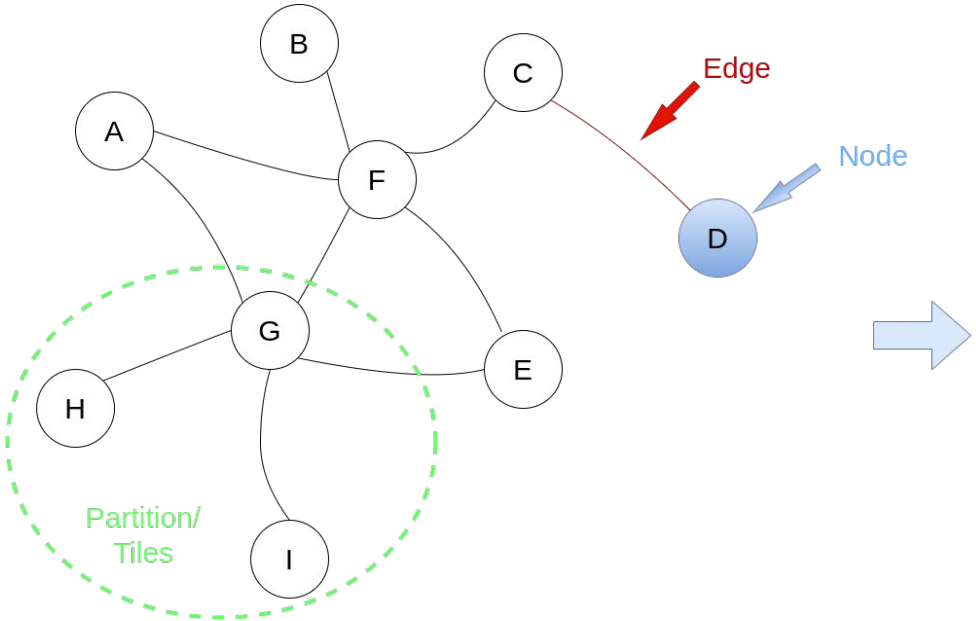
Parallel Graph Coloring in Shared Memory

Daniel Reith and Eric Kern

Outline

1. Terminology
2. Project Description
3. Implementation
    a. Tiling
    b. Collision Computation
4. Evaluation
    a. Data Movement
    b. Distance 1
    c. Distance 2

Terminology



	A	B	C	D	E	F	G	H	I
A	1	0	0	0	0	1	1	0	0
B	0	1	0	0	0	1	0	0	0
C	0	0	1	1	0	1	0	0	0
D	0	0	1	1	0	0	0	0	0
E	0	0	0	0	1	1	1	0	0
F	1	1	1	0	1	1	1	0	0
G	1	0	0	0	1	1	1	1	1
H	0	0	0	0	0	0	1	1	0
I	0	0	0	0	0	0	1	0	1

Project Description

- Max & Total collisions for Distance 1 & Distance 2 coloring
- Different hash functions for coloring
- Performance & Quality evaluation
- Comparison with cusparseDcsrcolor() and old ASC project

Tiling

- No need to access elements in different partitions
- Permutation not needed
- Only row pointer important
- Sequential

➜ Simply create partitions with constraint on maximum size

How size of partition is calculated may differ

Row_Ptr

0	5	9	13	16	20	...

Adjacent_Difference

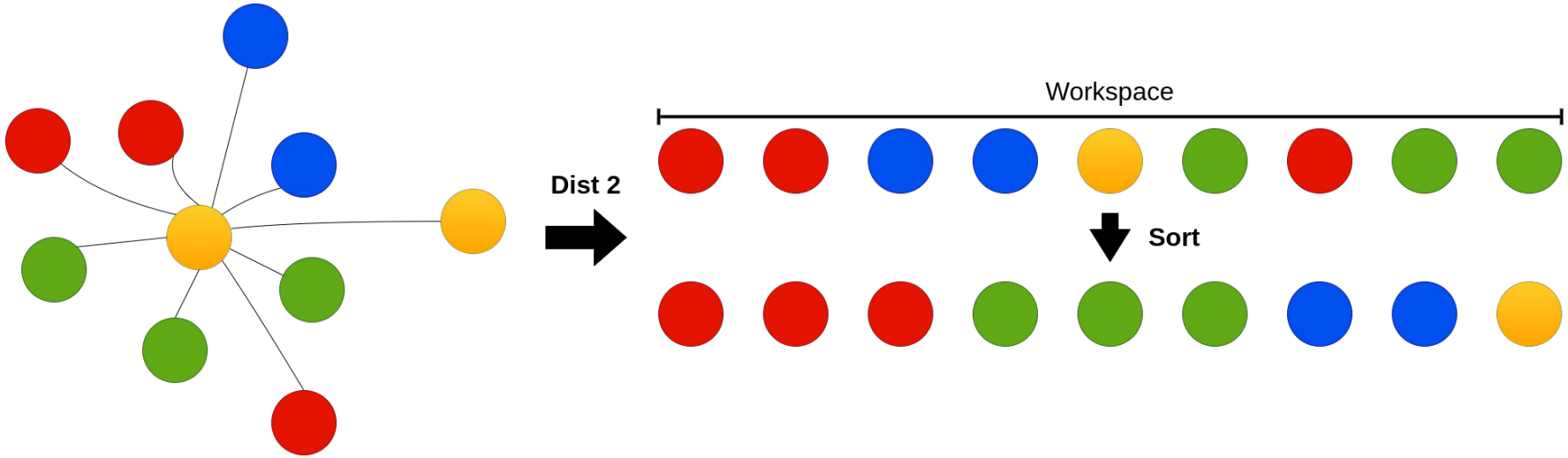
0	5	4	4	3	4	...

```
tile_start ← 0
tile_cols ← 0
i ← 1
while i < len(Row_Ptr) do
    tile_rows ← i − tile_start
    tile_cols ← tile_cols + Adjacent_Difference[i]

    tile_size ← foo(tile_rows, tile_cols)

    if tile_size > shared_mem_size then
        set_tile_start(i);
        tile_cols ← 0
    end if
    i ← i + 1
end while
```

Collision Counting

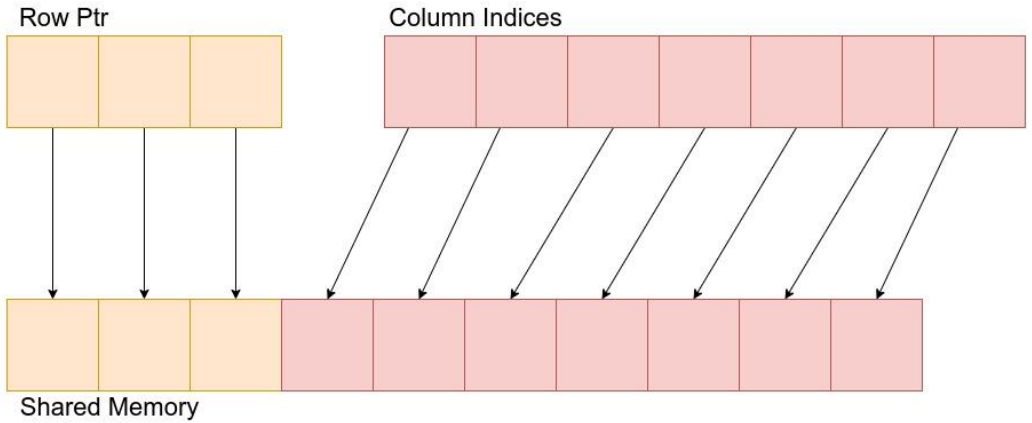


Distance 1:
1 Collision

Distance 2:
3 Collisions -> highest number of appearances

Shared Memory Load

- Coalesced load of both Row Ptr and Column Indices
- Possibly unaligned
- Barrier sync. after shared memory load



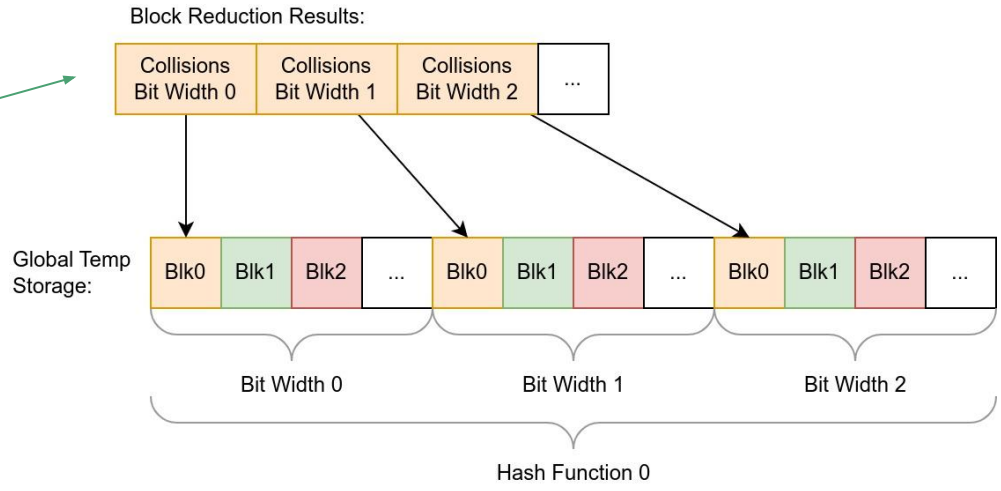
➜ No overlap of load and compute inside a single thread block

Distance 1 Kernel

Algorithm 1 Distance 1 Kernel

```
PartitionToSharedMemory(tile)
for hash_functions do
    ComputeLocalCollisions(tile)
    for bit_widths do
        BlockReduceTotal(local_collisions)
        BlockReduceMax(local_max_colisions)
        if threadID == 0 then
            global_block_total ← block_total
            global_block_max ← block_max
        end if
    end for
end for
threadfence()
if Thread == 0 then
    CountTickets()
end if
if last_block then
    ReduceAllBlocks()
end if
```

*2 * hash_functions * bit_widths* Counters
➔ 300 in our case



Hardware

RTX 2080 TI

Memory Size:	11 GB
Memory Bandwidth:	616 GB/s
Max Shared Memory per SM:	64 KB
Max Register Memory per SM:	256 KB
Max Threads per SM:	1024
Streaming Multiprocessors (SM):	68
Compute Capability	7.5

Evaluation

- We use bandwidth as metric
  - ➔ Reference to HW peak
  - ➔ Comparable
- Speedup in comparisons

Janna/Cube_Coup_dt0	
Rows:	2 164 760 (8 MiB)
NonZero Entries:	127 206 144 (485 MiB)
Avg Node degree:	59
Max Node degree:	68

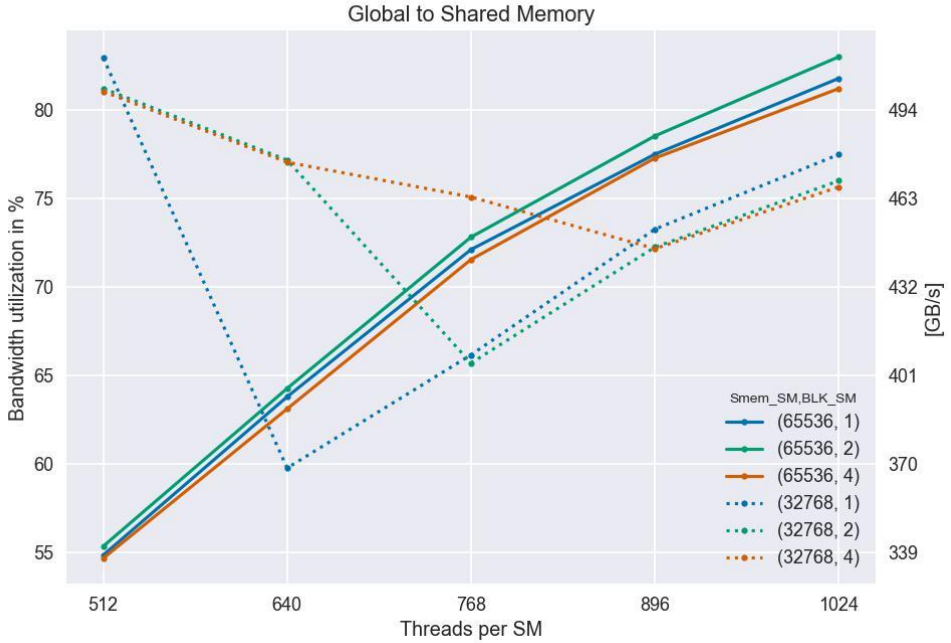
- Performance might vary with other matrices

Global to Shared Memory

Launch Bounds Configuration:

- Blocks per SM is **desired lower limit** (can be more)
- We adjust threads per block so that desired blocks fit on SM

- More Threads are better
- 2 Blocks per SM seems best

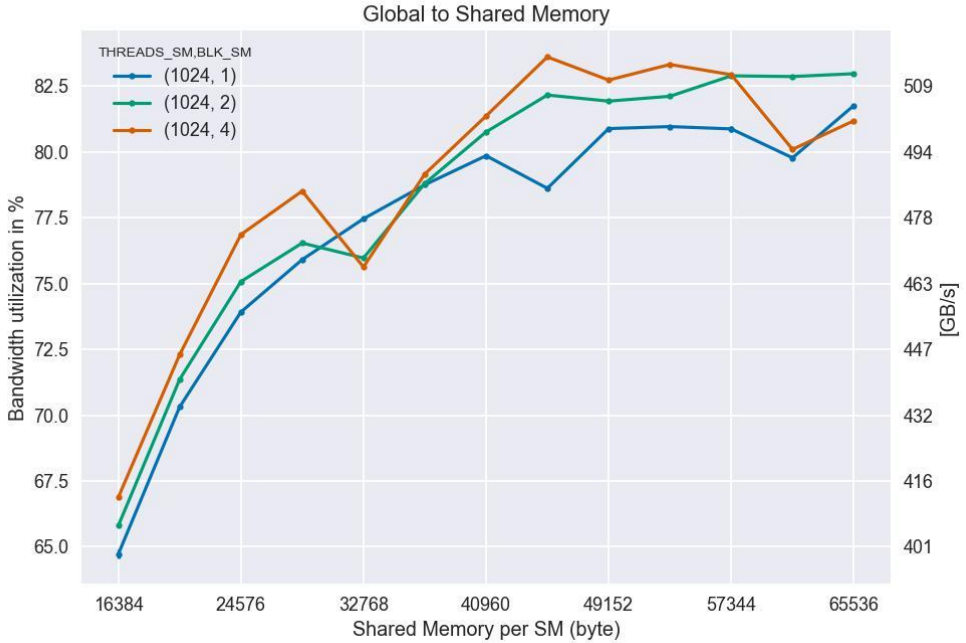


Global to Shared

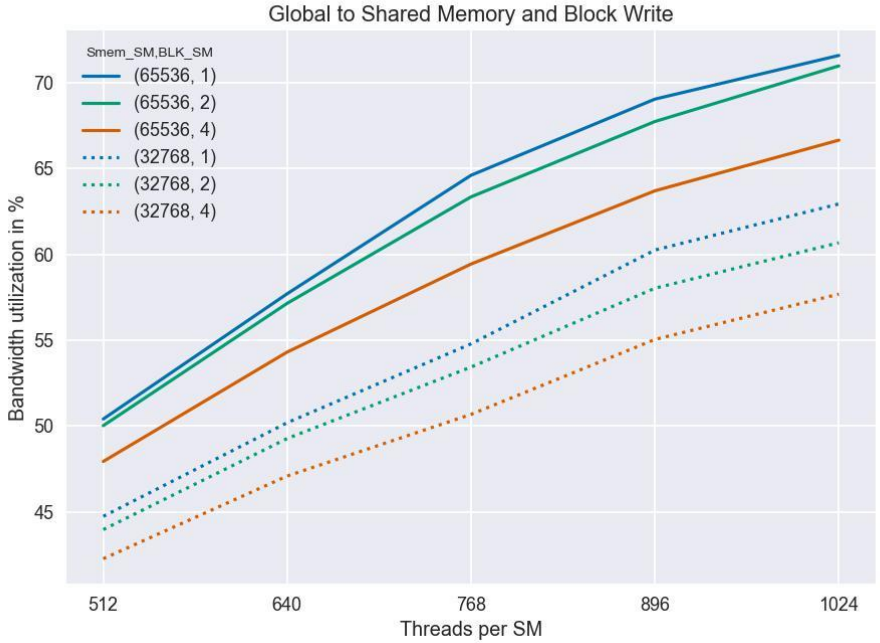
Partition Size:

- Partitions below 48 KiB loses performance
- Above 48 KiB no big change
- Distance 2 coloring
  ➡ only half shared memory for data

Less Shared Memory
⬇
More Partitions
⬇
More unaligned Memory Accesses

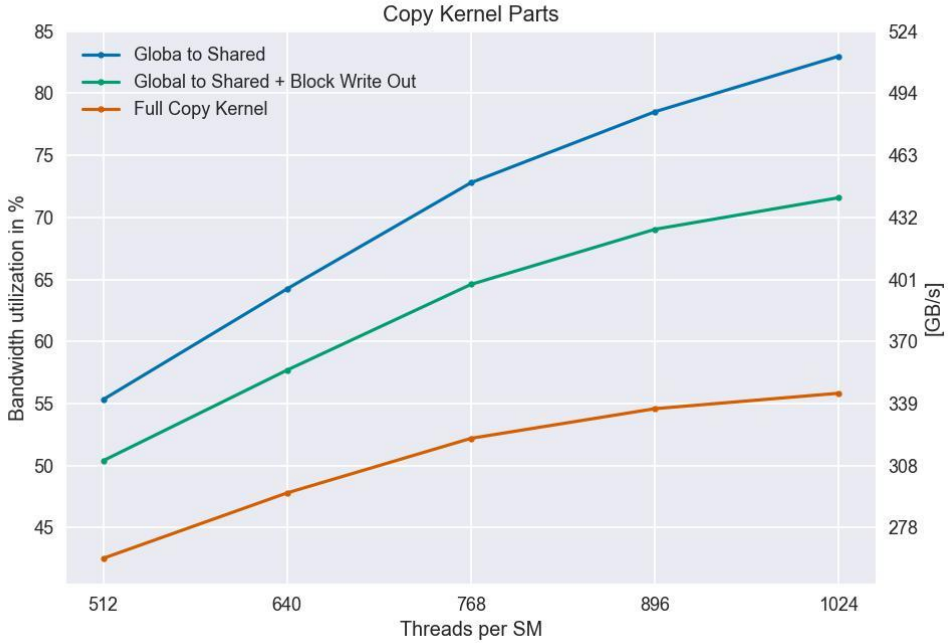


Global Load and Write of Block Results



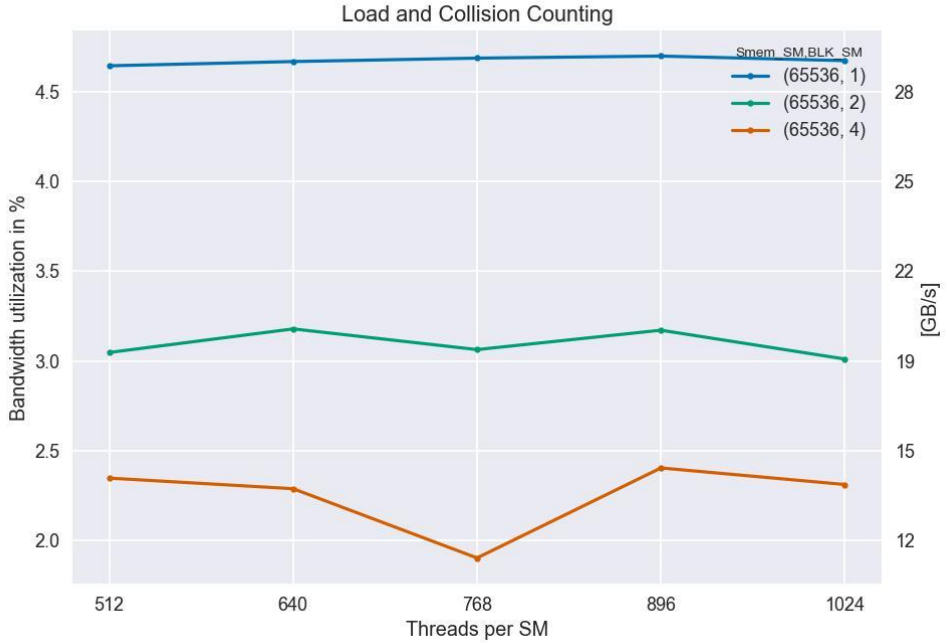
Copy Kernel Parts

We can achieve 56% of peak BW with the current global data movement



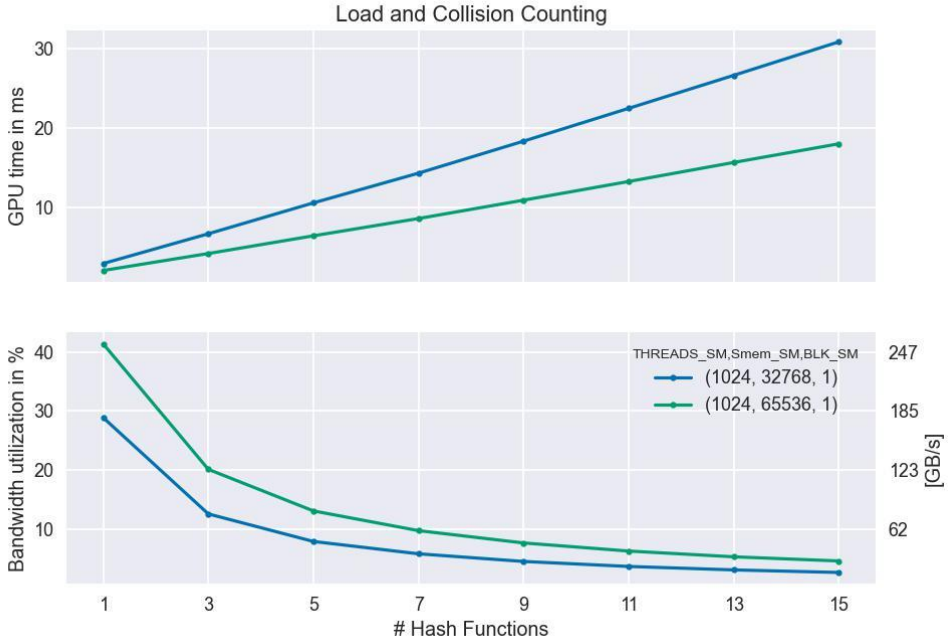
Load and Collision Counting

- Hashing significantly reduces Throughput



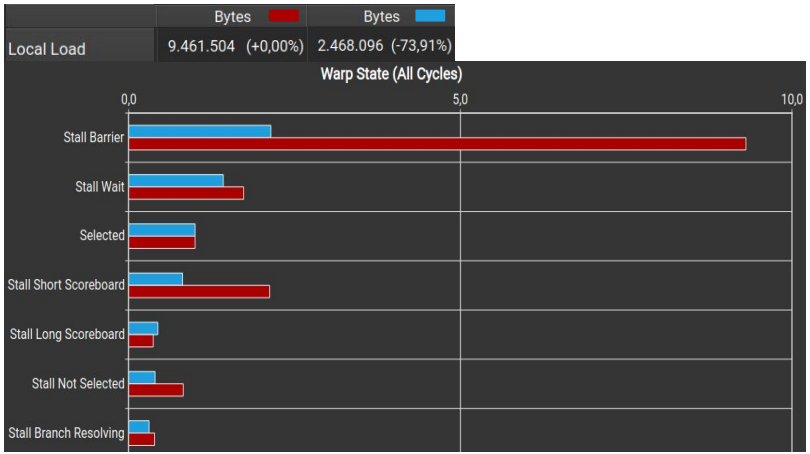
Load and Collision Counting

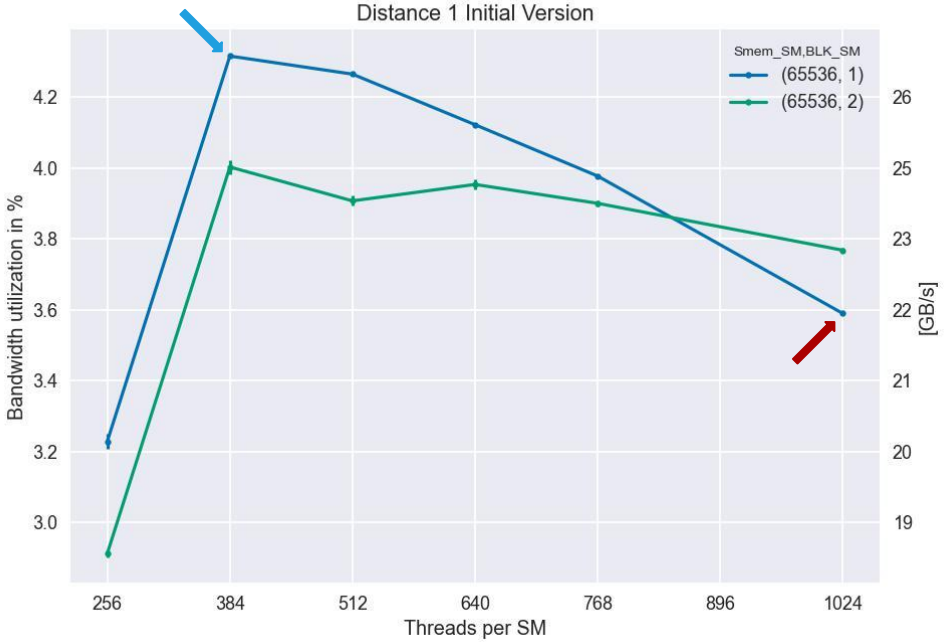
- We are compute-bound
- For 15 hash functions we spend only 4% of the time for loading data to shared memory



Distance 1 Initial Version

- Not enough parallelism for matrices with high node degree
- Only a bit register spilling with many threads





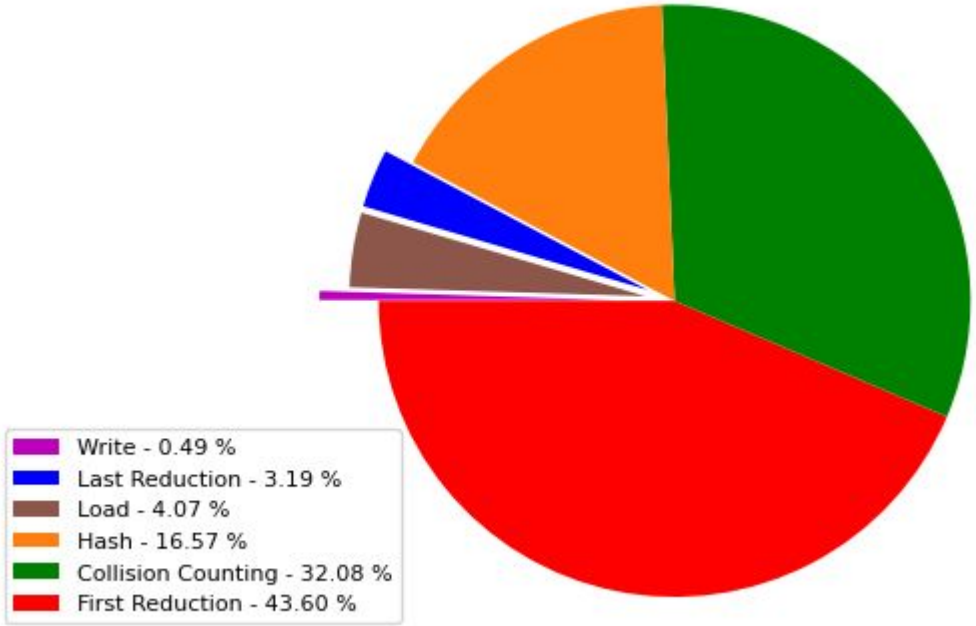
64 KiB, 1 Block, 1024 Threads per SM 64 KiB, 1 Block, 384 Threads per SM

Relative Execution Time

Main contributors:

- Block reduction
- Collision computation

➜ Focus of improvements

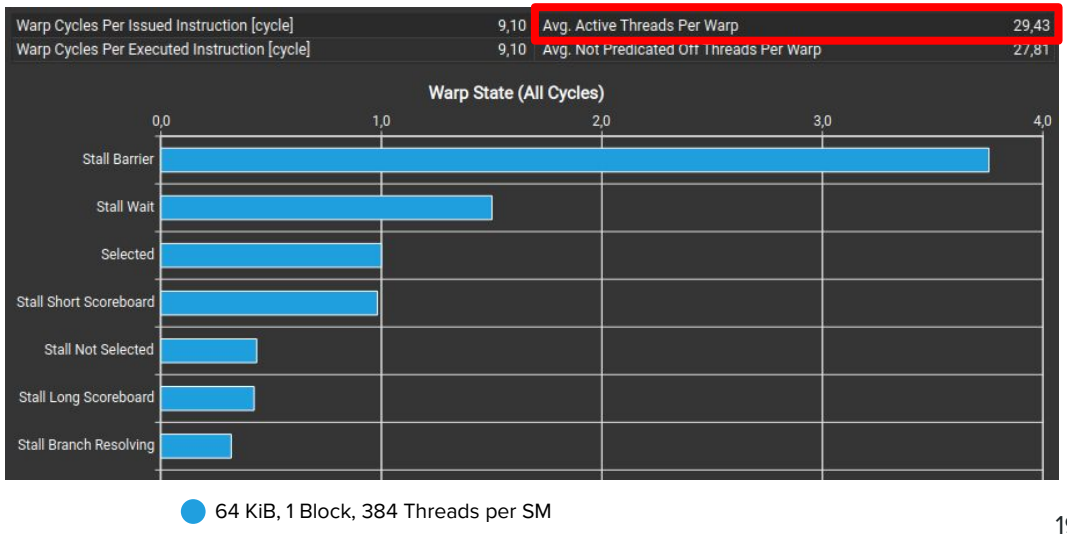


64 KiB, 1 Block, 384 Threads per SM

Constraints on Performance

Possible problems during computation:

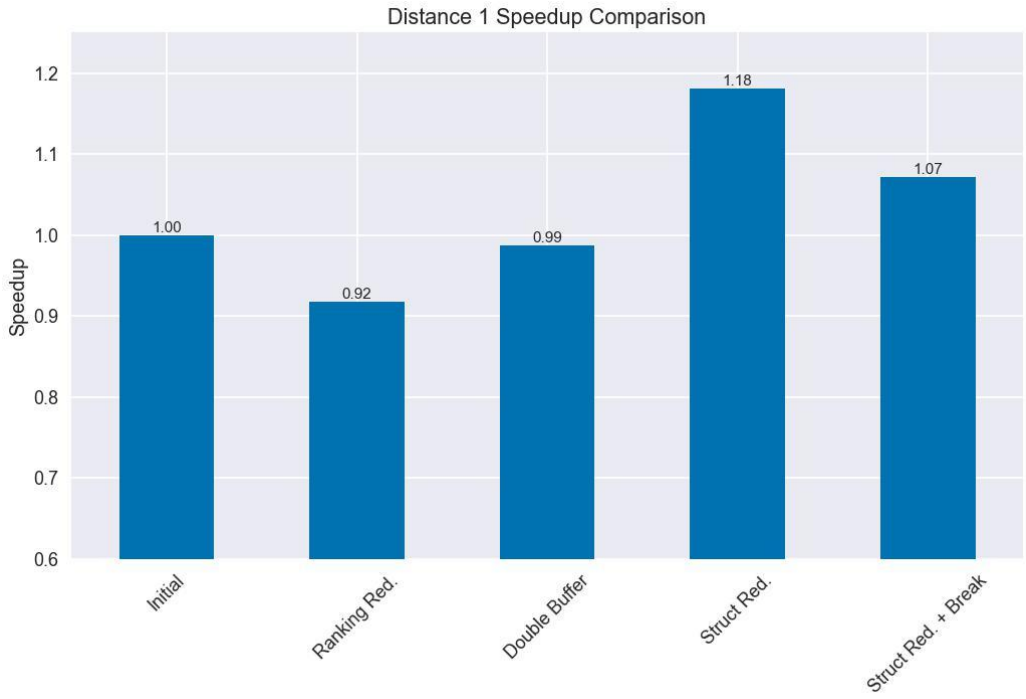
- Unnecessary computations (break loop, unrolling)
- Bank conflicts (read before hash)
- Too many synchronizations
- Other reduce algorithms
- Branch divergence



Speedup Comparison

- Warp Reduction (default) better than Ranking Reduction Algorithm (cub)
- Struct reduction decreases number of reduction calls
  ➡ less synchronization
- Breaking loop causes branch divergence

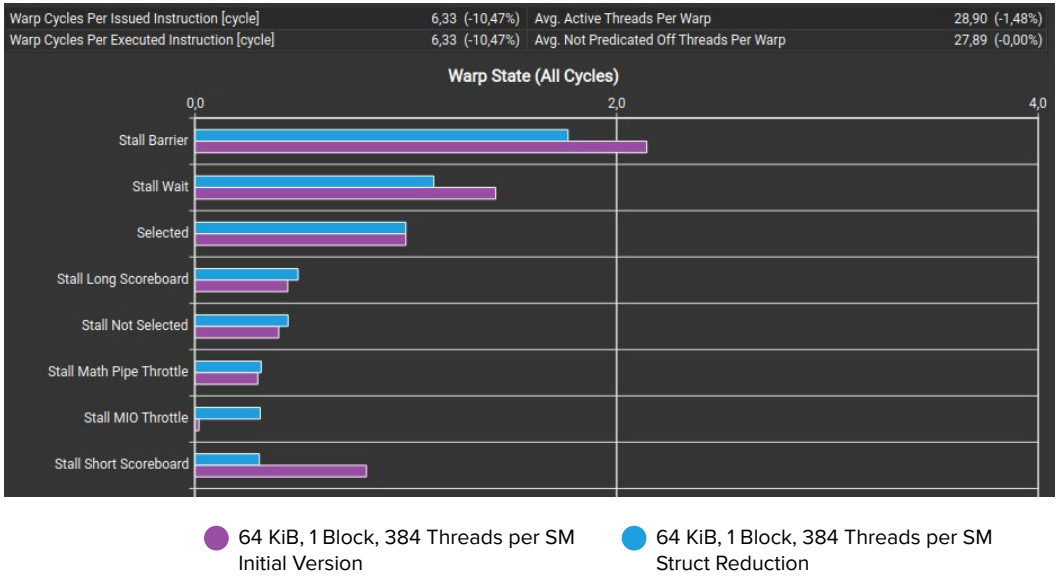
64 KiB, 1 Block, 384 Threads per SM



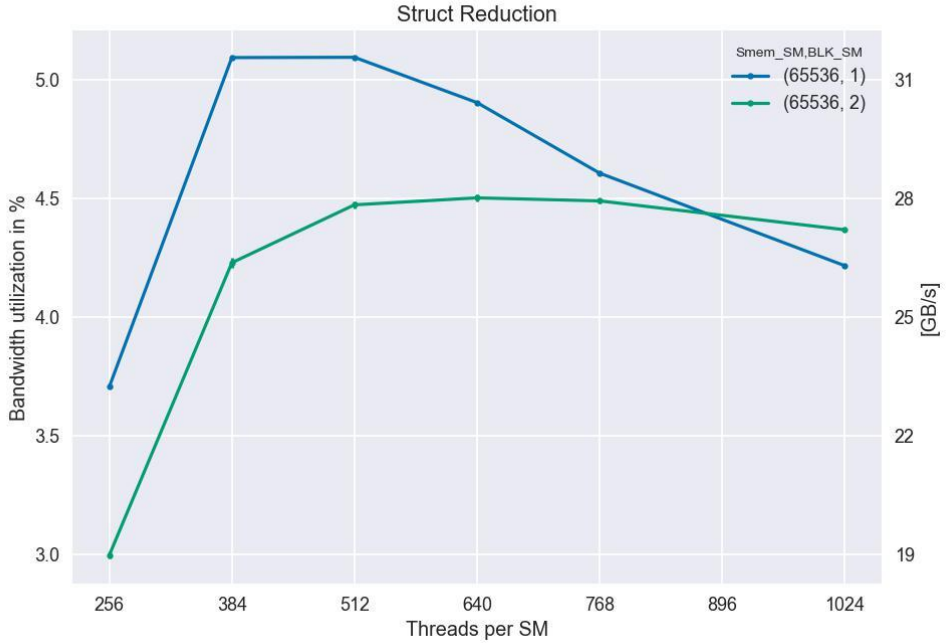
Distance 1 with Struct Reduction

- More work in single reduction
- More shared memory in single reduction (less reuse)

➜ Less stalls caused by barrier



Best version launch config

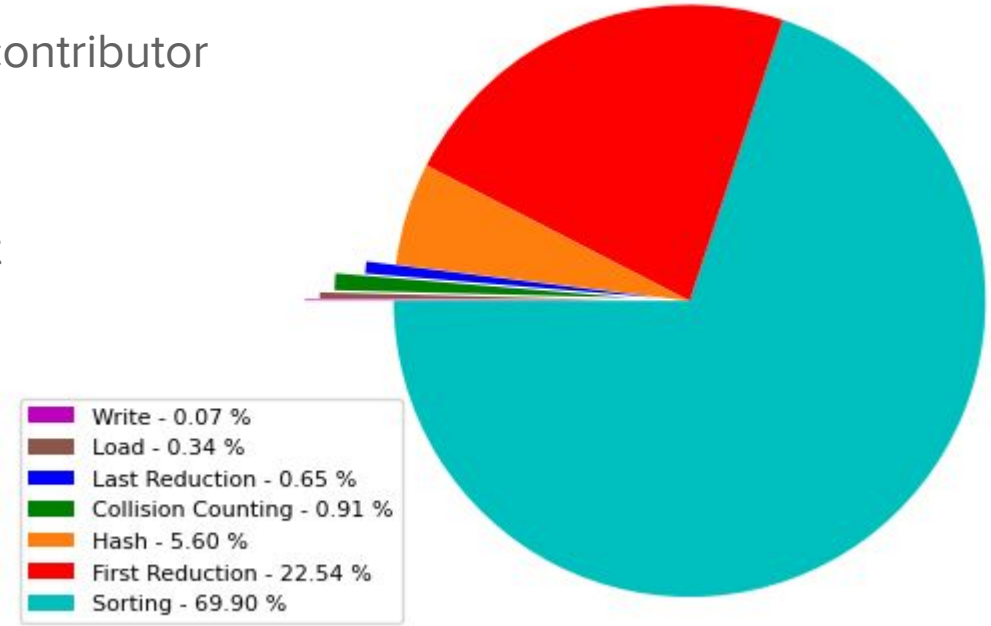


Relative Execution Time Distance 2

- Workspace sorting main contributor

  ➜ main focus

- Reduction also big impact



64 KiB, 2 Blocks, 384 Threads per SM

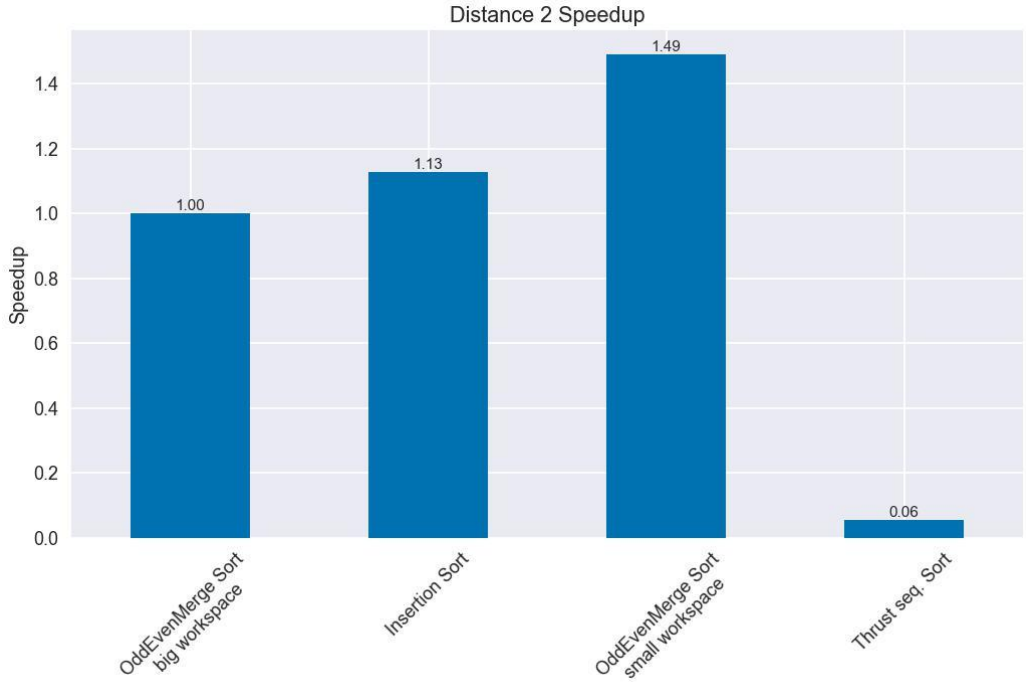
Distance 2 Speedup

64 KiB, 1 Blocks, 384 Threads per SM

- Big Workspace:
  - ➜ round up to multiple of 32 + 1

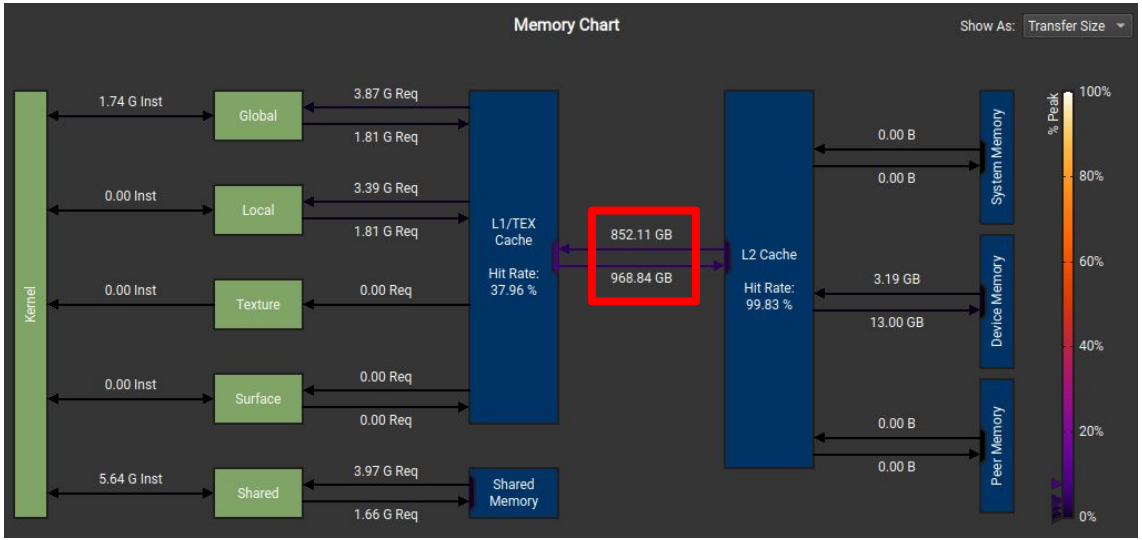
  <u>But:</u> makes partition smaller
- Sorting Network still useful

  ➜ Depending on matrix



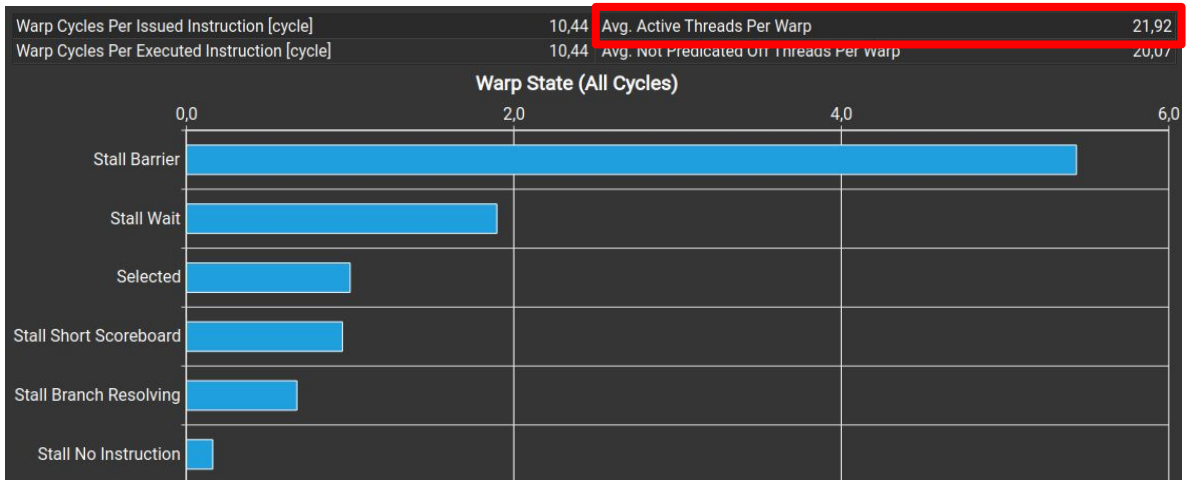
Thrust seq Sort

➜ global memory is used for temporary sorting storage



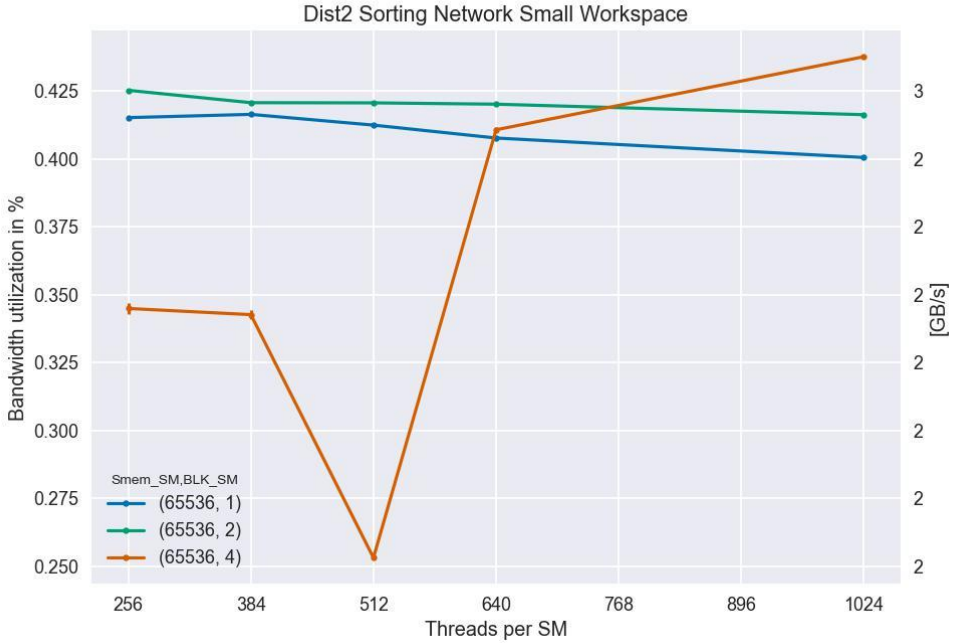
Profile Distance 2 Optimized

- Barrier still main stall reason
- Branch divergence

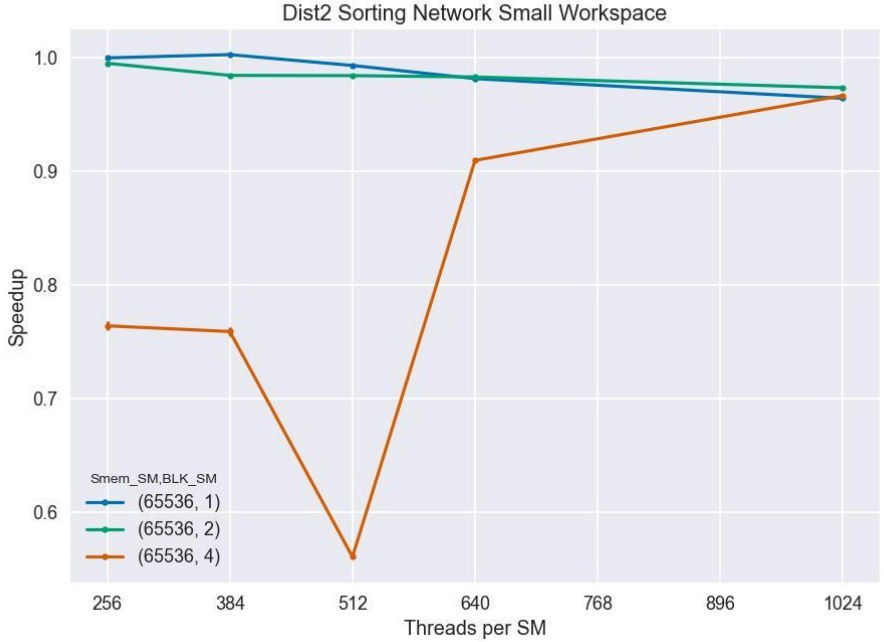


64 KiB, 1 Block, 384 Threads per SM

D2 launch config



D2 launch config

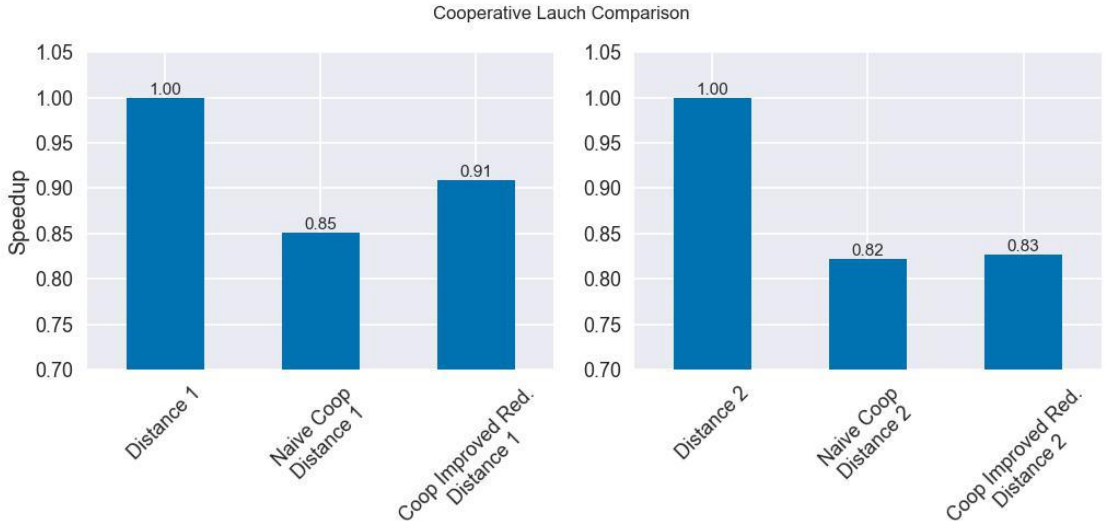


Cooperative Launch Comparison

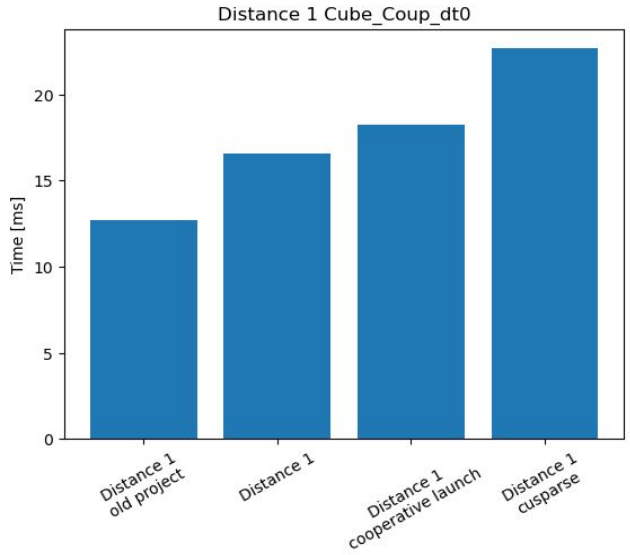
- Reduction of multiple partitions in single block can save global stores
  ➜ Improvements for Dist 1

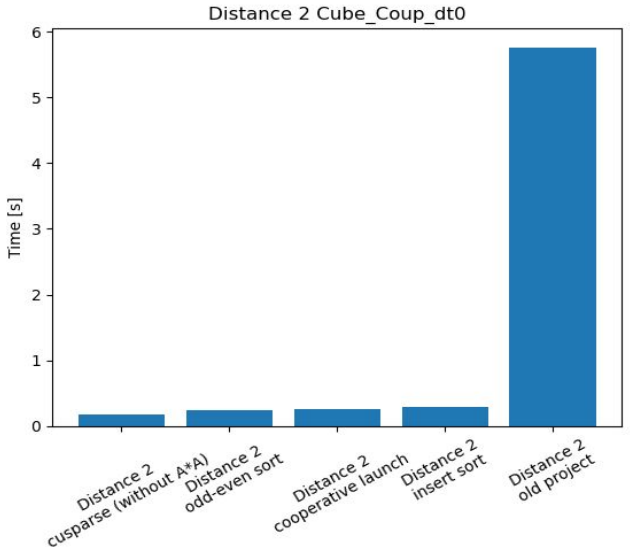
➜ Overall performance depending on matrix

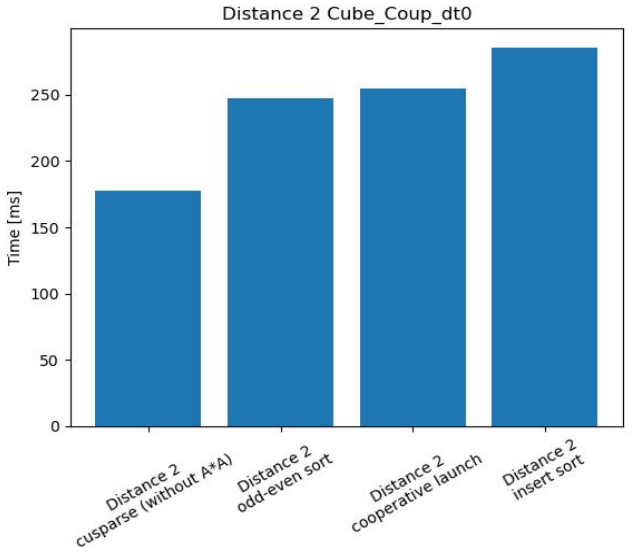
64 KiB, 1 Blocks, 384 Threads per SM



Best Version comparison

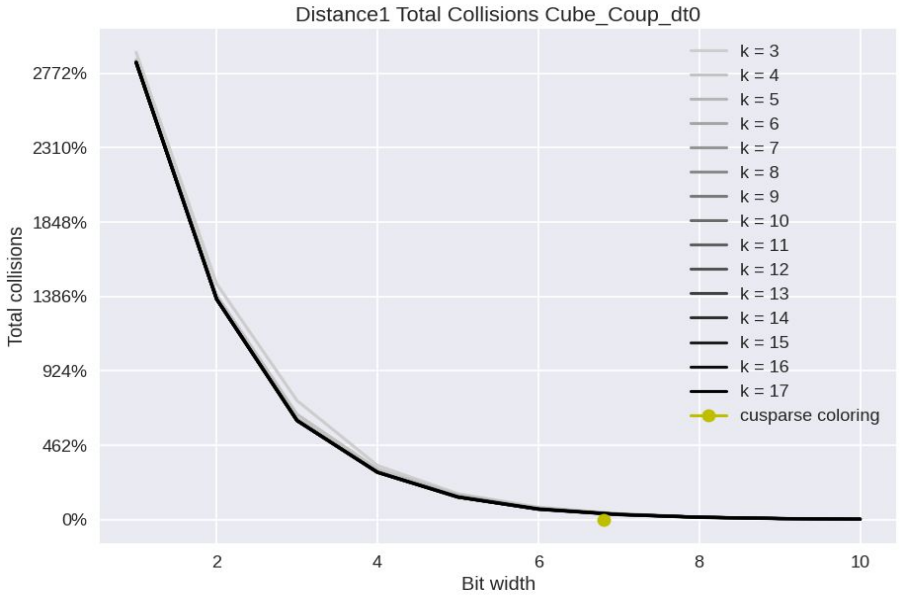


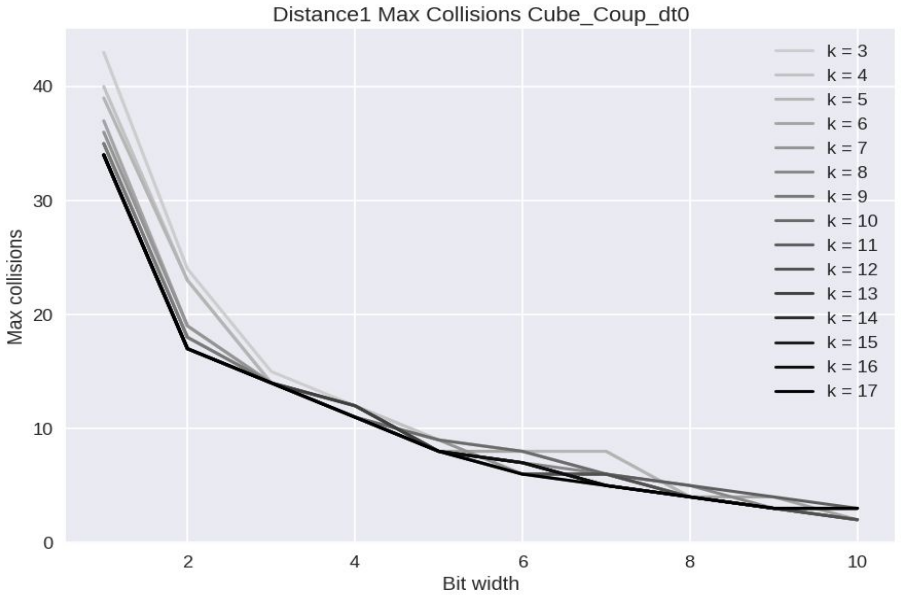




64 KiB, 1 Blocks, 384 Threads per SM

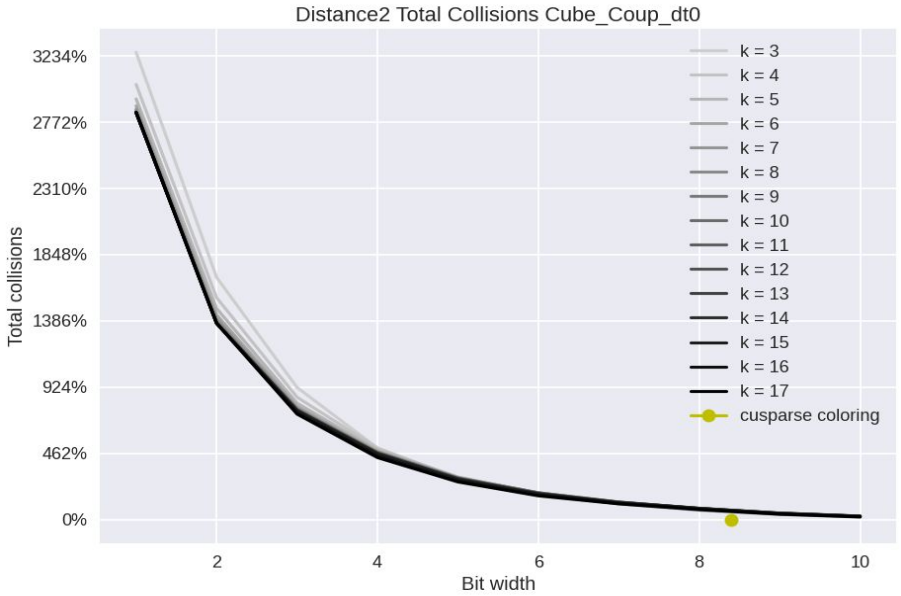
Coloring Quality Distance 1

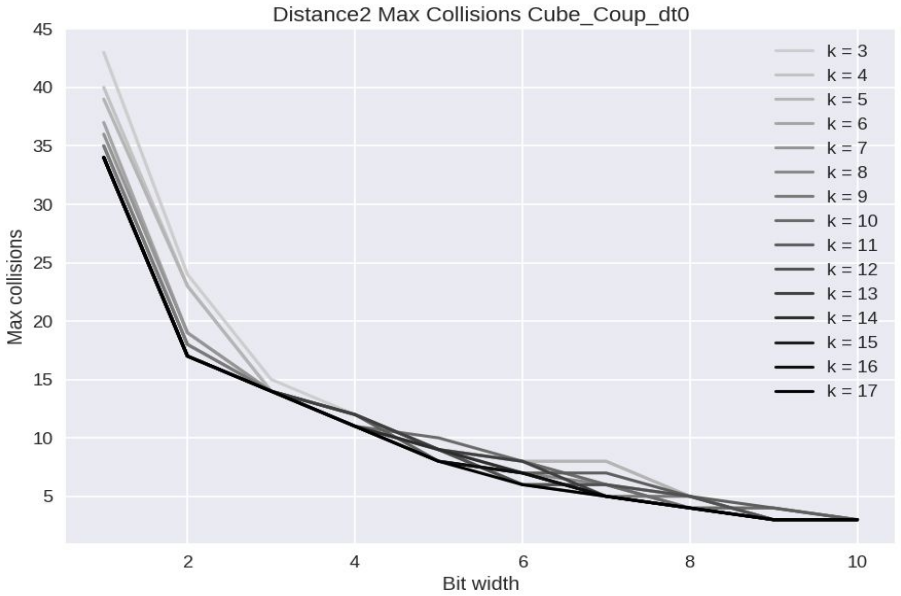




- old project: 32768 colors (k = 5)

Coloring Quality Distance 2





- old project: 65536 colors (k = 5)

Summary

- Collision computation can happen completely in shared memory
- Contrary to our expectations: compute bound
- Time spent primarily in collision computation and reduction
- Shared Memory size can limit performance
- Bank conflicts not that problematic for sorting

Further Improvements

- Trade Computation for Communication in global Memory
  ➜ Currently Global memory unused most of the time
- Think about using a whole Warp for a single Node
  ➜ Too few rows in partition for graphs with high node degree
  ➜ Might also help with branch divergence in very irregular graphs
- Use of better Hardware
  ➜ More Shared Memory and more SMs

Thank you!

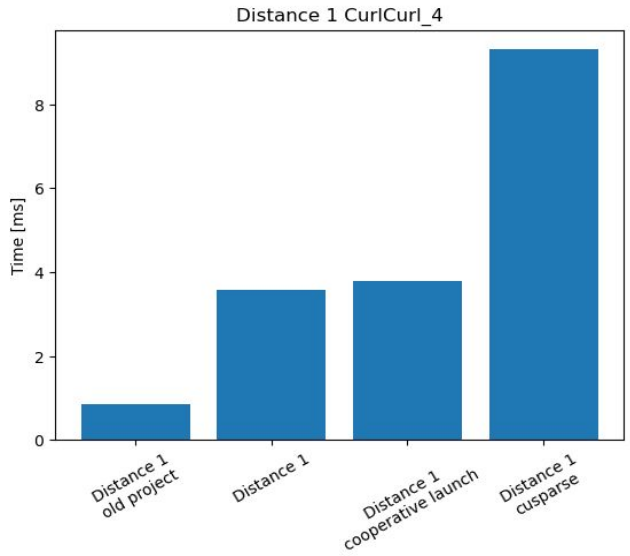
Remark on upcoming plots

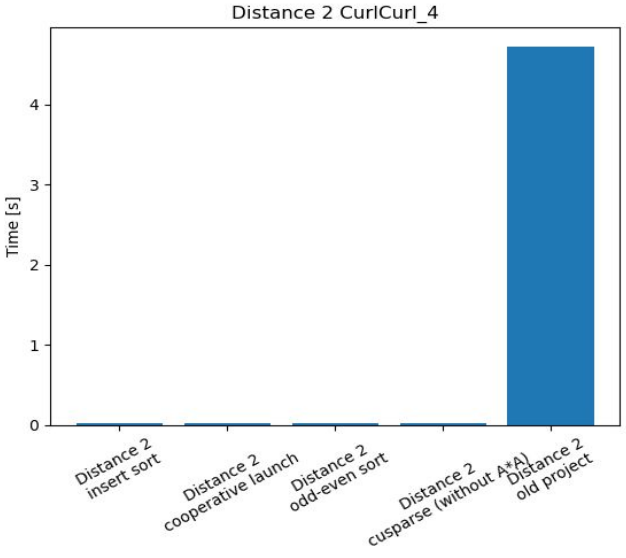
- all following plots created with best performing setup

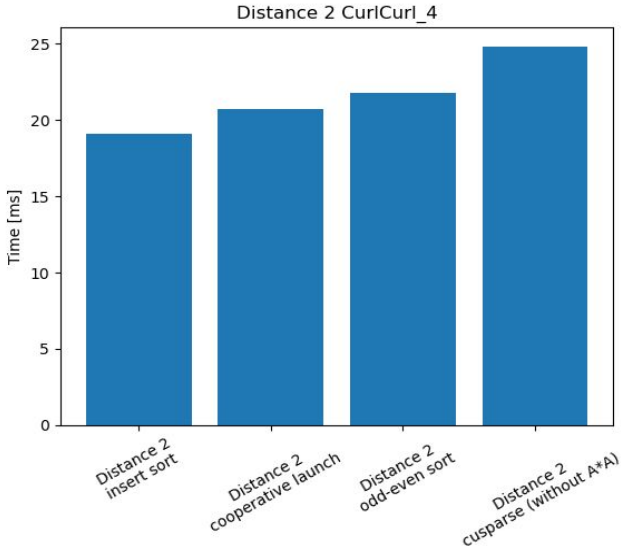
  ➔ 1 Thread Block per SM & 384 Threads per Block

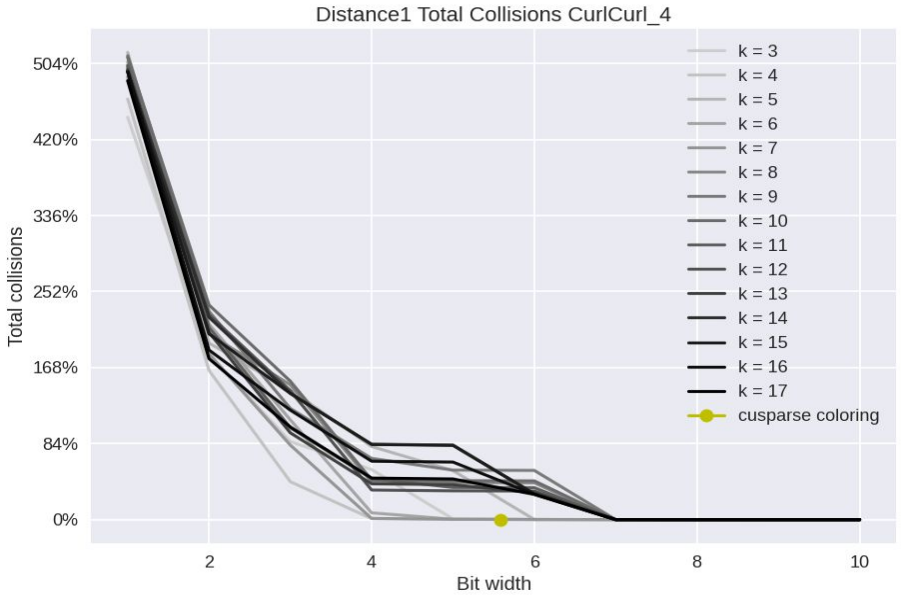
  ➔ For each Matrix a comparison of execution time and coloring quality for distance 1 & 2 with cusparseDcsrcolor() and the old project is made

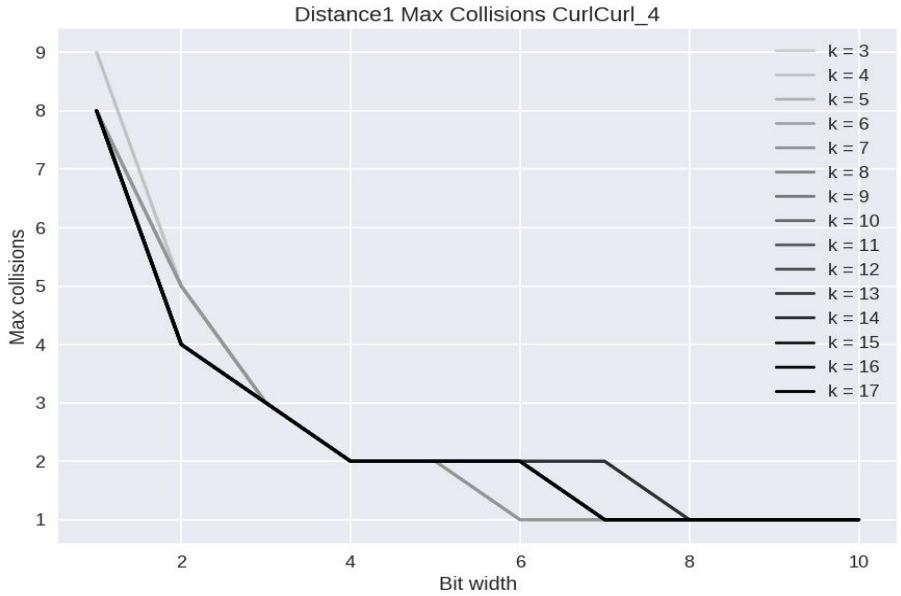
- As the for the cusparse distance 2 coloring required matrix multiplication was performed on the CPU we did not include this in our measurements


Distance 1 CurlCurl_4
Time [ms]
0
2
4
6
8
Distance 1 old project
Distance 1
Distance 1 cooperative launch
Distance 1 cusparse

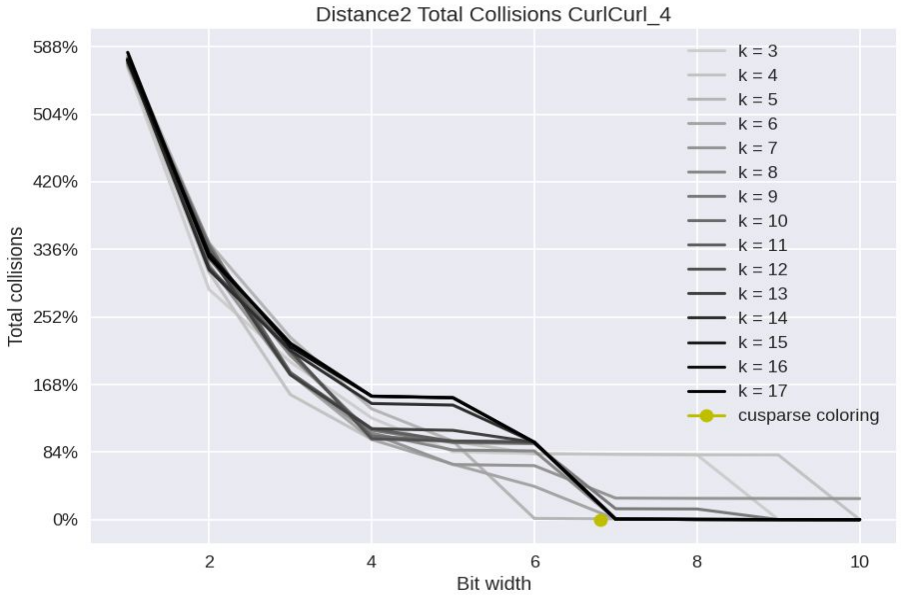

Distance 2 CurlCurl_4
Time [s]
0
1
2
3
4
Distance 2 insert sort
Distance 2 cooperative launch
Distance 2 odd-even sort
Distance 2 cusparse (without A*A)
Distance 2 old project

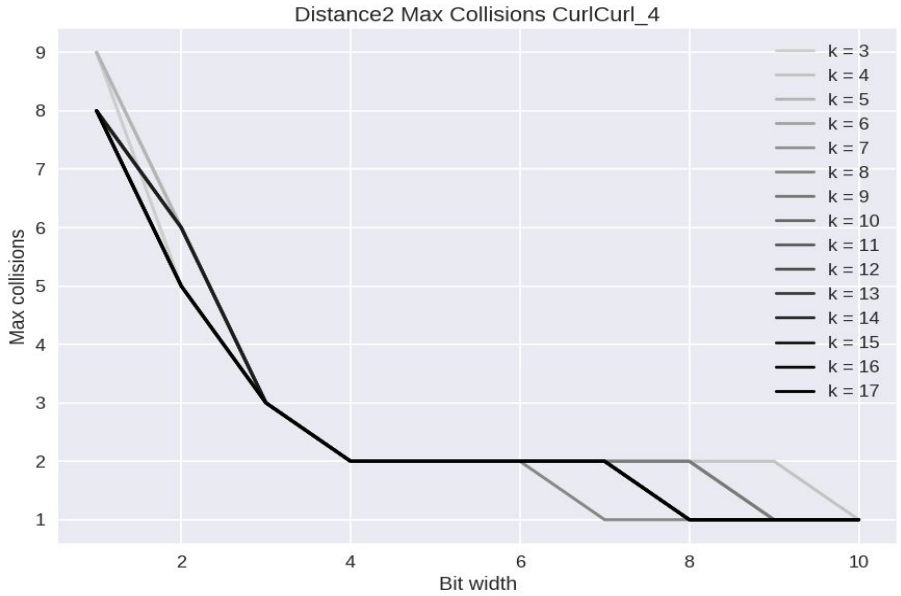

Distance 2 CurlCurl_4
Time [ms]
0
5
10
15
20
25
Distance 2 insert sort
Distance 2 cooperative launch
Distance 2 odd-even sort
Distance 2 cusparse (without A*A)



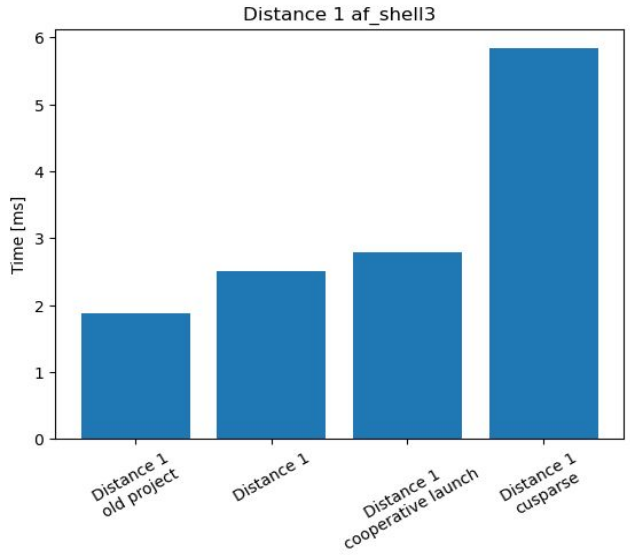


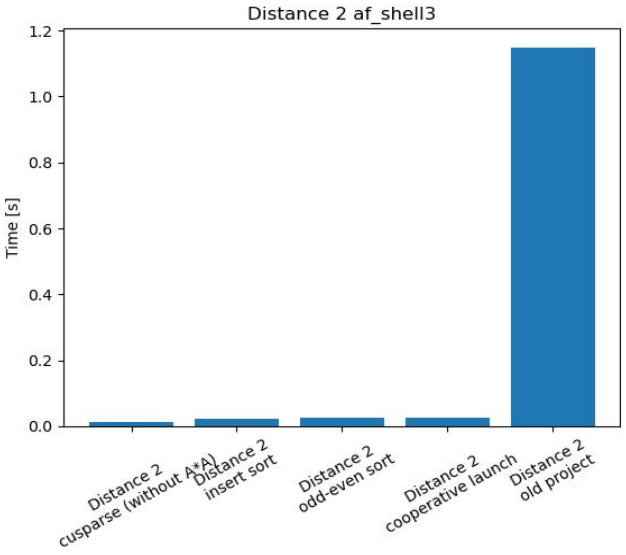
- old project: 65536 colors (k = 3)

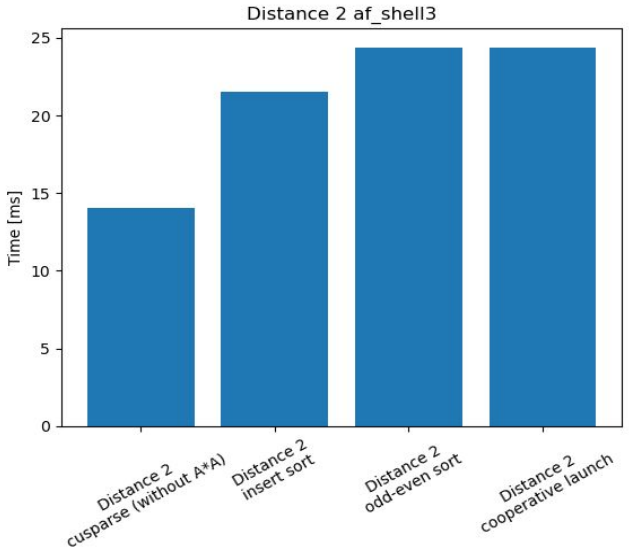


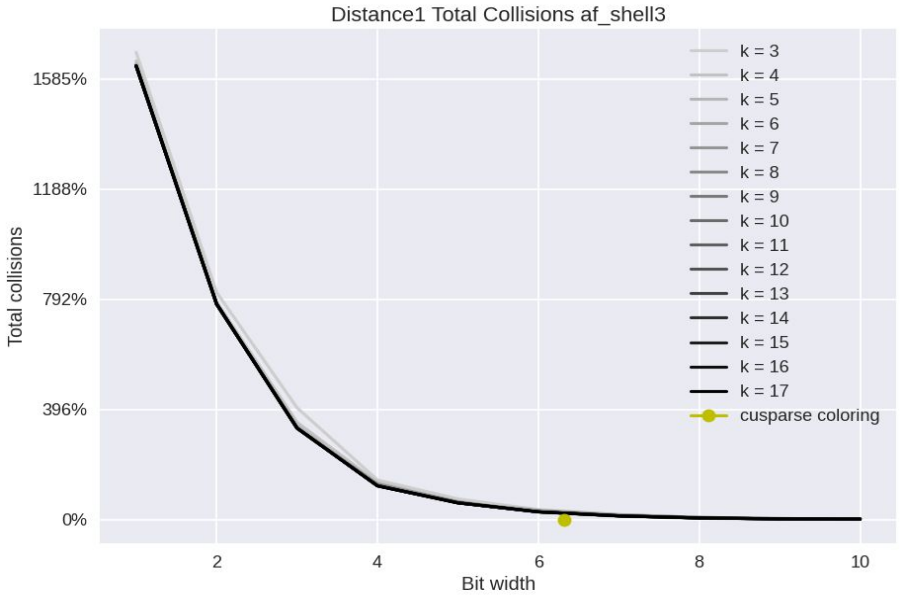


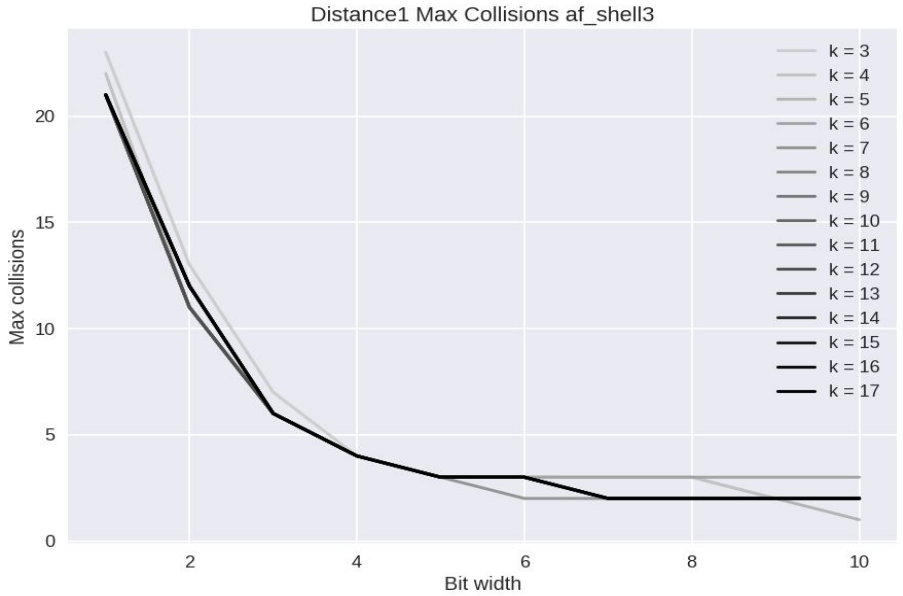
- old project: 65536 colors (k = 3)


Distance 1 af_shell3
Time [ms]
0
1
2
3
4
5
6
Distance 1 old project
Distance 1
Distance 1 cooperative launch
Distance 1 cusparse

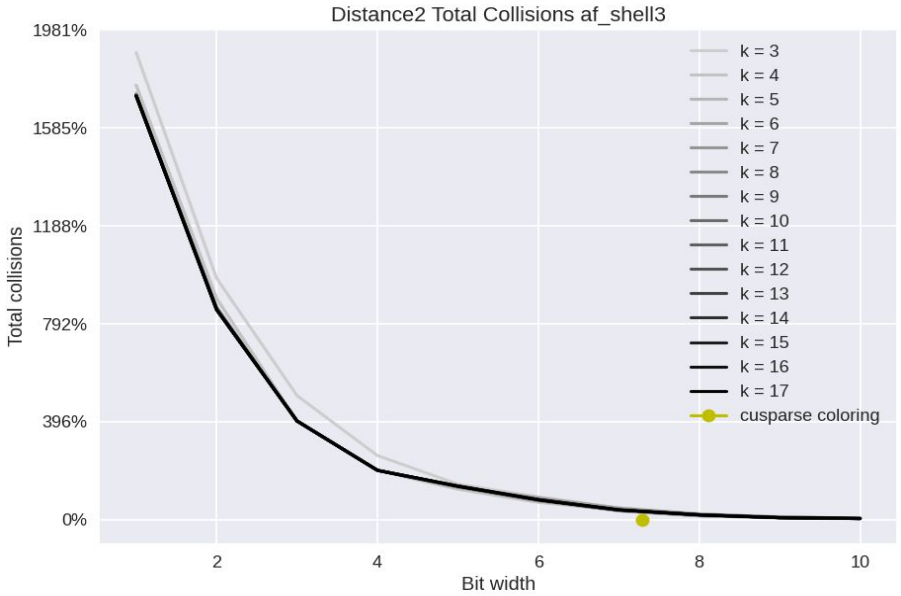

Distance 2 af_shell3
Time [s]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 old project

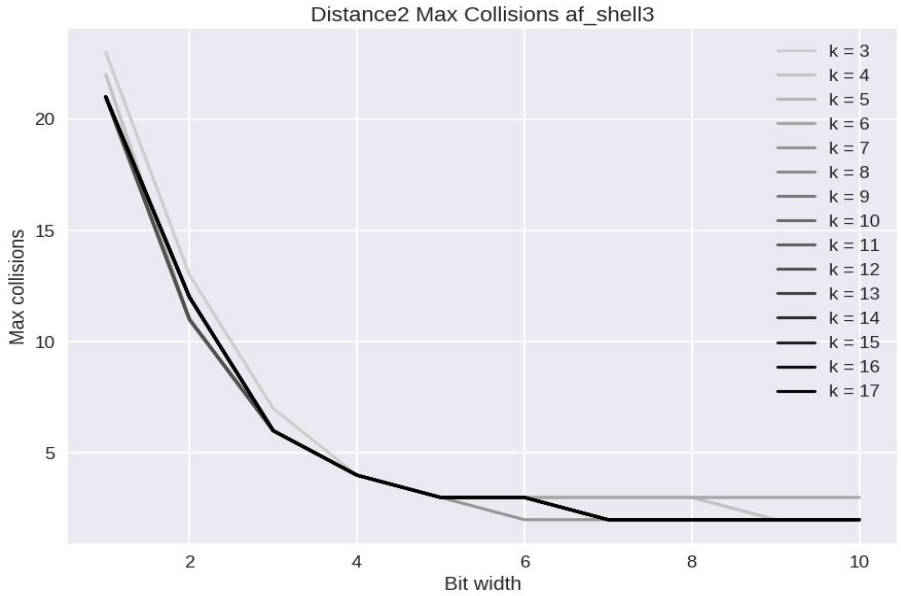

Distance 2 af_shell3
Time [ms]
0
5
10
15
20
25
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch



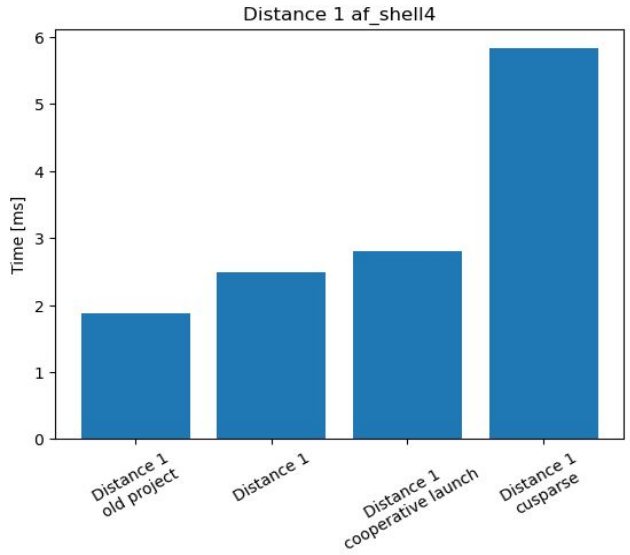


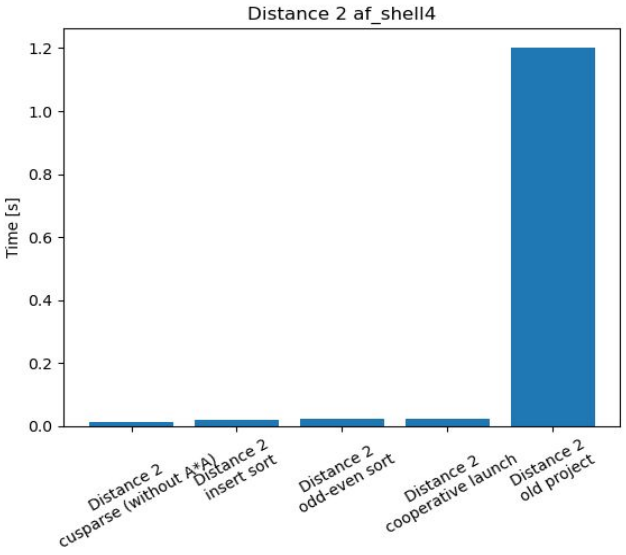
- old project: 4096 colors (k = 3)

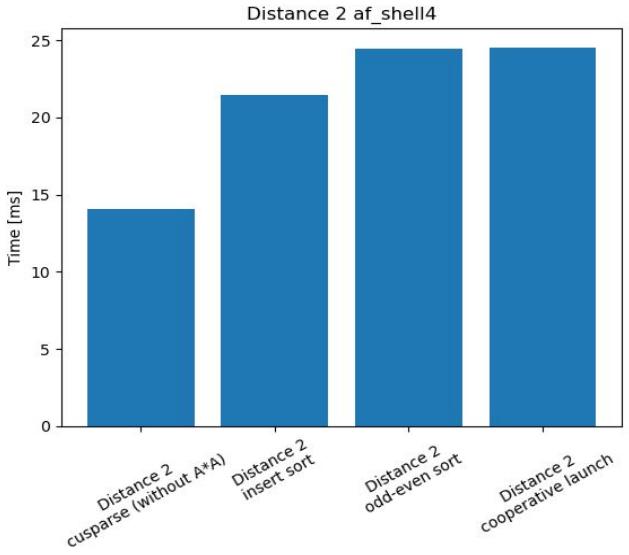


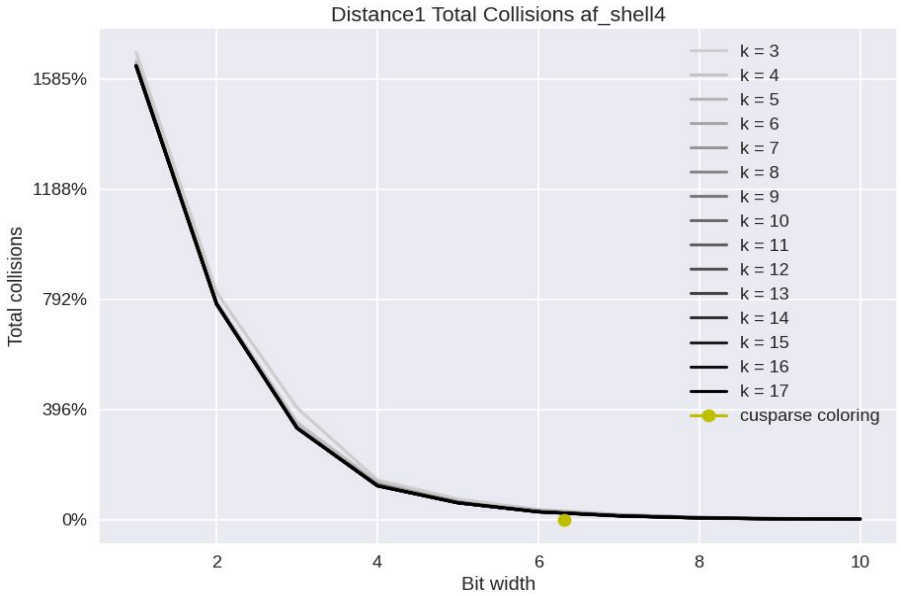


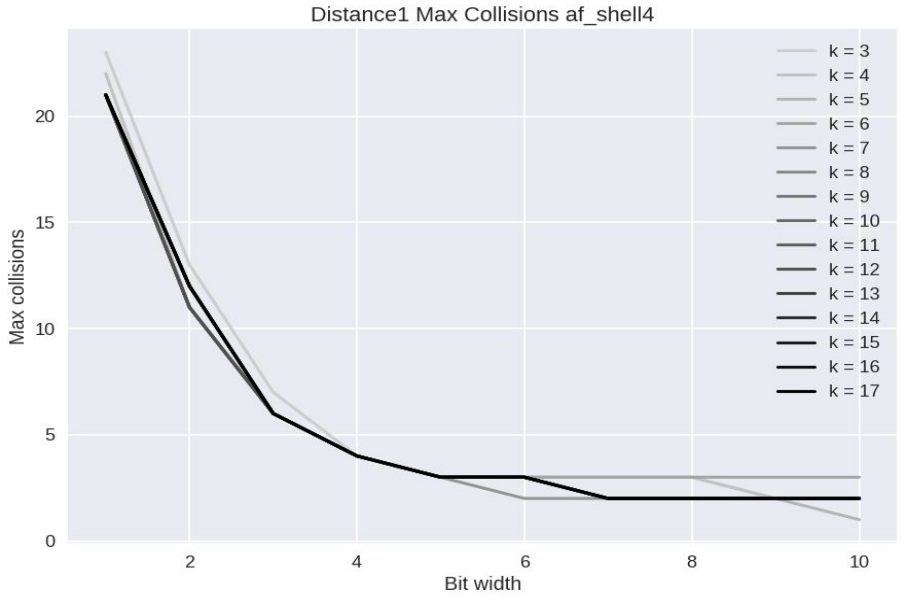
- old project: 8192 colors (k = 4)


Distance 1 af_shell4
Time [ms]
0
1
2
3
4
5
6
Distance 1 old project
Distance 1
Distance 1 cooperative launch
Distance 1 cusparse

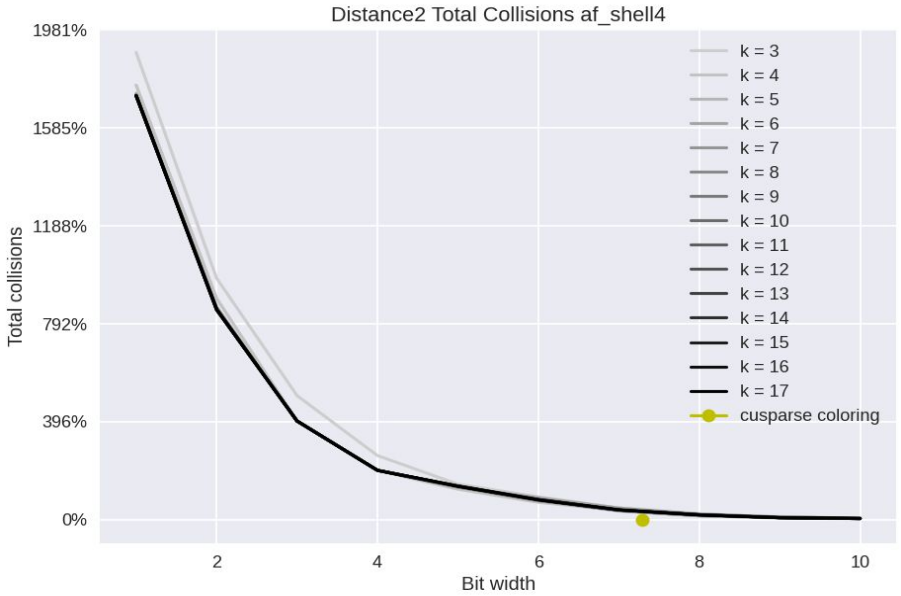

Distance 2 af_shell4
Time [s]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 old project

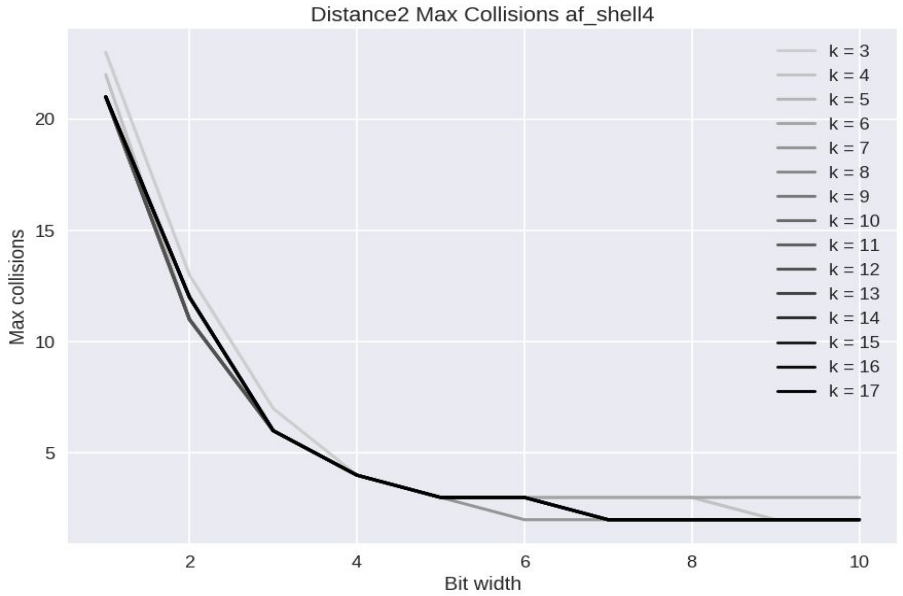

Distance 2 af_shell4
Time [ms]
0
5
10
15
20
25
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch



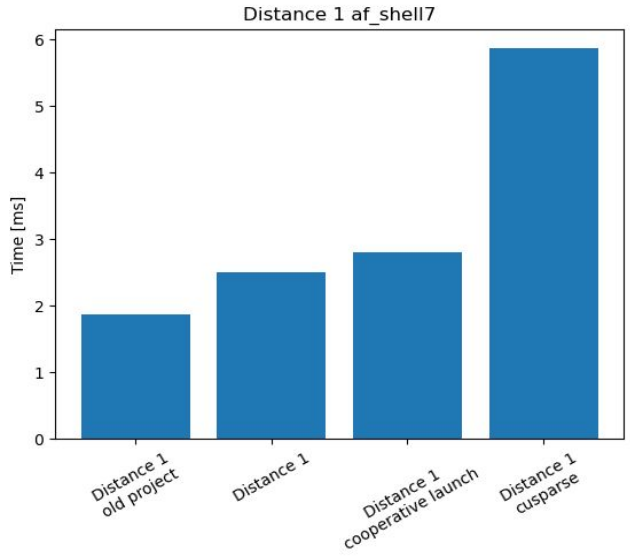


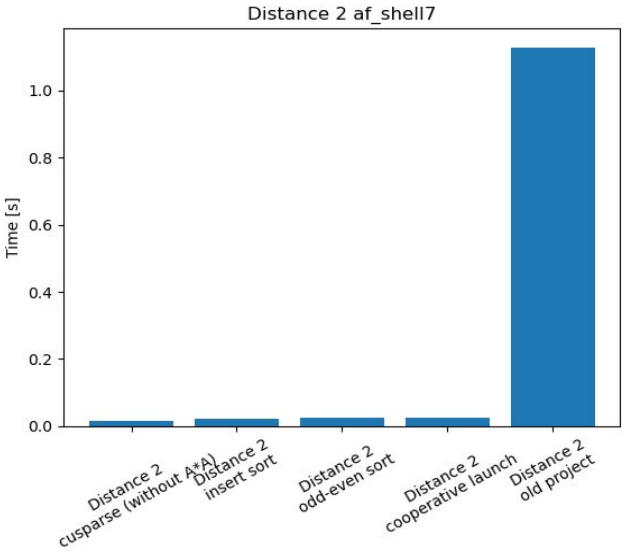
- old project: 4096 colors (k = 3)

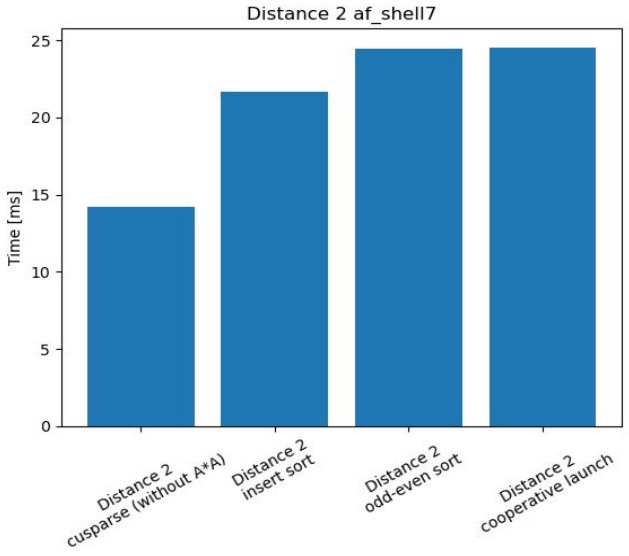


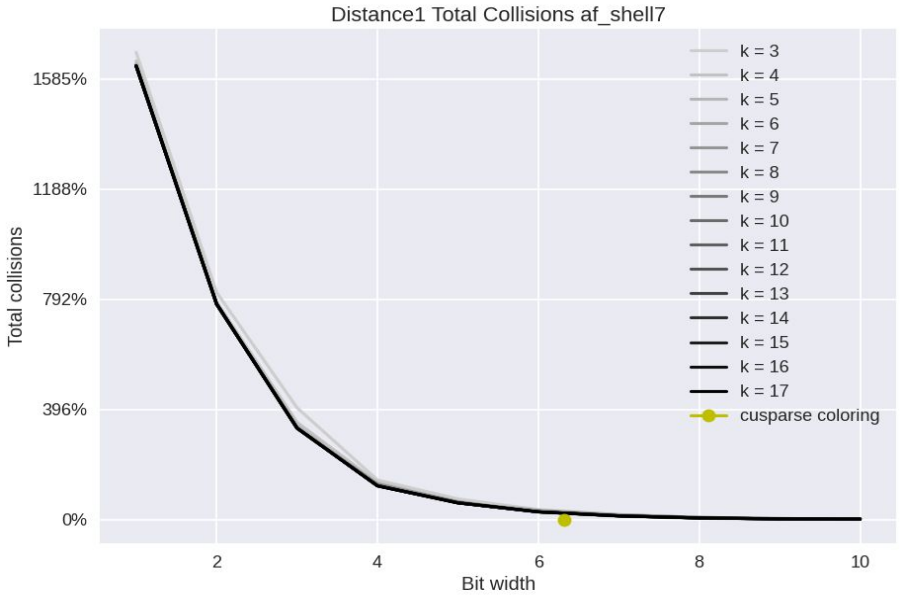


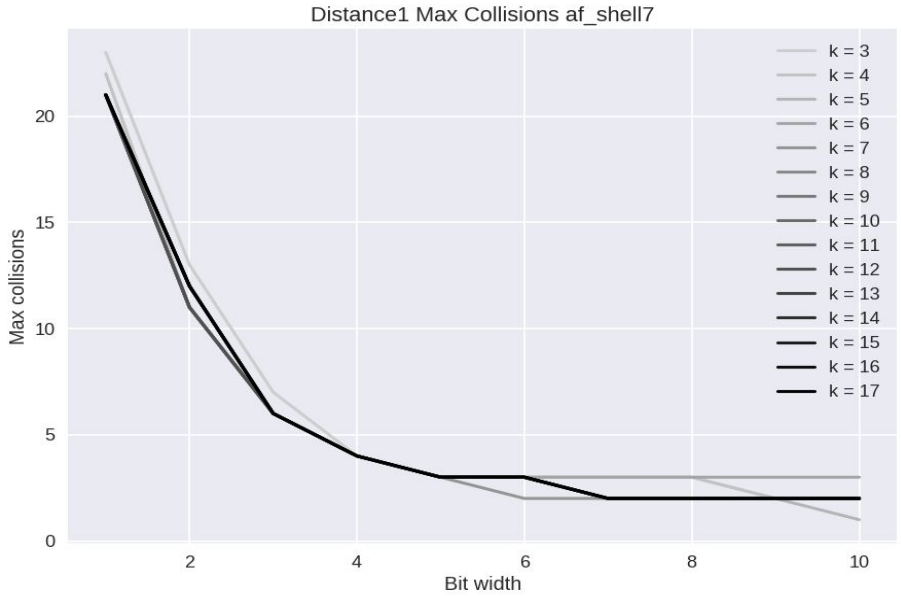
- old project: 8192 colors (k = 4)


Distance 1 af_shell7
Time [ms]
0
1
2
3
4
5
6
Distance 1 old project
Distance 1
Distance 1 cooperative launch
Distance 1 cusparse

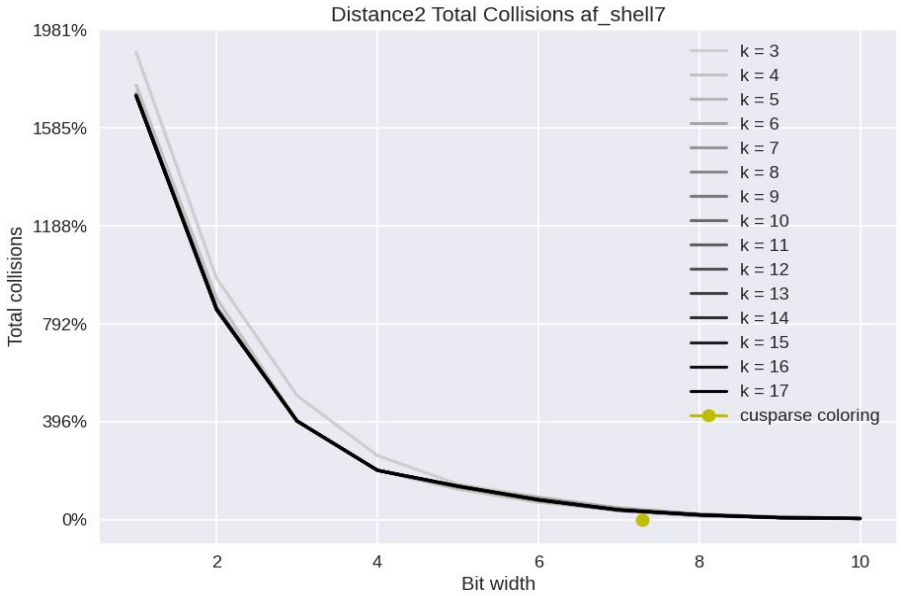

Distance 2 af_shell7
Time [s]
0.0
0.2
0.4
0.6
0.8
1.0
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 old project

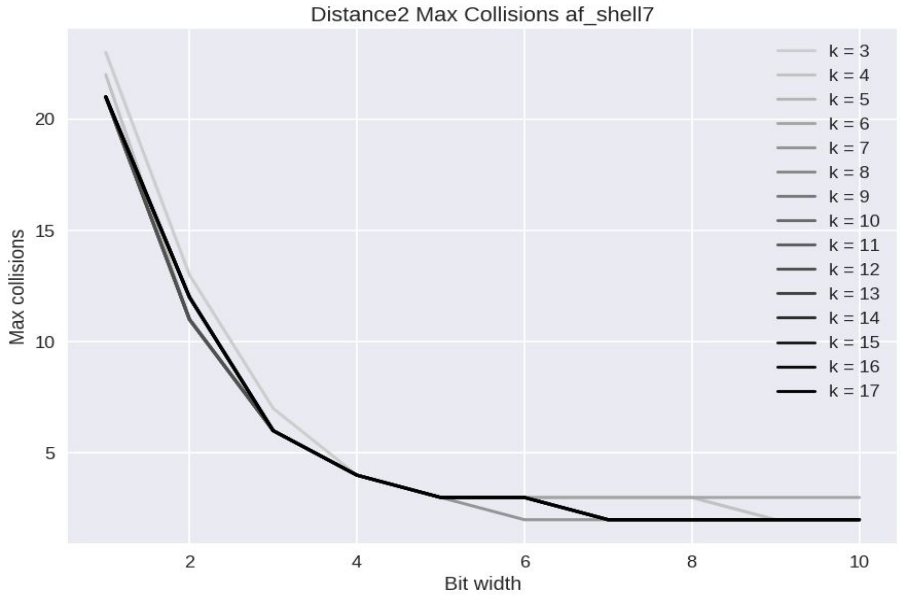

Distance 2 af_shell7
Time [ms]
0
5
10
15
20
25
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch



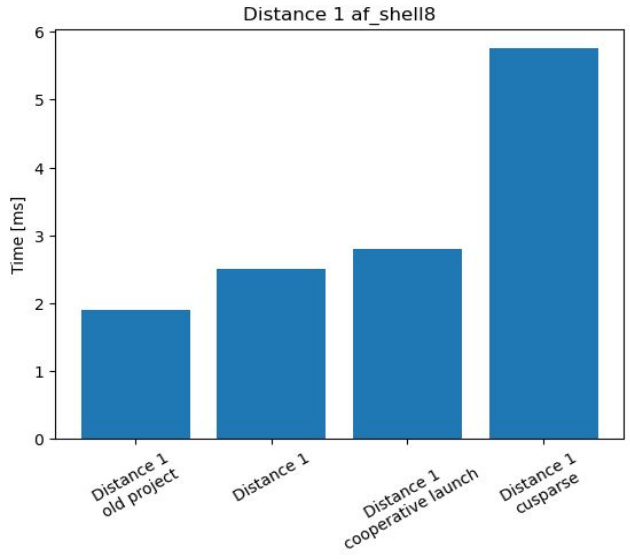


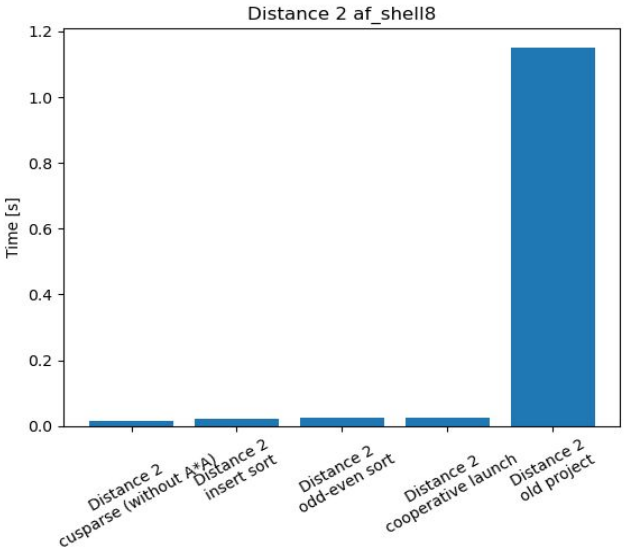
- old project: 4096 colors (k = 3)

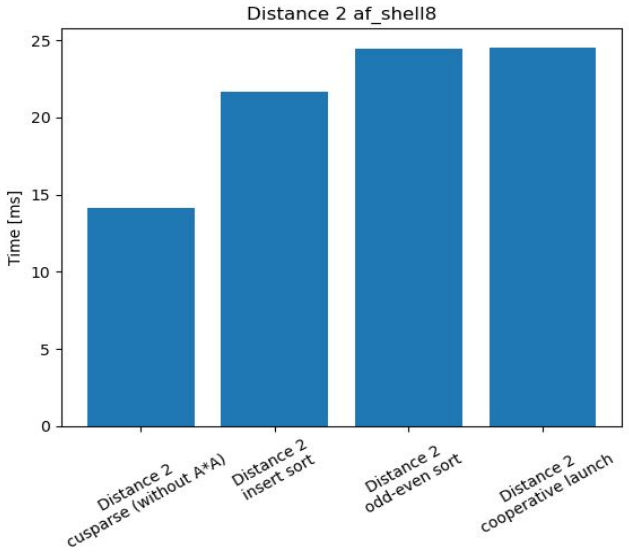


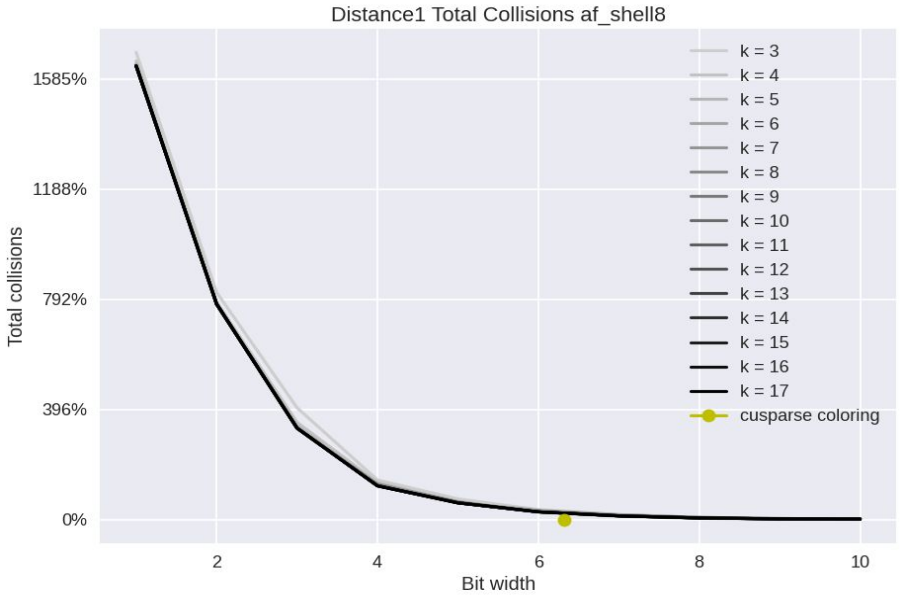


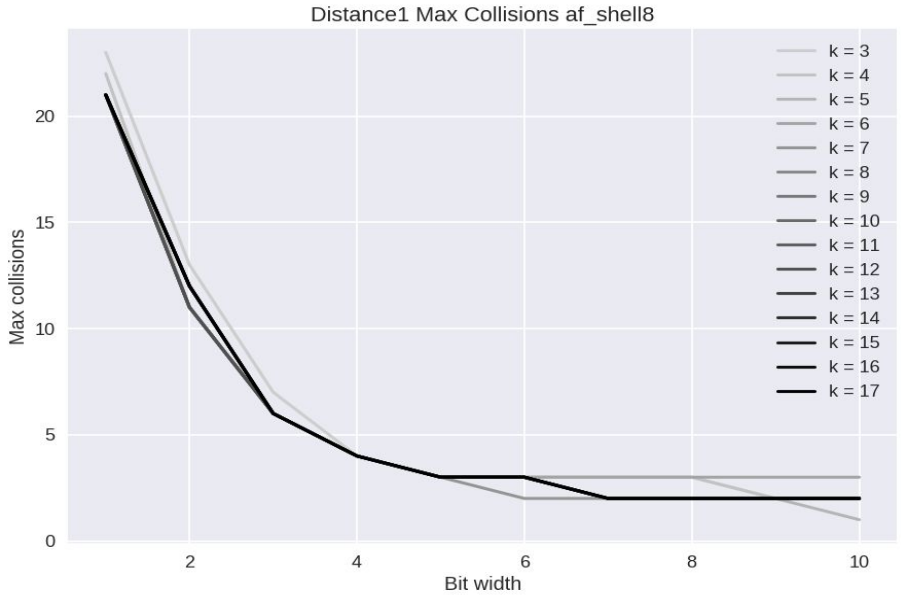
- old project: 8192 colors (k = 4)


Distance 1 af_shell8
Time [ms]
0
1
2
3
4
5
6
Distance 1 old project
Distance 1
Distance 1 cooperative launch
Distance 1 cusparse

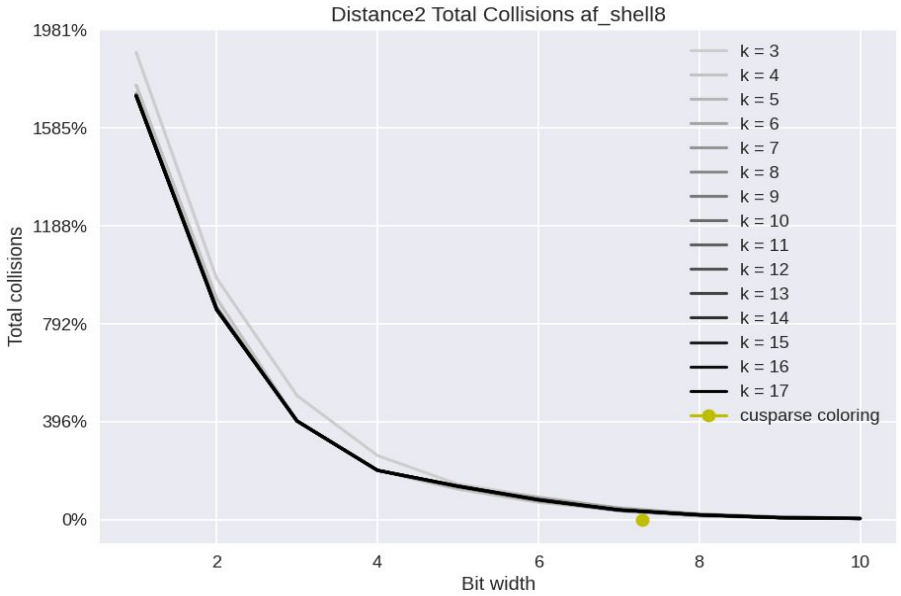

Distance 2 af_shell8
Time [s]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 old project

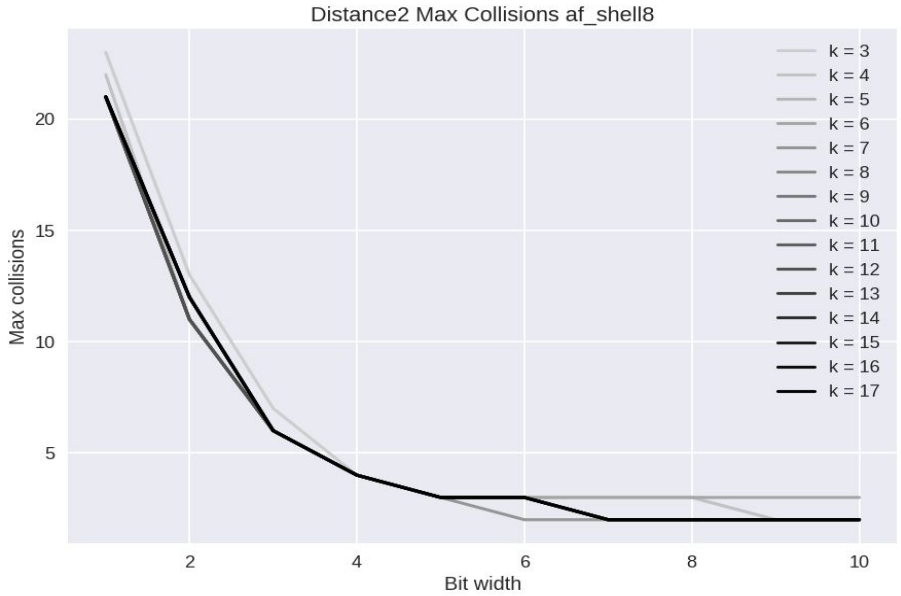

Distance 2 af_shell8
Time [ms]
0
5
10
15
20
25
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch



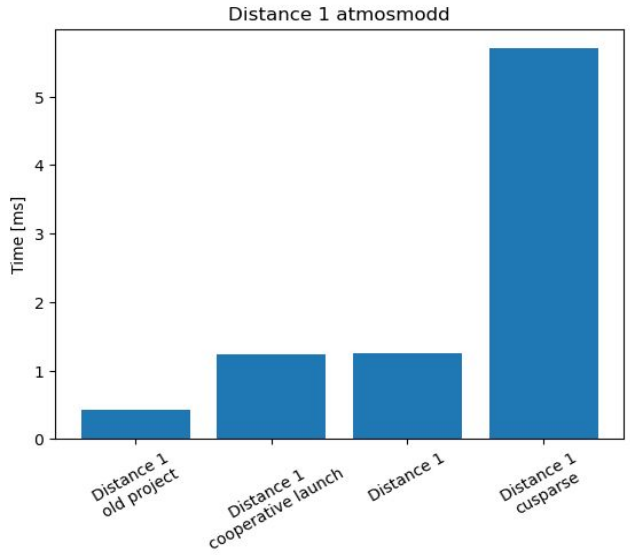


- old project: 4096 colors (k = 3)

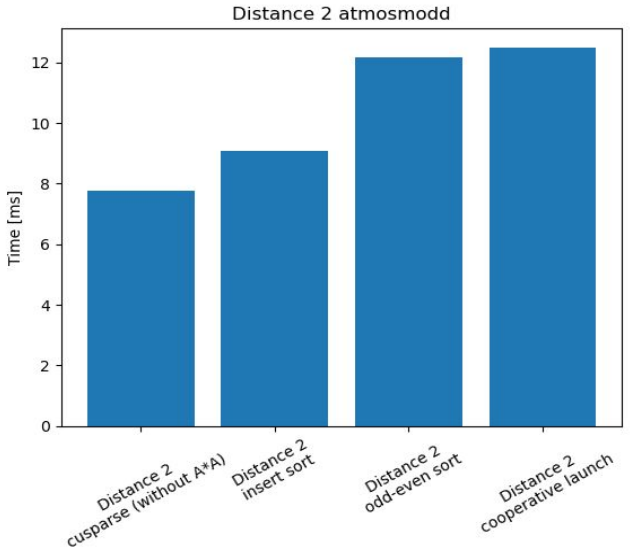


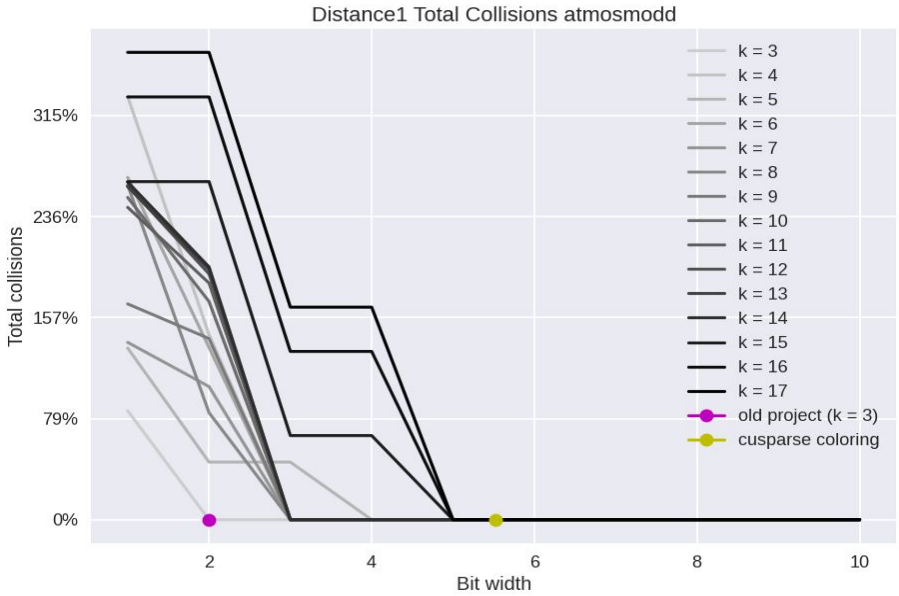


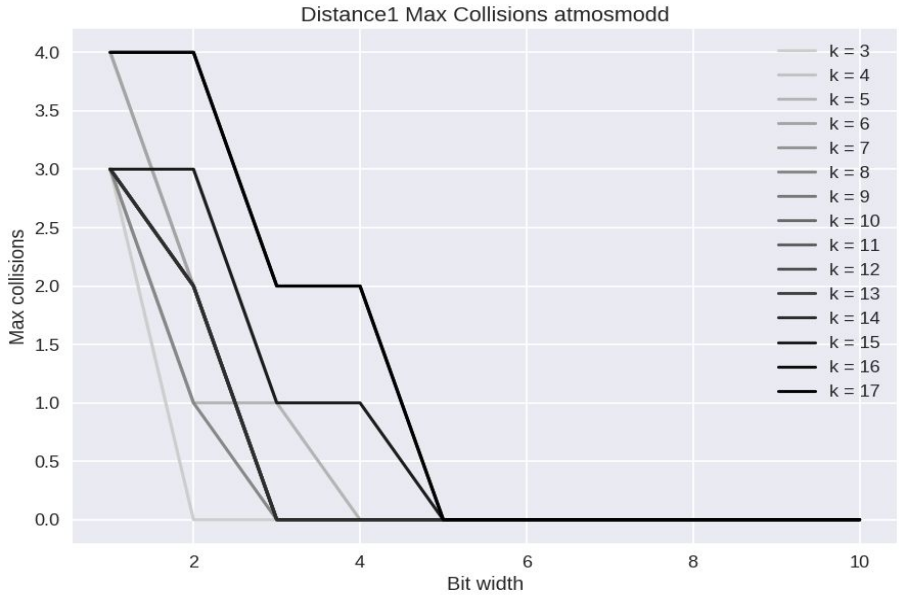
- old project: 8192 colors (k = 4)

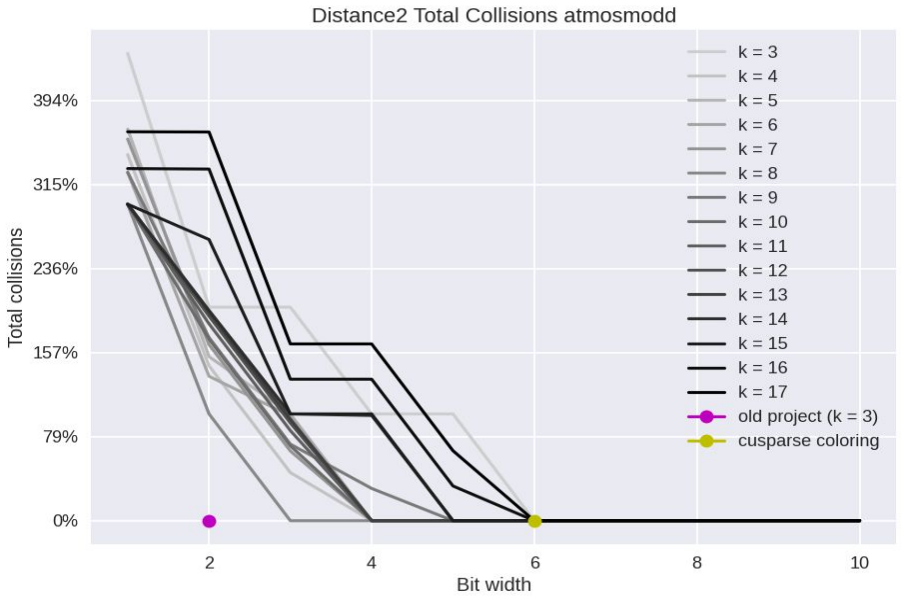

Distance 1 atmosmodd
Time [ms]
0
1
2
3
4
5
Distance 1 old project
Distance 1 cooperative launch
Distance 1
Distance 1 cusparse

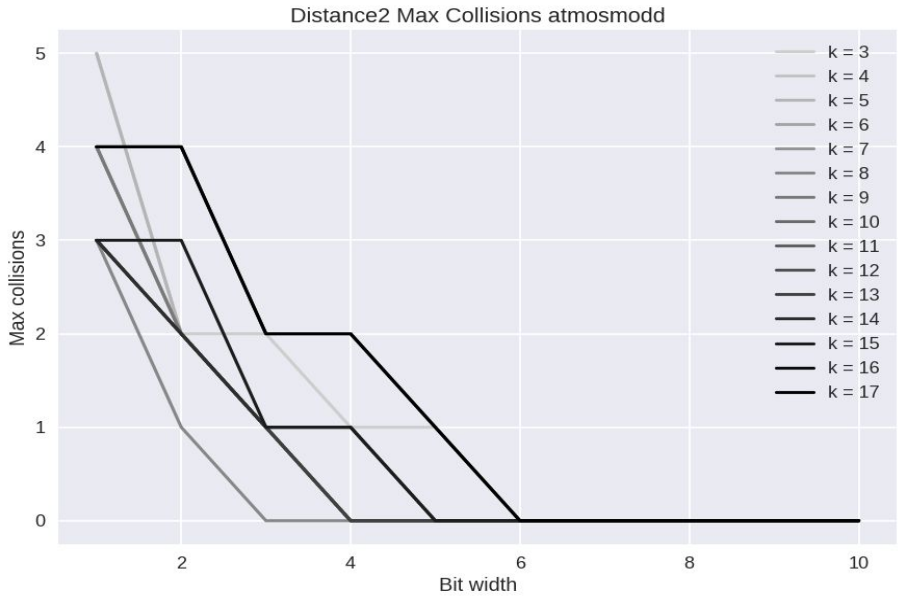

Distance 2 atmosmodd
Time [s]
0.0
0.5
1.0
1.5
2.0
2.5
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 old project

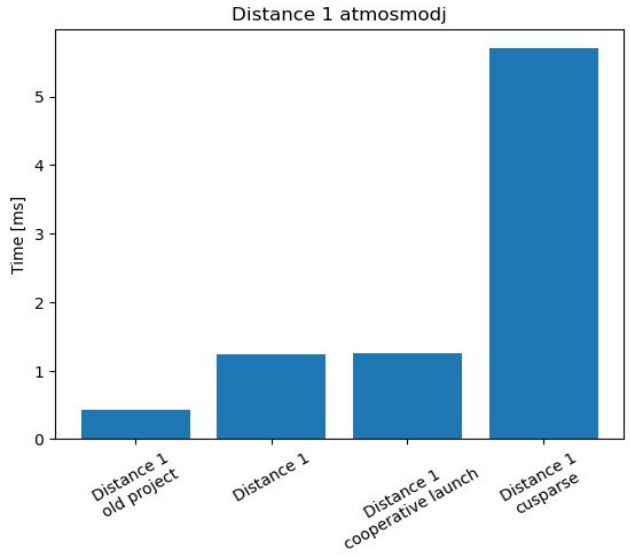

Distance 2 atmosmodd
Time [ms]
0
2
4
6
8
10
12
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch

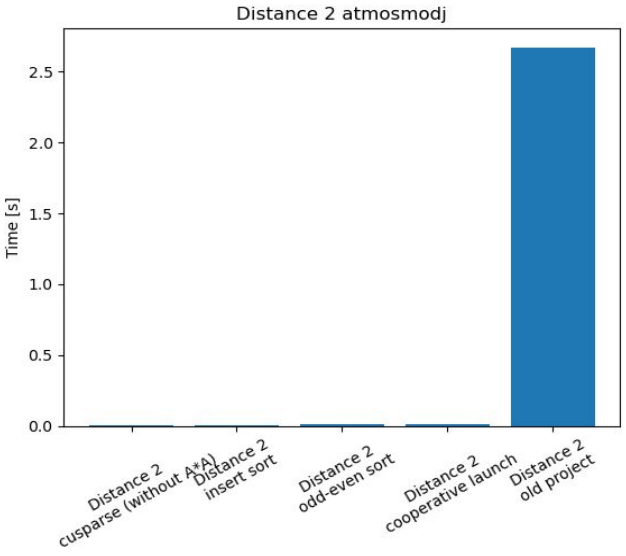

Distance1 Total Collisions atmosmodd
Total collisions
315%
236%
157%
79%
0%
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17
old project (k = 3)
cusparse coloring

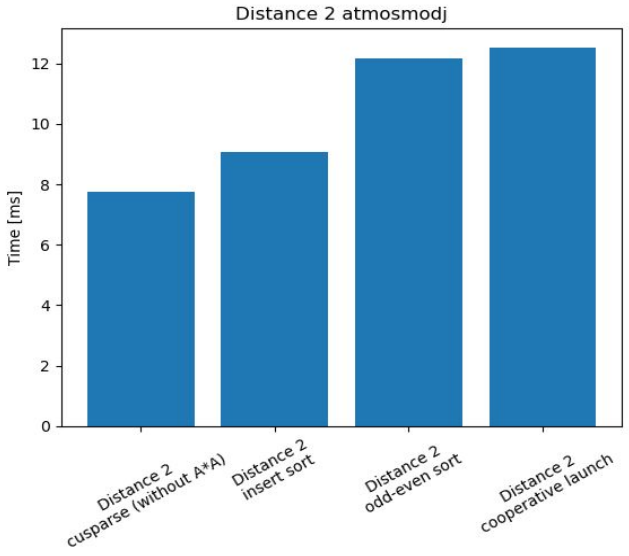

Distance1 Max Collisions atmosmodd
Max collisions
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17

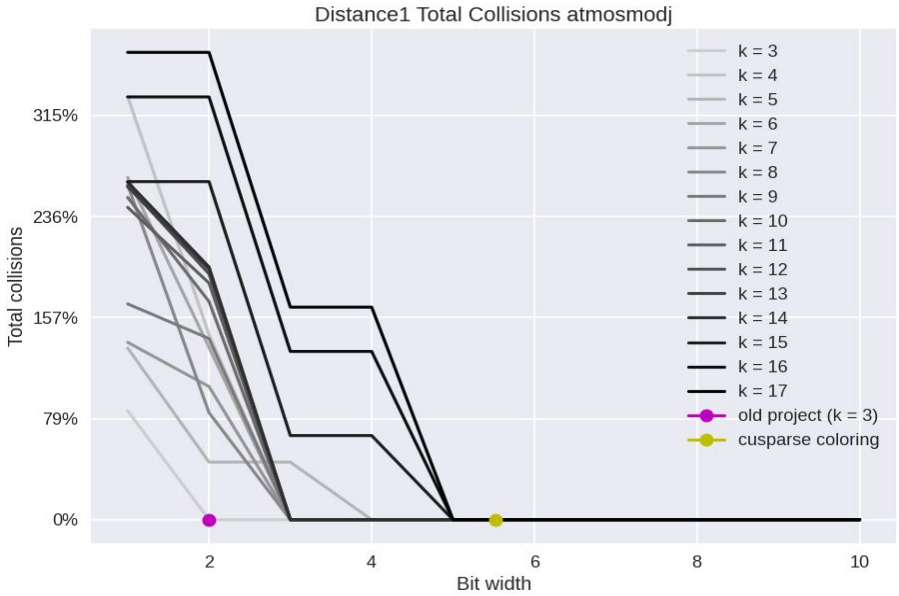

Distance2 Total Collisions atmosmodd
Total collisions
394%
315%
236%
157%
79%
0%
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17
old project (k = 3)
cusparse coloring

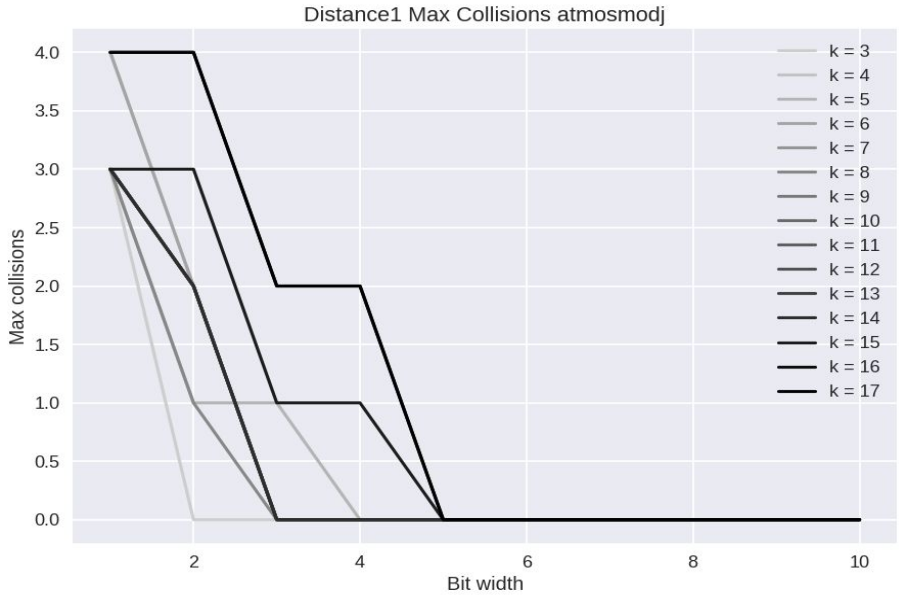

Distance2 Max Collisions atmosmodd
Max collisions
5
4
3
2
1
0
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17

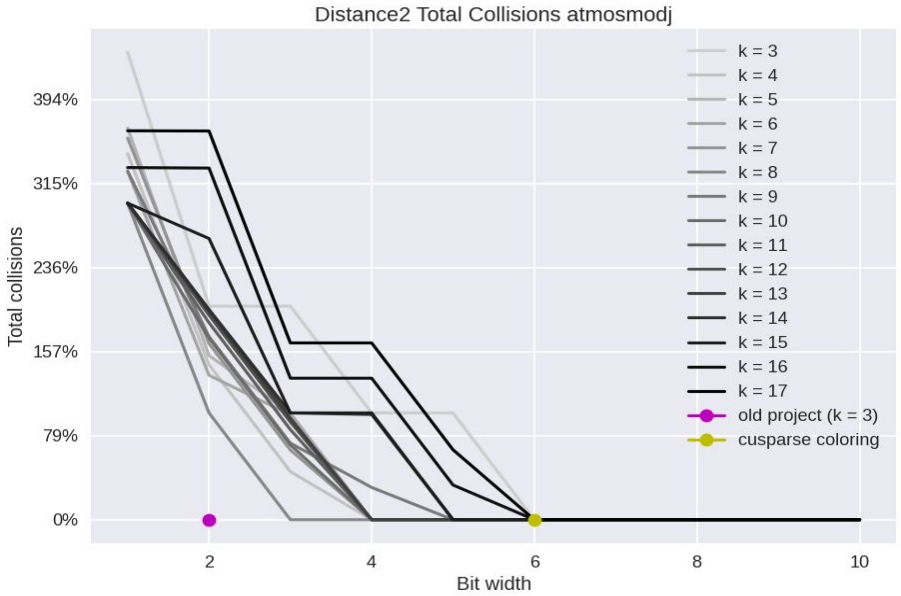

Distance 1 atmosmodj
Time [ms]
0
1
2
3
4
5
Distance 1 old project
Distance 1
Distance 1 cooperative launch
Distance 1 cusparse

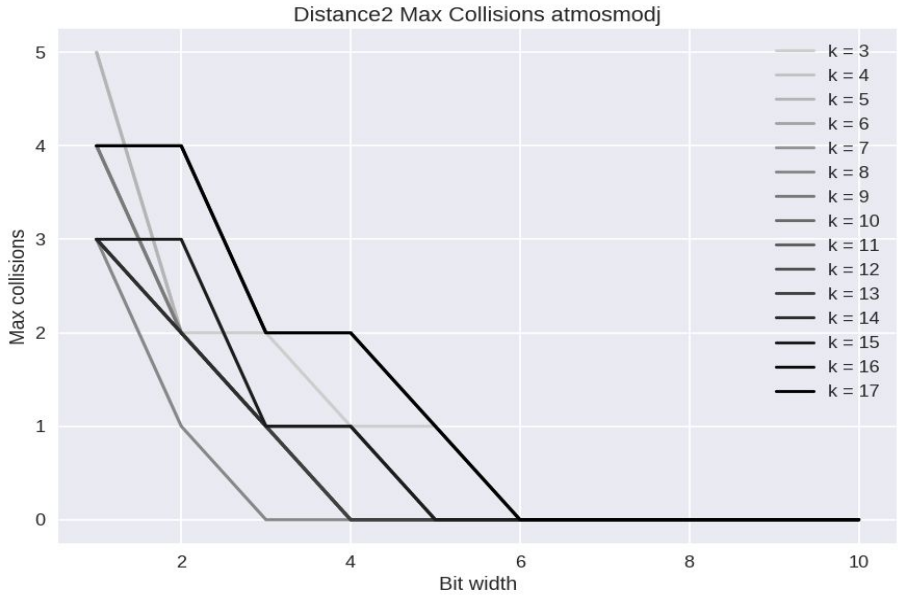

Distance 2 atmosmodj
Time [s]
0.0
0.5
1.0
1.5
2.0
2.5
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 old project

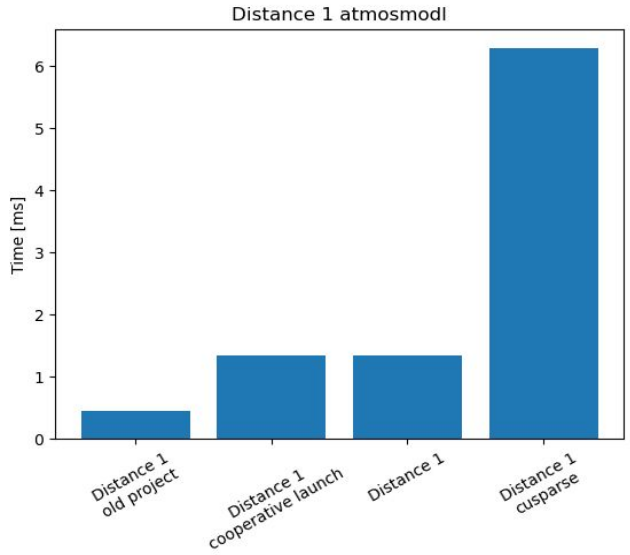

Distance 2 atmosmodj
Time [ms]
0
2
4
6
8
10
12
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch

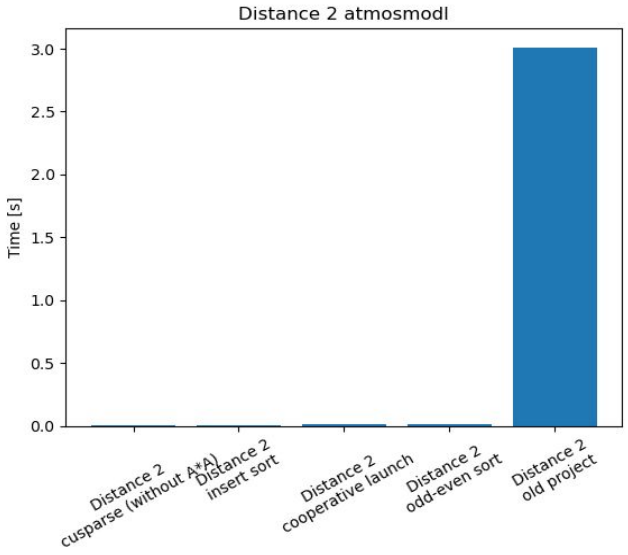

Distance1 Total Collisions atmosmodj
Total collisions
315%
236%
157%
79%
0%
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17
old project (k = 3)
cusparse coloring

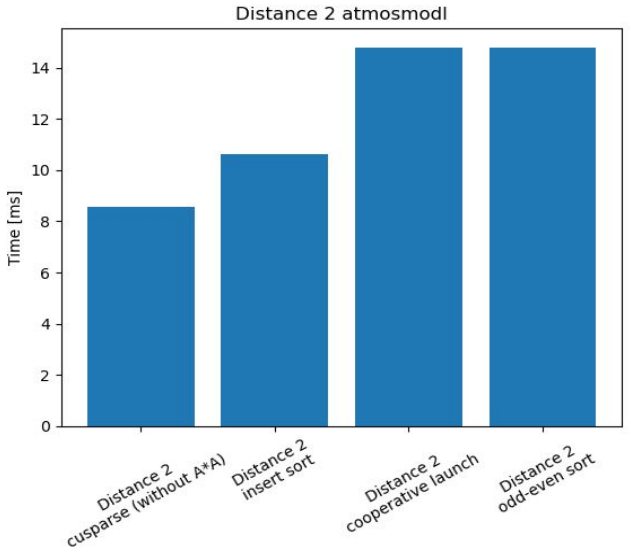

Distance1 Max Collisions atmosmodj
Max collisions
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17

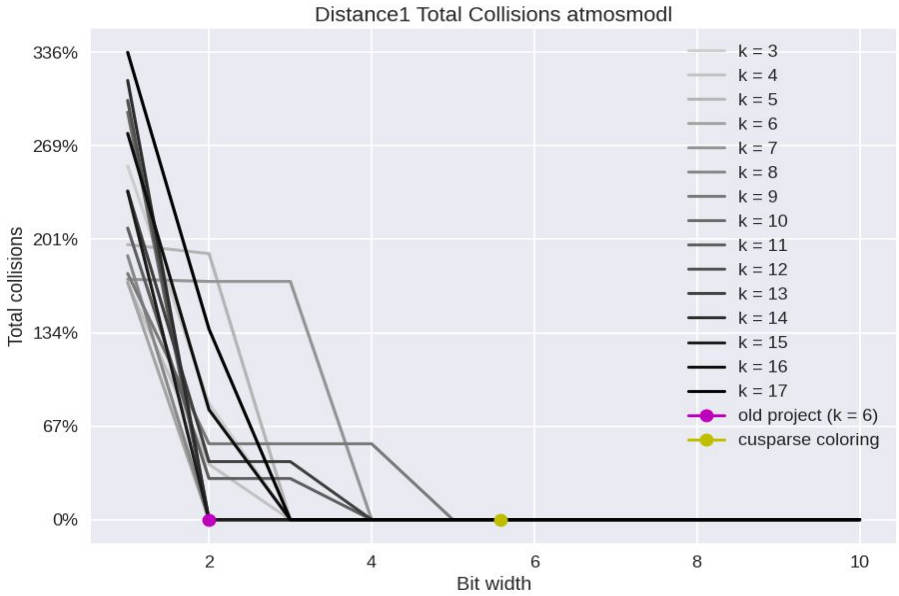

Distance2 Total Collisions atmosmodj
Total collisions
394%
315%
236%
157%
79%
0%
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17
old project (k = 3)
cusparse coloring

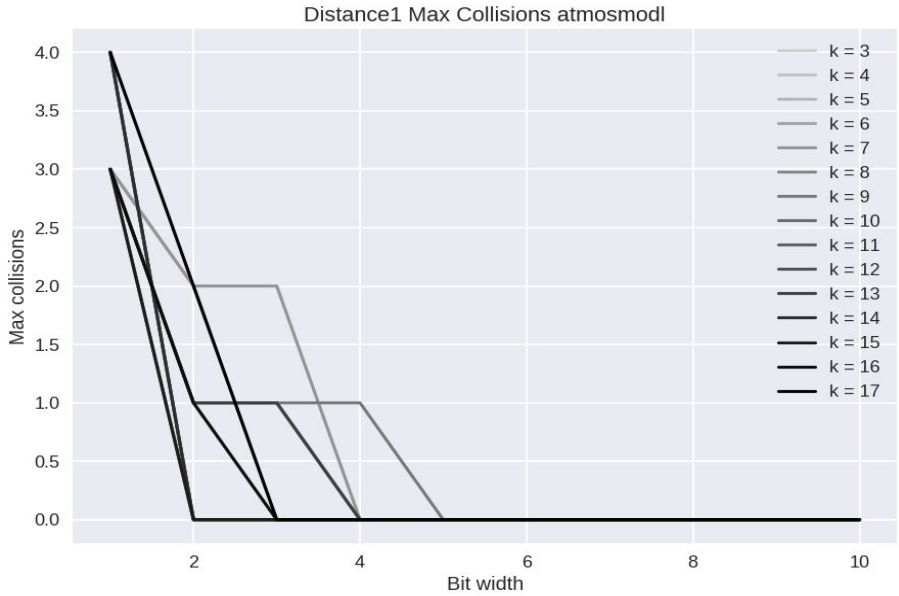

Distance2 Max Collisions atmosmodj
Max collisions
5
4
3
2
1
0
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17

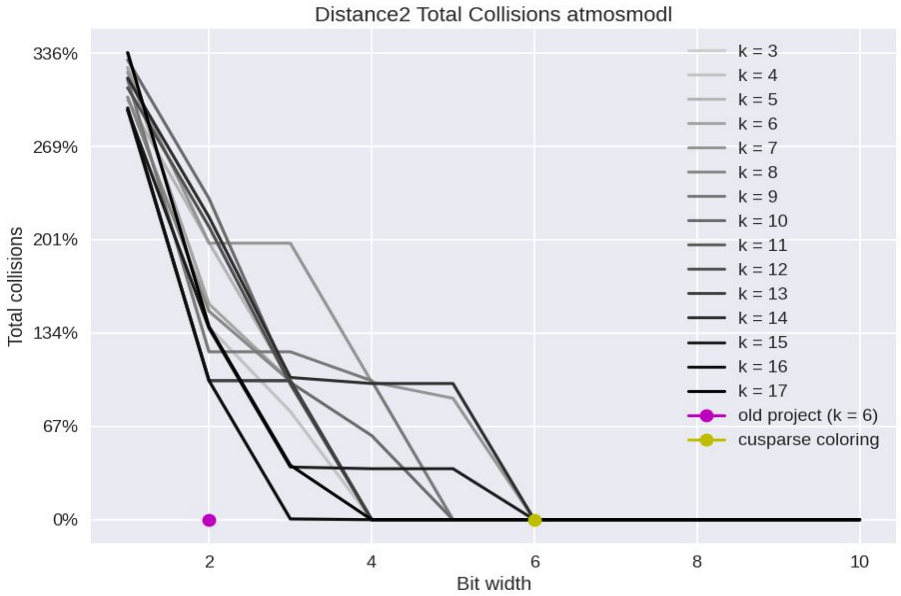

Distance 1 atmosmodl
Time [ms]
0
1
2
3
4
5
6
Distance 1 old project
Distance 1 cooperative launch
Distance 1
Distance 1 cusparse

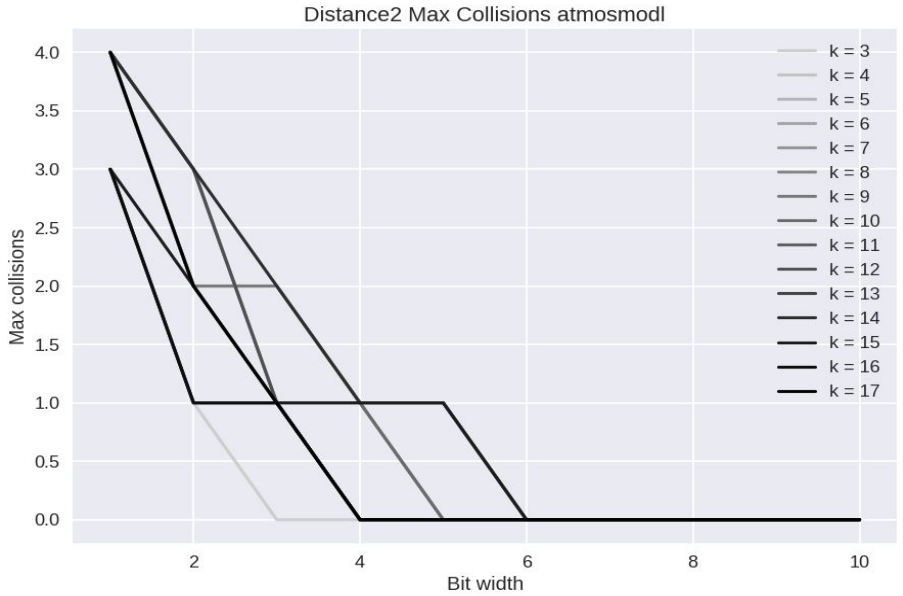

Distance 2 atmosmodl
Time [s]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 cooperative launch
Distance 2 odd-even sort
Distance 2 old project

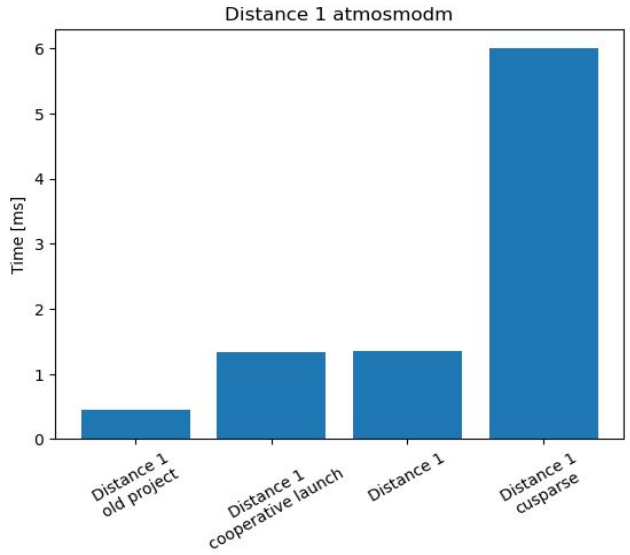

Distance 2 atmosmodl
Time [ms]
0
2
4
6
8
10
12
14
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 cooperative launch
Distance 2 odd-even sort

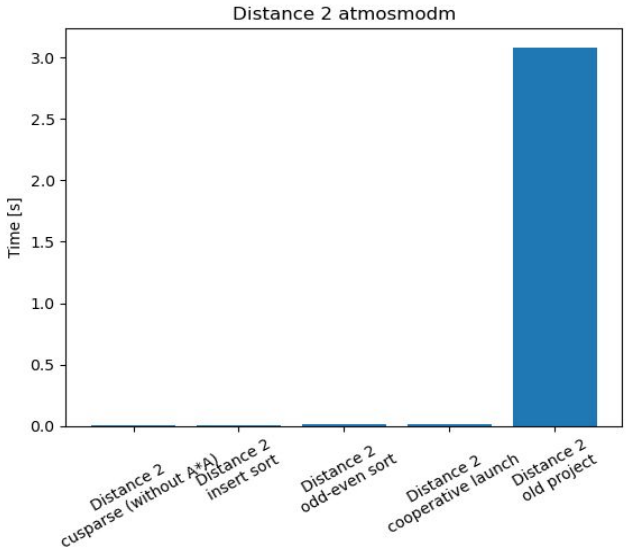

Distance1 Total Collisions atmosmodl
Total collisions
336%
269%
201%
134%
67%
0%
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17
old project (k = 6)
cusparse coloring

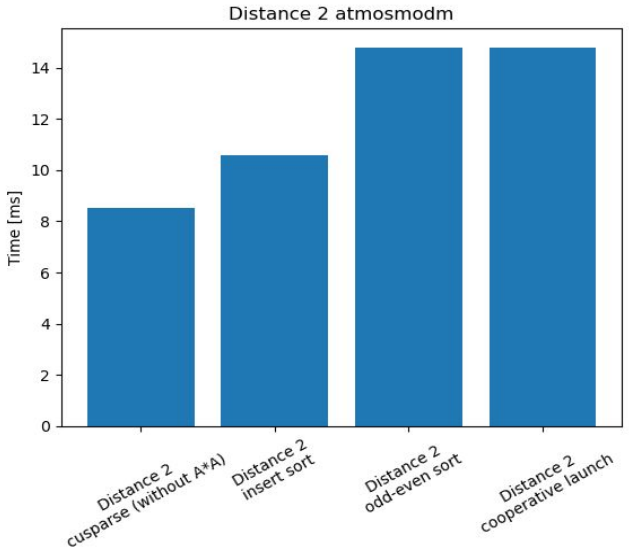

Distance1 Max Collisions atmosmodl
Max collisions
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17

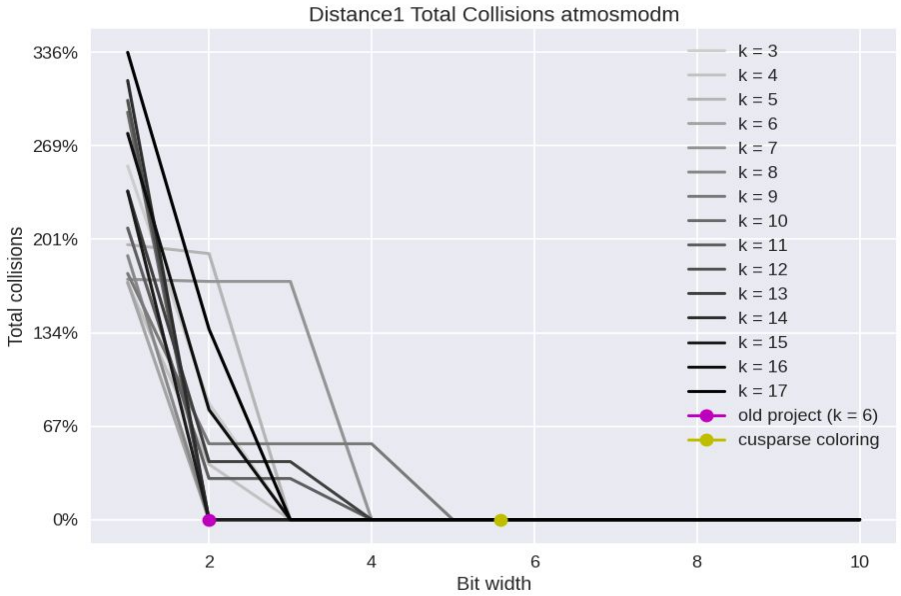

Distance2 Total Collisions atmosmodl
Total collisions
336%
269%
201%
134%
67%
0%
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17
old project (k = 6)
cusparse coloring

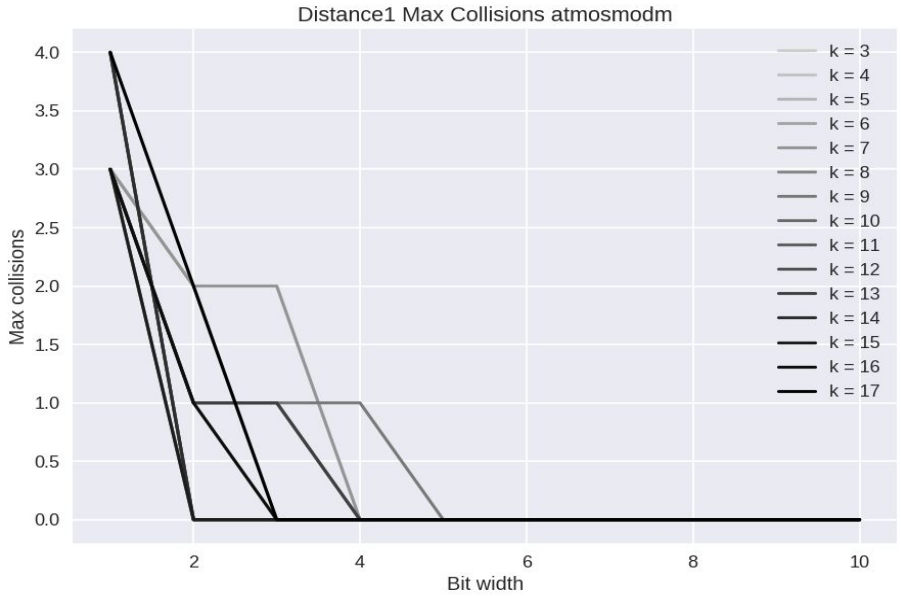

Distance2 Max Collisions atmosmodl
Max collisions
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17

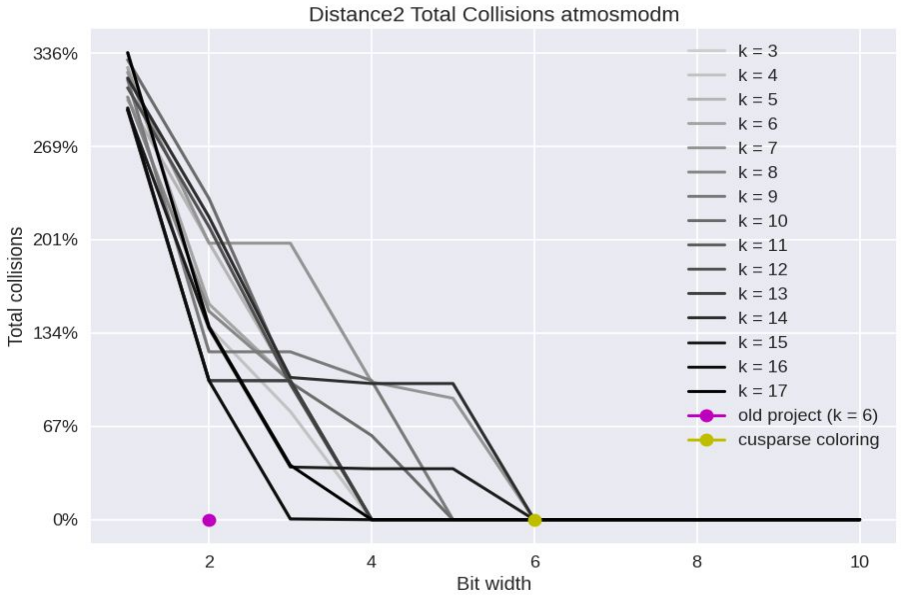

Distance 1 atmosmodm
Time [ms]
0
1
2
3
4
5
6
Distance 1 old project
Distance 1 cooperative launch
Distance 1
Distance 1 cusparse

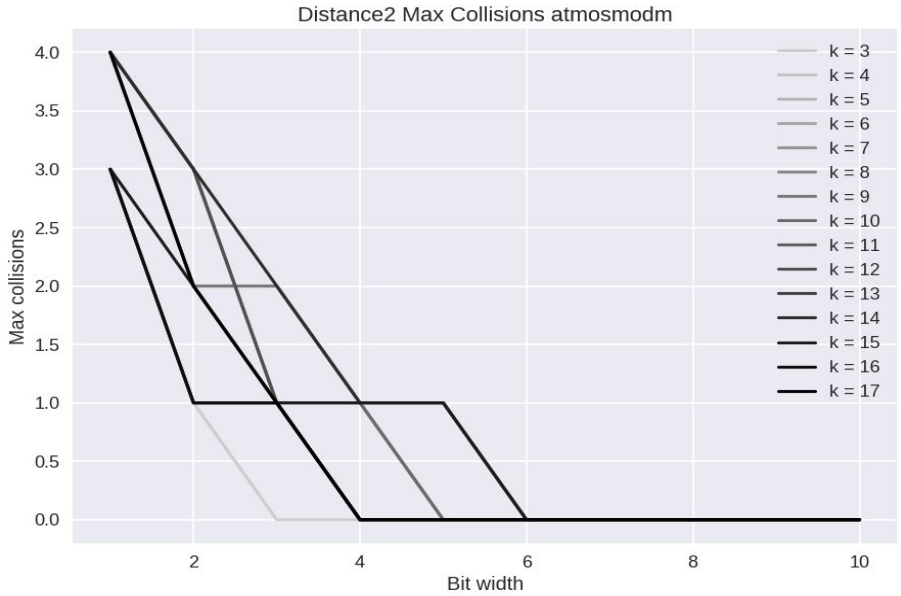

Distance 2 atmosmodm
Time [s]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 old project

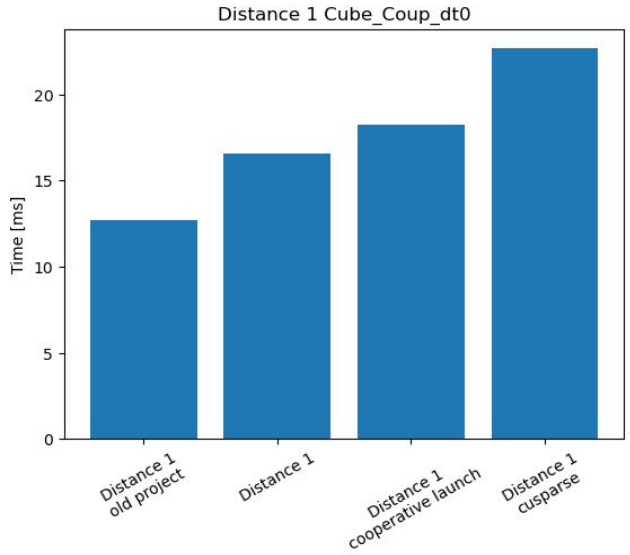

Distance 2 atmosmodm
Time [ms]
0
2
4
6
8
10
12
14
Distance 2 cusparse (without A*A)
Distance 2 insert sort
Distance 2 odd-even sort
Distance 2 cooperative launch

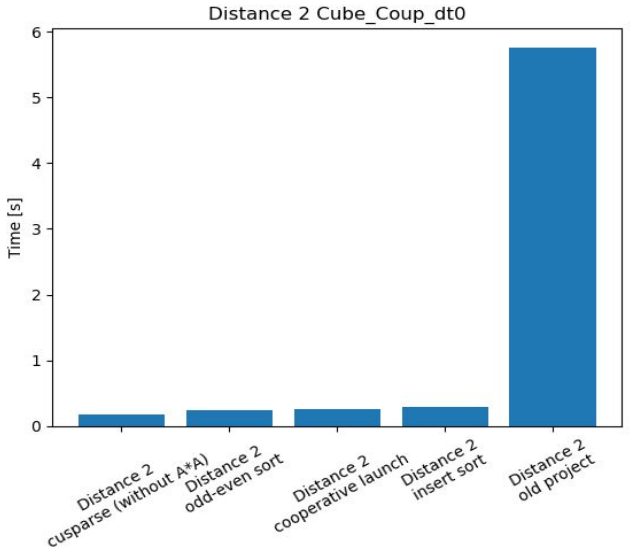

Distance1 Total Collisions atmosmodm
Total collisions
336%
269%
201%
134%
67%
0%
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17
old project (k = 6)
cusparse coloring

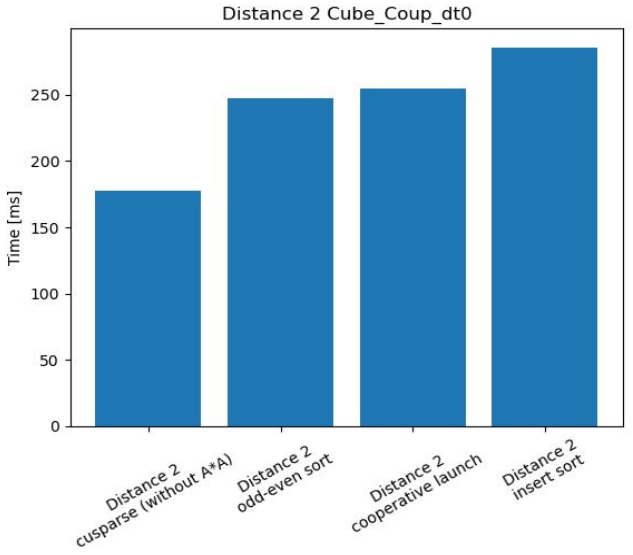

Distance1 Max Collisions atmosmodm
Max collisions
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17

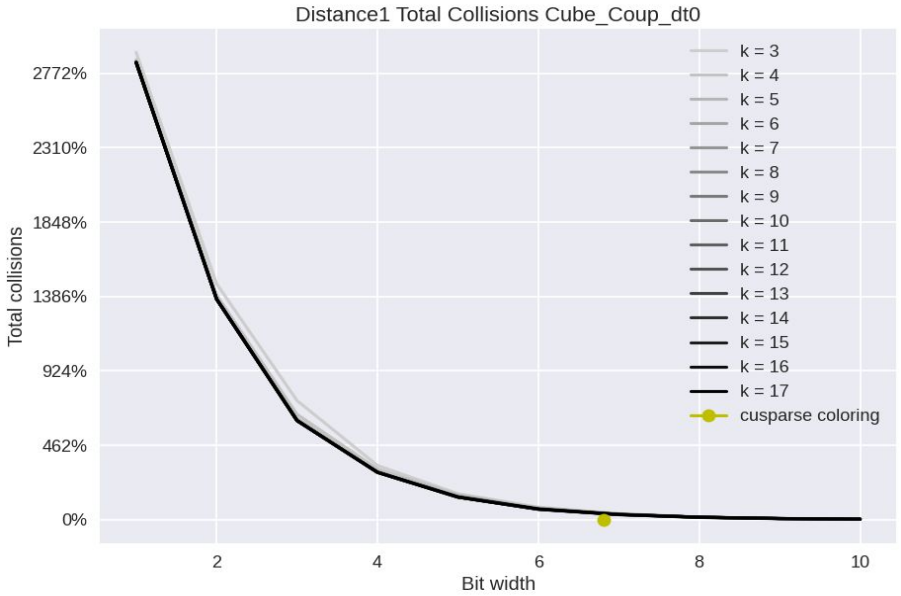

Distance2 Total Collisions atmosmodm
Total collisions
336%
269%
201%
134%
67%
0%
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17
old project (k = 6)
cusparse coloring

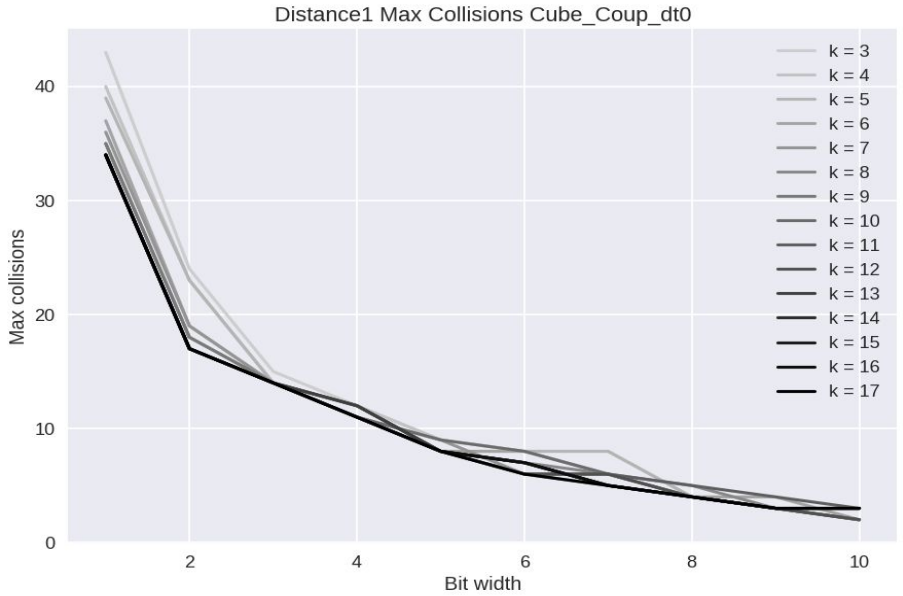

Distance2 Max Collisions atmosmodm
Max collisions
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10
Bit width
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
k = 13
k = 14
k = 15
k = 16
k = 17


Distance 1 Cube_Coup_dt0
Time [ms]
0
5
10
15
20
Distance 1 old project
Distance 1
Distance 1 cooperative launch
Distance 1 cusparse

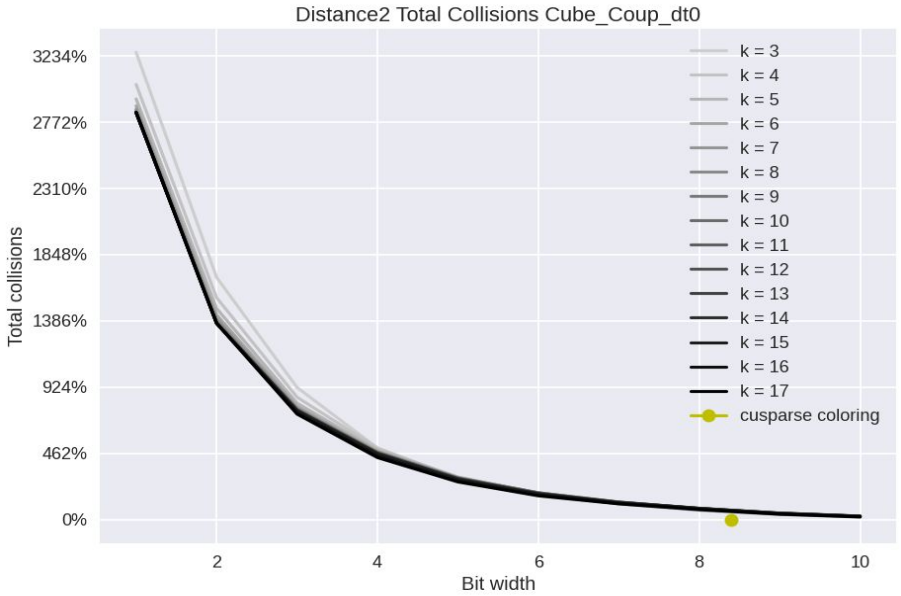

Distance 2 Cube_Coup_dt0
Time [s]
0
1
2
3
4
5
6
Distance 2 cusparse (without A*A)
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 insert sort
Distance 2 old project

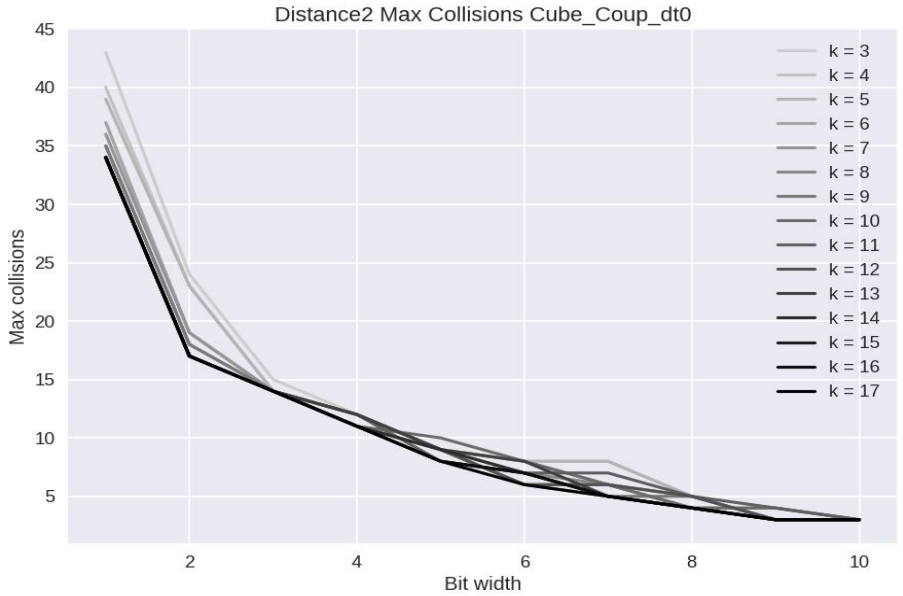

Distance 2 Cube_Coup_dt0
Time [ms]
0
50
100
150
200
250
Distance 2 cusparse (without A*A)
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 insert sort



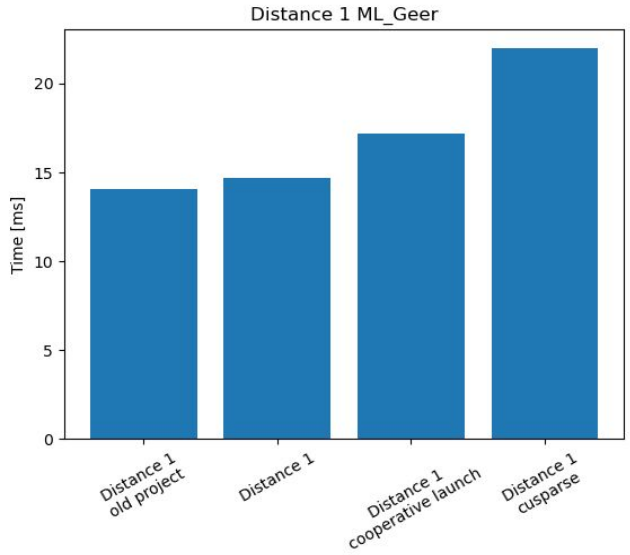


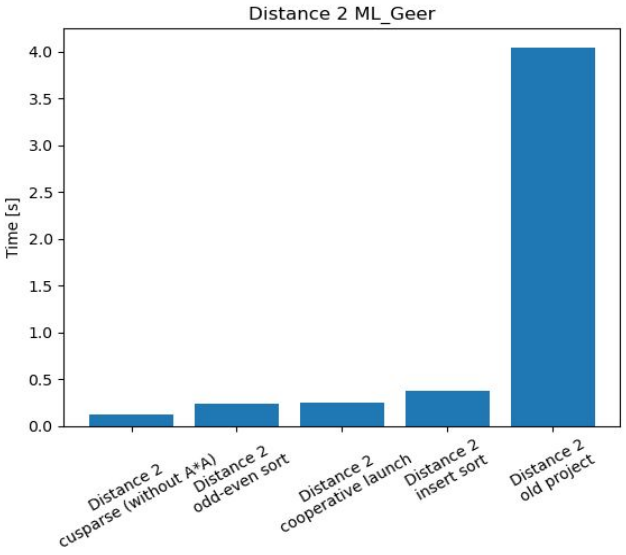
- old project: 32768 colors (k = 5)

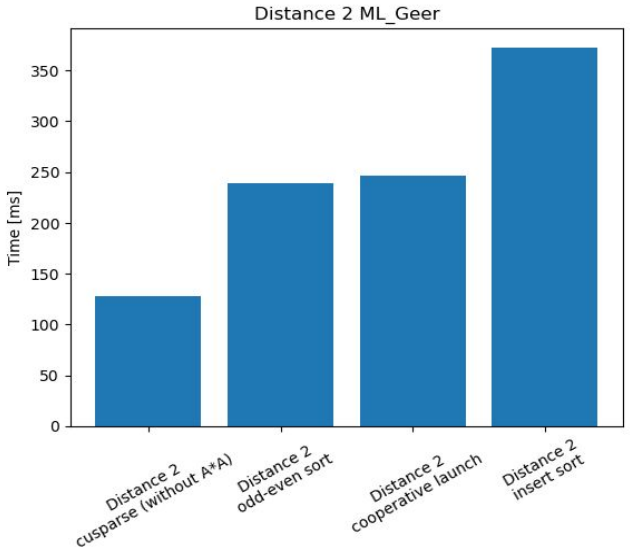


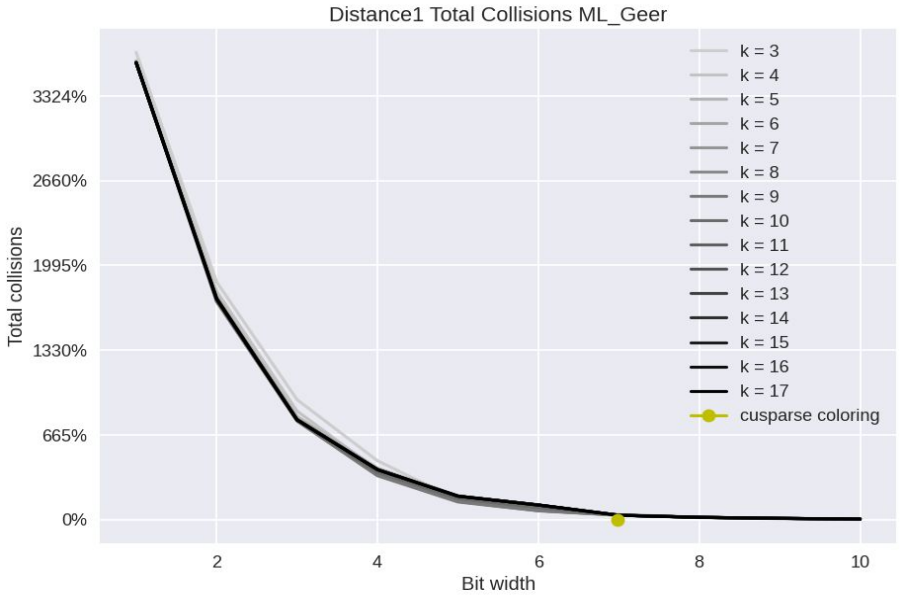


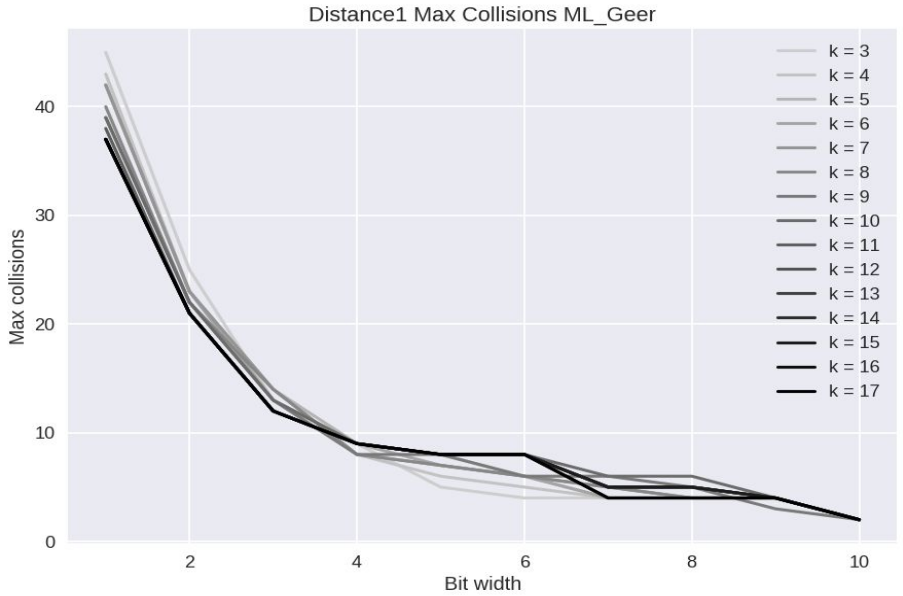
- old project: 65536 colors (k = 5)


Distance 1 ML_Geer
Time [ms]
0
5
10
15
20
Distance 1 old project
Distance 1
Distance 1 cooperative launch
Distance 1 cusparse

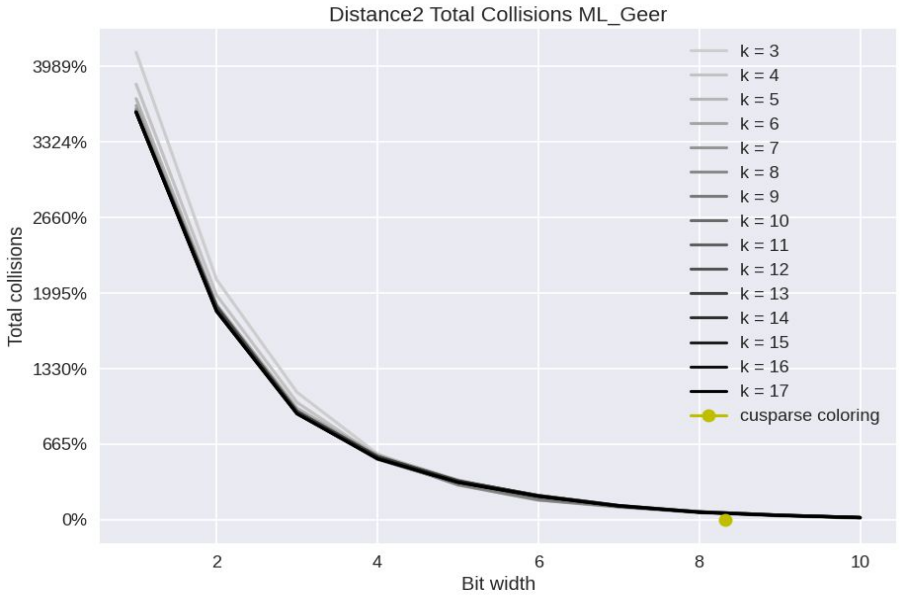

Distance 2 ML_Geer
Time [s]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Distance 2 cusparse (without A*A)
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 insert sort
Distance 2 old project

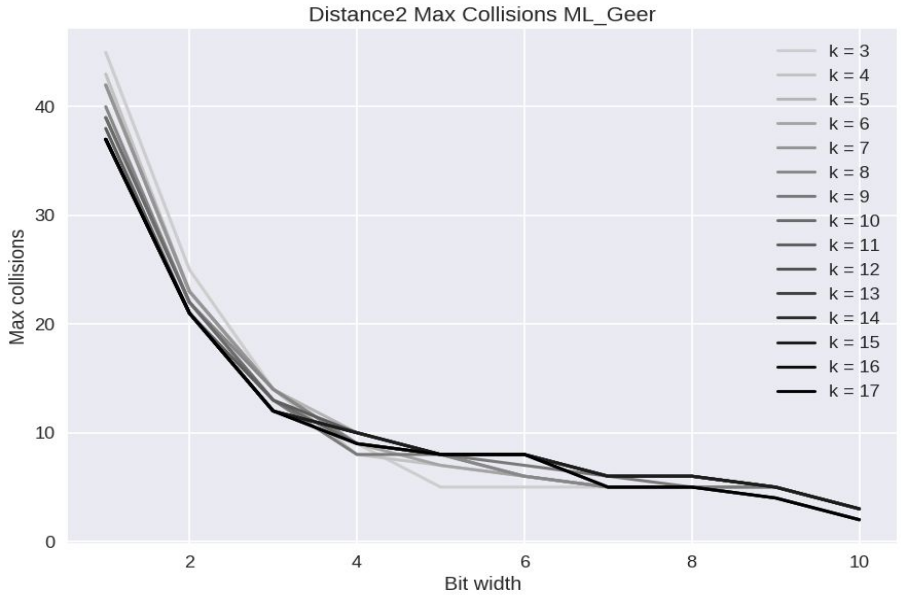

Distance 2 ML_Geer
Time [ms]
0
50
100
150
200
250
300
350
Distance 2 cusparse (without A*A)
Distance 2 odd-even sort
Distance 2 cooperative launch
Distance 2 insert sort





- old project: 8192 colors (k = 3)





- old project: 16384 colors (k = 3)