negatives completely but has false-positives.

Bloom Filter



- A bloom filter consists of
  - An bit-array B of size m, initially all set to zero
  - t hash functions $h_1, h_2, \cdots, h_t$, where each h_i maps $x \in U$ to an index of B, i.e., $\{1,2,\cdots, m\}$.
- Add(S, x): set $B[h_i(x)] = 1$, for all $i = 1,2,\cdots, t$
- Query(S, x): return true if $B[h_i(x)] = 1$ for all $i = 1,2,\cdots, t$; otherwise return false.
- A bloom filter allows for querying items without actually storing them!

Minimizing False Positives

- By choosing proper t: t should not be too small or too large
- **Theorem**: the probability of false positive is minimized when $t = (m/n)\ln 2$, where n is the number of items added.
- **Proof**: assume that the t hash functions map x to t positions randomly and independently.

B: 

- After a single item is added, the probability that a particular bit remains zero is: $Pr = (1 - 1/m)^t$
- After adding all n items, the probability that a particular bit is set to 1 is: $Pr = 1 - (1 - 1/m)^{t \cdot n}$

Proof (continued)

- We get a false-positive when all the t bits for a given element x, i.e., $h_1(x), \cdots, h_t(x)$, are set to 1; the probability is:

$$\Pr(FP) = \Pr(h_1(x)=1, h_2(x)=1, \cdots, h_t(x)=1)$$

$$= \left(1-\left(1-1/m\right)^{tn}\right)^t$$

$$\boxed{e = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n}$$

$$= \left(1-\boxed{\left(1-1/m\right)^m}^{\frac{tn}{m}}\right)^t \approx \left(1-e^{-tn/m}\right)^t.$$

$$\ln \Pr(FP) = t\cdot \ln\left(1-e^{-tn/m}\right).$$

Proof (continued)

$$\frac{d \ln \Pr(FP)}{dt} = 1 \cdot \ln\left(1 - e^{-tn/m}\right) + t \cdot \frac{e^{-tn/m} \cdot \frac{n}{m}}{1 - e^{-tn/m}}$$

let $\frac{d \ln \Pr}{dt} = 0 \Rightarrow \underline{\ln\left(1 - e^{-tn/m}\right) - e^{-tn/m} \cdot \ln\left(1 - e^{-tn/m}\right)}$

$$= -e^{-tn/m} \cdot \frac{nt}{m} = \underline{e^{-tn/m} \cdot \ln e^{-\frac{nt}{m}}}$$

$$\Rightarrow \ln\left(1 - e^{-tn/m}\right) \cdot \underline{\underline{\left(1 - e^{-tn/m}\right)}} = \ln e^{-\frac{nt}{m}} \cdot \underline{\underline{e^{-tn/m}}}$$

$$1 - e^{-tn/m} = e^{-tn/m} \Rightarrow t = \frac{m}{n} \ln 2 .$$