Facet-JFA: Faster computation of discrete Voronoi diagrams

Supplementary material

Talha Bin Masood
Hari Krishna Malladi
Vijay Natarajan

Dept. of Computer Science and Automation,
Indian Institute of Science, Bangalore.

Table S-1. Timing results for Facet-JFA and JFA for **2D grids on Kepler**.

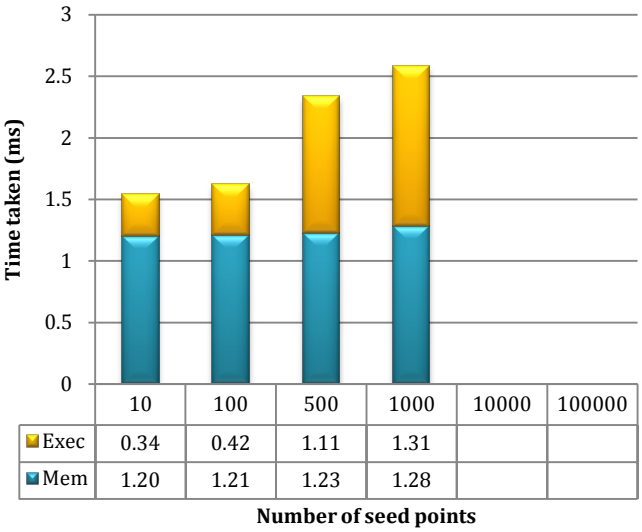
Grid size (n)	No. of seed points	Facet-JFA time (ms)			JFA time (ms)			Total Speed up	Exec. Speed up
		Mem.	Exec.	Total	Mem.	Exec.	Total		
256	10	1.20	0.34	1.54	1.22	0.55	1.77	**1.15**	**1.61**
	100	1.21	0.42	1.62	1.20	0.58	1.78	**1.10**	**1.40**
	500	1.23	1.11	2.34	1.25	1.10	2.35	**1.01**	**0.99**
	1000	1.28	1.31	2.58	1.24	1.39	2.62	**1.02**	**1.06**
512	10	1.81	0.66	2.47	1.79	2.08	3.88	**1.57**	**3.17**
	100	1.81	1.21	3.02	1.77	2.24	4.01	**1.33**	**1.86**
	500	1.82	3.87	5.69	1.75	3.41	5.16	**0.91**	**0.88**
	1000	1.77	4.41	6.19	1.80	4.19	6.00	**0.97**	**0.95**
1024	10	3.62	1.34	4.96	3.58	8.04	11.61	**2.34**	**5.99**
	100	3.61	2.43	6.04	3.51	8.39	11.90	**1.97**	**3.45**
	500	3.63	7.83	11.46	3.49	12.05	15.54	**1.36**	**1.54**
	1000	3.67	14.64	18.31	3.49	14.37	17.86	**0.98**	**0.98**
	10000	3.63	23.77	27.40	3.48	22.99	26.46	**0.97**	**0.97**
2048	10	10.66	3.73	14.38	9.75	28.95	38.70	**2.69**	**7.77**
	100	10.46	6.59	17.05	9.60	31.37	40.97	**2.40**	**4.76**
	500	10.38	15.33	25.71	9.74	47.83	57.57	**2.24**	**3.12**
	1000	10.27	27.62	37.89	9.74	57.02	66.76	**1.76**	**2.06**
	10000	10.38	90.38	100.76	10.71	95.38	106.09	**1.05**	**1.06**
4096	10	36.10	12.13	48.22	35.48	131.88	167.36	**3.47**	**10.87**
	100	35.77	18.52	54.28	35.38	140.51	175.89	**3.24**	**7.59**
	500	35.73	37.70	73.43	35.54	199.05	234.58	**3.19**	**5.28**
	1000	36.07	67.49	103.56	35.22	236.12	271.34	**2.62**	**3.50**
	10000	35.52	368.32	403.84	35.62	331.23	366.85	**0.91**	**0.90**
	100000	36.90	506.09	542.98	35.62	494.46	530.08	**0.98**	**0.98**

- Please see **Notes** section of Figure S-1 for explanation of *Mem* and *Exec* times.
- Speed-ups greater than 2 are highlighted in red.

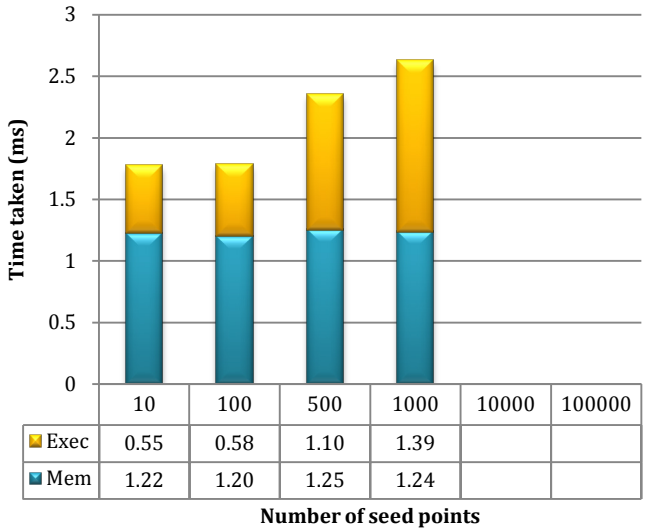
Figure S-1. Timing comparison of Facet-JFA and JFA for **2D case** on **Kepler** architecture (Left: Facet-JFA, Right: JFA)

Notes

- Following plots are for data presented in **Table S-1**.
- GPU: nVidia GTX660Ti (Kepler architecture).
- Facet-JFA timings plots are on the left while JFA timing plots for same grid size are shown on the right.
- *Exec.* is the time spent in execution of CUDA kernels, which also includes time to access the memory.
- *Mem.* is the time taken to transfer the input grid from host to device memory and the results back from device to host.

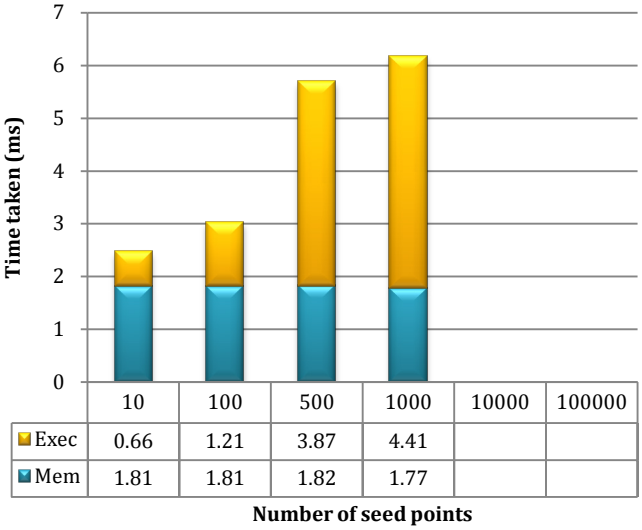


	10	100	500	1000	10000	100000
Exec	0.34	0.42	1.11	1.31		
Mem	1.20	1.21	1.23	1.28		

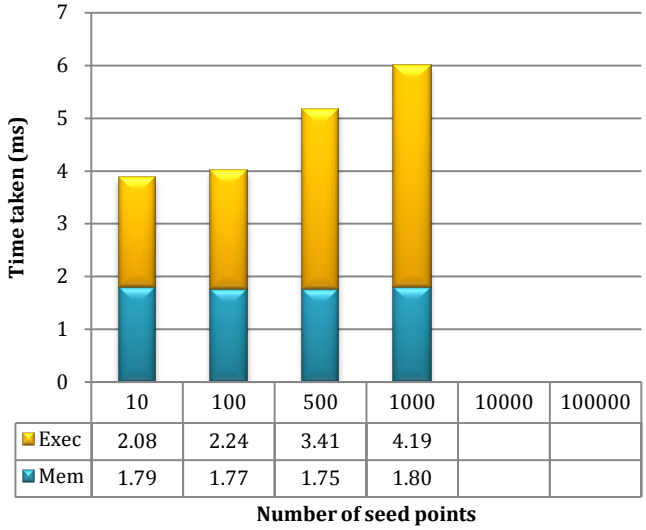


	10	100	500	1000	10000	100000
Exec	0.55	0.58	1.10	1.39		
Mem	1.22	1.20	1.25	1.24		

256 x 256



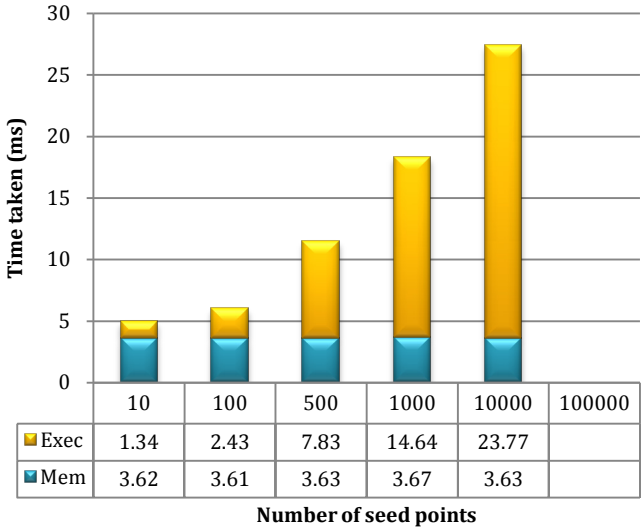
	10	100	500	1000	10000	100000
Exec	0.66	1.21	3.87	4.41		
Mem	1.81	1.81	1.82	1.77		



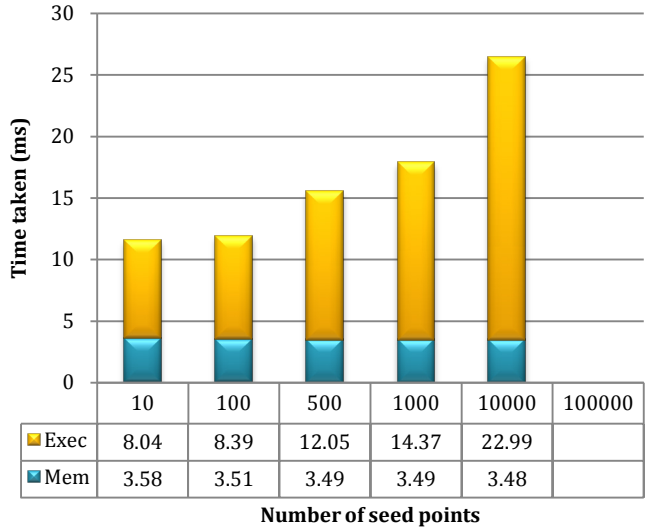
	10	100	500	1000	10000	100000
Exec	2.08	2.24	3.41	4.19		
Mem	1.79	1.77	1.75	1.80		

512 x 512

Left: Facet-JFA, Right: JFA

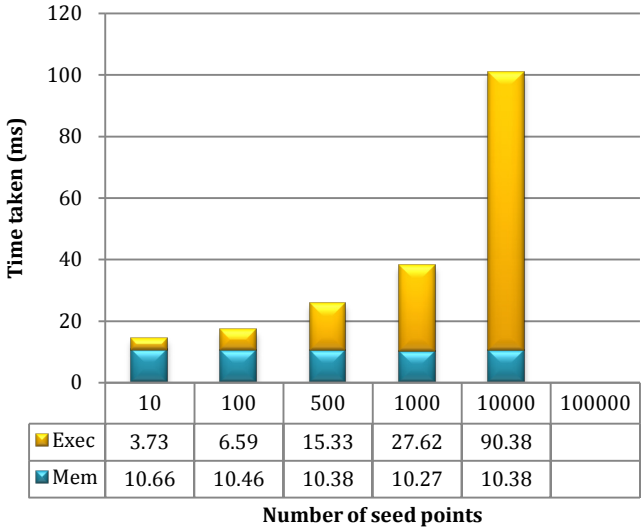


	10	100	500	1000	10000	100000
Exec	1.34	2.43	7.83	14.64	23.77	
Mem	3.62	3.61	3.63	3.67	3.63	

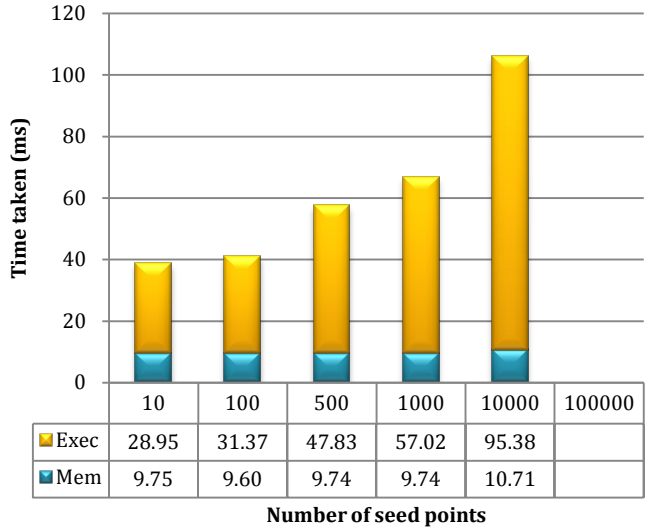


	10	100	500	1000	10000	100000
Exec	8.04	8.39	12.05	14.37	22.99	
Mem	3.58	3.51	3.49	3.49	3.48	

1024 x 1024

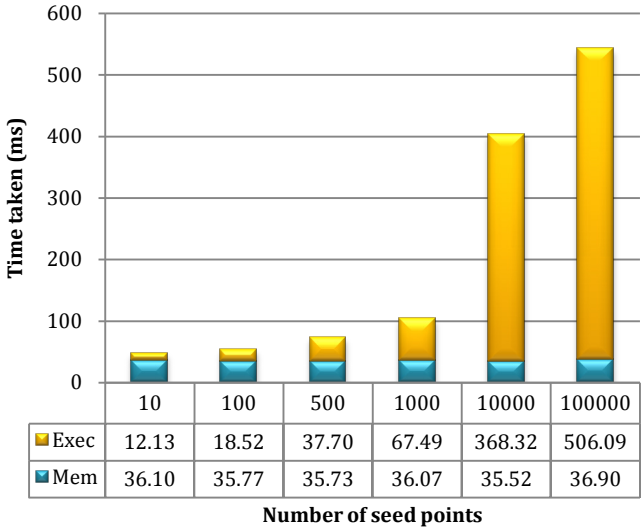


	10	100	500	1000	10000	100000
Exec	3.73	6.59	15.33	27.62	90.38	
Mem	10.66	10.46	10.38	10.27	10.38	

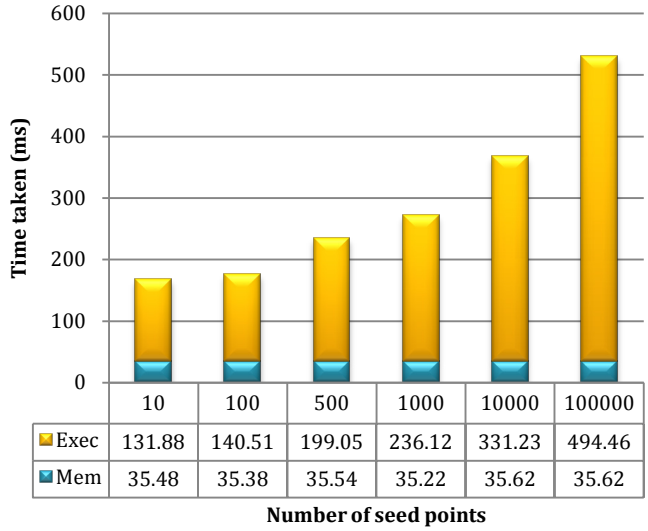


	10	100	500	1000	10000	100000
Exec	28.95	31.37	47.83	57.02	95.38	
Mem	9.75	9.60	9.74	9.74	10.71	

2048 x 2048



	10	100	500	1000	10000	100000
Exec	12.13	18.52	37.70	67.49	368.32	506.09
Mem	36.10	35.77	35.73	36.07	35.52	36.90



	10	100	500	1000	10000	100000
Exec	131.88	140.51	199.05	236.12	331.23	494.46
Mem	35.48	35.38	35.54	35.22	35.62	35.62

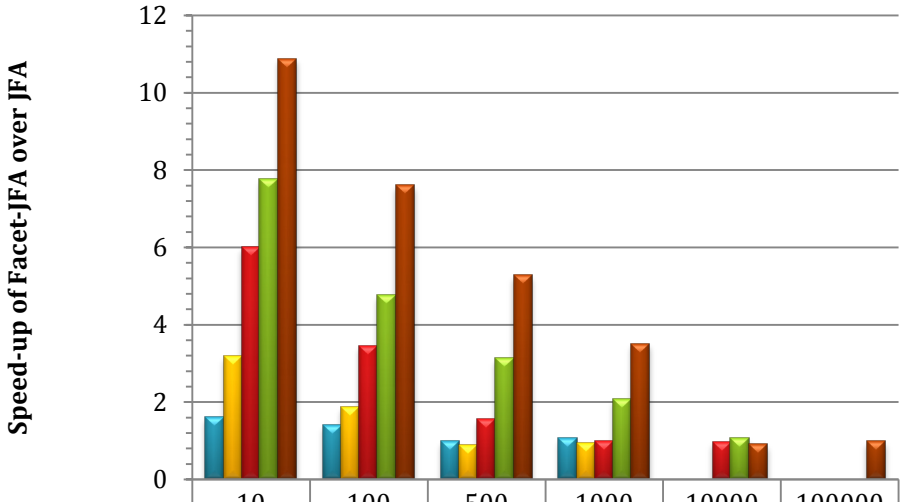
4096 x 4096

Table S-2. Unmarked pixels in Facet-JFA for 2D case (fewer the better).

Grid size (n)	No. of seed points	Total Unmarked pixels	Total pixels	*Savings ratio**
256	10	10,260	65,536	**6.39**
	100	36,300	65,536	**1.81**
	500	65,536	65,536	**1.00**
	1000	65,536	65,536	**1.00**
512	10	19,801	262,144	**13.24**
	100	83,055	262,144	**3.16**
	500	258,289	262,144	**1.01**
	1000	262,144	262,144	**1.00**
1024	10	50,180	1,048,576	**20.90**
	100	180,754	1,048,576	**5.80**
	500	552,137	1,048,576	**1.90**
	1000	1,048,576	1,048,576	**1.00**
	10000	1,048,576	1,048,576	**1.00**
2048	10	92,772	4,194,304	**45.21**
	100	376,677	4,194,304	**11.14**
	500	909,660	4,194,304	**4.61**
	1000	2,322,336	4,194,304	**1.81**
	10000	4,194,304	4,194,304	**1.00**
4096	10	190,093	16,777,216	**88.26**
	100	796,206	16,777,216	**21.07**
	500	2,420,779	16,777,216	**6.93**
	1000	5,595,679	16,777,216	**3.00**
	10000	16,777,216	16,777,216	**1.00**
	100000	16,777,216	16,777,216	**1.00**

***** *Savings ratio* is computed as the ratio of total pixels and the unmarked pixels. As, unmarked pixels correspond to working threads, it captures savings in work done due to Facet-JFA against JFA.

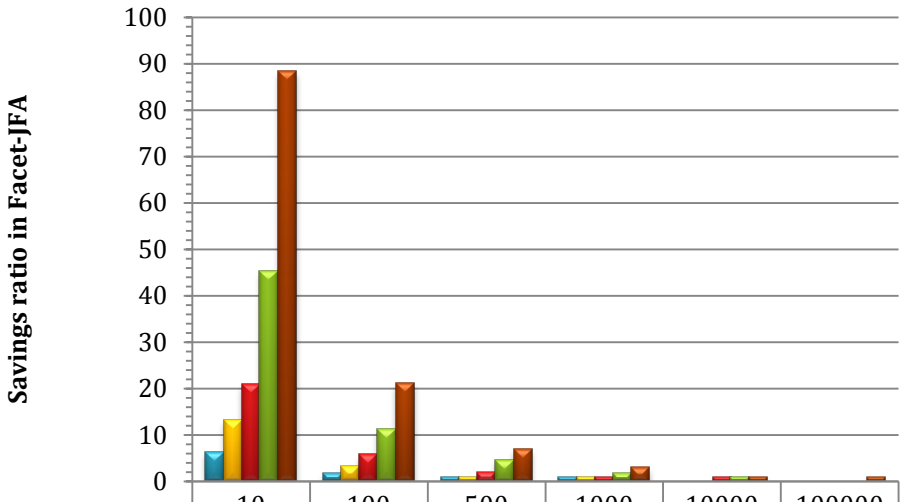
Figure S-2. Experimental confirmation that speed-up is consequence of marking. Observed speed-up is directly proportional to *savings ratio*.



	10	100	500	1000	10000	100000
256 x 256	1.61	1.40	0.99	1.06		
512 x 512	3.17	1.86	0.88	0.95		
1024 x 1024	5.99	3.45	1.54	0.98	0.97	
2048 x 2048	7.77	4.76	3.12	2.06	1.06	
4096 x 4096	10.87	7.59	5.28	3.50	0.90	0.98

Number of seed points

(a) Observed Speed-ups



	10	100	500	1000	10000	100000
256 x 256	6.39	1.81	1.00	1.00		
512 x 512	13.24	3.16	1.01	1.00		
1024 x 1024	20.90	5.80	1.90	1.00	1.00	
2048 x 2048	45.21	11.14	4.61	1.81	1.00	
4096 x 4096	88.26	21.07	6.93	3.00	1.00	1.00

Number of seed points

(b) Savings ratio in Facet-JFA

Table S-3. Timing results for Facet-JFA and JFA for **2D grids on Fermi**.

Grid size (n)	No. of seed points	Facet-JFA time (ms)			JFA time (ms)			Total Speed up	Exec. Speed up
		Mem.	Exec.	Total	Mem.	Exec.	Total		
256	10	3.08	0.40	3.47	2.97	0.46	3.44	**0.99**	**1.17**
	100	2.96	0.45	3.41	2.89	0.54	3.43	**1.00**	**1.20**
	500	2.84	0.75	3.60	3.06	0.66	3.72	**1.03**	**0.88**
	1000	2.96	0.84	3.79	2.98	0.73	3.72	**0.98**	**0.88**
512	10	3.64	0.77	4.41	3.90	1.76	5.66	**1.28**	**2.28**
	100	3.45	1.01	4.47	3.54	2.12	5.66	**1.27**	**2.09**
	500	4.00	2.89	6.89	3.65	2.47	6.12	**0.89**	**0.85**
	1000	3.44	3.22	6.66	3.90	2.73	6.62	**0.99**	**0.85**
1024	10	4.98	1.75	6.73	4.90	7.37	12.27	**1.82**	**4.22**
	100	5.75	2.67	8.42	5.11	8.69	13.80	**1.64**	**3.25**
	500	5.97	11.10	17.07	5.54	10.36	15.90	**0.93**	**0.93**
	1000	4.98	13.02	18.00	5.18	11.77	16.95	**0.94**	**0.90**
	10000	5.08	21.54	26.62	5.15	23.27	28.42	**1.07**	**1.08**
2048	10	13.67	4.90	18.56	13.40	30.72	44.12	**2.38**	**6.28**
	100	13.92	8.24	22.16	13.27	35.87	49.14	**2.22**	**4.35**
	500	14.45	13.87	28.32	13.93	40.78	54.71	**1.93**	**2.94**
	1000	13.42	28.19	41.62	14.94	46.50	61.44	**1.48**	**1.65**
	10000	13.67	89.31	102.98	13.28	84.74	98.01	**0.95**	**0.95**
4096	10	45.29	16.18	61.47	43.53	136.62	180.15	**2.93**	**8.44**
	100	43.53	22.20	65.73	43.77	153.36	197.12	**3.00**	**6.91**
	500	45.51	35.92	81.43	43.54	177.32	220.85	**2.71**	**4.94**
	1000	41.69	68.41	110.10	53.50	199.25	252.75	**2.30**	**2.91**
	10000	53.90	361.26	415.16	43.49	372.48	415.97	**1.00**	**1.03**
	100000	45.53	573.48	619.01	44.16	607.98	652.13	**1.05**	**1.06**

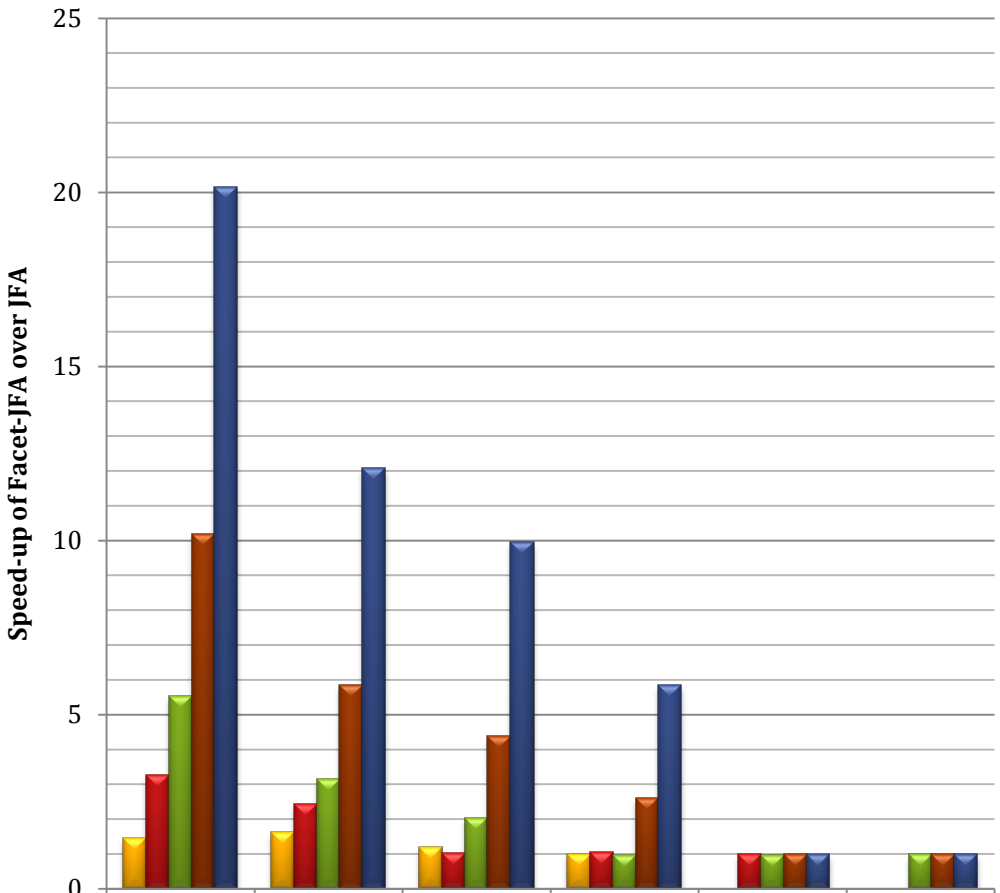
- Please see **Notes** section of Figure S-1 for explanation of *Mem* and *Exec* times.
- Speed-ups greater than 2 are highlighted in red.

Table S-4. Timing results for Facet-JFA and JFA for **3D grids on Kepler**.

Grid size (n)	No. of seed points	Facet-JFA time (ms)			JFA time (ms)			Total Speed up	Exec. Speed up
		Mem.	Exec.	Total	Mem.	Exec.	Total		
32	10	1.20	0.63	1.83	1.46	0.90	2.35	**1.29**	**1.44**
	100	1.19	0.71	1.90	1.49	1.14	2.64	**1.39**	**1.62**
	500	1.19	1.59	2.78	1.16	1.88	3.04	**1.09**	**1.18**
	1000	1.22	2.18	3.39	1.30	2.18	3.48	**1.03**	**1.00**
64	10	2.13	2.27	4.40	1.91	7.38	9.28	**2.11**	**3.26**
	100	1.91	2.86	4.76	1.93	6.96	8.89	**1.87**	**2.44**
	500	1.92	10.18	12.10	1.94	10.42	12.36	**1.02**	**1.02**
	1000	1.88	11.66	13.54	2.15	12.08	14.23	**1.05**	**1.04**
	100000	2.01	18.95	20.96	1.98	18.80	20.77	**0.99**	**0.99**
128	10	6.33	9.45	15.78	5.76	52.20	57.96	**3.67**	**5.52**
	100	6.01	16.73	22.74	6.01	52.30	58.31	**2.56**	**3.13**
	500	5.88	37.08	42.96	5.77	75.28	81.05	**1.89**	**2.03**
	1000	5.98	90.34	96.32	5.71	86.63	92.34	**0.96**	**0.96**
	10000	6.12	131.57	137.69	5.97	128.20	134.18	**0.97**	**0.97**
	100000	6.29	197.60	203.89	6.41	194.46	200.87	**0.99**	**0.98**
256	10	38.60	41.48	80.09	39.53	422.78	462.31	**5.77**	**10.19**
	100	38.89	73.31	112.20	39.30	427.80	467.09	**4.16**	**5.84**
	500	40.37	138.50	178.87	39.20	603.65	642.85	**3.59**	**4.36**
	1000	38.65	263.23	301.89	38.68	681.13	719.81	**2.38**	**2.59**
	10000	37.59	1007.48	1045.07	39.72	987.67	1027.39	**0.98**	**0.98**
	100000	35.13	1394.19	1429.32	38.18	1372.33	1410.51	**0.99**	**0.98**
512	10	278.03	179.64	457.67	277.91	3616.64	3894.55	**8.51**	**20.13**
	100	283.79	305.84	589.63	250.35	3694.71	3945.06	**6.69**	**12.08**
	500	283.55	514.93	798.47	263.49	5114.14	5377.63	**6.73**	**9.93**
	1000	248.13	974.90	1223.02	259.58	5700.99	5960.56	**4.87**	**5.85**
	10000	245.70	8137.42	8383.12	260.08	7952.83	8212.91	**0.98**	**0.98**
	100000	255.63	10812.28	11067.91	254.95	10669.75	10924.69	**0.99**	**0.99**

***** Please see **Notes** section of Figure S-1 for explanation of *Mem* and *Exec* times

Figure S-3. Speed-up of Facet-JFA over JFA for **3D grids** on Kepler.

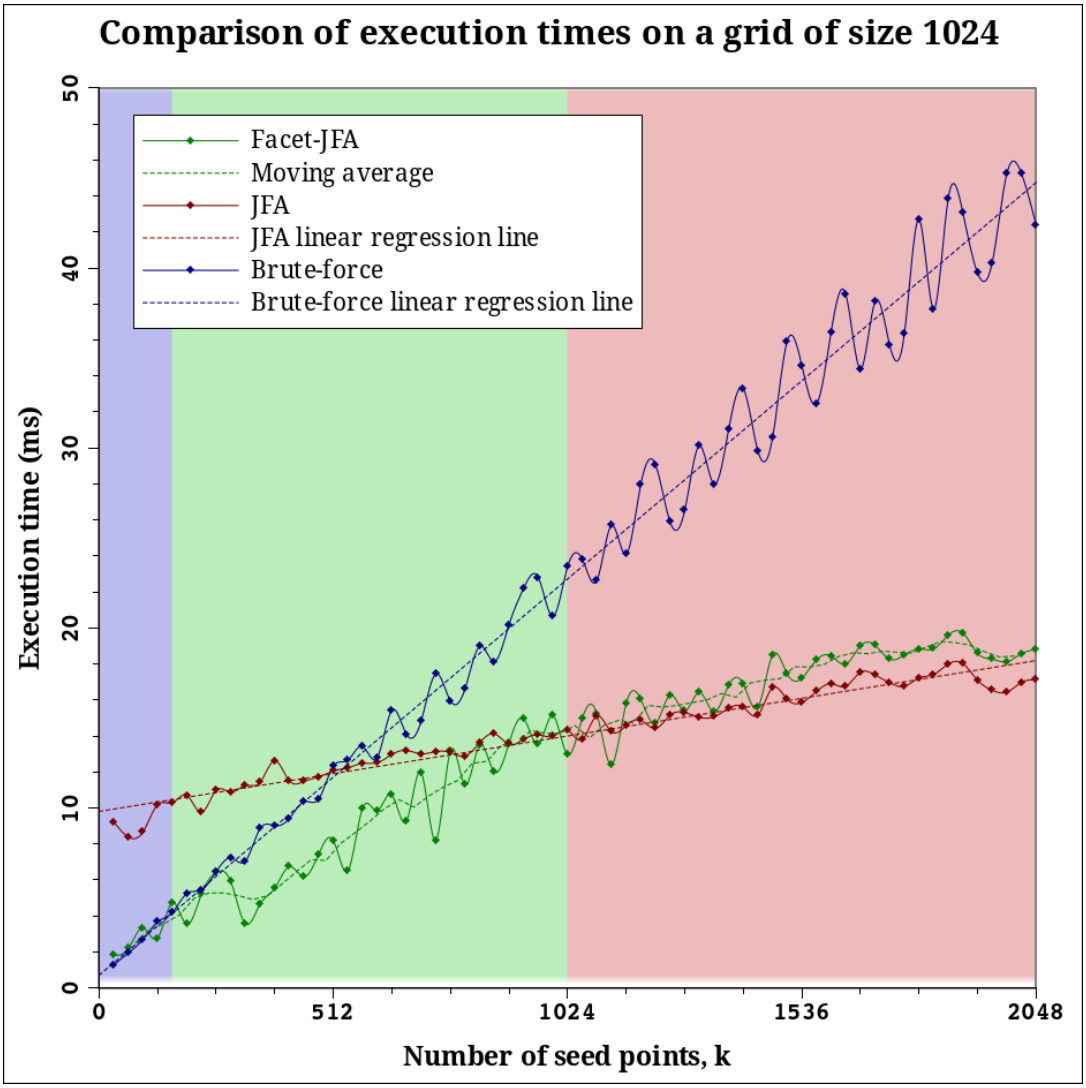


	10	100	500	1000	10000	100000
32	1.44	1.62	1.18	1.00		
64	3.26	2.44	1.02	1.04	0.99	
128	5.52	3.13	2.03	0.96	0.97	0.98
256	10.19	5.84	4.36	2.59	0.98	0.98
512	20.13	12.08	9.93	5.85	0.98	0.99

Comparison of three parallel algorithms for discrete Voronoi diagram computation

Here we compare, in detail, the Facet-JFA runtime performance against JFA and the parallel brute-force algorithm. In the brute-force algorithm, each pixel iterates over the seed set and finds the seed closest to itself. All pixels are processed in parallel. The running time of this algorithm is linear in k (the seed set size).

In this experiment, we fixed the grid size n to be 1024, while k was varied from 32 to 2048 with a step size of 32. The execution times for the three algorithms were averaged over 100 runs for each value of k. The seeds were placed randomly in the grid for each run, and all the algorithms processed the same input data.



From the plot above, it can be observed that for very small values of k (<150), the brute-force algorithm performs better (blue region). For higher values of k (>1024), Facet-JFA performance is similar to or slightly worse than that of JFA (red region). The window in which Facet-JFA out-performs the other two algorithms ($150<k<1024$) is colored green. Designing an automatic decider to select an algorithm based on the values of k and n is an interesting challenge that can be explored in future.

Pre-processing in Facet-JFA: Computation of *m*

Facet-JFA begins by invoking JFA on an initial grid of size *m*. In the paper, we do not explore automatic methods to compute *m.* Instead, we assume that *m* is provided by the user. Here we briefly discuss a few possible approaches to compute *m*.

Problem: Given seed set $S=\{s_1, s_2, \dots, s_k\}$ of size *k* and maximum grid size *n*. Compute the smallest grid size $m=2^p$, $p \in \mathbf{N}$, which ensures projection of the *k* seeds without conflict. i.e. no two seeds should share the same pixel in the *m x m* grid after projection.

Solutions:

1. **Closest pair method:** The grid size that does not cause any conflicts may be computed by estimating the distance between the closest pair of seeds. The value of *m* can be computed in constant time given the closest pair distance.
2. **Domain specific:** The closest pair information can be domain specific. For example, in case of molecules, it is known that two atoms can not be physically closer than a threshold distance. That threshold can be directly used to compute *m* in constant time. The domain expert can also provide an error tolerance threshold in terms of distance below which two seeds can be considered to be the same. This information can again be used to compute *m* in constant time.
3. **Brute force with grid:** We have to allocate an *n x n* grid for computation of Voronoi diagram using Facet-JFA. We can exploit this available space to determine *m*. For each seed point, set the pixel to which it is projected as 1. If after the projection it is found that the pixel is already set, then we have to consider higher values of *m*. Also, we can use binary search to identify a valid value of *m.* This approach can therefore determine *m* and simultaneously place the seeds in ***O(k log log n)*** time.

The performance of above methods can be improved further by exploiting parallelism. For example, a brute force comparison of all pairs of seed points can be performed in parallel.