Designing small universal k-mer hitting sets for improved analysis of high throughput sequencing

Orenstein Y, Pellow D, Marçais G, Shamir R, Kingsford C

PLOS Computational Biology. 2017 October; 13(10): e1005777

Hung-Yu Chen, R06945024
Vincent Hwang, B05902122
Outline

- Background
- Methods and results
- Conclusion
· Sequencing datasets are larger and larger.
· New computational ideas are essential to manage and analyze data.
Minimizer

- Given a sequence of length $L$, the minimizer is the lexicographically smallest $k$-mer in it.
- Given a sequence $S$ of any length, the minimizer set is the set of minimizers of every $L$-long subsequence in $S$.

\[\implies\] Every $L$-long subsequence in $S$ is represented in the set.
Application of Minimizers

- Hashing for read overlapping
- Sparse suffix arrays
- Bloom filters to speed up sequence search
$L = 6, \ k = 3$

**R1**: CATCGACA

minimizers: ATC, ACA

**R2**: ACTCGACA

minimizers: ACT, CGA, ACA

**R3**: GAGCTTGC

minimizers: AGC, CTT

---

Table:

<table>
<thead>
<tr>
<th></th>
<th>AAA</th>
<th>AAC</th>
<th>...</th>
<th>ACA</th>
<th>...</th>
<th>ACT</th>
<th>AGC</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Test R1 and R2.
Sparse suffix arrays

<table>
<thead>
<tr>
<th>Index</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AGTCGACT</td>
</tr>
<tr>
<td>2</td>
<td>GTCGACT</td>
</tr>
<tr>
<td>3</td>
<td>TCGACT</td>
</tr>
<tr>
<td>4</td>
<td>CGACT</td>
</tr>
<tr>
<td>5</td>
<td>GACT</td>
</tr>
<tr>
<td>6</td>
<td>ACT</td>
</tr>
<tr>
<td>7</td>
<td>CT</td>
</tr>
<tr>
<td>8</td>
<td>T</td>
</tr>
</tbody>
</table>

**Suffix Array**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>6</td>
</tr>
<tr>
<td>AGTCGACT</td>
<td>1</td>
</tr>
<tr>
<td>CGACT</td>
<td>4</td>
</tr>
<tr>
<td>CT</td>
<td>7</td>
</tr>
<tr>
<td>GACT</td>
<td>5</td>
</tr>
<tr>
<td>GTCGACT</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>8</td>
</tr>
<tr>
<td>TCGACT</td>
<td>3</td>
</tr>
</tbody>
</table>

**Sparse Suffix Array**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>6</td>
</tr>
<tr>
<td>CGACT</td>
<td>4</td>
</tr>
<tr>
<td>GTCGACT</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>8</td>
</tr>
</tbody>
</table>

To query a string $q$, perform at most $s$ queries starting from indices 0, ..., $s-1$ in $q$.

$s = 2$

To query a string $q$, find $q$'s minimizers and search strings starting with these minimizers.

When $L = 6$, $k = 3$, minimizers: AGT, CGA, ACT
Bloom filter

- A bit array.
- A constant number of different hash functions are defined to map elements to the array.
- Supports two operations: “storing an element in the set” and “checking if an element is in the set.”
- Can generate false positives during querying.

![Bloom filter diagram](image)
Universal hitting set (UHS)

- For integers $k$, $L$, a set $U_{k,L}$ is called a UHS of $k$-mers if every possible sequence of length $L$ must contain at least one $k$-mer in $U_{k,L}$.
- For example, the set of all $k$-mers is a trivial UHS.
- **Problem 1.** Given $k$ and $L$, find a smallest UHS of $k$-mers.
A $k$-mer $w$ hits string $S$, denoted $w \subseteq S$, if $w$ is a substring in $S$.

$k$-mer set $X$ hits string $S$ if there exists $w \in X$ such that $w \subseteq S$.

The UHS in Problem 1 is a set of $k$-mers $U_{k,L}$ which hits every possible sequence of length $L$. 

Advantages of UHS over minimizers

- The set of minimizers may be as large as the complete set of $k$-mers. The method in this paper can often generate UHSs smaller by a factor of nearly $k$.
- UHS is universal.
  - For any $k$ and $L$, a UHS needs to be computed only once for every dataset.
  - The data structures created for different datasets will contain a comparable set of $k$-mers.
Problem 2. Given a complete de Bruijn graph $D_k$ of order $k$ and an integer $L$, find a smallest set of vertices $U_{k,L}$ such that any path in $D_k$ of length $l = L - k$ passes through at least one vertex of $U_{k,L}$.
A complete de Bruijn graph of order $k$ over alphabet $\Sigma$: 

- $V$: $|\Sigma|^k$ vertices, each labelled with a unique $k$-mer.
- $E$: If there is an edge $(u, v)$ with a $(k + 1)$-mer label $l$, then the label of vertex $u$ is the $k$-suffix of $l$ and the label of vertex $v$ is the $k$-prefix of $l$. A complete de Bruijn graph contains all possible $|\Sigma|^{k+1}$ edges of this type.

A complete de Bruijn graph of order $3$ over alphabet $\{0, 1\}$

$= B(2, 4)$

A complete de Bruijn graph of order $k$ over alphabet $\Sigma$

$= B(|\Sigma|, k+1)$

Image from Genome Reconstruction by Phillip E. C. Compeau and Pavel A. Pevzner
How to find the UHS?

- NP-hard in general (supporting information in the paper).
- Heuristic approaches. (DOCKS, DOCKSany, DOCKSanyX)
How to find UHS?

1. Generate a complete de Bruijn graph of order \( k \), set \( l = L - k \).
2. Find the decycling vertex set (V set), \( X \).
3. Remove \( X \) from the graph, result in \( G' \).
4. Remove vertices from \( G' \) and add them to \( S \) to hit the remained \( L \) length sequences.
   (i) DOCKS
   (ii) DOCKSany
   (iii) DOCKSanyX
5. \( X \) is the universal hitting set we’re searching for.
Decycling de Bruijn graph

- Vertices labeling
- Factor
- Pure cycling register($PCR_k$)
- V-set
Decycling de Bruijn graph

Designing small universal k-mer hitting sets for improved analysis of high throughput sequencing Orenstein Y, Pellow D, Marçais G, Shamir R, Kingsford C

PLoS Computational Biology. 2017 October; 13(10): e1005777
Vertices labeling

For a vertex $v(s_0, s_1, \ldots, s_{k-1})$, calculate the center of mass. According to the center of mass position in the coordinate system, label the vertex $I$ if $x = 0$, $L$ if $x < 0$, $R$ if $x > 0$,
Vertex labeling example

\[ v = 010111, \text{ the center of mass' } x \text{ value } > 0. \implies R. \]
A factor is a set of cycles such that all vertices in the graph are in exactly one of the cycles. Each cycle has a unique feedback function $f(s_0, s_1, \ldots, s_{k-1}) = s_k$. 
Pure cycling register($PCR_k$)

- $PCR_k$ is a factor.
- Each cycle has a unique function $f(s_0, s_1, \ldots, s_{k-1}) = s_k = s_0$, that is, for every arc $<u, v>$, $u = (s_0, s_1, \ldots, s_{k-1}) \implies v = (s_1, s_2, \ldots, s_k) = (s_1, s_2, \ldots, s_0)$.
- The number of cycles in $PCR_k$ is $Z(k)$, which converges to $\frac{|\Sigma|^k}{k}$.
- It is proved that any circle in the $PCR_k$ must be either all $I$’s or a block of $L$’s and a block of $R$’s separated by at most two $I$’s.
$PCR_k$ example
Factor but not $PCR_k$ example
Why $PCR_k$?

Lemmas tell us:

- All cycles are in the form of all $I$’s or at least a $L$ and a $R$.
- Cycles with all $I$’s are in $PCR_k$.
- For each cycle with at least a $L$ and a $R$, there exist exactly one cycle in $PCR_k$ such that the first vertex of $L$ block of the two cycles are the same one.

$\implies$ We only need to deal with cycles in $PCR_k$. 

A minimum set of vertices which when removed leaves a graph with no cycles.
V-set

Naïve algorithm:
1. Choose a vertex \( v \), find the cycle belongs to \( PCR_k \) that contains \( v \).
2. Choose a certain vertex \( u \) and add it to the V-set:
   - Arbitrary one, if the cycle is all \( l \)'s.
   - The first vertex in the \( L \) block, otherwise.
3. Remove the cycle from the graph.
4. Repeat until all cycles belong to \( PCR_k \) are tested.
V-set example

V-set example

000  010  111  110
Time complexity analysis

There are $Z(k)$ iterations. Find the vertex to be added with $O(k)$ time cost in every iteration.

$$\implies O(kZ(k)) = O(|\Sigma|^k) \text{ in total.}$$
How to find Minimum UHS?

1. Generate a complete de Bruijn graph of order $k$, set $l = L - k$.
2. Find the decycling vertex set (V set), $X$.
3. Remove $X$ from the graph, result in $G'$.
4. Remove vertices from $G'$ and add them to $S$ to hit the remained $L$ length sequences.
   (i) DOCKS
   (ii) DOCKSany
   (iii) DOCKSanyX
5. $X$ is the universal hitting set we’re searching for.
Define:

\[ D(v, i) = \text{the number of } i\text{-long paths starting at } v \]
\[ F(v, i) = \text{the number of } i\text{-long paths ending at } v \]

\[ T(v, l) = \text{the number of } l\text{-long paths through } v \]
\[ = \sum_{i=0}^{l} F(v, i) \cdot D(v, l - i) \]

- Calculate \( D(-, -) \), \( F(-, -) \) to find \( T(-, l) \).
- Choose the one has the largest \( T(-, l) \) and extract it.
- Repeat until no such vertex \((p \text{ iterations})\).
- \( O((1 + p)|\Sigma|^{k+1} \cdot l) \)
DOCKS performance (set size)

Fig A

A

GreedyL / DOCKS

set size

k

5

6

7

8

9

10

L - sequence length

20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200
DOCKS performance (runtime)
DOCKS performance (memory)

Define:

\[ D(v) = \text{the number of paths start at } v \]
\[ F(v) = \text{the number of paths end at } v \]

\[ T(v) = \text{the number of paths through } v \]

\[ = F(v) \cdot D(v) \]

\[ T(v) = \text{the number of paths through } v \]

- Calculate \( D(\_), F(\_ \_ \) to find \( T(\_ \_ \) .
- Choose the one has the largest \( T(\_ \_ \) and extract it.
- Repeat until no paths of length \( l \) (\( p \) iterations).
- \( O((1 + p)|\Sigma|^{k+1}) \)
DOCKSany performance (set size)

Fig C
DOCKSany performance (runtime)

Designing small universal k-mer hitting sets for improved analysis of high throughput sequencing Orenstein Y, Pellow D, Marçais G, Shamir R, Kingsford C

PLoS Computational Biology. 2017 October; 13(10): e1005777
DOCKSany performance (memory)
Same calculation as DOCKSany.
Extract at most $x$ such vertices instead of just one.
DOCKSanyX performance (set size)

Fig D

A

L - sequence length

DOCKSanyX / DOCKS
set size

的设计和优化分析方法为高通量测序提供改进。

DOCKSanyX performance (runtime)
DOCKSanyX performance (memory)
DOCKS can generate compact sets of $k$-mers that hit all $L$-long sequences for any $k \leq 13$ and $L$.

These compact sets can improve many of the applications that currently use minimizers.