Hobbes: Optimizing Gram-Based Methods for Read Alignment

1 Supplementary Information

1.1 Pseudo-Code

In this section, we present pseudo-code for computing the optimal gram prefix (from Section 2.2), using the bitvector filter (from Section 2.3) and for the overall read-mapping procedure that utilizes the former two techniques for Hamming distance.

1.1.1 Optimal Prefix Grams

See Figure 1.

1.1.2 Read-Mapping Procedure

See Figure 2.

1.1.3 Bitvector Filtering Procedures

See Figures 3, and 4.

1.2 Multi-Threading

Figure 5 shows the mapping-time speed-up as we increased the number of threads used by Hobbes, BWA, and Bowtie. Hobbes scales very well when asked to find 100 (a), 1000 (b), or all (c) mappings per read. BWA's relative speed-up becomes worse as the number of requested mappings per read increases. One explanation could be that BWA only supports multiple threads in the second step of their mapping procedure. Notice that Bowtie's mapping time increases beyond 8 threads. We conjecture that both BWA's and Bowtie's speed in this scenario is limited by the disk bandwidth for writing the output file, because we observed disk thrashing at times. Since finding all mappings is not their design point this behavior is acceptable. However, as the number of cores in modern CPUs increase dramatically, disk usage may become a performance bottleneck even when asking for only a few mappings per read. We have shown that Hobbes utilizes multiple threads well, and still outperforms BWA and Bowtie.

```
Input:
    Gram length $q$.
    Inverted lists $L$ corresponding to a read $s$'s $q$-grams $G(s)$.
    Hamming-distance threshold $d$.
Output:
    A set $P$ of $d + 1$ non-overlapping $q$-gram positions,
    minimizing the sum of their inverted-list lengths.
Method: FindOptimalPrefix
1.  $r = d + 1 + 1$; // rows
2.  $c = |L| - d * q + 1$; // columns
3.  Create matrix $M(r, c)$;
4.  // initialize matrix
5.  FOR $i = 1$ TO $c$
6.      $M(i, 0) = \infty$;
7.  END FOR
8.  FOR $j = 1$ TO $r$
9.      $M(0, j) = 0$;
10. END FOR
11. // perform dynamic programming
12. FOR $i = 1$ TO $c$
13.     FOR $j = 1$ TO $r$
14.         $cmp = M(i - 1, j) + L[j + (i - 1) * q].len$;
15.         IF $cmp < M(i, j - 1)$ THEN
16.             $M(i, j) = cmp$;
17.             $P[i] = j + (i - 1) * q$;
18.         ELSE
19.             $M(i, j) = M(i, j - 1)$;
20.         ENDIF
21.     END FOR
22. END FOR
23. RETURN $P$;
```

Figure 1: Dynamic programming algorithm for finding an optimal set of prefix grams (Section 2.2).

1.3 Application in RNA-seq analysis

```
Input:
    Gram length $q$.
    Read $s$.
    Inverted lists $L$ corresponding to a read $s$'s $q$-grams $G(s)$.
    Note: Inverted lists contain pairs $(p, b)$ of position and bitvector.
    Hamming-distance threshold $d$.
    Bitvector width in bits $w$.
Output:
    A set $R$ of mapping positions within Hamming distance $d$ of read $s$.
Method: FindMappings
1.  $b_s = $ GetBitvector($s$);
2.  $P = $ FindOptimalPrefix($q$, $L$, $d$);
3.  SortByListLength($P$);
4.  $C = \{\}$; // candidate mappings
5.  FOR $i = 1$ TO $|P|$
6.      $b = $ AlignBitvector($b_s$, $w$, $q$, $P[i]$);
7.      $m = $ GetBitMask($|s|$, $w$, $q$, $P[i]$);
8.      $C_{new} = \{\}$, $k = 1$, $j = 1$;
9.      $list = L[P[i]]$;
10.     WHILE $k \leq |C|$ OR $j \leq |list|$ DO
11.         IF $k \leq |C|$ OR ($j \leq |list|$ AND $C[k] > list[j].p$) THEN
12.             IF Popcount($b$ XOR ($list[j].b$ AND $m$)) $\leq d$ THEN
13.                 $C_{new}.add(list[j].p)$;
14.             END IF
15.             $j++$;
16.         ELSE IF $j \leq |list|$ OR ($k \leq |C|$ AND $C[k] < list[j].p$) THEN
17.             $C_{new}.add(C[k])$;
18.             $k++$;
19.         ELSE
20.             IF Popcount($b$ XOR ($list[j].b$ AND $m$)) $\leq d$ THEN
21.                 $C_{new}.add(C[k])$;
22.             END IF
23.             $k++$;
24.             $j++$;
25.         END IF
26.     END WHILE
27.     $C = C_{new}$;
28. END FOR
29. $R$ = VerifyCandidates($C$);
30. RETURN $R$;
```

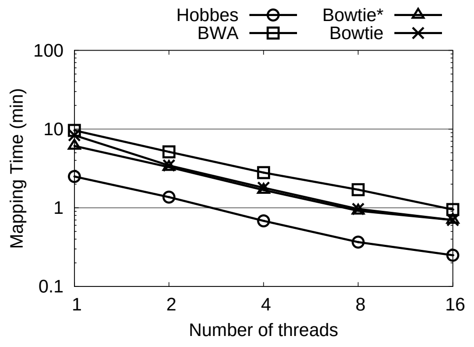
Figure 2: Read-mapping procedure for finding all mappings within a Hamming-distance threshold. The procedure uses the optimal prefix grams, and bitvector filtering.

```
Input:
    Read bitvector $b_s$.
    Bitvector width in bits $w$.
    Gram length $q$.
    Position of matching $q$-gram $p$.
Output:
    A read's aligned bitvector.
Method: AlignBitvector
1.  $M_L = (p-1)$ "1" bits followed by $(|b_s| - p + 1)$ "0" bits;
2.  $M_R = (p-1)$ "0" bits followed by $(|b_s| - p + 1)$ "1" bits;
3.  $L = (b_s \text{ AND } M_L) >> (w/2 - (p-1))$;
4.  $R = (b_s \text{ AND } M_R) << (p + q - 1) - w/2))$;
5.  $b = L + R$;
6.  **RETURN** First $w$ bits of $b$;
```

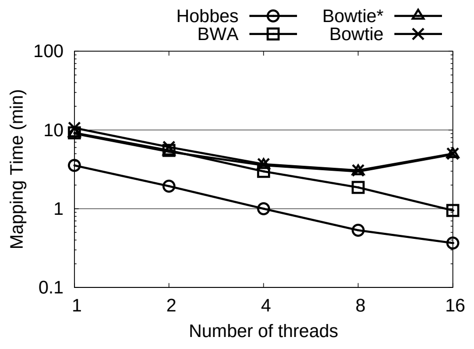
Figure 3: Procedure for aligning a read's bitvector to the bits in the bitvectors of the reference sequence, based on the position of a matching q-gram in the read.

```
Input:
    Read length $|s|$.
    Bitvector width in bits $w$.
    Gram length $q$.
    Position of matching $q$-gram $p$.
Output:
    Bitmask to remove invalid bits.
Method: GetBitMask
1.  $M = w$ "1" bits;
2.  $L = w/2 - (p-1)$;
3.  $R = max(0, (p + q - 1) - w/2 - (|s| - w))$;
4.  Set first $L$ bits of $M$ to "0".
5.  Set last $R$ bits of $M$ to "0".
6.  **RETURN** $M$;
```

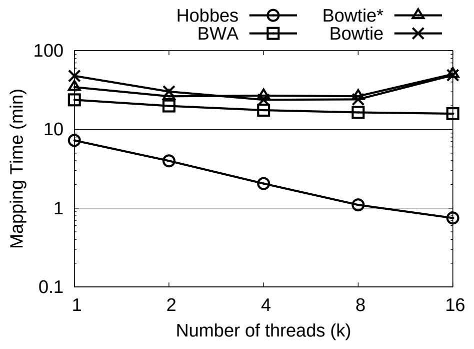
Figure 4: Procedure for computing a bitmask to remove invalid bits from bitvectors of the reference sequence, based on the position of a matching q-gram in a read.



(a) output at most 100 mappings per read



(b) output at most mappings 1000 per read



(c) output all mappings per read

Figure 5: Mapping times of Bowtie, BWA and Hobbes using multiple threads on 51bp read within 3 errors (Hamming distance). Notice the log-log scale. mrsFast, and RazerS2 do not support multi-threading (we could not find it in their manual).

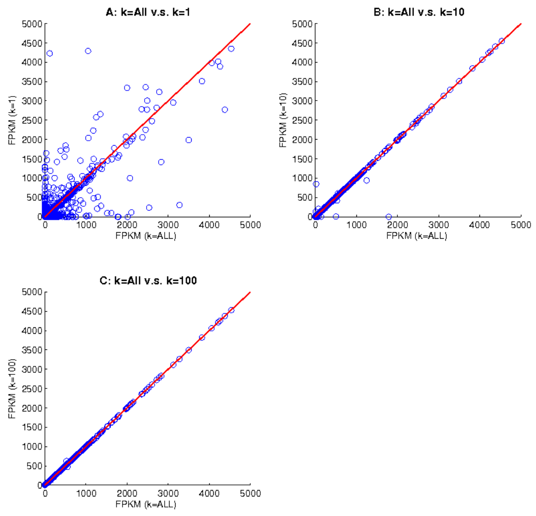


Figure 6: Scatter plot of fragments per kilobase per million (FPKM) value when Hobbes returns at most k mappings versus all mappings. In subfigure A, there is a lot of variability when k=1; in subfigure B, the amount of variablity reduces when k=10; in subfigure C, when k=100, the estimated FPKM value become more close to that estimated based on all mapping locations.