SEA 2018 report

Contents

1 Experimental setup

1.1 Time measurements

We run a program k times with and without the optimization and recorded the sum construction time of the MS and RUNS vectors. The plots report (median, with quartile ranges) of the speedup of each optimized time $t_i^{\texttt{opt}}$ relative to the average non-optimized time in the construction time of the MS vector. In other words

$$d^{(i)} = \frac{\bar{t}_{\texttt{non\_opt}}}{t_{\texttt{opt}}^{(i)}}$$

with $\bar{t}_{\texttt{non\_opt}} = 1/n \sum t_{\texttt{non\_opt}}^{(i)}$, and $i = 1, \ldots, k$.

The boxplots report the raw times.

2 WL tests

2.1 Input data

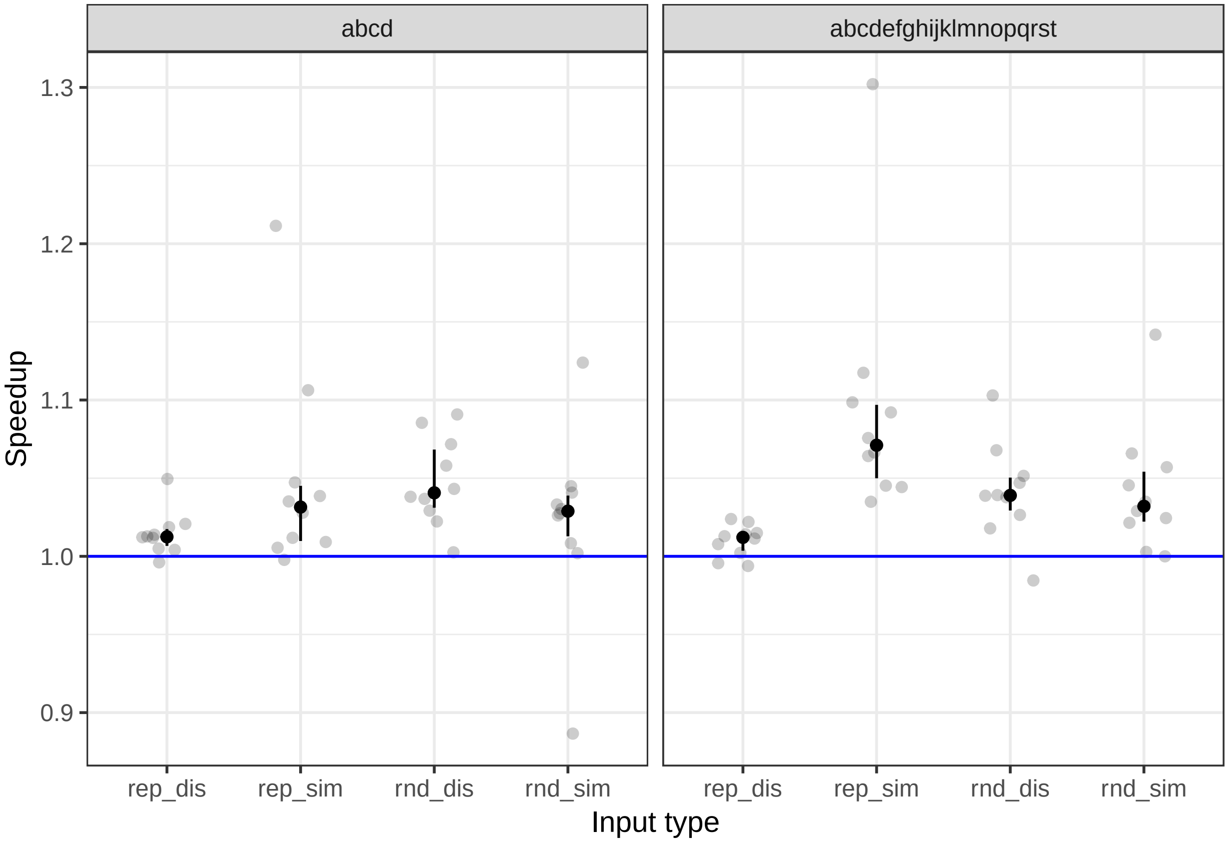
We perform tests on the Weiner Link optimizations on 4 types of input.

- Index string with repeats, query string random (code: `rep_dis`)

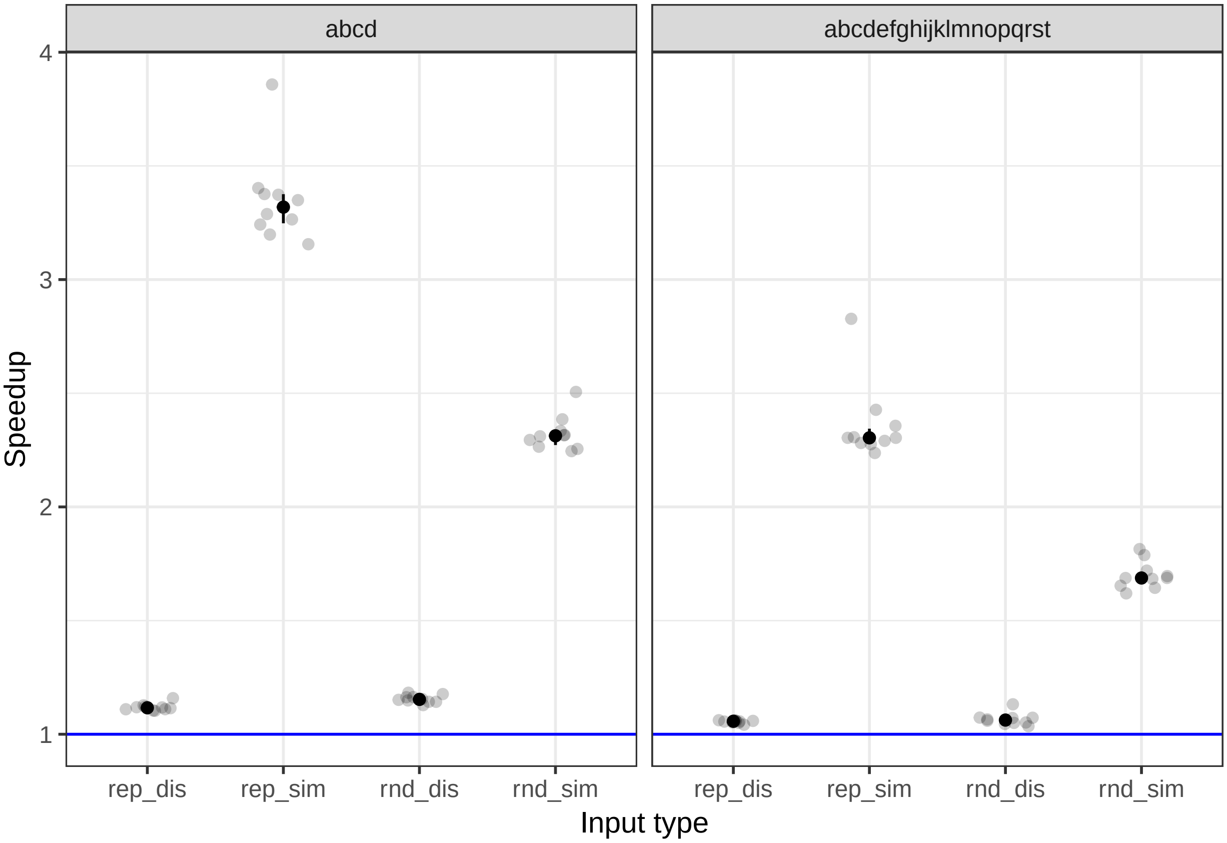
- Index string with repeats, query string similar to index (code: `rep_sim`)
- Index string random, query string random (code: `rnd_dis`)
- Index string random, query string similar to index (code: `rnd_sim`)

Further, we generate all of the above input data for two alphabet sizes: $\Sigma_1| = 4$ and $\Sigma_2| = 20$. For all input types, the index string is of length 100MB and the query 500KB.

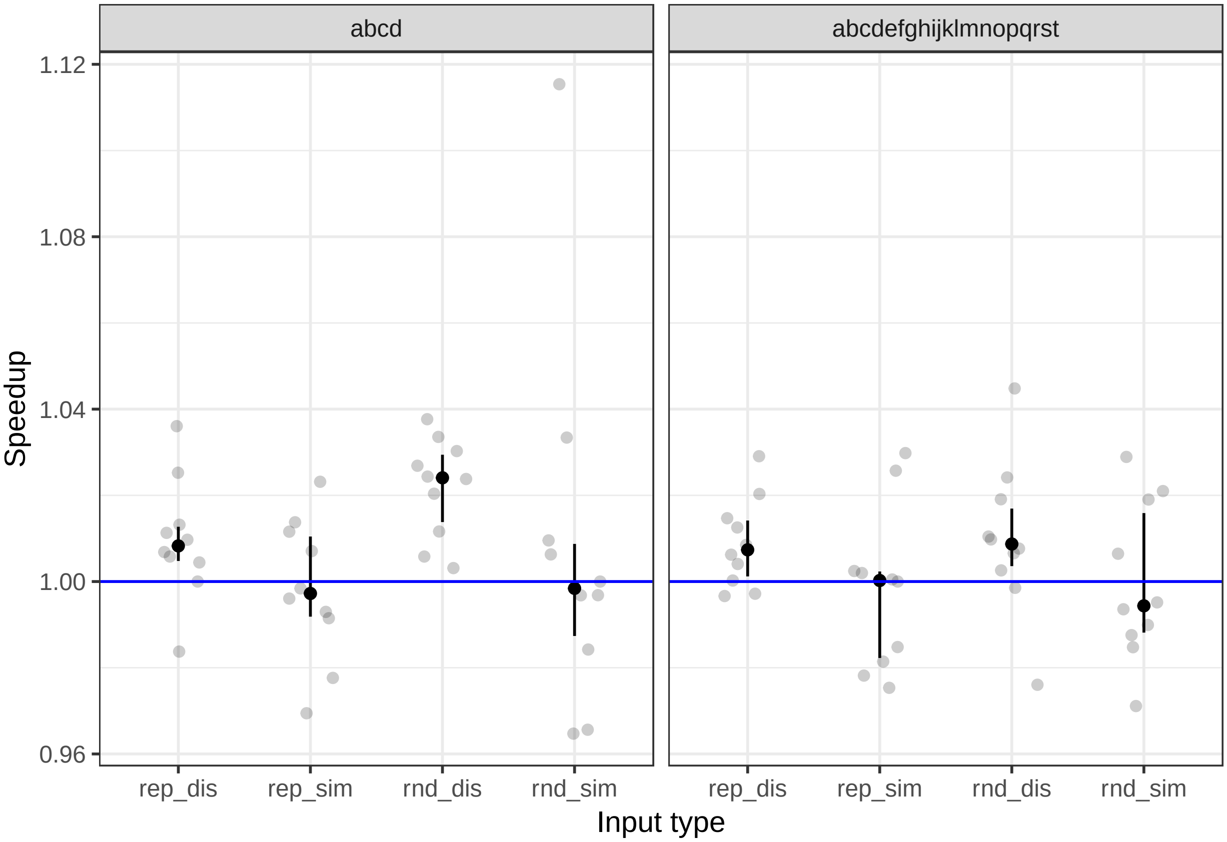
2.2 Double rank versus single rank



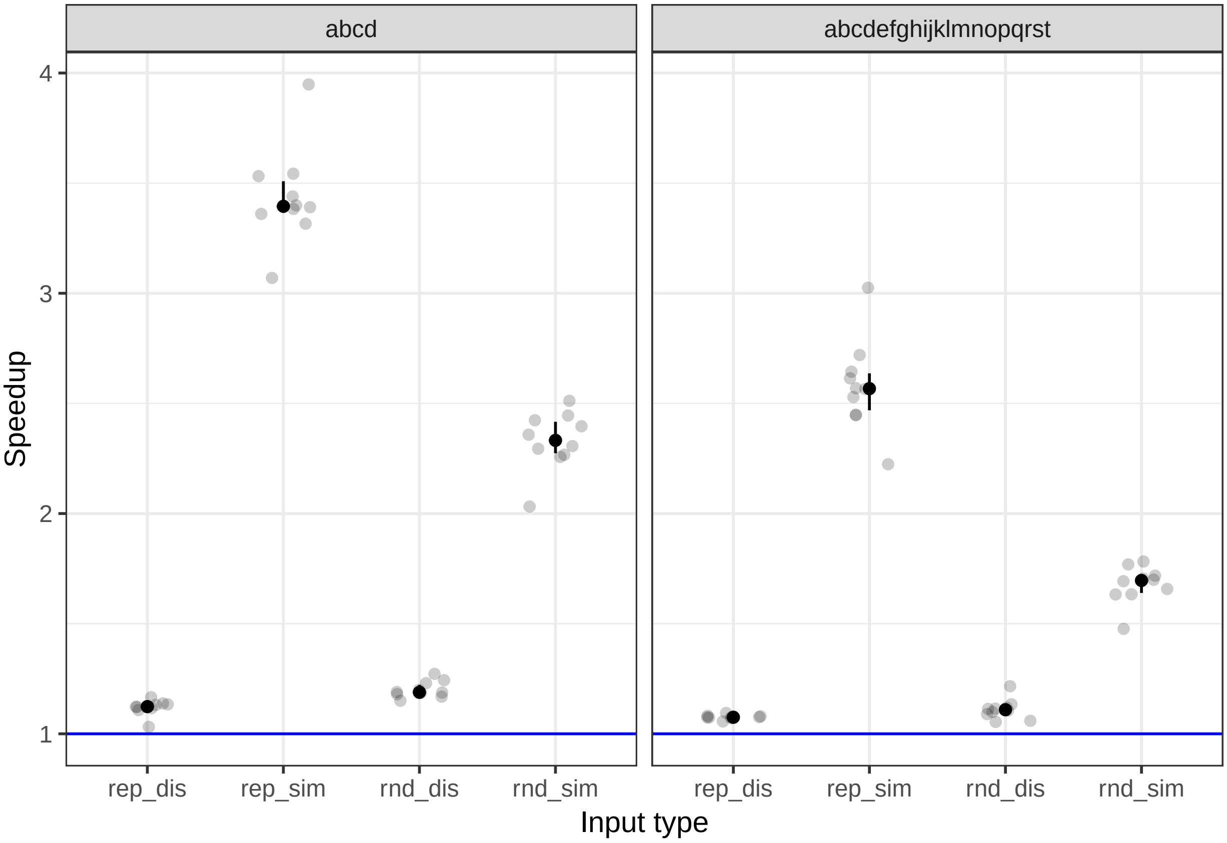
2.3 Lazy versus nonlazy



2.4 Double rank and fail versus double rank

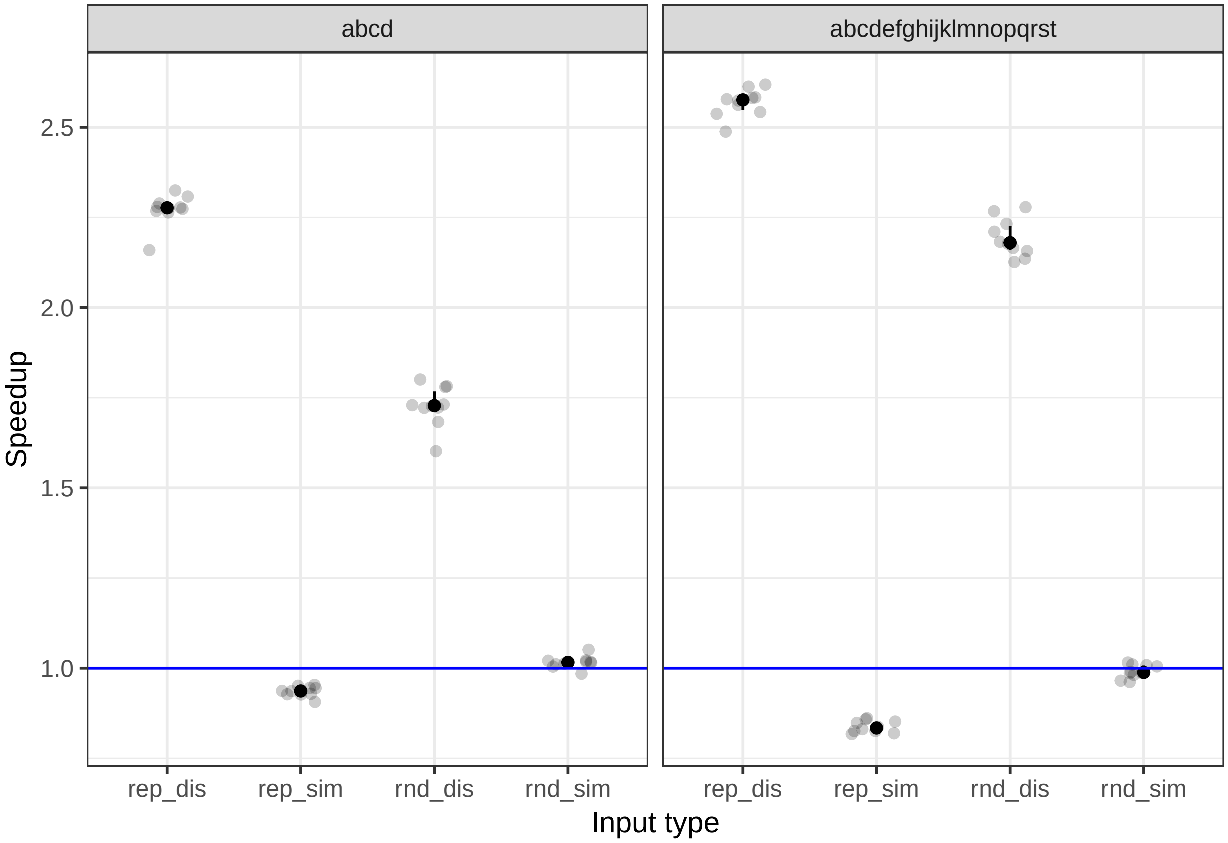


2.5 Double rank, Lazy, and Rank and fail versus Single rank, nonlazy, and nonfail

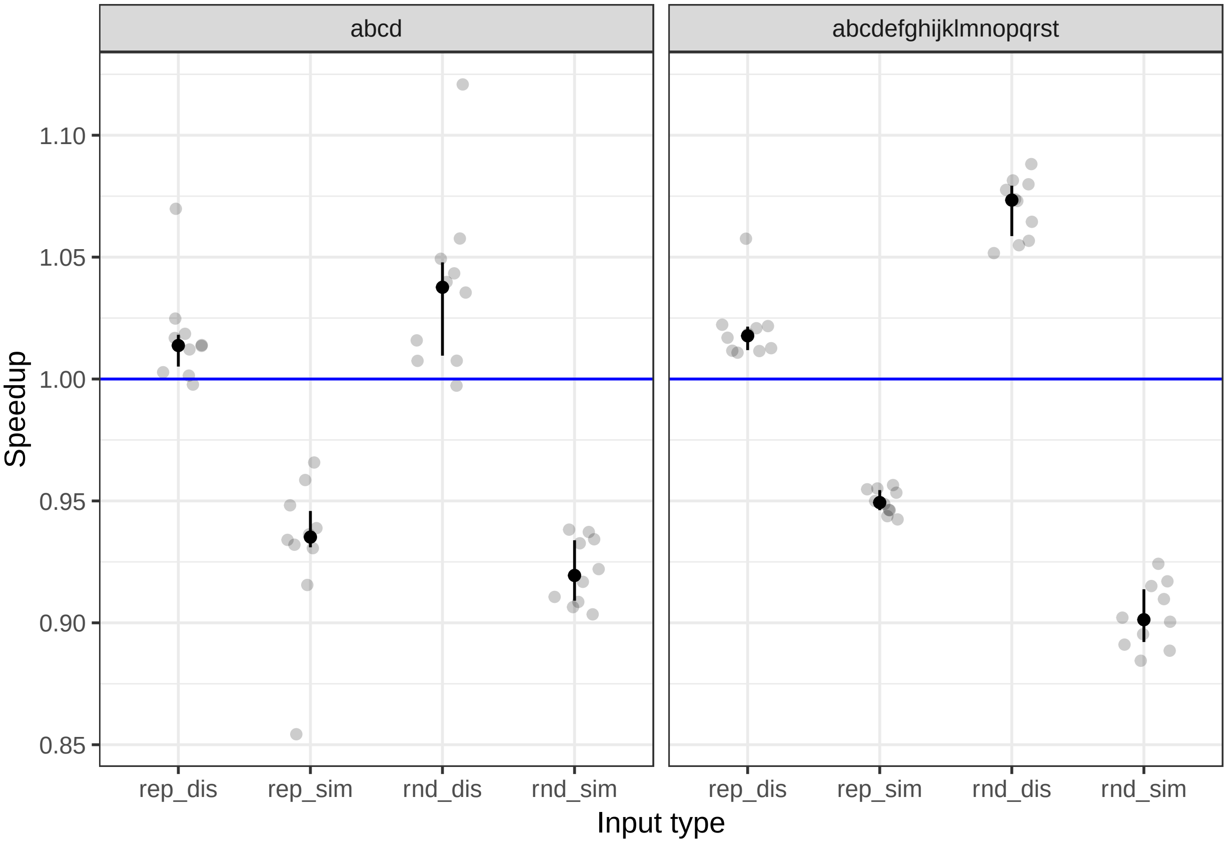


2.6 Maxrep

2.6.1 Vanilla Maxrep vs. non-maxrep



2.6.2 Rank&check Maxrep versus Vanilla Maxrep



3 Optimizations on parent operations

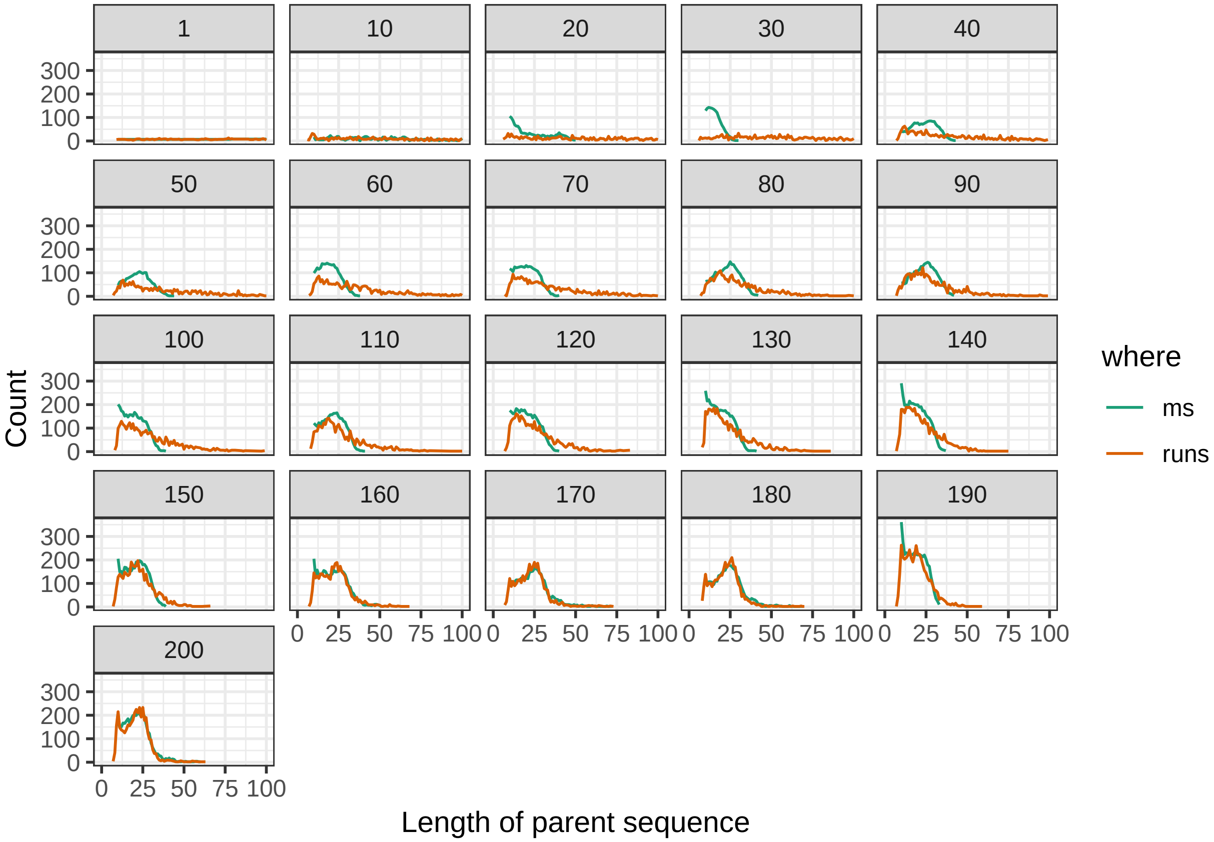
3.1 Input data

We generate the index input string with repetitions as follows. We generate a random seed block b of length 200. Next, we generate blocks of the same length b_k by introducing k mutations on b. The index string of length 10MB is $b \circ b_k^{(1)} \ldots \circ b_k^{(4999)}$.

The query string is obtained as a concatenation of labels from nodes of the suffix tree of s. We select nodes with node depth of at least 10 and string length at most 170 for a total string length of 103KB. We separate the labels with a sentinel character that does not appear in s.

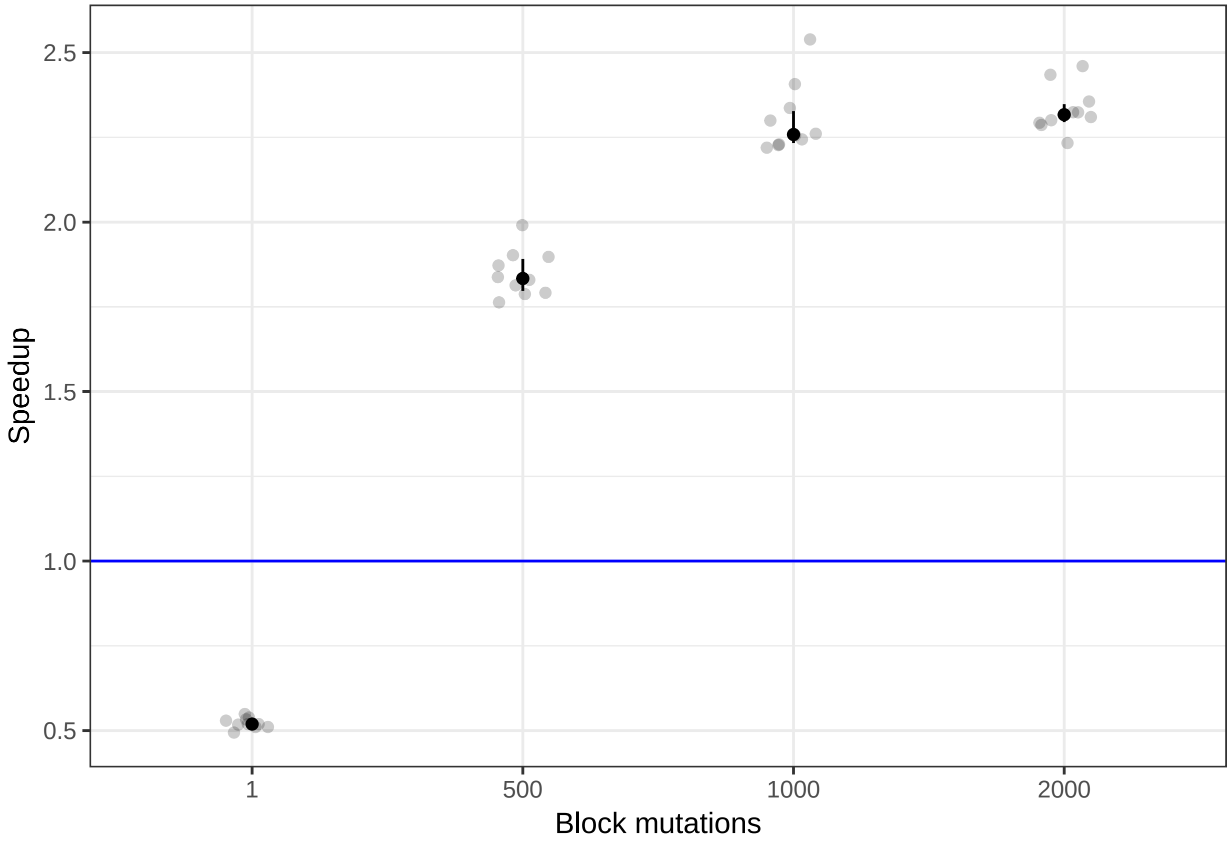
Furthermore, we perform experiments for various choices of $1 \geq k \geq |b|$.

The plot below shows a histogram of the length of consecutive parent operations. This quantity is important since the speedup of this optimization is proportional to the length of sequence of parent operations. Importantly, the optimization might not even be beneficial if the length of the sequence of parent operations is less than 3.



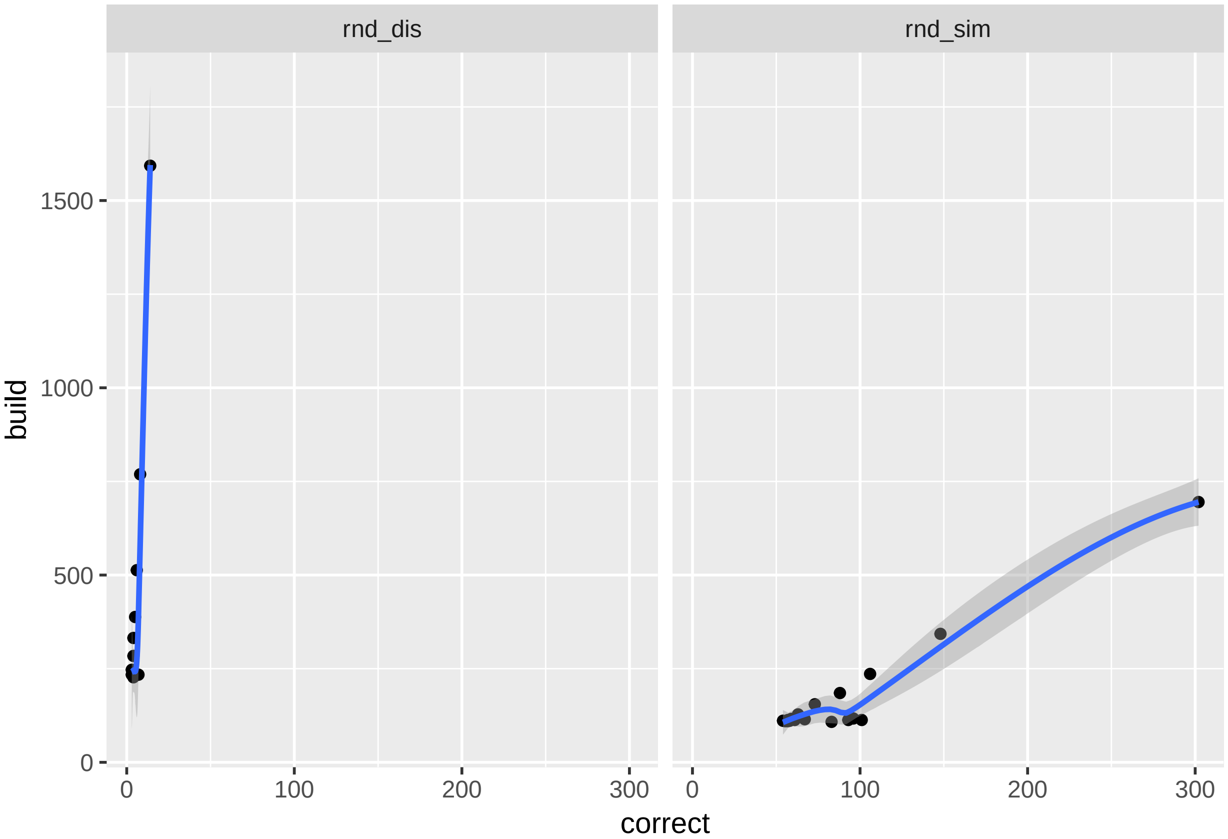
3.2 LCA versus parent sequence

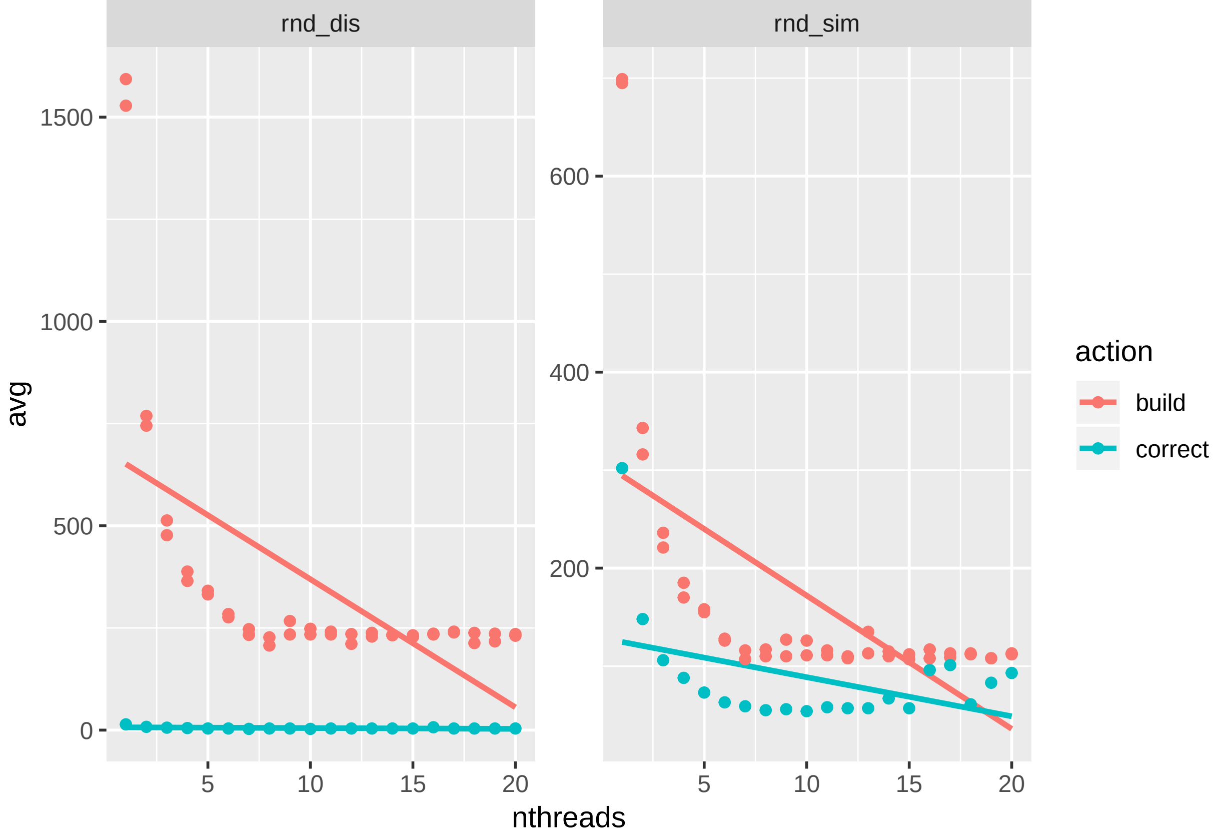
```
## # A tibble: 40 x 5
## # Groups:   ntrial [10]
##    ntrial k       lca  pseq value
##     <int> <fct> <dbl> <dbl> <dbl>
##  1      1 1       887   459 0.517
##  2      1 1000    207   476 2.30
##  3      1 2000    213   487 2.29
##  4      1 500     225   408 1.81
##  5      2 1       863   448 0.519
##  6      2 1000    209   469 2.24
##  7      2 2000    216   502 2.32
##  8      2 500     234   430 1.84
##  9      3 1       861   459 0.533
## 10      3 1000    205   479 2.34
## # ... with 30 more rows
```

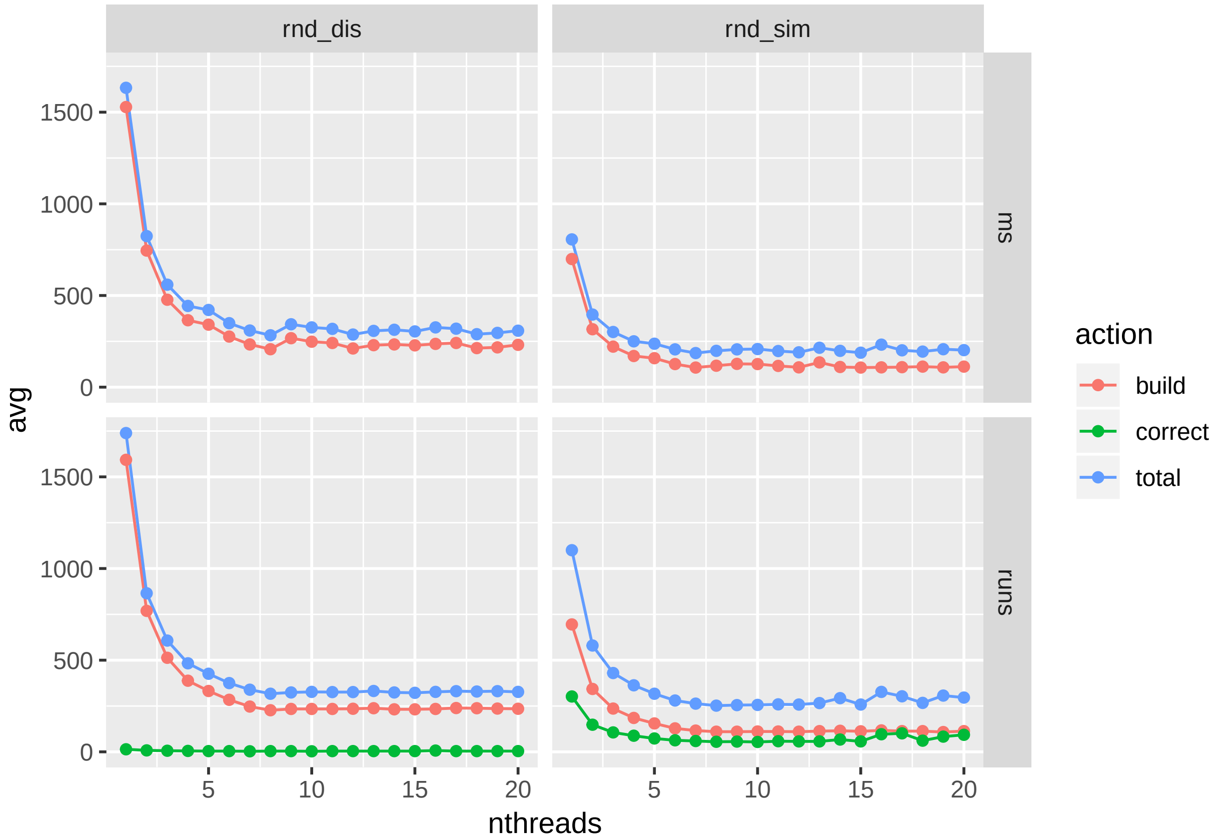


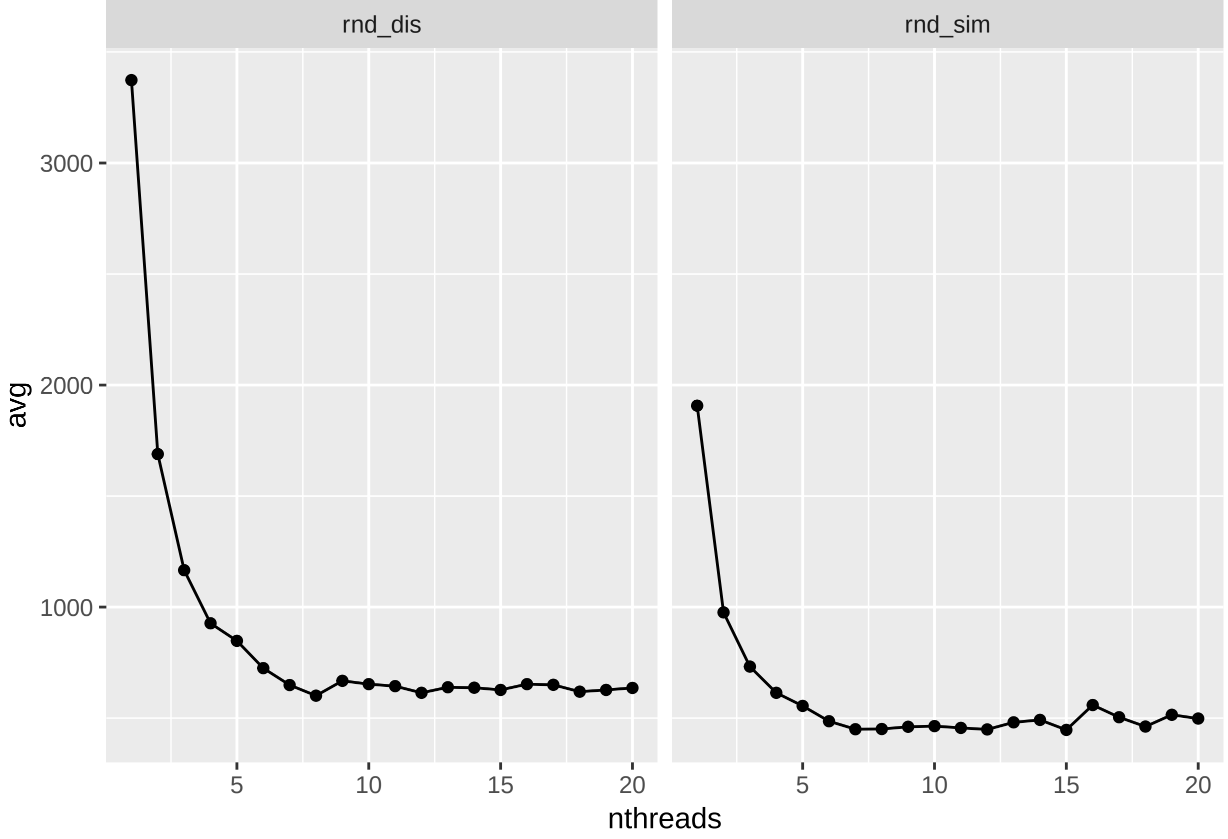
4 Parallelization

```
## # A tibble: 240 x 5
##    inp_type nthreads section action    avg
##    <chr>       <int> <chr>   <chr>   <dbl>
##  1 rnd_dis         1 comp    total    3373
##  2 rnd_dis         1 ms      build    1528
##  3 rnd_dis         1 ms      total    1633
##  4 rnd_dis         1 runs    build    1593
##  5 rnd_dis         1 runs    correct    14
##  6 rnd_dis         1 runs    total    1739
##  7 rnd_dis         2 comp    total    1689
##  8 rnd_dis         2 ms      build     745
##  9 rnd_dis         2 ms      total     824
## 10 rnd_dis         2 runs    build     769
## # ... with 230 more rows
```


rnd_dis
rnd_sim
build
1500
1000
500
0
0
100
200
300
0
100
200
300
correct

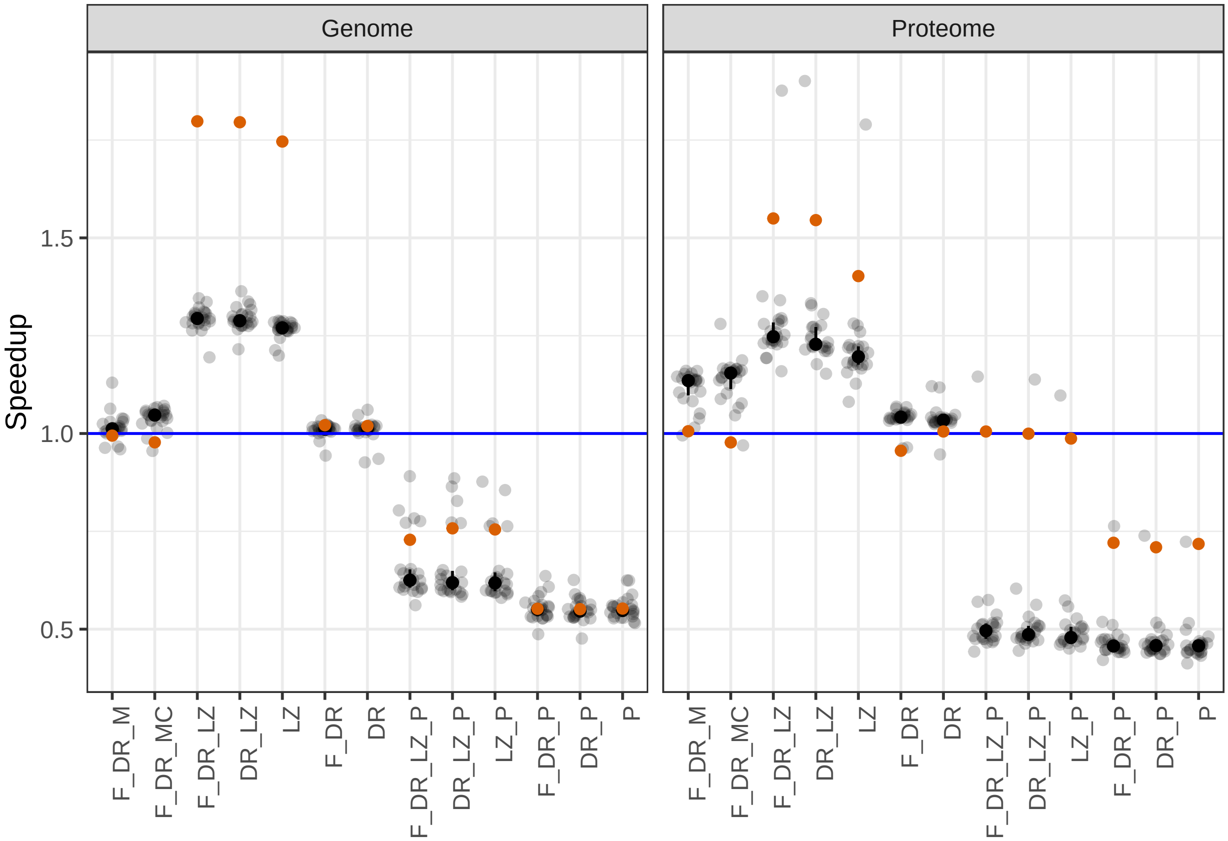

rnd_dis
rnd_sim
avg
1500
1000
500
0
600
400
200
5
10
15
20
5
10
15
20
nthreads
action
build
correct


rnd_dis
rnd_sim
ms
runs
avg
1500
1000
500
0
5
10
15
20
nthreads
action
build
correct
total

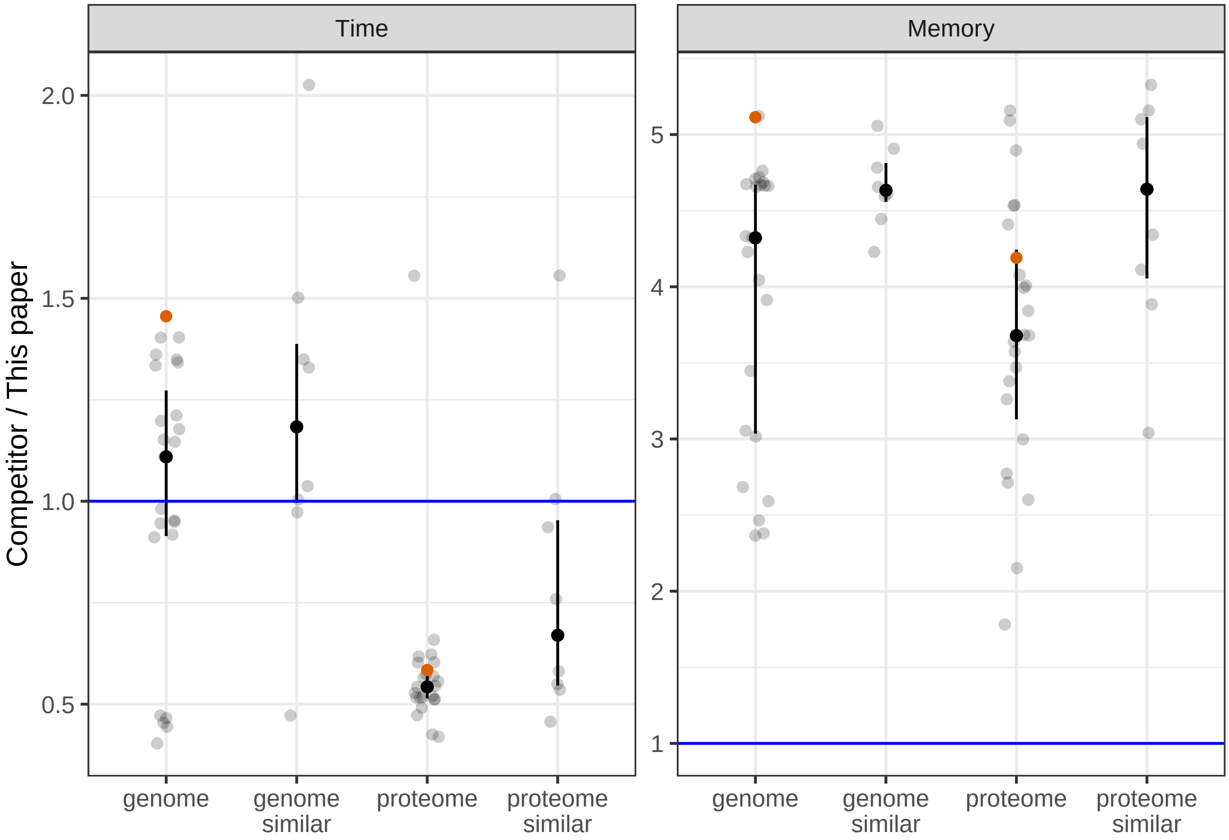

rnd_dis
rnd_sim
avg
3000
2000
1000
5
10
15
20
nthreads

5 Genome tests

5.1 Figure 4



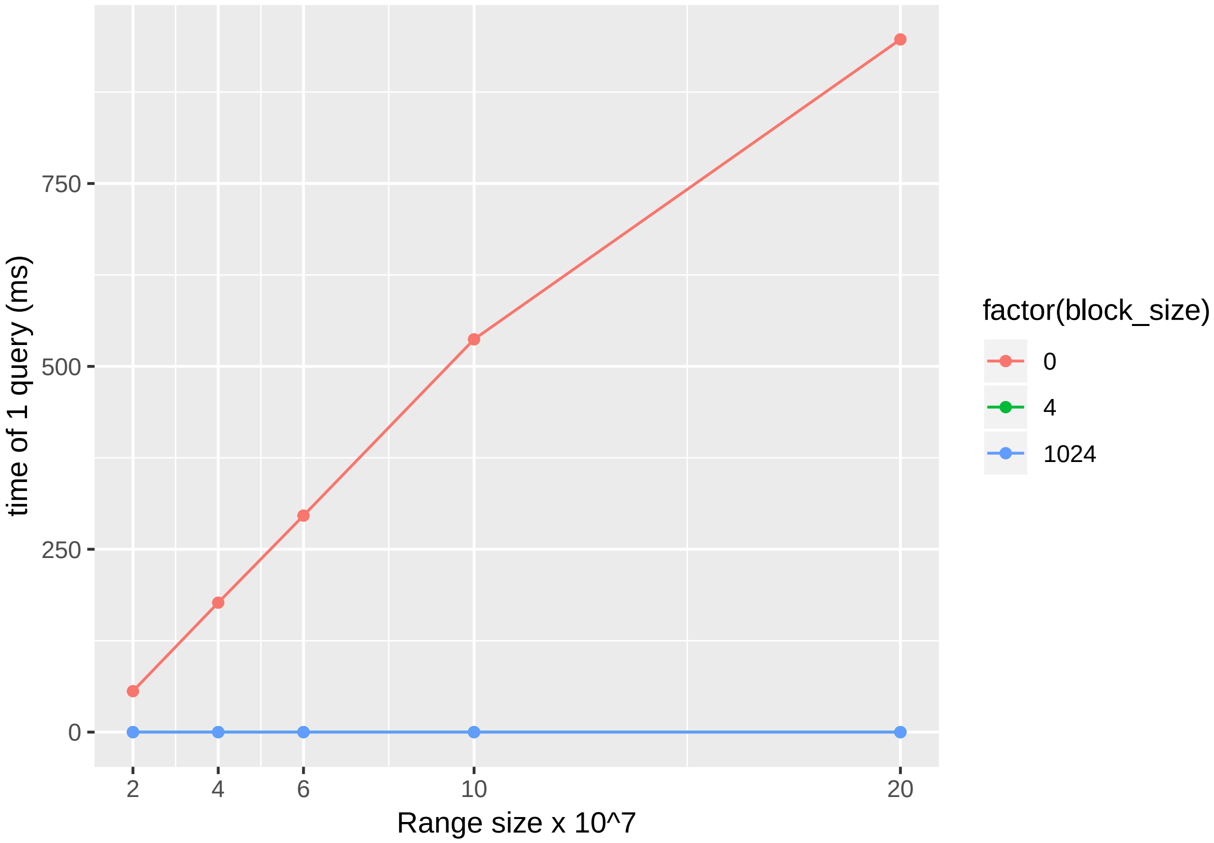
5.2 Figure 5



5.3 parallel on real data

6 Range queries

```
## # A tibble: 15 x 5
##    block_size range_size nqueries time_ms time_per_query
##         <dbl>      <dbl>    <dbl>   <dbl>          <dbl>
##  1          0   20000000        1      56      56
##  2          0   40000000        1     177     177
##  3          0   60000000        1     296     296
##  4          0  100000000        1     537     537
##  5          0  200000000        1     947     947
##  6          4   20000000  1000000     328       0.000328
##  7          4   40000000  1000000     352       0.000352
##  8          4   60000000  1000000     372       0.000372
##  9          4  100000000  1000000     391       0.000391
## 10          4  200000000  1000000     391       0.000391
## 11       1024   20000000  1000000     327       0.000327
## 12       1024   40000000  1000000     344       0.000344
## 13       1024   60000000  1000000     366       0.000366
## 14       1024  100000000  1000000     387       0.000387
## 15       1024  200000000  1000000     392       0.000392
```


time of 1 query (ms)
750
500
250
0
2
4
6
10
20
Range size x 10^7
factor(block_size)
0
4
1024