Fast detection of maximal exact matches via fixed sampling of query k-mers and Bloom filtering of index k-mers

Yuansheng Liu[1], Leo Yu Zhang[2], and Jinyan Li[1]

[1]Advanced Analytics Institute, University of Technology Sydney, PO Box 123, Broadway, NSW 2007, Australia
[2]School of Information Technology, Deakin University, Victoria 3216, Australia

March 31, 2019

Table S1: Details of genome datasets used in the experiments

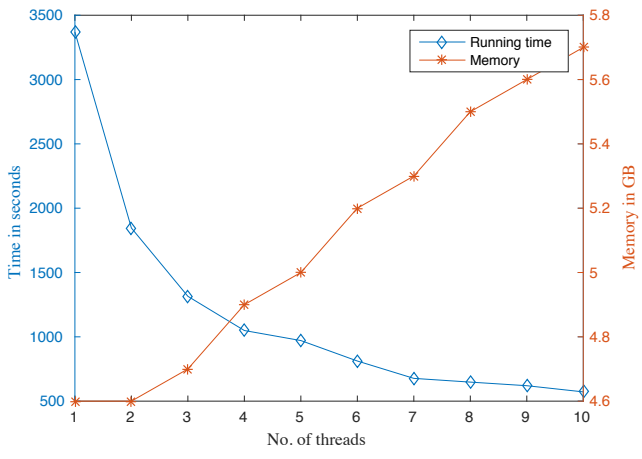
Dataset	No. of sequences	URL
Homo sapiens	93	`ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz`
Mus musculus	66	`ftp://hgdownload.cse.ucsc.edu/goldenPath/mm10/bigZips/chromFa.tar.gz`
Pan troglodytes	24,132	`ftp://hgdownload.cse.ucsc.edu/goldenPath/panTro3/bigZips/panTro3.fa.gz`
Triticum aestivum	731,921	`ftp://ftp.ensemblgenomes.org/pub/plants/release-22/fasta/triticum_aestivum/dna/Triticum_aestivum.IWGSP1.22.dna.genome.fa.gz`
Triticum durum	5,671,204	`https://urgi.versailles.inra.fr/download/iwgsc/TGAC_WGS_assemblies_of_other_wheat_speciesTGAC_WGS_durum_v1.fasta.gz`

Table S2: Three pairs of reference and query genomes along with their number of sequences used in the experiments

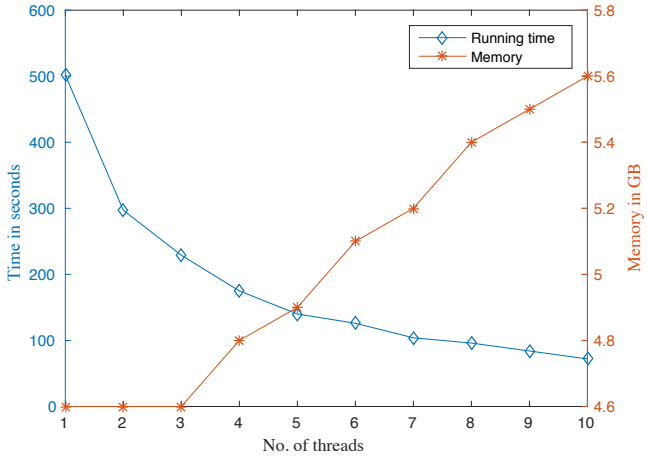
Reference genome	Query genome
Homo sapiens (93)	Mus musculus (66)
Homo sapiens (93)	Pan troglodytes (24,132)
Triticum aestivum (731,921)	Triticum durum (5,671,204)

Table S3: Number of k-mers on sampling and indexing stages of bfMEM

Methods	H. sapiens vs M. musculus				T. aestivum vs T. durum			
	$L = 40$	$L = 50$	$L = 80$	$L = 100$	$L = 40$	$L = 50$	$L = 80$	$L = 100$
Number of sampled k-mers	294, 751, 204	241, 159, 540	91, 473, 837	64, 700, 785	310, 246, 823	247, 157, 111	88, 480, 971	60, 185, 537
Cardinality of sampled k-mers set	260, 682, 668	218, 899, 183	86, 099, 573	61, 678, 211	296, 539, 245	239, 430, 641	82, 779, 191	54, 685, 008
Number of filtered k-mers	20, 925, 278	19, 001, 992	20, 196, 055	21, 007, 666	513, 266, 575	198, 611, 456	50, 524, 801	32, 473, 459
Cardinality of filtered k-mers set	13, 798, 228	15, 559, 803	18, 540, 613	19, 797, 603	127, 097, 761	93, 678, 754	38, 276, 573	27, 626, 651



(a) $L = 50$



(b) $L = 80$

Figure S1: Effect of increasing number of threads on running time and memory.

1 Result comparison

As we hash matches into some small files and sort them independently to remove duplicates, the order of matches is different from E-MEM and copMEM. For the convenience of the result comparison, we provide a tool "formatconvert" to reorder the MEMs generated by our tool bfMEM. After reordering, the order is the same as that of E-MEM and copMEM. The usages are added in the GitHub page https://github.com/yuansliu/bfMEM. Table S4 lists the running time and memory usage of the tool "formatconvert". It can be seen from the table that the reorder procedure is time- and memory-consuming when large number of MEMs are detected.

Table S4: Running time (in second) and memory usage (in GB) of the tool "formatconvert"

Format convert	*H. sapiens* vs *M. musculus*			*H. sapiens* vs *P. troglodytes*					*T. aestivum* vs *T. durum*		
	$L = 40$	$L = 50$	$L \geq 80$	$L = 40$	$L = 50$	$L = 80$	$L = 100$	$L \geq 150$	$L = 40$	$L = 50$	$L \geq 80$
Time	6,824	599	< 3	7,703	1,989	208	82	< 10	143	27	< 4
Memory	73.5	7.1	< 0.1	102.5	22.5	3.0	1.2	< 0.2	0.03	0.02	< 0.02

2 False positive probability of two Bloom filters

In our method, we first assume the false positive probability (FPP) $p = 0.01$ and then calculate the bits length s by cooperating with the number of sampled k-mers and the number of hash functions $h = -\log_2 0.01 = 6$. As we mentioned in Sec. 2.1.1 that some sampled k-mers on the query sequence are duplicated. The number of sampled k-mers and the cardinality of sampled k-mers set are shown in rows 1 and 2 in Tab. S3. We can see that the number of k-mers inserted into the Bloom filter f_q is little smaller than the number of sampled k-mers actually. And the bit array length of f_q is calculated by the number of sampled k-mers. Therefore, the FPP of f_q is little smaller than 0.01, i.e.,

$$\left(1 - e^{-\frac{6\times u'}{s}}\right)^6 < \left(1 - e^{-\frac{6\times u}{s}}\right)^6 \approx 0.01,$$

where u' is the cardinality of sampled k-mers set and $u' < u$.

In the second Bloom filter f_r, we set the same bit array length of f_q. The FPP of f_r is determined by the cardinality of filtered k-mers set. The number of filtered k-mers and the cardinality of associated k-mers set are shown in rows 3 and 4 in Tab. S3. We can see from the Tab. S3, the cardinality of filtered k-mers set is at least 2 times smaller than the number of sampled k-mers. Therefore, the FPP of f_r is:

$$\left(1 - e^{-\frac{6\times u'/2}{s}}\right)^6 < \left(1 - \left(1 - 0.01^{1/6}\right)^{1/2}\right)^6 = 0.00037.$$

3 The effect of Bloom filter FPP

Table S5 presents the performance comparison of bfMEM under different FPP of Bloom filter in the discovery of MEMs between *H. sapiens* vs *M. musculus* with $L = 80$ and 100. We can see from the results, higher FPP normally leads to more memory consumption as more k-mers from the reference sequence passing the Bloom filter test which are then added into the hash table for extension. Moreover, higher FPP (≥ 0.03) slows down the running time much. On the other hand, when small values of FPP (≤ 0.01) are used, the running time and memory usage will fluctuate around their mean value. In our algorithm, we set FPP = 0.01 as default.

Table S5: Comparison of running time (in second) and memory usage (in GB) under different FPP of Bloom filter (*H. sapiens* vs *M. musculus* with $L = 80$ and 100

FPP		0.003	0.005	0.007	0.009	0.01	0.015	0.02	0.025	0.03	0.05	0.10
$L = 80$	Running time	82	83	87	80	81	85	84	92	103	106	123
	Memory usage	5.3	5.3	5.5	5.6	5.6	5.9	6.2	6.5	7.0	9.2	16.5
$L = 100$	Running time	64	69	67	63	70	71	78	89	87	102	113
	Memory usage	5.2	5.3	5.4	5.6	5.6	6.0	6.2	6.6	7.2	9.6	17.0

4 Other features of bfMEM

Our tool bfMEM also supports to detecting reverse-complement matches or both forward and reverse-complement matches by setting the parameter '`-s`':

```
./bfmem -l <n> -r <ref> -q <query> -o <output> -s r
./bfmem -l <n> -r <ref> -q <query> -o <output> -s b
```

5 Selection of k

Given values of L and k, the sampling step is $w = L - k + 1$ and the number of querying k-mer is $u = \lceil (m - k + 1)/w \rceil$, where m is the length of query sequence. We can see that if k is set to small, less number of querying k-mers are sampled. However, if k is too small, such as $1 \leq k \leq 20$, large number of duplicate k-mers on the reference sequence will pass the Bloom filter test. It leads to more k-mers to be stored in the hash table. If k is too large, such as $k = L$ or $L - 1$, large number of querying k-mers are generated. The value of k affects the memory usage and running speed. In our implementation, we set some default values according to different values of L as following:

$$k = \begin{cases} 200; \text{if } L \geq 300. \\ 160; \text{if } L \geq 200; \\ 100; \text{if } L \geq 150; \\ 60; \text{if } L \geq 100; \\ 52; \text{if } L \geq 80; \\ 40; \text{if } L \geq 50; \\ 32; \text{if } L \geq 40; \\ 28; \text{if } L \geq 34; \end{cases}$$

6 Exact command used in experiment

- **bfMEM**
  `./bfmem -l <n> -r <ref> -q <query> -o <output> -t <threads>`
- **essaMEM**
  `./essaMEM -maxmatch -l <n> -F -n -threads 10 -k 10 <ref> <query>`
- **E-MEM**
  `./e-mem -n -l <n> -F -t <threads> <ref> <query>`
- **copMEM**
  `./copmem -l <n> -o <output> <ref> <query>`

Here, we give part of details on the two genomes "`hg19.all.fa`" and "`mus.all.fa`". For others, we only need to change the number of threads and the input genome files.

- **bfMEM**

```
./bfmem -l 300 -r hg19.all.fa -q mus.all.fa -o bfmem-300-out -t 10
./bfmem -l 200 -r hg19.all.fa -q mus.all.fa -o bfmem-200-out -t 10
./bfmem -l 150 -r hg19.all.fa -q mus.all.fa -o bfmem-150-out -t 10
./bfmem -l 100 -r hg19.all.fa -q mus.all.fa -o bfmem-100-out -t 10
./bfmem -l 80 -r hg19.all.fa -q mus.all.fa -o bfmem-80-out -t 10
./bfmem -l 50 -r hg19.all.fa -q mus.all.fa -o bfmem-50-out -t 10
./bfmem -l 40 -r hg19.all.fa -q mus.all.fa -o bfmem-40-out -t 10
```

- **essaMEM**

```
./essaMEM -maxmatch -l 300 -F -n -qthreads 10 -k 10 hg19.all.fa mus.all.fa > essamem-300-out
./essaMEM -maxmatch -l 200 -F -n -qthreads 10 -k 10 hg19.all.fa mus.all.fa > essamem-200-out
./essaMEM -maxmatch -l 150 -F -n -qthreads 10 -k 10 hg19.all.fa mus.all.fa > essamem-150-out
./essaMEM -maxmatch -l 100 -F -n -qthreads 10 -k 10 hg19.all.fa mus.all.fa > essamem-100-out
./essaMEM -maxmatch -l 80 -F -n -qthreads 10 -k 10 hg19.all.fa mus.all.fa > essamem-80-out
./essaMEM -maxmatch -l 50 -F -n -qthreads 10 -k 10 hg19.all.fa mus.all.fa > essamem-50-out
./essaMEM -maxmatch -l 40 -F -n -qthreads 10 -k 10 hg19.all.fa mus.all.fa > essamem-40-out
```

- **E-MEM**

```
./e-mem -n -l 300 -F -t 10 hg19.all.fa mus.all.fa > emem-300-t10-out
./e-mem -n -l 200 -F -t 10 hg19.all.fa mus.all.fa > emem-200-t10-out
./e-mem -n -l 150 -F -t 10 hg19.all.fa mus.all.fa > emem-150-t10-out
./e-mem -n -l 100 -F -t 10 hg19.all.fa mus.all.fa > emem-100-t10-out
./e-mem -n -l 80 -F -t 10 hg19.all.fa mus.all.fa > emem-80-t10-out
./e-mem -n -l 50 -F -t 10 hg19.all.fa mus.all.fa > emem-50-t10-out
./e-mem -n -l 40 -F -t 10 hg19.all.fa mus.all.fa > emem-400-t10-out
```

- **copMEM**

```
./copmem -o copmem-300-out -l 300 hg19.all.fa mus.all.fa
./copmem -o copmem-200-out -l 200 hg19.all.fa mus.all.fa
./copmem -o copmem-150-out -l 150 hg19.all.fa mus.all.fa
./copmem -o copmem-100-out -l 100 hg19.all.fa mus.all.fa
./copmem -o copmem-80-out -l 80 hg19.all.fa mus.all.fa
./copmem -o copmem-50-out -l 50 hg19.all.fa mus.all.fa
```