PERM: EFFICIENT MAPPING OF SHORT SEQUENCING READS WITH PERIODIC FULL SENSITIVE SPACED SEEDS

Yangho Chen, Tade Souaiaia and Ting Chen

presenters:
蔡誠軒 黃子容
王柏易 蔡博倫
翁健庭 何恩 王舜玄
OUTLINE

- Introduction
- Methods & algorithm
- Results
- Discussion
INTRODUCTION

R00922053 黃子容
R00922005 蔡誠軒
INTRODUCTION

- Definition of the Nouns
- Current Technologies
- Contribution of PerM
Full sensitive to 'k' mismatches

- If $k = 2$, and each read has size = 10.
- For each alignment as above, we check the following:
Full sensitive to 'k' mismatches (cont.)

- For each "two mismatches" case in this alignment (two because $k = 2$).
Full sensitive to 'k' mismatches (cont.)

If this two mismatches can be cover by at least one read, such that all other symbols in this read are matches, ...
Full sensitive to 'k' mismatches (cont.)

The system must return at least one "hit" for this "two mismatches" case.
Full sensitive to 'k' mismatches (cont.)

- If a system supports full sensitive to 'k' mismatches, it supports full sensitive to 'm' mismatches for all the $m < k$ as well.

- There may also be hits for mismatches greater than $k$, but it's not guaranteed.
INTRODUCTION

Target - 1

• We want to design system that supports full sensitivity.
INTRODUCTION

BLAST

• Suitable for long reads.
• Shortcomings:
  ○ Can't support full sensitive to larger 'k'.
  ○ Inefficient for large amounts of short reads.

• Since many datasets produce short reads and require full sensitive to at least three mismatches, the solution need to be improved.
INTRODUCTION

Target - 2

- We want to support full sensitive to 'k' mismatches for larger 'k'.
Introducing "seeds"

- Method used by ELAND, MAQ, SOAP, Corona Lite, and SOCS...

- A "seed" is a set of positions within a window that must be matches to produce a hit.

- Advantage: Support full sensitive to more than three mismatches.
This method is displayed in Figure 1. The read is divided into four substrings. PatternHunter suggested 111*1**1*1**11*111 as a good spaced care' positions, annotated as '1's and '*'s, respectively. For example, the conventional consecutive seeds used by many other programs. For any full sensitive seed family, there exists two performance criteria, the conventional consecutive seeds used by many other programs.

For any alignment within two mismatches, at least one of six pairs of substrings will match exactly. This method requires three equal size fragments, resulting in the hashing of six substring pairs used by ELAND, MAQ (Li et al., 2008c) chooses mismatches. Then chosen for the reference with matches in the different combinations of each read into three index tables.

Biosystems, 2008) chooses for filtering: only when a read and a genomic substring share the same subsequence will a 'hit' be declared and further examination to form a fixed length subsequence which can be used to extract about 88% and 67% partial sensitivity for four and five mismatches a free color error. Periodic seeds allow PerM to maximize efficiency.

For large genomes and datasets, these 'random hits' will often be probability of subsequences matching by chance becomes greater. be carried out to determine the actual similarity level. When the same subsequence will a 'hit' be declared and further examination for any alignment within two mismatches, at least one of six pairs of substrings will match exactly. This method requires three equal size fragments, resulting in the hashing of six substring pairs used by ELAND, MAQ (Li et al., 2008c) chooses mismatches. Then chosen for the reference with matches in the different combinations of each read into three index tables.

Conventional Read Mapping Seeds

32bp Read:

- ACGTACGT CCCCTTTT ACGTACGT AAAAGGGG
- ACGTACGT CCCCTTTT
- ACGTACGT CCCCTTTT ACGTACGT
- ACGTACGT AAAAGGGG

Lookup Table 1 (3 cases):

- ACGTACGT CCCCTTTT
- ACGTACGT
- ACGTACGT AAAAGGGG

Lookup Table 2 (2 cases):

- ACGTACGT
- ACGTACGT
- ACGTACGT AAAAGGGG

Lookup Table 3 (1 case):

- ACGTACGT
- ACGTACGT AAAAGGGG

In Section 2, we describe how PerM's periodic seeds allow to repeating pattern (seven in the example) is equal to the number of repeating the pattern (111*1**).
Introducing "seeds" (cont.)

- The above example uses three kinds of seeds to ensure full sensitive to two mismatches.

- Shortcomings:
  - There are many duplicated hits.
  - Large scale of spaces are required.
Introducing "spaced seeds" (1/2)

- Used by PatternHunter.
- Change the pattern of seed into a set of "care (1)" and "don't care (*)" positions.
- The number of "cares" in a seed is the "weight" of this seed.
- For example, '1*11*1*11*1' has weight 7.
Introducing "spaced seeds" (2/2)

- Pros: More sensitive than consecutive seeds.

- Cons: When the requirement of full sensitive mismatches (value of 'k') increase, the number of seeds and look-up tables also increase.
What does PerM improve?

- Use a **single seed** to achieve full sensitive to 'k' mismatches.

- The seed is **weight-maximized**, which means that it can satisfy full sensitivity and maximize the number of matches in each hit. Hence, it can reduce the number of duplicated hits.
What does PerM improve? (cont.)

• Smaller data structure
  ○ only 4.5 bytes per base

• Mapping sensitivity
  ○ up to three mismatches with weight maximized periodic seed

• Mapping efficiency
  ○ allowing entire genomes to be loaded to memory
  ○ multiple processors
OUTLINE

- Introduction
- Methods & algorithm
- Results
- Discussion
METHODS & ALGORITHM

R00922001 王柏易
R00922153 蔡博倫
Seed Notation

$C_k$: the conventional seed family which divides reads into $k+2$ fragments (used in ELAND, MAQ and SOAP) to provide full sensitivity to $k$ mismatches.

$F_k$: the maximum-weight periodic spaced seed family which is full sensitive to $k$ mismatches.

$S_{x,k}$: the special weight maximized periodic seed family for mapping SOLiD reads, full sensitive to $x$ SNP candidates (consecutive mismatches) and $k$ free mismatches.
Periodic Spaced Seed Design

$111*1**$

$111*1** 111*1** 111*1** 111*1**$
Periodic Spaced Seed Design (cont.)

ACGTACGTCCCCCTTTTTACGTACGTAAAAAGGGG
ACG*A**TCC*C**TTA*G**CGT*A*****
*CGT*C**CCC*T**TTA*G**CGT*A*****
**GTA*G**CCC*T**ACG*A**TAA*A****
***TAC*T**CCT*T**CGT*C**AAA*G***
****ACG*C**CTT*T**GTA*G**AAA*G**
*****CGT*C**TTT*A**TAC*T**AAG*G*
******GTC*C**TTT*C**ACG*A**AGG*G
Periodic Spaced Seed Design (cont.)

Seed: 111*1**111*1**111*1**111*1

Read: ACGTACGTCCCCCTTTTTACGTACGTAA
      AAGGGG

<table>
<thead>
<tr>
<th>.</th>
<th>.</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>
### Periodic Spaced Seed Design (cont.)

**Seed:** 111*1**1**111*1**1**111*1**1**111*1

**Read:** ACGTACGTCCCCCTTTTACGTACGTAA

AAGGGG

```
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
```

$W=16$
Periodic Spaced Seed Design (cont.)

Seed: \[111*1**111*1**111*1**111*1\] \(W=16\)

Read: ACGTACGTCCCTTTTACGTACGTAA
      AAGGGG
Periodic Spaced Seed Design (cont.)

Seed: 111*1**111*1**111*1**111*1 W=16

Read: ACGTACGTCCCCCTTTTACGTACGTAA
      AAGGGG

<table>
<thead>
<tr>
<th>.</th>
<th></th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td></td>
<td>.</td>
</tr>
</tbody>
</table>
### Periodic Spaced Seed Design (cont.)

**Seed:** 111*1**111*1**111*1**111*1**111*1  \[ W = 16 \]

**Read:** ACGTACGTCCCCTTTTACGTACGTAA

AAGGGG

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACGATCCCTTAGCGTA</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

*Note: The table shows the periodic spaced seed design with the given seed and read.*
Periodic Spaced Seed Design (cont.)

Seed: $111*1**111*1**111*1**111*1$  \hspace{1cm} W=16

Read: ACGTACGTCCCCTTTTACGTACGTAA
      AAGGGG

\[
\begin{array}{cccc}
  \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot \\
  \text{ACGATCCCTTAGCGTA} & 1 & \\
  \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot \\
\end{array}
\]
### Periodic Spaced Seed Design (cont.)

**Seed:** \(111^*1**111^*1**111^*1**111^*1**111^*1\) \(W=16\)

**Read:** ACGTACGTCCCCCTTTTACGTACGTAA

AAGGGG

<table>
<thead>
<tr>
<th>ACGTACGTCCCCCTTTTACGTACGTAA</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGTCCCCCTTTACTGTAA</td>
<td>2</td>
</tr>
</tbody>
</table>
Periodic Spaced Seed Design (cont.)

1234567890123456789012345678901234
111*1**111*1**111*1**111*1**
111*1**111*1**111*1**111*1**
111*1**111*1**111*1**111*1**
111*1**111*1**111*1**111*1**
111*1**111*1**111*1**111*1**
111*1**111*1**111*1**111*1**
111*1**111*1**111*1**111*1**
111*1**111*1**111*1**111*1**
111*1**111*1**111*1**111*1**
111*1**111*1**111*1**111*1**
Periodic Spaced Seed Design (cont.)

Table 1. The periodic spaced seed, applied to a read and slid through positions 8–14 six times, covers all the 21 pair of positions exactly once.

<table>
<thead>
<tr>
<th>Positions</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>Covering 21 pairs of positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slide 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>(11,13) (11,14) (13,14)</td>
</tr>
<tr>
<td>Slide 1</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>(8,12) (8,14) (12,14)</td>
</tr>
<tr>
<td>Slide 2</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>(8,9) (8,13) (9,13)</td>
</tr>
<tr>
<td>Slide 3</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>(9,10) (9,14) (10,14)</td>
</tr>
<tr>
<td>Slide 4</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(8,10) (8,11) (10,11)</td>
</tr>
<tr>
<td>Slide 5</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>(9,11) (9,12) (11,12)</td>
</tr>
<tr>
<td>Slide 6</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>(10,12) (10,13) (12,13)</td>
</tr>
</tbody>
</table>
Periodic Spaced Seed Generalization

- $|P|$: length of pattern.
- To get $|P|-1$ slides on a Read of length $|R|$, we need:
  - $\#$ Repeated Patterns = $(|R| - |P| + 1) / |P|$.
  - Appended Length = $(|R| - |P| + 1) \mod |P|$.
Periodic Spaced Seed Extension

Extended Single Periodic Spaced Seed

34-color SOLiD Read:

ACGTACGTCCCCCTTTTACGTACGTAAAAGGGGAAA

Colors:

1313131200020003131313130002000200

The seed generated by extending $S_{1,1}$ pattern

$W=19$

$\begin{array}{cccccccccccccccc}
1313 & * & * & 1 & * & * & 0 & 2 & 0 & 0 & * & * & 1 & * & * & 1313 & * & 0 & * & * & 0 & 2 & 0 & 0
\end{array}$

$W=18$

$\begin{array}{cccccccccccccccc}
* & 3131 & * & 2 & * & * & 2 & 0 & 0 & 0 & * & * & 3 & * & * & 3130 & * & 2 & * & * & 2 & 0 & 0
\end{array}$

$W=17$

$\begin{array}{cccccccccccccccc}
** & 1313 & * & 0 & * & * & 0 & 0 & 0 & 3 & * & 1 & * & * & 1300 & * & 0 & * & * & 0 & 2 & 0 & 0
\end{array}$

$\vdots$

$\vdots$

$\vdots$

$W=14$

$\begin{array}{cccccccccccccccc}
******** & 0 & 0 & 2 & * & 0 & * & * & 1313 & * & 0 & * & * & 0 & 0 & 2 & 0
\end{array}$

$W=14$

$\begin{array}{cccccccccccccccc}
******** & 0 & 0 & 2 & 0 & * & 3 & * & 3 & 1313 & * & 0 & * & * & 0 & 2 & 0 & 0
\end{array}$

29
Periodic Spaced Seed Extension

Extended Single Periodic Spaced Seed

34-color SOLiD Read:
ACGTACGTCCCCCTTTTACGTACGTAAAAAGGGGAAA

Colors: 1313131200020003131313130002000200

The seed generated by extending $S_{1,1}$ pattern

$W=19$ \[1313**1***0200**1***1313**0***0200\]

$W=18$ \[**3131**2***2000**3***3130**2***200\]

$W=17$ \[**1313**0***003**1***1300**0***00\]

$W=14$ \[********002**0***1313**0***002**\]

$W=14$ \[********0020**3***3131**0***0200\]

...
Periodic Spaced Seed Extension

Extended Single Periodic Spaced Seed

34-color SOLiD Read:

ACGTACGTCCCCCTTTTTACGTACGTTAAAGGGGAAA
Colors: 1313131200020003131313130002000200

The seed generated by extending S1,1 pattern

\[
\begin{align*}
W=19 & : 1313**1***0200**1***1313**0***0200 \\
W=18 & : *3131**2***2000**3***3130**2***200 \\
W=17 & : **1313**0***003**1***1300**0***00 \\
W=16 & : \\
W=15 & : \\
W=14 & : ******002**0***1313**0***002** \\
W=13 & : \\
W=12 & : \\
W=11 & : \\
W=10 & : \\
\end{align*}
\]

5 Times Faster!
Efficient indexing for extension

<table>
<thead>
<tr>
<th>13131020011313</th>
<th>0002 002 00200 0021 010</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|                |                         | 30
Efficient indexing for extension

<table>
<thead>
<tr>
<th>13131020011313</th>
<th>0002 002 00200 0021 010</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**METHODS & ALGORITHM**

**Efficient indexing for extension**

| 13131020011313 | 0002 |
| 002 | 002 |
| 00200 | 0021 |
| 010 | 010 |
| . | . |
| . | . |
| . | . |
| . | 30 |
**METHODS & ALGORITHM**

Efficient indexing for extension

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>13131020011313</th>
<th></th>
<th>0002</th>
<th>002</th>
<th>00200</th>
<th>0021</th>
<th>010</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0021</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Efficient indexing for extension

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13131020011313</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00200</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0021</td>
<td></td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>
How to find such seed?

- Exhaustive search:
- Given seed length, for each \((x, k)\), enumerating all patterns of length \(|P|\) which satisfy full sensitivity \(k\) and has \(x\) consecutive mismatches.
- Find the pattern with maximum weight.
METHODS & ALGORITHM

How to find such seed? (cont.)

Table 2. The maximum weights of patterns that are full sensitivity to $x$ SNPs and $k$ free mismatches

| Sensitivity threshold | Periodic pattern length $|P|$ |
|-----------------------|-----------------------------|
|                       | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
| $k = 2$               | 3  | 4  | 4  | 5  | 6  | 7  | 8  | 9  | 9  | 10 |
| $x = 1, k = 1$        | 2  | 2  | 3  | 4  | 5  | 5  | 6  | 7  | 8  | 8  |
| $k = 3$               | 2  | 2  | 3  | 3  | 4  | 5  | 5  | 6  | 6  | 7  |
| $x = 2, k = 0$        | 1  | 2  | 2  | 3  | 4  | 5  | 5  | 6  | 7  | 8  |
| $k = 4$               | 1  | 1  | 1  | 2  | 3  | 3  | 3  | 4  | 4  | 5  |
How to find such seed? (cont.)

- Which pattern length provides the best seed given $x, k$?
- Consider the shortest pattern whose weight is large enough, i.e. find the pattern with reasonably high maximum-weight / length ratio.
How to find such seed? (cont.)

The weight–length ratios of the single periodic spaced seed patterns

Fig. 3. This figure shows the optimal weight–length ratios for different pattern lengths.
How to find such seed? (cont.)

- Choose $|P| = 7$ for less queries per read.
  - only 6 queries
Implementation detail

• Traditional two-bit base encoding:
  – A = 00, C = 01, G = 10, T = 11
  – Ex. ATGGA = 00 11 10 10 00
    • Most significant bit string U = 01110.
    • Least significant bit string V = 01000.
Color encoding

- SOLiD
  - Parallel sequencing.
  - Each probe determines two base positions at a time, represented by four colors to encode the 16 possible two-base combinations.
  - A single color encodes two adjacent bases.
  - Every base affects two adjacent colors.
Color encoding (cont.)

- Encoding for SOLiD reads:
  - B = 00, G = 01, Y = 10, R = 11
- Base to color:
  - S = U XOR (U >> 1), T = V XOR (V >> 1)
  - Ex. ATGGA = 00 11 10 10 00
    S = 01110 XOR 0111 = 1001
    T = 01000 XOR 0100 = 1100
    Color string of ATGGA is BGYR (11 01 00 10).
Color encoding (cont.)

• Most significant bit can be used to distinguish between \{A:00,C:01\} and \{G:10,T:11\}.

• Least significant bit can be used to distinguish between \{A:00,G:10\} and \{C:01,T:11\}.

• Blue:00 means no difference between two consecutive bases while Red:11 means total difference.

• Yellow:10 means different most significant bit, while Green:01 means different least significant bit.
Biological meaning of mismatch: Point Mutation or Substitution
Two consecutive mismatches of color

• Three types of base substitutions (valid mismatches):
  – Transversion 1: A:00 <> T:11 or G:10 <> C:01
    • B:00 <> R:11 or G:01 <> Y:10
  – Transversion 2: A <> C or G <> T
    • B <> G or R <> Y
  – Transition: A <> G or C <> T
    • B <> Y or G <> R
Two consecutive mismatches of color (cont.)

- Ex. **BRRB** mapped to **BBBB** (possibly, AATAA maps to AAAAAA).
  - A <> T causes two R <> B
  - a valid SNP
- Invalid SNP:
  - BRRB (AATAA) vs BBGB (AAACC)
- Both color mismatches are of the same type if it indicates a valid SNP.
Two consecutive mismatches of color (cont.)

- Given traditional two-bit base encoding:
- Transversion 1: B:00 <> R:11 or Y:10 <> G:01
  - (MSB1 XOR MSB2) AND (LSB1 XOR LSB2)
- Transversion 2: B:00 <> G:01 or R:11 <> Y:10
  - (NOT (MSB1 XOR MSB2)) AND (LSB2 XOR LSB2)
- Transition: B:00 <> Y:10 or G:01 <> R:11
  - (MSB1 XOR MSB2) AND (NOT (LSB2 XOR LSB2))
INTRODUCTION

Methods & algorithm

Results

Discussion
EXPERIMENTAL RESULTS (1/3)

R00922152 翁健庭
RESULTS

Table 3. PerM’s single periodic spaced seeds for SOLiD 34-color reads

<table>
<thead>
<tr>
<th>Seed name</th>
<th>Seed patterns parenthesized according to their repeats</th>
<th>Seed weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_2$</td>
<td>(111<em>1**)(111</em>1**)(111<em>1**)(111</em>1**)</td>
<td>16</td>
</tr>
<tr>
<td>$S_{1,1}$</td>
<td>(1111<strong>1</strong>*)(1111<strong>1</strong>*)(1111*)</td>
<td>14</td>
</tr>
<tr>
<td>$F_3$</td>
<td>(111<em>1<strong>1</strong></em>)(111<em>1<strong>1</strong></em>)(11)</td>
<td>12</td>
</tr>
<tr>
<td>$S_{2,0}$</td>
<td>(1111<strong>1****)(1111</strong>1****)(11)</td>
<td>12</td>
</tr>
<tr>
<td>$F_4$</td>
<td>(11<strong><strong>1*****)(11</strong></strong>1*****)(11****)</td>
<td>8</td>
</tr>
</tbody>
</table>

- The periodic spaced seeds used in PerM outperform the seeds used in MAQ in terms of mapping speed and sensitivity for both Illumina and SOLiD data.
RESULTS

Table 3. PerM’s single periodic spaced seeds for SOLiD 34-color reads

<table>
<thead>
<tr>
<th>Seed name</th>
<th>Seed patterns parenthesized according to their repeats</th>
<th>Seed weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_2$</td>
<td>(111<em>1**)(111</em>1**)(111<em>1**)(111</em>1**)(111*1**)</td>
<td>16</td>
</tr>
<tr>
<td>$S_{1,1}$</td>
<td>(1111<strong>1</strong>*)(1111<strong>1</strong>*)(1111*)</td>
<td>14</td>
</tr>
<tr>
<td>$F_3$</td>
<td>(111<em>1<strong>1</strong></em>)(111<em>1<strong>1</strong></em>)(11)</td>
<td>12</td>
</tr>
<tr>
<td>$S_{2,0}$</td>
<td>(1111<strong>1</strong>***)(1111<strong>1</strong>***)(11)</td>
<td>12</td>
</tr>
<tr>
<td>$F_4$</td>
<td>(11<em><strong>1</strong></em>**)(11<em><strong>1</strong></em>**)(11****)</td>
<td>8</td>
</tr>
</tbody>
</table>

- $F_k$ denotes a seed full sensitive to $k$ mismatches,
- $S_{x,k}$ denotes a SOLiD-specific seed full sensitive to $x$ consecutive color mismatches (SNPs) and $k$ free color mismatches.
RESULTS

- Memory
- Running Time
Table 4. Three seed families are compared in their ability to map 34-color SOLiD reads to a preprocessed human genome.

<table>
<thead>
<tr>
<th>Seed name</th>
<th>No. of index tables</th>
<th>No. of queries per read</th>
<th>Seed weight</th>
<th>Extended weights</th>
<th>E(Random Hits) per read</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_2$</td>
<td>1</td>
<td>7</td>
<td>16</td>
<td>16–20</td>
<td>1.89</td>
</tr>
<tr>
<td>$C_2$</td>
<td>3</td>
<td>6</td>
<td>16</td>
<td></td>
<td>8.38</td>
</tr>
<tr>
<td>$S_{1,1}$</td>
<td>1</td>
<td>10</td>
<td>14</td>
<td>14–19</td>
<td>68.91</td>
</tr>
<tr>
<td>$F_3$</td>
<td>1</td>
<td>11</td>
<td>12</td>
<td>12–16</td>
<td>627.25</td>
</tr>
<tr>
<td>$C_3$</td>
<td>4</td>
<td>10</td>
<td>12</td>
<td></td>
<td>3576.28</td>
</tr>
<tr>
<td>$S_{2,0}$</td>
<td>1</td>
<td>11</td>
<td>12</td>
<td>12–16</td>
<td>534.42</td>
</tr>
<tr>
<td>$C_4$</td>
<td>5</td>
<td>15</td>
<td>10</td>
<td></td>
<td>85.830</td>
</tr>
<tr>
<td>$F_4$</td>
<td>1</td>
<td>10</td>
<td>8</td>
<td>8–11</td>
<td>216.007</td>
</tr>
</tbody>
</table>

- PerM: a single index table.
- Convention Method: 3~5 index tables.
- It allows us to preprocess the human genome efficiently into 4.5 bytes per base, and load it to 14 GB of memory, without the swapping of index tables between disk and memory.
Preprocessing:
- The time to preprocess the reference genome (or the reads set) into one or more index tables.

Mapping:
- The total time to find matches in the index tables for all queried subsequences, and the time to examine all matches using the full read-genome substring alignments.
RESULT-RUNNING TIME

• Preprocessing:
  • A single index table results in faster preprocessing time than methods.

• Mapping:
  • Query each seed-induced subsequence and validate matches which result in true alignments.
  • Examine and ignore matches that result from random hits.

(related to seed weight)
Table 4. Three seed families are compared in their ability to map 34-color SOLiD reads to a preprocessed human genome

<table>
<thead>
<tr>
<th>Seed name</th>
<th>No. of index tables</th>
<th>No. of queries per read</th>
<th>Seed weight</th>
<th>Extended weights</th>
<th>$E$(Random Hits) per read</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_2$</td>
<td>1</td>
<td>7</td>
<td>16</td>
<td>16–20</td>
<td>1.89</td>
</tr>
<tr>
<td>$C_2$</td>
<td>3</td>
<td>6</td>
<td>16</td>
<td></td>
<td>8.38</td>
</tr>
<tr>
<td>$S_{1,1}$</td>
<td>1</td>
<td>10</td>
<td>14</td>
<td>14–19</td>
<td>68.91</td>
</tr>
<tr>
<td>$F_3$</td>
<td>1</td>
<td>11</td>
<td>12</td>
<td>12–16</td>
<td>627.25</td>
</tr>
<tr>
<td>$C_3$</td>
<td>4</td>
<td>10</td>
<td>12</td>
<td></td>
<td>3576.28</td>
</tr>
<tr>
<td>$S_{2,0}$</td>
<td>1</td>
<td>11</td>
<td>12</td>
<td>12–16</td>
<td>534.42</td>
</tr>
<tr>
<td>$C_4$</td>
<td>5</td>
<td>15</td>
<td>10</td>
<td></td>
<td>85.830</td>
</tr>
<tr>
<td>$F_4$</td>
<td>1</td>
<td>10</td>
<td>8</td>
<td>8–11</td>
<td>216.007</td>
</tr>
</tbody>
</table>

- If the seed weight is insufficient, the examination of random hits will dominate the running time.
EXPERIMENTAL RESULTS (2/3)

D96922010 何 恩
EXPERIMENTAL RESULTS

• Genome-scale comparison
  – MAQ and Bowtie
• Illumina and SOLiD reads
  – The 100 Genomes Project
• PerM vs. SOCS
  – SOCS: designed for ABI SOLiD reads
Genome-scale mapping with SOLiD reads

Table 5. The results of mapping 5 million 34-color SOLiD reads to the whole human genome

<table>
<thead>
<tr>
<th>Seed name</th>
<th>Mapped reads</th>
<th>Unique SNP-supporting reads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 mis</td>
<td>4 mis</td>
</tr>
<tr>
<td>$F_2$</td>
<td>298 898</td>
<td>167 048</td>
</tr>
<tr>
<td>$S_{1,1}$</td>
<td>465 460</td>
<td>348 416</td>
</tr>
<tr>
<td>$F_3$</td>
<td>496 401</td>
<td>379 936</td>
</tr>
</tbody>
</table>

All PerM seeds provide a minimum of full sensitivity to two mismatches and report 637 681 exact matches, and 583 363 and 561 029 reads with one and two mismatches, respectively.
EXPERIMENTAL RESULTS

Genome-scale mapping with SOLiD reads

Table 6. Running time comparison of mapping the 35 bp SOLiD reads to the whole human genome

<table>
<thead>
<tr>
<th>Program</th>
<th>Seed/mode</th>
<th>weight</th>
<th>(Full) Sensitivity</th>
<th>Speed (M/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PerM</td>
<td>$F_2$</td>
<td>16–20</td>
<td>2 colors</td>
<td>3.53</td>
</tr>
<tr>
<td>PerM</td>
<td>$S_{1,1}$</td>
<td>14–19</td>
<td>1 base + 1 color</td>
<td>1.17</td>
</tr>
<tr>
<td>PerM</td>
<td>$F_3$</td>
<td>12–16</td>
<td>3 colors</td>
<td>0.75</td>
</tr>
<tr>
<td>MAQ</td>
<td>-c</td>
<td>14</td>
<td>2 colors</td>
<td>0.56</td>
</tr>
</tbody>
</table>
EXPERIMENTAL RESULTS

Genome-scale mapping with Illumina reads

Table 7. Running time comparison of mapping the Illumina reads with different read lengths and seeds to the whole human genome

<table>
<thead>
<tr>
<th>Length</th>
<th>36 bp</th>
<th>40 bp</th>
<th>47 bp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Weight</td>
<td>Reads/h</td>
<td>Weight</td>
</tr>
<tr>
<td>Seed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>18–21</td>
<td>5.92 M</td>
<td>20–24</td>
</tr>
<tr>
<td>MAQ</td>
<td>14</td>
<td>0.49 M</td>
<td>14</td>
</tr>
<tr>
<td>Bowtie -v2*</td>
<td>4.43 M</td>
<td>3.87 M</td>
<td>14</td>
</tr>
<tr>
<td>F3</td>
<td>13–18</td>
<td>1.69 M</td>
<td>15–19</td>
</tr>
<tr>
<td>Bowtie -v3*</td>
<td>4.28 M</td>
<td>3.38 M</td>
<td>14</td>
</tr>
<tr>
<td>Bowtie default</td>
<td>9.27 M</td>
<td>7.95 M</td>
<td>24–28</td>
</tr>
</tbody>
</table>

The default mode of Bowtie is equivalent to -k 1. The -v k mode is set with -a -best -strata. The tests are performed on Sun, X4600, Opteron, 2.6 GHz, using 15 GB single node and thread.
EXPERIMENTAL RESULTS (3/3)

R00944050 王舜玄
Comparison: PerM and MAQ

Table 6. Running time comparison of mapping the 35 bp SOLiD reads to the whole human genome

<table>
<thead>
<tr>
<th>Program</th>
<th>Seed/mode</th>
<th>weight</th>
<th>(Full) Sensitivity</th>
<th>Speed (M/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PerM</td>
<td>$F2$</td>
<td>16–20</td>
<td>2 colors</td>
<td>3.53</td>
</tr>
<tr>
<td>PerM</td>
<td>$S_{1,1}$</td>
<td>14–19</td>
<td>1 base + 1 color</td>
<td>1.17</td>
</tr>
<tr>
<td>PerM</td>
<td>$F3$</td>
<td>12–16</td>
<td>3 colors</td>
<td>0.75</td>
</tr>
<tr>
<td>MAQ</td>
<td>-c</td>
<td>14</td>
<td>2 colors</td>
<td>0.56</td>
</tr>
</tbody>
</table>

- PerM is significant fast than MAQ, benefitted from
- **extendable periodic spaced seeds.**
  - Provides greater seed weight than fix-cont. seeds.
Comparison: PerM and MAQ (cont.)

- PerM is significantly faster than MAQ, benefitted from
- **extendable periodic spaced seeds**.
  - Provides greater seed weight than fix-cont. seeds.
- PerM **avoids the bottleneck from the many random hits** on large genome.
- MAQ builds **index tables for each mapping project**, while PerM **reuses the same index** because it preprocesses the genome.
EXPERIMENTAL RESULTS

Comparison: PerM and Bowtie

<table>
<thead>
<tr>
<th>Length</th>
<th>36 bp</th>
<th>40 bp</th>
<th>47 bp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Weight</td>
<td>Reads/h</td>
<td>Weight</td>
</tr>
<tr>
<td>Seed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>18–21</td>
<td>5.92 M</td>
<td>20–24</td>
</tr>
<tr>
<td>MAQ</td>
<td>14</td>
<td>0.49 M</td>
<td>14</td>
</tr>
<tr>
<td>Bowtie -v2*</td>
<td>4.43 M</td>
<td>3.87 M</td>
<td>2.64 M</td>
</tr>
<tr>
<td>F3</td>
<td>13–18</td>
<td>1.69 M</td>
<td>15–19</td>
</tr>
<tr>
<td>Bowtie -v3*</td>
<td>4.28 M</td>
<td>3.38 M</td>
<td>1.63 M</td>
</tr>
<tr>
<td>Bowtie default</td>
<td>9.27 M</td>
<td>7.95 M</td>
<td>7.20 M</td>
</tr>
</tbody>
</table>

- Bowtie slows down when long reads occur, because backtracking required to find inexact alignments.
- PerM's performance is just a result of seed weight.
EXPERIMENTAL RESULTS

Comparison: PerM and Bowtie (cont.)

- Bowtie slows down when long reads occur, because
  - backtracking required to find inexact alignments.
- PerM’s performance is just a result of seed weight.

- Both index the genome.

- PerM finds full sensitive alignments by seed matching,
  - while Bowtie uses modified exact matching and backtracking algorithms.
EXPERIMENTAL RESULTS

Comparison: PerM and Bowtie (cont.)

- **Bowtie** slows down when long reads occur, because
  - backtracking required to find inexact alignments.
- **PerM**’s performance is just a **result of seed weight**.
- Both index the genome.
- **PerM** finds **full sensitive alignments by seed matching**, while **Bowtie uses modified exact matching and backtracking algorithms**.
EXPERIMENTAL RESULTS

Comparison: PerM and SOCS

<table>
<thead>
<tr>
<th>Full sensitivity</th>
<th>PerM</th>
<th>SOCS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Running time</td>
<td>Weight</td>
</tr>
<tr>
<td>2 color mis</td>
<td>11 min 46 s</td>
<td>16–20</td>
</tr>
<tr>
<td>1 base + 1 color mis</td>
<td>23 min 0 s</td>
<td>14–19</td>
</tr>
<tr>
<td>3 color mis</td>
<td>32 min 41 s</td>
<td>12–16</td>
</tr>
</tbody>
</table>

The running time includes preprocessing and I/O. The memory usage of both the programs is <2 GB. The tests are performed on Sun, X4600, Opteron, 2.6 GHz, using single node and thread.

- SOCS is dedicated to SOLiD reads.
- PerM is fast than SOCS because the higher seed weight.
Comparison: PerM and SOCS (cont.)

- SOCS is dedicated to SOLiD reads.
- PerM is fast than SOCS because the higher seed weight.
- Both provide full sensitivity to 3 mismatches.
- While SOCS does not provide sufficient seed weight to map reads to the entire genome.
- Conducting 5 million 35bp SOLiD reads to chromosome X,
- and 8% reads including mapping with <3 substitutions in the experiment that highlights this weakness of SOCS.
EXPERIMENTAL RESULTS

Genome preprocessing

• Genome preprocessing time is linear to the reference’s size,
  • regardless of the number of used seed.

• To index the human genome:
  • PerM uses 3h 30min with 14GB memory.
  • Bowtie uses 4h 47min with 2.7GB memory.

• Preprocessing time << Mapping time
• Conducting mapping under the multiple-core architecture.
DISCUSSION

R00922152 翁健庭
OUTLINE

- Introduction
- Methods & algorithm
- Experimental Results
- Discussion
DISCUSSION

- PerM provides highly efficient mapping solutions for genome-scale mapping projects involving Illumina or SOLiD data.

- Require full sensitivity mismatches ($k \geq 4$) on a short read.

  - May incapable of providing efficient mapping performance.

    - Hashing to multiple index tables may be necessary to increase seed weight and eliminate a bottleneck in the checking step.
FIN.

- Introduction
- Methods & algorithm
- Experimental Results
- Discussion
Questions?