Irregular alignment of arbitrarily long DNA sequences using GPUs

Supplementary Material

Esteban Pérez-Wohlfeil[1,2], Oswaldo Trelles[2] and Nicolás Guil[2]
1. Dynatrace Research, Linz, Austria
2. Computer Architecture Department, University of Málaga, Málaga, Spain.

From words to alignments

Figure 1 shows the processing flow of GECKO from words to seeds and alignments. The filtering of seeds is not shown, but as explained in the manuscript, it is performed by removing those seeds which are in the same diagonal (*i.e.* $d = x_q - y_r$) and whose separation is less than twice the size of the seed.

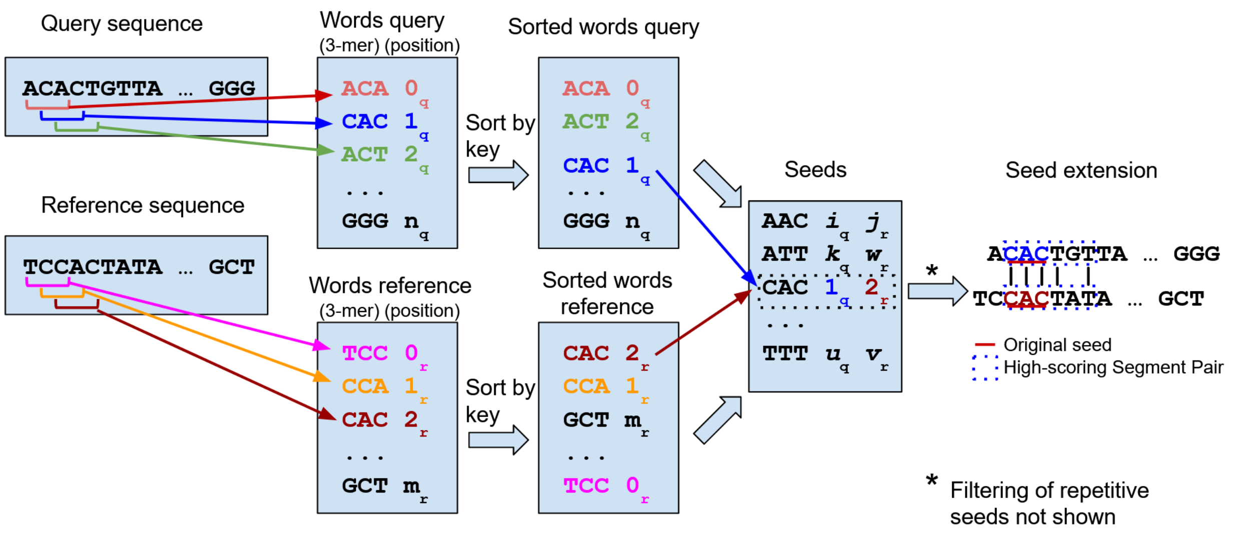


Figure 1. Seed-and-extend algorithm in GECKO. Note that the filtering of close and repetitive seeds is not shown.

Probability of splitting HSPs

In order to find the probability of a random HSP falling within the boundaries of a subsequence, we ran a Monte Carlo simulation using the following parameters as obtained from a *Homo sapiens* chr. X versus *Mus musculus* chr. X comparison:

1. We generated $n = 15,199,400$ HSPs as detected by GPUGECKO using a minimum of 64 bp of length.
2. The position of the HSPs followed a uniform distribution.
3. The length of the HSPs followed an exponential distribution as obtained from the comparison (see Figure 2).
4. The 2-D cartesian map of the comparison was split along the x and y axis every 21,538,133 base pairs, which is the default boundary size for subsequences for a GPU device with 4 GB using a factor of 0.125.

Each HSP was then placed randomly according to the uniform distribution in the comparison of size 151,099,878 by 163,484,862 base pairs.

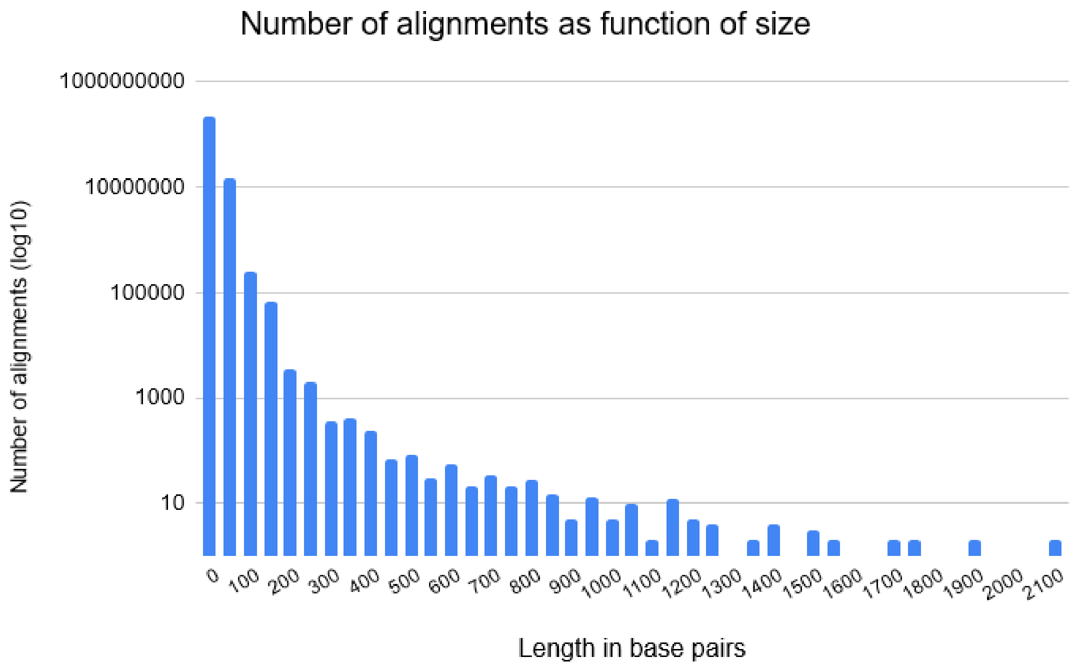


Figure 2. Distribution of the length of HSPs in the comparison between Homo sapiens chr. X and Mus musculus chr. X. Notice that the y-axis is in $\log_{10}$ scale.

Length	Count
0-50	221,316,314
50-100	14,804,333
100-150	256,911
150-200	67,192
200-250	3,424
250-300	2,003
300-350	353
350-400	407
400-450	230
More than 450	430

Table 1. Exact values of HSP count in the Homo sapiens chr. X and Mus musculus chr. X comparison.

We used the following formula to calculate the quadrant at which an HSP begins (the same formula applies in the case of the ending quadrant but changing x_{start} and y_{start} to x_{end} and y_{end}):

$$total_{quadrants} = x_{len} / window_{size}$$
$$x_{slice} = x_{start} / window_{size}$$
$$y_{slice} = y_{start} / window_{size}$$
$$quadrant = total_{quadrants} * x_{slice} * y_{slice}$$

This formula indexes quadrants by columns, *i.e.* the quadrants of the first column are numbered from 0 to n, the ones in the second are numbered from n to $2n$, etc. We ran ten simulations to average the proportion of HSPs being split in half (Table 2).

Execution number	Split HSPs	Total	Percentage (%)
1	179	15,199,400	0.00117767806 %
2	168	15,199,400	0.00110530678 %
3	166	15,199,400	0.00109214837 %
4	202	15,199,400	0.00132899982 %
5	179	15,199,400	0.00117767806 %
6	166	15,199,400	0.00109214837 %
7	181	15,199,400	0.00119083648 %
8	156	15,199,400	0.00102635630 %
9	178	15,199,400	0.00117109885 %
10	164	15,199,400	0.00107898996 %
Average	173.9 (174)	15,199,400	0.00114478203 %

Table 2. Ten executions of the Monte Carlo simulation of the probability of HSPs overlapping the boundaries of subsequences in a GPUGECKO execution.

As we can see, the average percentage of HSPs that are split in half is 0.00114478203 %. While it cannot be generalized to all other sequence comparisons (in fact it should be taken into account if sequences are more or less related), we believe this value is small enough in order to ignore split HSPs especially when, if needed, these split HSPs can be merged back (with a single linear pass in the size of the HSPs) into one since they are in the same diagonal and have a distance between them of a maximum of 64 bp.

A table containing two examples of split HSPs is shown below in Table 3. These were extracted from the Monte Carlo simulation.

xStart	yStart	xEnd	yEnd	Start quadrant	End quadrant
48,622	43,076,209	48,750	43,076,337	7 * 0 + 1 = 1	7 * 0 + 2 = 2
129,228,761	39,444,015	129,228,889	39,444,143	7 * 5 + 1 = 36	7 * 6 + 1 = 43

Table 3. Calculation of subsequence quadrants in example HSPs.

Custom memory allocator in MGPU mergesort

The following listing shows the changes performed to the mergesort algorithm of MGPU:

```
#include <inttypes.h>

typedef struct mem_pool{

        char * mem_ptr;
        uint64_t address;
        uint64_t limit;

} Mem_pool;

virtual void* alloc(size_t size, memory_space_t space) {
        void* p = nullptr;
        if(size) {
        cudaError_t result;
        if(memory_space_device == space)
        {
        if(mptr != NULL)
        {
        uint64_t new_address = mptr->address + 32 - (mptr->address % 32);
        if(new_address + (uint64_t) size >= mptr->limit) { fprintf(stderr, "Not enough memory in
pool\n"); exit(-1); }
        uint64_t t_occupied = (new_address + (uint64_t) size) -  mptr->address;
        allocs.push(t_occupied);
        p = (void *) (mptr->mem_ptr + new_address);
        mptr->address += t_occupied;
        result = cudaSuccess;
        }
        else
        result = cudaMalloc(&p, size);
        }
        else
        result = cudaMallocHost(&p, size);

        if(cudaSuccess != result) throw cuda_exception_t(result);
        }
        return p;
  }

virtual void free(void* p, memory_space_t space) {
        if(p) {
        cudaError_t result;
        if(memory_space_device == space)
        {
        if(mptr != NULL)
        {
        uint64_t last_size = allocs.top();
        allocs.pop();
        mptr->address -= last_size;
        result = cudaSuccess;
        }
        else
        result = cudaFree(p);
        }
        else
        result = cudaFreeHost(p);
        if(cudaSuccess != result) throw cuda_exception_t(result);
        }
  }
```
Listing 1. Custom memory allocator in MGPU mergesort.

Reference numbers of the sample dataset

Table 4 includes the reference identifier number for each sequence employed in the sample dataset along with the species name and the database from which it was obtained.

Species name	Database	Reference number
Mycoplasma hyopneumoniae 232	RefSeq	gi|54019969|ref|NC_006360.1|
Mycoplasma hyopneumoniae 7422	RefSeq	gi|525903163|ref|NC_021831.1|
Escherichia coli B	RefSeq	NZ_CP014268.2 Escherichia coli B

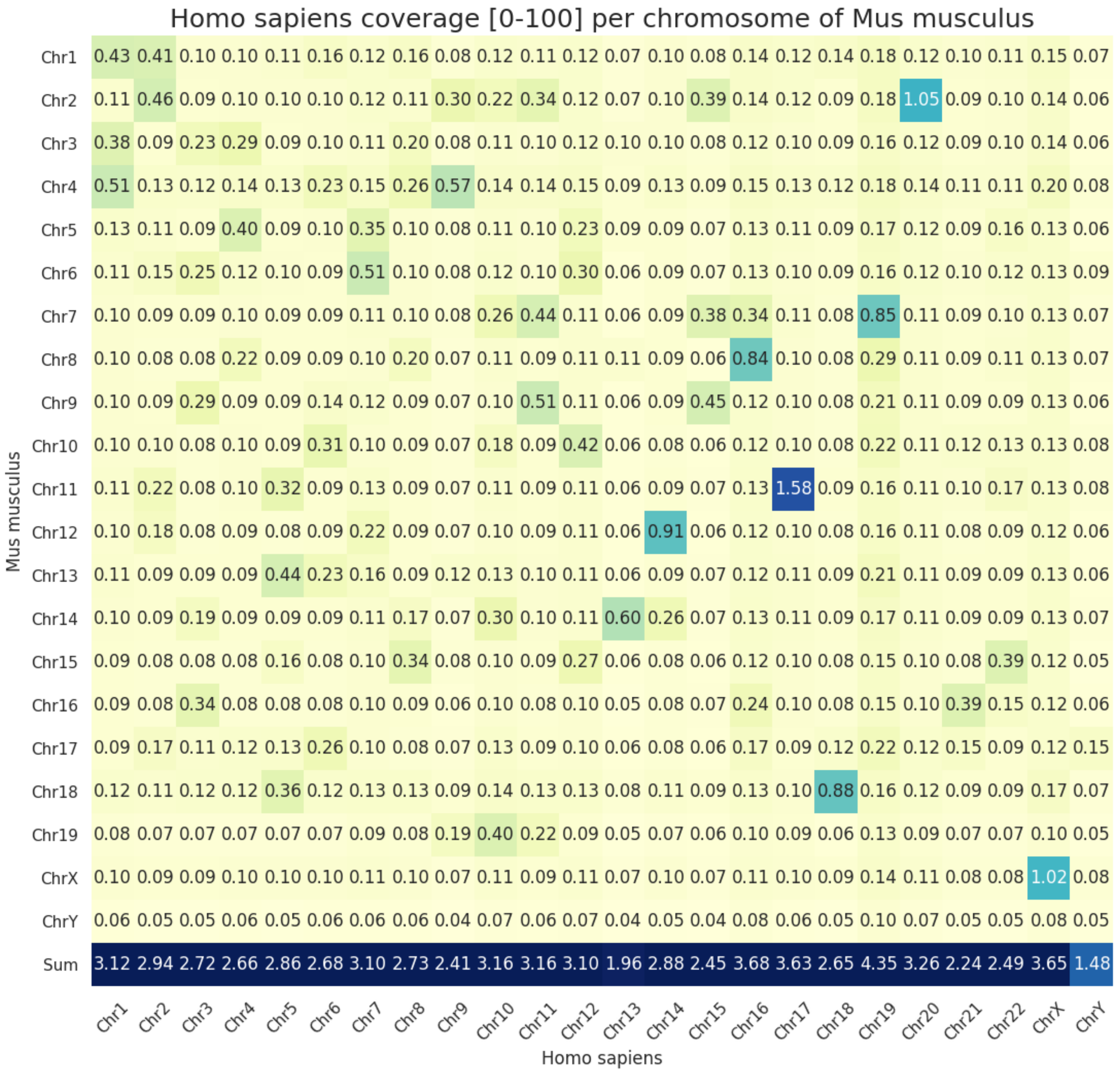
		strain C2566, complete genome
Escherichia coli K12	RefSeq	NZ_CP009789.1 Escherichia coli K-12 strain ER3413, complete genome
Gallus gallus 18	Ensembl	18 dna:chromosome chromosome:WASHUC2:18:1:10925261:1 REF
Meleagris gallopavo 20	Ensembl	20 dna:chromosome chromosome:UMD2:20:1:11078015:1 REF
Oryzias latipes 6	Ensembl	6 dna:chromosome chromosome:MEDAKA1:6:1:26576615:1 REF
Danio rerio 25	Ensembl	25 dna:chromosome chromosome:GRCz11:25:1:37502051:1 REF
Sus scrofa 11	Ensembl	11 dna:chromosome chromosome:Sscrofa11.1:11:1:79169978:1 REF
Bos taurus 12	Ensembl	12 dna:chromosome chromosome:UMD3.1:12:1:91163125:1 REF
Homo sapiens X	Ensembl	X dna:chromosome chromosome:GRCh37:X:1:155270560:1 REF
Mus musculus X	Ensembl	X dna:chromosome chromosome:GRCm38:X:1:171031299:1 REF
Homo sapiens 1	Ensembl	1 dna:chromosome chromosome:GRCh37:1:1:249250621:1 REF
Gorilla gorilla 1	Ensembl	1 dna:chromosome chromosome:gorGor3.1:1:1:229507203:1 REF

Table 4. Access reference numbers for each sequence in the sample dataset.

All-vs-all chromosome comparison

As described in the main manuscript, one of the capabilities of GPUGECKO is to perform massive all-vs-all chromosome comparisons. For demonstration purposes, we show two cases here, namely (1) *Homo sapiens* vs *Mus musculus* and (2) *Homo sapiens* vs *Gorilla gorilla*. Figure 1 shows the heatmap for the *Homo sapiens* vs *Mus musculus* comparison. Notice that coverage is computed only for the query, *i.e.* how much of the query can be found in the reference, and it is calculated as the number of shared bases in the alignments (with a minimum length of 128 and 512 for both sequences, respectively) and divided by the length of the query. Coverage is a well-known metric to estimate how similar two sequences are [1].

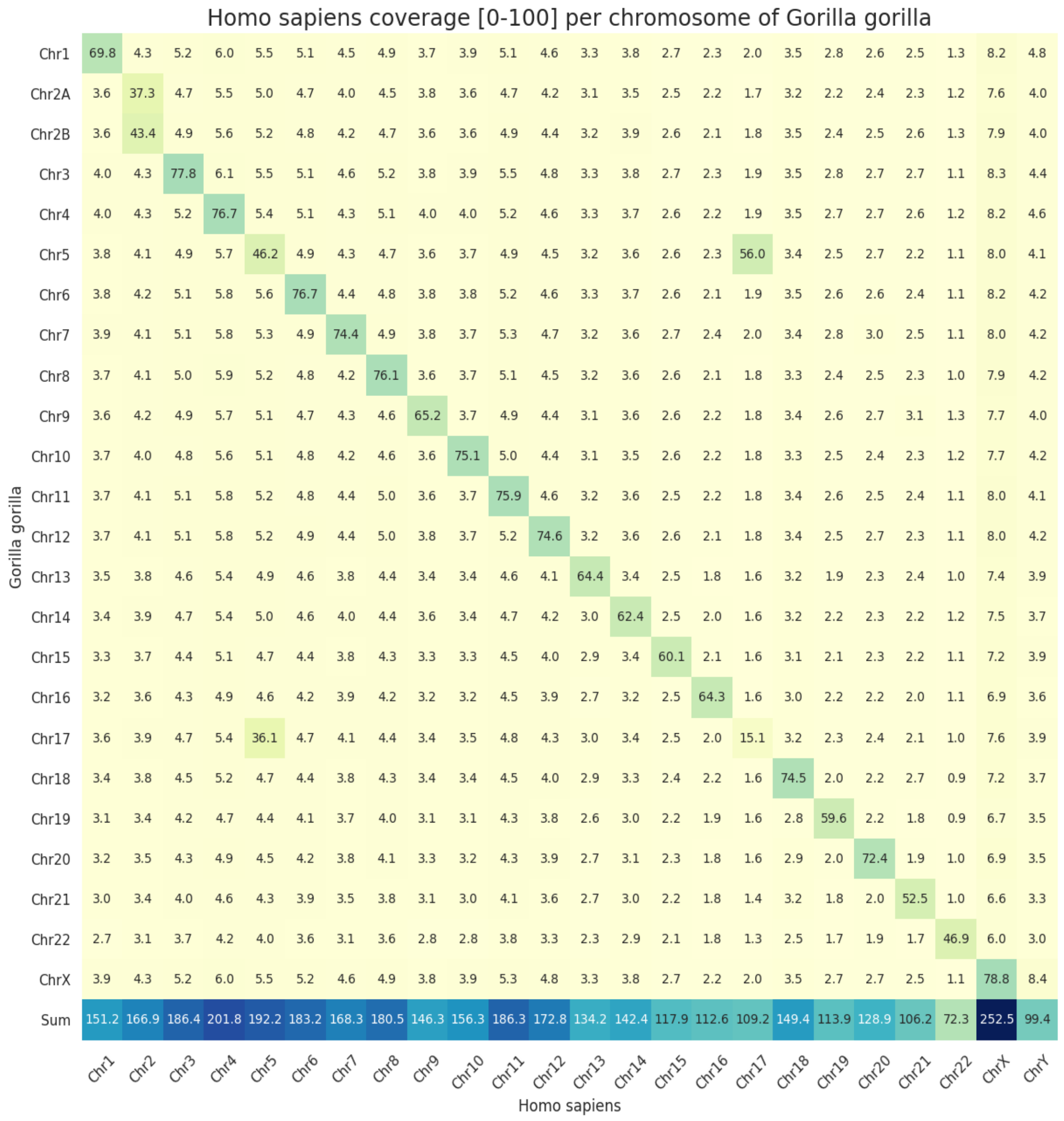
The exhaustive comparison of *Homo sapiens* vs *Mus musculus* took 3 hours and 39 minutes whereas the *Homo sapiens* vs *Gorilla gorilla* took 4 hours and 18 minutes. The increase in runtime is due to the much larger number of alignments found between primates, as can be seen in an increase in coverage (see Figure 2). On the contrary, the executions including seed-skipping policies took 1 hour and 50 minutes and 2 hours and 24 minutes, respectively. In both cases, the speedup is approximately up to ~4x without sacrificing coverage significantly (see the differences in coverage in section “All-vs-all chromosome comparison employing seed-skipping policies” later on).



	Chr1	Chr2	Chr3	Chr4	Chr5	Chr6	Chr7	Chr8	Chr9	Chr10	Chr11	Chr12	Chr13	Chr14	Chr15	Chr16	Chr17	Chr18	Chr19	Chr20	Chr21	Chr22	ChrX	ChrY
Chr1	0.43	0.41	0.10	0.10	0.11	0.16	0.12	0.16	0.08	0.12	0.11	0.12	0.07	0.10	0.08	0.14	0.12	0.14	0.18	0.12	0.10	0.11	0.15	0.07
Chr2	0.11	0.46	0.09	0.10	0.10	0.10	0.12	0.11	0.30	0.22	0.34	0.12	0.07	0.10	0.39	0.14	0.12	0.09	0.18	1.05	0.09	0.10	0.14	0.06
Chr3	0.38	0.09	0.23	0.29	0.09	0.10	0.11	0.20	0.08	0.11	0.10	0.12	0.10	0.10	0.08	0.12	0.10	0.09	0.16	0.12	0.09	0.10	0.14	0.06
Chr4	0.51	0.13	0.12	0.14	0.13	0.23	0.15	0.26	0.57	0.14	0.14	0.15	0.09	0.13	0.09	0.15	0.13	0.12	0.18	0.14	0.11	0.11	0.20	0.08
Chr5	0.13	0.11	0.09	0.40	0.09	0.10	0.35	0.10	0.08	0.11	0.10	0.23	0.09	0.09	0.07	0.13	0.11	0.09	0.17	0.12	0.09	0.16	0.13	0.06
Chr6	0.11	0.15	0.25	0.12	0.10	0.09	0.51	0.10	0.08	0.12	0.10	0.30	0.06	0.09	0.07	0.13	0.10	0.09	0.16	0.12	0.10	0.12	0.13	0.09
Chr7	0.10	0.09	0.09	0.10	0.09	0.09	0.11	0.10	0.08	0.26	0.44	0.11	0.06	0.09	0.38	0.34	0.11	0.08	0.85	0.11	0.09	0.10	0.13	0.07
Chr8	0.10	0.08	0.08	0.22	0.09	0.09	0.10	0.20	0.07	0.11	0.09	0.11	0.11	0.09	0.06	0.84	0.10	0.08	0.29	0.11	0.09	0.11	0.13	0.07
Chr9	0.10	0.09	0.29	0.09	0.09	0.14	0.12	0.09	0.07	0.10	0.51	0.11	0.06	0.09	0.45	0.12	0.10	0.08	0.21	0.11	0.09	0.09	0.13	0.06
Chr10	0.10	0.10	0.08	0.10	0.09	0.31	0.10	0.09	0.07	0.18	0.09	0.42	0.06	0.08	0.06	0.12	0.10	0.08	0.22	0.11	0.12	0.13	0.13	0.08
Chr11	0.11	0.22	0.08	0.10	0.32	0.09	0.13	0.09	0.07	0.11	0.09	0.11	0.06	0.09	0.07	0.13	1.58	0.09	0.16	0.11	0.10	0.17	0.13	0.08
Chr12	0.10	0.18	0.08	0.09	0.08	0.09	0.22	0.09	0.07	0.10	0.09	0.11	0.06	0.91	0.06	0.12	0.10	0.08	0.16	0.11	0.08	0.09	0.12	0.06
Chr13	0.11	0.09	0.09	0.09	0.44	0.23	0.16	0.09	0.12	0.13	0.10	0.11	0.06	0.09	0.07	0.12	0.11	0.09	0.21	0.11	0.09	0.09	0.13	0.06
Chr14	0.10	0.09	0.19	0.09	0.09	0.09	0.11	0.17	0.07	0.30	0.10	0.11	0.60	0.26	0.07	0.13	0.11	0.09	0.17	0.11	0.09	0.09	0.13	0.07
Chr15	0.09	0.08	0.08	0.08	0.16	0.08	0.10	0.34	0.08	0.10	0.09	0.27	0.06	0.08	0.06	0.12	0.10	0.08	0.15	0.10	0.08	0.39	0.12	0.05
Chr16	0.09	0.08	0.34	0.08	0.08	0.08	0.10	0.09	0.06	0.10	0.08	0.10	0.05	0.08	0.07	0.24	0.10	0.08	0.15	0.10	0.39	0.15	0.12	0.06
Chr17	0.09	0.17	0.11	0.12	0.13	0.26	0.10	0.08	0.07	0.13	0.09	0.10	0.06	0.08	0.06	0.17	0.09	0.12	0.22	0.12	0.15	0.09	0.12	0.15
Chr18	0.12	0.11	0.12	0.12	0.36	0.12	0.13	0.13	0.09	0.14	0.13	0.13	0.08	0.11	0.09	0.13	0.10	0.88	0.16	0.12	0.09	0.09	0.17	0.07
Chr19	0.08	0.07	0.07	0.07	0.07	0.07	0.09	0.08	0.19	0.40	0.22	0.09	0.05	0.07	0.06	0.10	0.09	0.06	0.13	0.09	0.07	0.07	0.10	0.05
ChrX	0.10	0.09	0.09	0.10	0.10	0.10	0.11	0.10	0.07	0.11	0.09	0.11	0.07	0.10	0.07	0.11	0.10	0.09	0.14	0.11	0.08	0.08	1.02	0.08
ChrY	0.06	0.05	0.05	0.06	0.05	0.06	0.06	0.06	0.04	0.07	0.06	0.07	0.04	0.05	0.04	0.08	0.06	0.05	0.10	0.07	0.05	0.05	0.08	0.05
Sum	3.12	2.94	2.72	2.66	2.86	2.68	3.10	2.73	2.41	3.16	3.16	3.10	1.96	2.88	2.45	3.68	3.63	2.65	4.35	3.26	2.24	2.49	3.65	1.48

Figure 3. Coverage heatmap of the all-vs-all chromosome comparison between *Homo sapiens* and *Mus musculus*. The coverage is in the range $[0, 100]$. The last row is the sum of the coverage per each human chromosome which is **not** in the range $[0, 100]$ but rather $[0, n * 100]$ where n is the number of chromosomes of *Mus musculus*.

The same comparison is performed exhaustively for *Homo sapiens* and *Gorilla gorilla*, see Figure 4 below.



	Chr1	Chr2	Chr3	Chr4	Chr5	Chr6	Chr7	Chr8	Chr9	Chr10	Chr11	Chr12	Chr13	Chr14	Chr15	Chr16	Chr17	Chr18	Chr19	Chr20	Chr21	Chr22	ChrX	ChrY
Chr1	69.8	4.3	5.2	6.0	5.5	5.1	4.5	4.9	3.7	3.9	5.1	4.6	3.3	3.8	2.7	2.3	2.0	3.5	2.8	2.6	2.5	1.3	8.2	4.8
Chr2A	3.6	37.3	4.7	5.5	5.0	4.7	4.0	4.5	3.8	3.6	4.7	4.2	3.1	3.5	2.5	2.2	1.7	3.2	2.2	2.4	2.3	1.2	7.6	4.0
Chr2B	3.6	43.4	4.9	5.6	5.2	4.8	4.2	4.7	3.6	3.6	4.9	4.4	3.2	3.9	2.6	2.1	1.8	3.5	2.4	2.5	2.6	1.3	7.9	4.0
Chr3	4.0	4.3	77.8	6.1	5.5	5.1	4.6	5.2	3.8	3.9	5.5	4.8	3.3	3.8	2.7	2.3	1.9	3.5	2.8	2.7	2.7	1.1	8.3	4.4
Chr4	4.0	4.3	5.2	76.7	5.4	5.1	4.3	5.1	4.0	4.0	5.2	4.6	3.3	3.7	2.6	2.2	1.9	3.5	2.7	2.7	2.6	1.2	8.2	4.6
Chr5	3.8	4.1	4.9	5.7	46.2	4.9	4.3	4.7	3.6	3.7	4.9	4.5	3.2	3.6	2.6	2.3	56.0	3.4	2.5	2.7	2.2	1.1	8.0	4.1
Chr6	3.8	4.2	5.1	5.8	5.6	76.7	4.4	4.8	3.8	3.8	5.2	4.6	3.3	3.7	2.6	2.1	1.9	3.5	2.6	2.6	2.4	1.1	8.2	4.2
Chr7	3.9	4.1	5.1	5.8	5.3	4.9	74.4	4.9	3.8	3.7	5.3	4.7	3.2	3.6	2.7	2.4	2.0	3.4	2.8	3.0	2.5	1.1	8.0	4.2
Chr8	3.7	4.1	5.0	5.9	5.2	4.8	4.2	76.1	3.6	3.7	5.1	4.5	3.2	3.6	2.6	2.1	1.8	3.3	2.4	2.5	2.3	1.0	7.9	4.2
Chr9	3.6	4.2	4.9	5.7	5.1	4.7	4.3	4.6	65.2	3.7	4.9	4.4	3.1	3.6	2.6	2.2	1.8	3.4	2.6	2.7	3.1	1.3	7.7	4.0
Chr10	3.7	4.0	4.8	5.6	5.1	4.8	4.2	4.6	3.6	75.1	5.0	4.4	3.1	3.5	2.6	2.2	1.8	3.3	2.5	2.4	2.3	1.2	7.7	4.2
Chr11	3.7	4.1	5.1	5.8	5.2	4.8	4.4	5.0	3.6	3.7	75.9	4.6	3.2	3.6	2.5	2.2	1.8	3.4	2.6	2.5	2.4	1.1	8.0	4.1
Chr12	3.7	4.1	5.1	5.8	5.2	4.9	4.4	5.0	3.8	3.7	5.2	74.6	3.2	3.6	2.6	2.1	1.8	3.4	2.5	2.7	2.3	1.1	8.0	4.2
Chr13	3.5	3.8	4.6	5.4	4.9	4.6	3.8	4.4	3.4	3.4	4.6	4.1	64.4	3.4	2.5	1.8	1.6	3.2	1.9	2.3	2.4	1.0	7.4	3.9
Chr14	3.4	3.9	4.7	5.4	5.0	4.6	4.0	4.4	3.6	3.4	4.7	4.2	3.0	62.4	2.5	2.0	1.6	3.2	2.2	2.3	2.2	1.2	7.5	3.7
Chr15	3.3	3.7	4.4	5.1	4.7	4.4	3.8	4.3	3.3	3.3	4.5	4.0	2.9	3.4	60.1	2.1	1.6	3.1	2.1	2.3	2.2	1.1	7.2	3.9
Chr16	3.2	3.6	4.3	4.9	4.6	4.2	3.9	4.2	3.2	3.2	4.5	3.9	2.7	3.2	2.5	64.3	1.6	3.0	2.2	2.2	2.0	1.1	6.9	3.6
Chr17	3.6	3.9	4.7	5.4	36.1	4.7	4.1	4.4	3.4	3.5	4.8	4.3	3.0	3.4	2.5	2.0	15.1	3.2	2.3	2.4	2.1	1.0	7.6	3.9
Chr18	3.4	3.8	4.5	5.2	4.7	4.4	3.8	4.3	3.4	3.4	4.5	4.0	2.9	3.3	2.4	2.2	1.6	74.5	2.0	2.2	2.7	0.9	7.2	3.7
Chr19	3.1	3.4	4.2	4.7	4.4	4.1	3.7	4.0	3.1	3.1	4.3	3.8	2.6	3.0	2.2	1.9	1.6	2.8	59.6	2.2	1.8	0.9	6.7	3.5
Chr20	3.2	3.5	4.3	4.9	4.5	4.2	3.8	4.1	3.3	3.2	4.3	3.9	2.7	3.1	2.3	1.8	1.6	2.9	2.0	72.4	1.9	1.0	6.9	3.5
Chr21	3.0	3.4	4.0	4.6	4.3	3.9	3.5	3.8	3.1	3.0	4.1	3.6	2.7	3.0	2.2	1.8	1.4	3.2	1.8	2.0	52.5	1.0	6.6	3.3
Chr22	2.7	3.1	3.7	4.2	4.0	3.6	3.1	3.6	2.8	2.8	3.8	3.3	2.3	2.9	2.1	1.8	1.3	2.5	1.7	1.9	1.7	46.9	6.0	3.0
ChrX	3.9	4.3	5.2	6.0	5.5	5.2	4.6	4.9	3.8	3.9	5.3	4.8	3.3	3.8	2.7	2.2	2.0	3.5	2.7	2.7	2.5	1.1	78.8	8.4
Sum	151.2	166.9	186.4	201.8	192.2	183.2	168.3	180.5	146.3	156.3	186.3	172.8	134.2	142.4	117.9	112.6	109.2	149.4	113.9	128.9	106.2	72.3	252.5	99.4

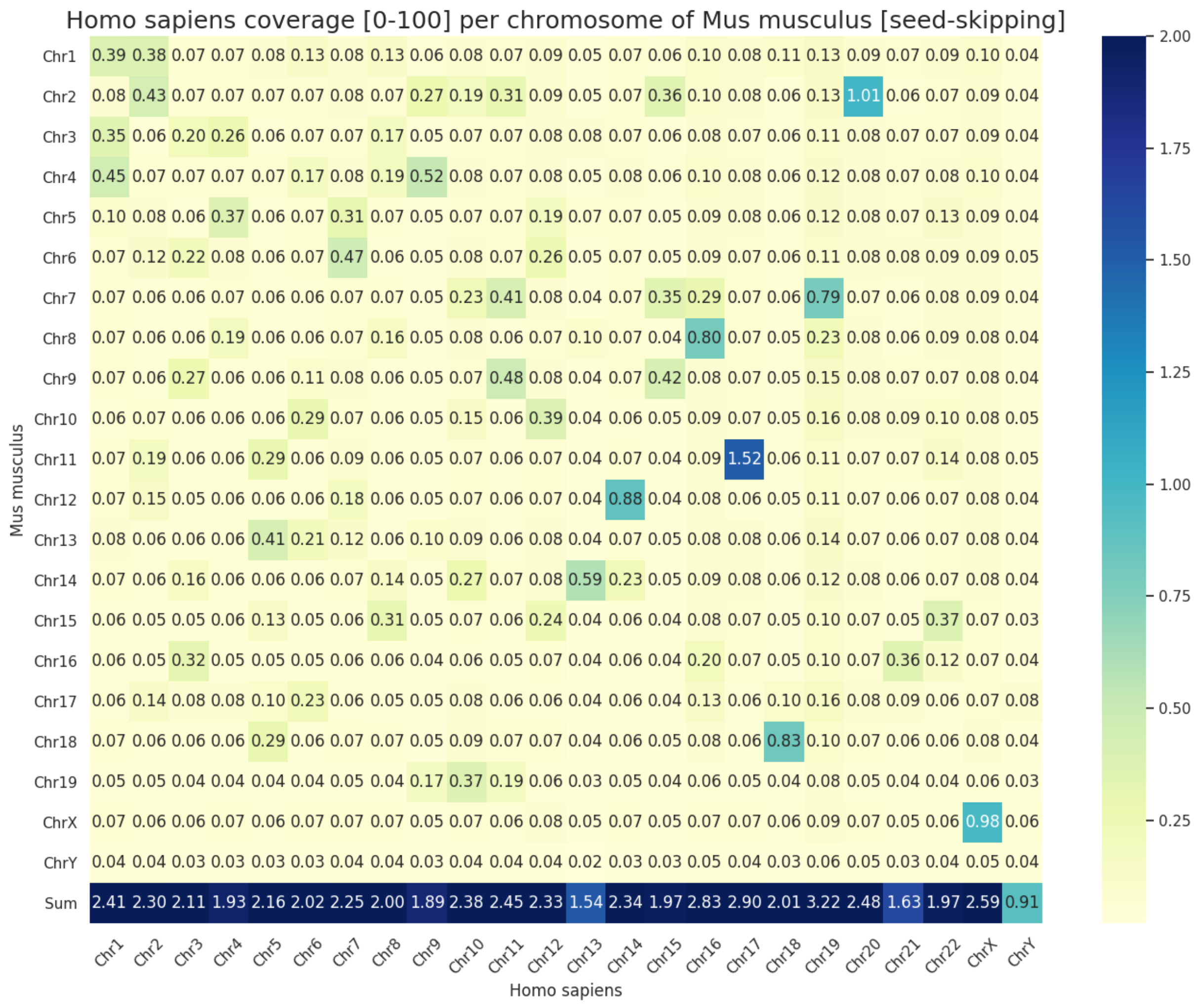
Figure 4. Coverage heatmap of the all-vs-all chromosome comparison between *Homo sapiens* and *Gorilla gorilla*. The coverage is in the range $[0, 100]$. The last row is the sum of the coverage per each human chromosome which is **not** in the range $[0, 100]$ but rather $[0, n * 100]$ where n is the number of chromosomes of *Gorilla gorilla*.

As can be seen, less evolutionary rearrangements have occurred between *Homo sapiens* vs *Gorilla gorilla* compared to *Homo sapiens* and *Mus musculus*, as expected given previous phylogenies. Notice that to run the whole species comparison, the following executions between chromosomes required fine-tuning of the seeds factor parameter (due to the massive number of repetitions found): h4-g4 ($f = 0.0085$), h10-g4 ($f = 0.04$), h21-g4 ($f = 0.06$), hY-g4 ($f = 0.02$

), where *h* stands for *Homo sapiens* and *g* stands for *Gorilla gorilla*. Notice that all comparisons involved the chromosome 4 of *Gorilla gorilla*.

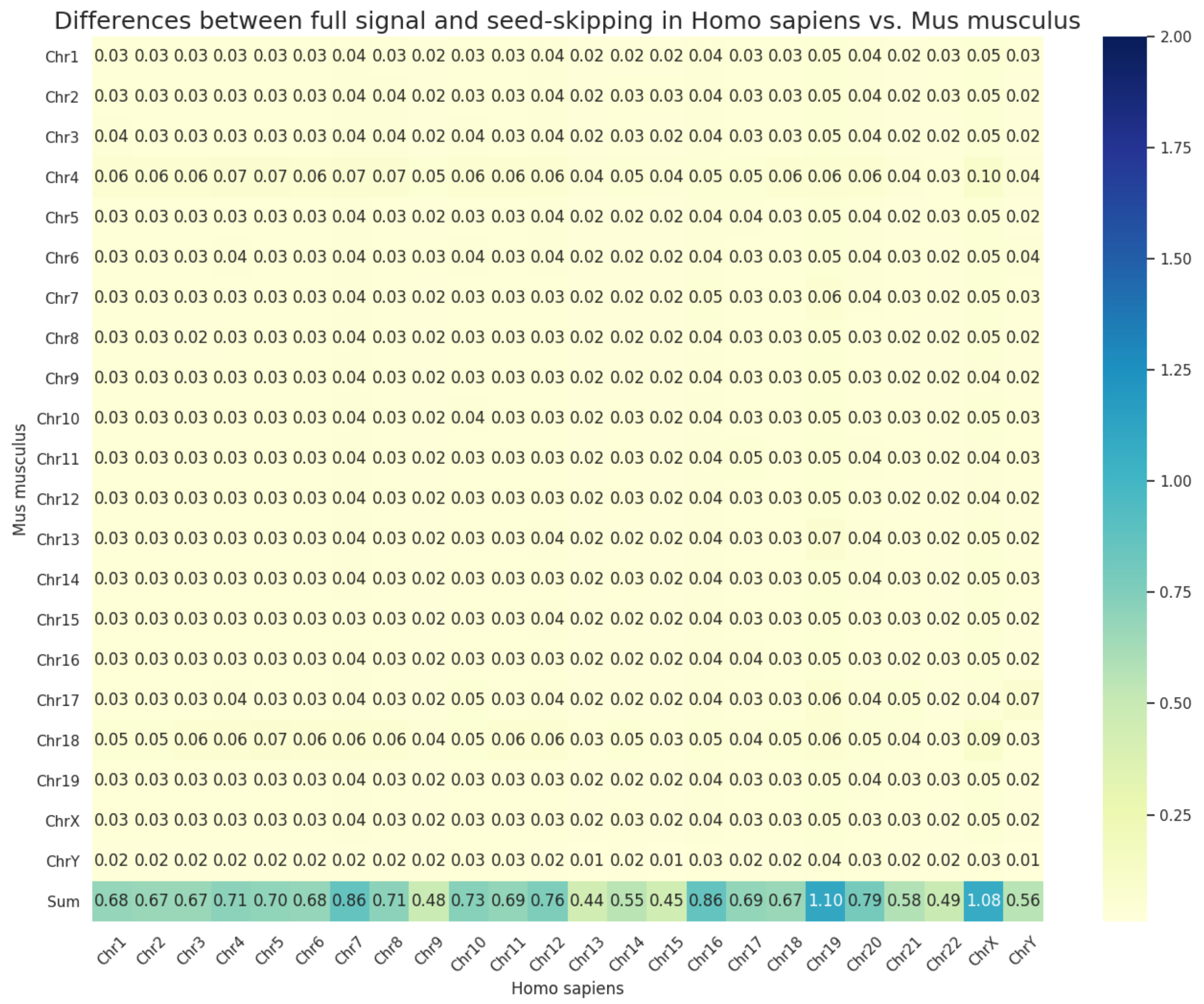
All-vs-all chromosome comparison employing seed-skipping policies

In this section, we compare the differences in coverage between the full signal analysis and the seed-skipping policy. Figure 5 shows the *Homo sapiens* vs *Mus musculus* comparison performed in less than two hours (110 minutes) with the seed-skipping policy. As can be seen, it is highly similar to the original comparison. To make the analysis easier, the same heatmap is used to plot the differences in coverage between the full and seed-skipping comparison, shown in Figure 6. As can be observed, the differences are in the range of 0.01 to 0.1 of signal lost (from 0% to 50% of the original signal). While the relative value of 50% might seem to be a lot of signal lost, these values are in fact nearly constant and can be credited to the amount of repeating regions lost rather than the main syntenies.



	Chr1	Chr2	Chr3	Chr4	Chr5	Chr6	Chr7	Chr8	Chr9	Chr10	Chr11	Chr12	Chr13	Chr14	Chr15	Chr16	Chr17	Chr18	Chr19	Chr20	Chr21	Chr22	ChrX	ChrY
Chr1	0.39	0.38	0.07	0.07	0.08	0.13	0.08	0.13	0.06	0.08	0.07	0.09	0.05	0.07	0.06	0.10	0.08	0.11	0.13	0.09	0.07	0.09	0.10	0.04
Chr2	0.08	0.43	0.07	0.07	0.07	0.07	0.08	0.07	0.27	0.19	0.31	0.09	0.05	0.07	0.36	0.10	0.08	0.06	0.13	1.01	0.06	0.07	0.09	0.04
Chr3	0.35	0.06	0.20	0.26	0.06	0.07	0.07	0.17	0.05	0.07	0.07	0.08	0.08	0.07	0.06	0.08	0.07	0.06	0.11	0.08	0.07	0.07	0.09	0.04
Chr4	0.45	0.07	0.07	0.07	0.07	0.17	0.08	0.19	0.52	0.08	0.07	0.08	0.05	0.08	0.06	0.10	0.08	0.06	0.12	0.08	0.07	0.08	0.10	0.04
Chr5	0.10	0.08	0.06	0.37	0.06	0.07	0.31	0.07	0.05	0.07	0.07	0.19	0.07	0.07	0.05	0.09	0.08	0.06	0.12	0.08	0.07	0.13	0.09	0.04
Chr6	0.07	0.12	0.22	0.08	0.06	0.07	0.47	0.06	0.05	0.08	0.07	0.26	0.05	0.07	0.05	0.09	0.07	0.06	0.11	0.08	0.08	0.09	0.09	0.05
Chr7	0.07	0.06	0.06	0.07	0.06	0.06	0.07	0.07	0.05	0.23	0.41	0.08	0.04	0.07	0.35	0.29	0.07	0.06	0.79	0.07	0.06	0.08	0.09	0.04
Chr8	0.07	0.06	0.06	0.19	0.06	0.06	0.07	0.16	0.05	0.08	0.06	0.07	0.10	0.07	0.04	0.80	0.07	0.05	0.23	0.08	0.06	0.09	0.08	0.04
Chr9	0.07	0.06	0.27	0.06	0.06	0.11	0.08	0.06	0.05	0.07	0.48	0.08	0.04	0.07	0.42	0.08	0.07	0.05	0.15	0.08	0.07	0.07	0.08	0.04
Chr10	0.06	0.07	0.06	0.06	0.06	0.29	0.07	0.06	0.05	0.15	0.06	0.39	0.04	0.06	0.05	0.09	0.07	0.05	0.16	0.08	0.09	0.10	0.08	0.05
Chr11	0.07	0.19	0.06	0.06	0.29	0.06	0.09	0.06	0.05	0.07	0.06	0.07	0.04	0.07	0.04	0.09	1.52	0.06	0.11	0.07	0.07	0.14	0.08	0.05
Chr12	0.07	0.15	0.05	0.06	0.06	0.06	0.18	0.06	0.05	0.07	0.06	0.07	0.04	0.88	0.04	0.08	0.06	0.05	0.11	0.07	0.06	0.07	0.08	0.04
Chr13	0.08	0.06	0.06	0.06	0.41	0.21	0.12	0.06	0.10	0.09	0.06	0.08	0.04	0.07	0.05	0.08	0.08	0.06	0.14	0.07	0.06	0.07	0.08	0.04
Chr14	0.07	0.06	0.16	0.06	0.06	0.06	0.07	0.14	0.05	0.27	0.07	0.08	0.59	0.23	0.05	0.09	0.08	0.06	0.12	0.08	0.06	0.07	0.08	0.04
Chr15	0.06	0.05	0.05	0.06	0.13	0.05	0.06	0.31	0.05	0.07	0.06	0.24	0.04	0.06	0.04	0.08	0.07	0.05	0.10	0.07	0.05	0.37	0.07	0.03
Chr16	0.06	0.05	0.32	0.05	0.05	0.05	0.06	0.06	0.04	0.06	0.05	0.07	0.04	0.06	0.04	0.20	0.07	0.05	0.10	0.07	0.36	0.12	0.07	0.04
Chr17	0.06	0.14	0.08	0.08	0.10	0.23	0.06	0.05	0.05	0.08	0.06	0.06	0.04	0.06	0.04	0.13	0.06	0.10	0.16	0.08	0.09	0.06	0.07	0.08
Chr18	0.07	0.06	0.06	0.06	0.29	0.06	0.07	0.07	0.05	0.09	0.07	0.07	0.04	0.06	0.05	0.08	0.06	0.83	0.10	0.07	0.06	0.06	0.08	0.04
Chr19	0.05	0.05	0.04	0.04	0.04	0.04	0.05	0.04	0.17	0.37	0.19	0.06	0.03	0.05	0.04	0.06	0.05	0.04	0.08	0.05	0.04	0.04	0.06	0.03
ChrX	0.07	0.06	0.06	0.07	0.06	0.07	0.07	0.07	0.05	0.07	0.06	0.08	0.05	0.07	0.05	0.07	0.07	0.06	0.09	0.07	0.05	0.06	0.98	0.06
ChrY	0.04	0.04	0.03	0.03	0.03	0.03	0.04	0.04	0.03	0.04	0.04	0.04	0.02	0.03	0.03	0.05	0.04	0.03	0.06	0.05	0.03	0.04	0.05	0.04
Sum	2.41	2.30	2.11	1.93	2.16	2.02	2.25	2.00	1.89	2.38	2.45	2.33	1.54	2.34	1.97	2.83	2.90	2.01	3.22	2.48	1.63	1.97	2.59	0.91

Figure 5. Coverage heatmap of the all-vs-all chromosome comparison between *Homo sapiens* and *Mus musculus* employing seed-skipping policies corresponding to the fast mode in GPUGECKO. The coverage is in the range $[0, 100]$. The last row is the sum of the coverage per each human chromosome which is **not** in the range $[0, 100]$ but rather $[0, n * 100]$ where n is the number of chromosomes of *Mus musculus*.



	Chr1	Chr2	Chr3	Chr4	Chr5	Chr6	Chr7	Chr8	Chr9	Chr10	Chr11	Chr12	Chr13	Chr14	Chr15	Chr16	Chr17	Chr18	Chr19	Chr20	Chr21	Chr22	ChrX	ChrY
Chr1	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.04	0.02	0.02	0.02	0.04	0.03	0.03	0.05	0.04	0.02	0.03	0.05	0.03
Chr2	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.04	0.02	0.03	0.03	0.04	0.02	0.03	0.03	0.04	0.03	0.03	0.05	0.04	0.02	0.03	0.05	0.02
Chr3	0.04	0.03	0.03	0.03	0.03	0.03	0.04	0.04	0.02	0.04	0.03	0.04	0.02	0.03	0.02	0.04	0.03	0.03	0.05	0.04	0.02	0.02	0.05	0.02
Chr4	0.06	0.06	0.06	0.07	0.07	0.06	0.07	0.07	0.05	0.06	0.06	0.06	0.04	0.05	0.04	0.05	0.05	0.06	0.06	0.06	0.04	0.03	0.10	0.04
Chr5	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.04	0.02	0.02	0.02	0.04	0.04	0.03	0.05	0.04	0.02	0.03	0.05	0.02
Chr6	0.03	0.03	0.03	0.04	0.03	0.03	0.04	0.03	0.03	0.04	0.03	0.04	0.02	0.02	0.02	0.04	0.03	0.03	0.05	0.04	0.03	0.02	0.05	0.04
Chr7	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.03	0.02	0.02	0.02	0.05	0.03	0.03	0.06	0.04	0.03	0.02	0.05	0.03
Chr8	0.03	0.03	0.02	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.03	0.02	0.02	0.02	0.04	0.03	0.03	0.05	0.03	0.02	0.02	0.05	0.02
Chr9	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.03	0.02	0.02	0.02	0.04	0.03	0.03	0.05	0.03	0.02	0.02	0.04	0.02
Chr10	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.04	0.03	0.03	0.02	0.03	0.02	0.04	0.03	0.03	0.05	0.03	0.03	0.02	0.05	0.03
Chr11	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.03	0.02	0.03	0.02	0.04	0.05	0.03	0.05	0.04	0.03	0.02	0.04	0.03
Chr12	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.03	0.02	0.03	0.02	0.04	0.03	0.03	0.05	0.03	0.02	0.02	0.04	0.02
Chr13	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.04	0.02	0.02	0.02	0.04	0.03	0.03	0.07	0.04	0.03	0.02	0.05	0.02
Chr14	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.03	0.02	0.03	0.02	0.04	0.03	0.03	0.05	0.04	0.03	0.02	0.05	0.03
Chr15	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.03	0.03	0.04	0.02	0.02	0.02	0.04	0.03	0.03	0.05	0.03	0.03	0.02	0.05	0.02
Chr16	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.03	0.02	0.02	0.02	0.04	0.04	0.03	0.05	0.03	0.02	0.03	0.05	0.02
Chr17	0.03	0.03	0.03	0.04	0.03	0.03	0.04	0.03	0.02	0.05	0.03	0.04	0.02	0.02	0.02	0.04	0.03	0.03	0.06	0.04	0.05	0.02	0.04	0.07
Chr18	0.05	0.05	0.06	0.06	0.07	0.06	0.06	0.06	0.04	0.05	0.06	0.06	0.03	0.05	0.03	0.05	0.04	0.05	0.06	0.05	0.04	0.03	0.09	0.03
Chr19	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.03	0.02	0.02	0.02	0.04	0.03	0.03	0.05	0.04	0.03	0.03	0.05	0.02
ChrX	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.02	0.03	0.03	0.03	0.02	0.03	0.02	0.04	0.03	0.03	0.05	0.03	0.03	0.02	0.05	0.02
ChrY	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.03	0.03	0.02	0.01	0.02	0.01	0.03	0.02	0.02	0.04	0.03	0.02	0.02	0.03	0.01
Sum	0.68	0.67	0.67	0.71	0.70	0.68	0.86	0.71	0.48	0.73	0.69	0.76	0.44	0.55	0.45	0.86	0.69	0.67	1.10	0.79	0.58	0.49	1.08	0.56

Figure 6. Differences in coverage of the full comparison and the seed-skipping policy in the all-vs-all chromosome comparison between *Homo sapiens* and *Mus musculus*. The coverage is in the range $[0, 100]$. The last row is the sum of the coverage per each human chromosome which is **not** in the range $[0, 100]$ but rather $[0, n * 100]$ where n is the number of chromosomes of *Mus musculus*.

In the case of *Homo sapiens* vs *Gorilla gorilla*, GPUGECKO takes 2 hours and 24 minutes while using the seed-skipping policy (as opposed to 9 hours and 6 minutes for the exhaustive comparison). Figure 7 shows the seed-skipping comparison and Figure 8 shows the differences in coverage.

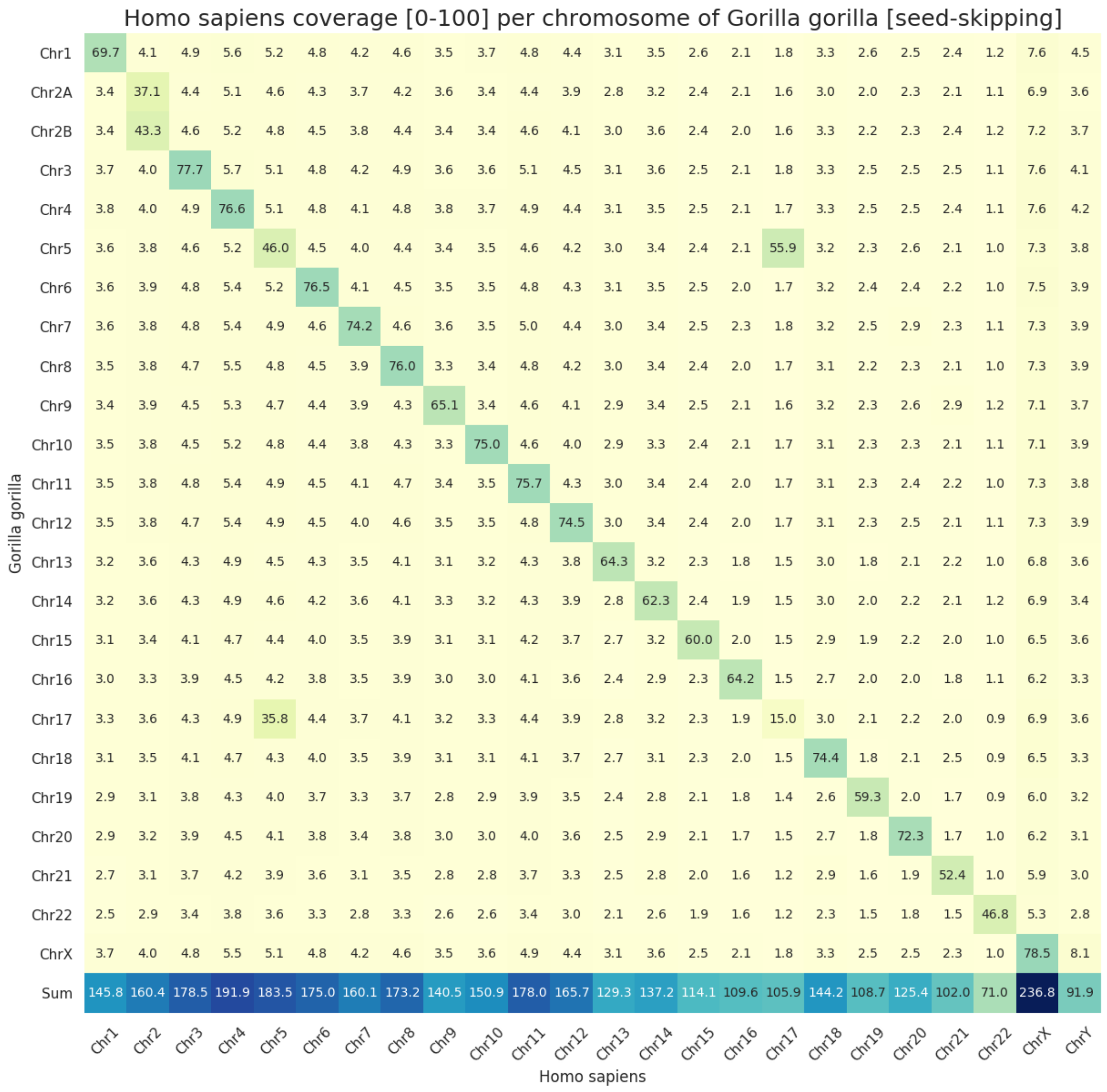


Figure 7. Coverage heatmap of the all-vs-all chromosome comparison between *Homo sapiens* and *Gorilla gorilla* using seed-skipping policies. The coverage is in the range $[0, 100]$. The last row is the sum of the coverage per each human chromosome which is **not** in the range $[0, 100]$ but rather $[0, n * 100]$ where n is the number of chromosomes of *Gorilla gorilla*.

Notice how the achieved coverage is nearly the same as in the exhaustive comparison while reducing runtime an additional ~4x. In fact, the differences (see Figure 4) are relatively smaller compared to the differences of *Homo sapiens* and *Mus musculus*, where up to 50% differences in coverage were achieved. In the case of *Homo sapiens* vs *Gorilla gorilla*, there are enough seeds (and final alignments are long enough) to make seed-skipping policy select, in most cases, a seed that yields the best alignment.

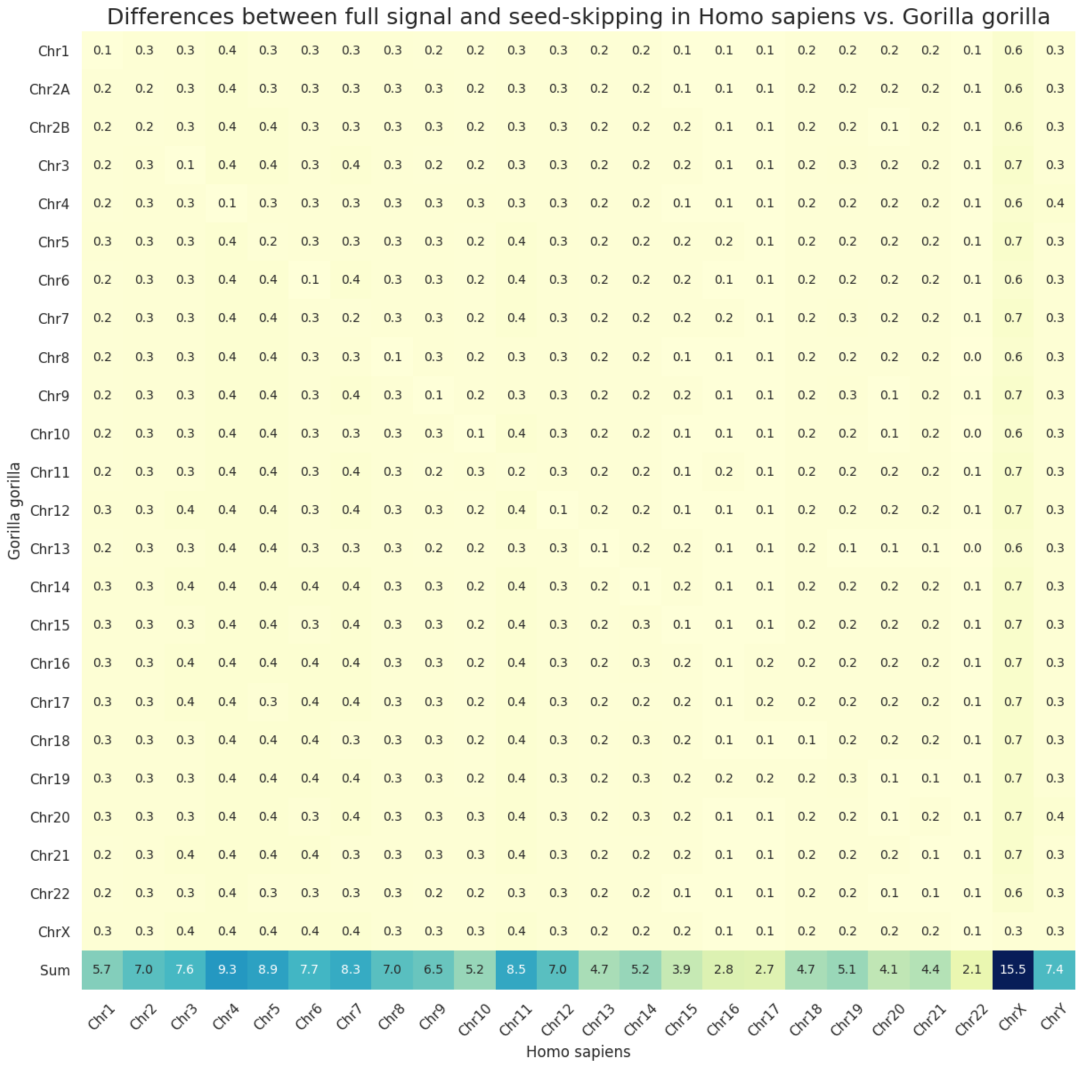


Figure 8. Differences in coverage of the full comparison and the seed-skipping policy in the all-vs-all chromosome comparison between *Homo sapiens* and *Gorilla gorilla*. The coverage is in the range $[0,\ 100]$. The last row is the sum of the coverage per each human chromosome which is **not** in the range $[0,\ 100]$ but rather $[0,\ n * 100]$ where n is the number of chromosomes of *Gorilla gorilla*.

Seed-skipping policies

Both GECKO and BLASTN (and GBLASTN as well) employ seed-skipping policies by default. These are aimed at removing highly repetitive seeds that do not contribute to conserved syntenies, but rather to repetitions. In the case of GECKO, highly-repetitive seeds (above a fixed threshold) are processed by selecting evenly spaced seeds following a uniform distribution. In the case of BLASTN, low-complexity regions are removed with the DUST program [2]. While GPUGECKO does not filter any repetitive seed by default (as this can lead to false beliefs regarding the comparison of two sequences), an option is also available for the case in which additional speed is desired and there is no interest in repeating regions. In this line, GPUGECKO implements two filtering modes based on one assumption: a seed which appears more than once can be considered a repetition (see [3]) and single seeds are considered as belonging to the conserved syntenies. The filtering modes proceed in two different ways:

1. A uniform selection of seeds based on the relative number of seeds (no threshold). Given a word that appears x and y times in each sequence (which in sensitive mode would result in $x * y$ seeds, quadratic complexity), the square root is taken for both so that the number of seeds is $\sqrt{x} * \sqrt{y}$, which keeps the amount of seeds in the same order of magnitude as the number of words. This means that the step in which words are skipped is the square root of the number of appearances of the particular word in each sequence, which enables to select them in an evenly-spaced fashion. For instance, consider the same word being repeated 10 and 20 times, respectively. While the original number of spawned seeds would rise to $10 * 20 = 200$, the suggested policy only requires$\sqrt{10} * \sqrt{20} \approx 14$. This number is furtherly reduced by dividing by a constant factor which was empirically determined. Overall, the equation of the number of seeds for the fast mode is shown below:
$$s = \left(\sqrt{x} * \sqrt{y}\right) \div 2$$
This working mode is able to reduce runtimes by 5x while only losing from 1 to 15% of coverage. This policy corresponds to the "--fast" parameter.

2. An only-first seed policy. This policy simply matches the query and reference words once, *i.e.* no word can be used twice. This working mode corresponds to the "--hyperfast" parameter, and should only be used to find the main syntenies.

State-of-the-art comparison employing seed-skipping policies

The comparison between GECKO, GBLASTN and GPUGECKO depicted in the main manuscript is now repeated employing the seed-skipping policies from each algorithm. Table 5 shows the runtime and coverage values.

Comparison	GECKO	GECKO	GBLASTN	GBLASTN	GPUGECKO	GPUGECKO
HYO-HYO	2.69 / 2.10	90.10	0.90 / 0.82	92.22	3.59 / 1.10	94.91
B-K12	10.01 / 1.85	93.14	2.61 / 1.51	91.06	3.75 / 1.12	94.65
GAL-MEL	13.90 / 2.22	49.82	44.79 / 1.15	58.23	4.01 / 1.07	59.29
ORY-DAN	39.44 / 3.98	0.21	11.09 / IND	0.15	4.72 / 1.93	0.22
SUS-BOS	98.18 / 2.47	5.48	67.78 / IND	5.59	8.29 / 1.66	5.00
HOM-MUS	179.10 / 7.66	0.96	57.88 / IND	0.87	16.58 / 3.65	0.98

HOM-GOR	2098.03 / 4.87	66.29	DNF	DNF	41.52 / 4.68	69.66

Table 5. Sequence comparison between GECKO, GBLASTN and GPUGECKO. The first column comprises the abbreviation of the sequence comparison. The following three column pairs represent the runtime (along with the speedup in regards to the full comparison) as well as the coverage of the reported alignments for each of the programs. The abbreviation “DNF” stands for “Did Not Finish”, whereas “IND” stands for “Indeterminate”.

There are several comments to be made on Table 5. In particular:

1. GBLASTN is still incapable of handling large comparisons and thus does not finish the *Homo sapiens* chr 1 vs *Gorilla gorilla* chr 1 comparison.
2. GPUGECKO is still the fastest except for small sequences (bacterial).
3. GPUGECKO achieves the highest coverage in most of the comparisons (4/6).

Finally, regarding speedup, GPUGECKO achieves the highest in long sequences when using seed-skipping policies, whereas GBLASTN achieves the highest for shorter sequences.

Comparison between the sequential GECKO and GPUGECKO

In this section, the speedup between the sequential CPU algorithm is compared against the proposed parallel GPU implementation. The per-kernel speedup is measured for each stage of the algorithm.

It can be observed from Table 6 and 7 that GPUGECKO achieves speedup in every kernel, ranging from around ~6x to ~300x, and averaging approximately an overall ~99x speedup (times include data transfers from host to device and vice versa). On one hand, the smallest speedup is obtained from the reverse complement kernel, which is expected, since the kernel requires virtually no work but the acceleration still needs to compensate for the host to device transfer.

Comparison	M. hyopneumoniae 232 - 7442			E. coli B12 - K		
Kernel	CPU	GPU	μ_s	CPU	GPU	μ_s
Reverse complement	0.087	0.0078	11.17	0.344	0.045	7.59
Words Dictionary	1.162	0.0085	136.76	4.248	0.014	297.30
Words Sorting	0.891	0.0041	216.14	3.668	0.023	157.55
Seeds Generation	1.621	0.0223	72.75	5.989	0.039	154.44

Seeds Sorting	0.187	0.0006	330.05	1.350	0.006	235.85
Seeds Filtering	0.020	0.0008	24.87	0.158	0.008	18.70
Seeds Extension	0.330	0.0019	176.70	0.514	0.021	24.25

Table 6. Per-kernel speedup comparison between GECKO and the proposed implementation GPUGECKO for the first two sequences of the sample dataset. Each multi-column represents a comparison and is further divided in 3 columns, namely (1) CPU time, (2) GPU time and (3) average speedup μ_s.

On the other hand, the sorting of both words and seeds, the extension of seeds and the generation of word dictionaries attain the highest acceleration (on average around ~160x and ~270x for the words and seed sorting, ~260x for the dictionary generation and ~140x for the seed extension). The sorting is expected to achieve such speedup (as it is a well-studied subject); particularly in the case of seeds where enough data is available (since the number of seeds is generally quadratic in terms of the number of words). Moreover, the custom memory allocator increases the acceleration which otherwise would be reduced by almost 10% due to the allocation calls caused by repetitive subsequence batching. The creation of the dictionary benefits not only from balanced and homogeneous computation but also from the fact that in the original CPU algorithm the complete dictionary must be written to disk. In the case of seed extension, the speedup is due to nearly-optimal coalescence of the kernel and the lockstep computation of warps. Lastly, the Escherichia coli case sees less speedup due to the amount of repetitive seeds that result in duplicate HSPs, since parallel threads can not communicate with others to check for overlapping HSPs. The static partitioning mitigates this effect partially and raises the speedup from an initial ~4x to up to ~24x in such cases.

Comparison	G. gallus 18 - M. gallo. 20			O. latipes 6 - D. rerio 25		
Kernel	CPU	GPU	μ_s	CPU	GPU	μ_s
Reverse complement	0.728	0.102	7.16	1.914	0.284	6.74
Words Dictionary	8.613	0.025	349.40	22.929	0.082	280.33
Words Sorting	7.889	0.060	132.28	24.159	0.177	136.54
Seeds Generation	8.606	0.057	151.26	24.264	0.409	59.39

Seeds Sorting	0.266	0.001	331.26	22.303	0.123	181.72
Seeds Filtering	0.034	0.001	25.92	2.167	0.158	13.68
Seeds Extension	0.661	0.003	237.80	7.055	0.056	125.04

Table 7. Per-kernel speedup comparison between GECKO and the proposed implementation GPUGECKO for the second two sequences of the sample dataset. Each multi-column represents a comparison and is further divided in 3 columns, namely (1) CPU time, (2) GPU time and (3) average speedup μ_s.

References

1. Dewey C.N. (2019) Whole-Genome Alignment. In: Anisimova M. (eds) Evolutionary Genomics. Methods in Molecular Biology, vol 1910. Humana, New York, NY
2. Kuzio, J., R. Tatusov, and D. J. Lipman. "Dust." Unpublished but briefly described in: Morgulis A, Gertz EM, Schäffer AA, Agarwala R. A Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences. Journal of Computational Biology 13, no. 5 (2006): 1028-1040.
3. Pérez-Wohlfeil, Esteban, Sergio Diaz-del-Pino, and Oswaldo Trelles. "Ultra-fast genome comparison for large-scale genomic experiments." Scientific reports 9, no. 1 (2019): 1-10.
4. Official CUB repository. Revision c3cceac. https://github.com/NVlabs/cub
5. Official ModernGPU repository. Revision 2b39855 on master branch. https://github.com/moderngpu/moderngpu