An experimental assessment of three point-insertion sequences for 3-D incremental Delaunay tessellations

Sanderson L. Gonzaga de Oliveira[1], Diogo T. Robaina[2], Diego N. Brandão[3], Mauricio Kischinhevsky[2], and Gabriel Oliveira[1]

[1] Universidade Federal de Lavras, Lavras, Minas Gerais, Brasil
sanderson@dcc.ufla.br,g.oliveira@computacao.ufla.br
[2] Universidade Federal Fluminense, Niterói, Rio de Janeiro, Brazil
drobaina,kisch@ic.uff.br
[3] CEFET-RJ, Nova Iguaçu, Rio de Janeiro, Brazil
diego.brandao@eic.cefet-rj.br

Abstract

Currently, state-of-the-art algorithms for building 3-D Delaunay tessellations are incremental. Thus, their execution costs depend on the order of point insertion. This work evaluates three point-insertion sequences in incremental algorithms for building 3-D Delaunay tessellations. An incremental algorithm with point-insertion sequence provided by the cut-longest-edge kd–tree is evaluated against the BRIO–Hilbert order in conjunction with spatial middle and median policies employed in the 4.11 version of the Computational Geometry Algorithms Library. The results of computational costs (time and space) of these three algorithms are evaluated experimentally. Extensive results show that the incremental algorithm with a point-insertion sequence provided by the BRIO–Hilbert order with spatial middle policy employed in the latest version of the Computational Geometry Algorithms Library shows lower execution and storage costs than the two other algorithms evaluated.

1 Introduction

Delaunay tessellations have been employed in various scientific and engineering applications, including FEM analysis, computer graphics, medical applications, the modeling of deformable objects, and terrain modeling [8]. In present day, incremental algorithms are considered as state-of-the-art methods to build Delaunay tessellations in various point distributions [7].

The efficiency of an incremental algorithm for generating Delaunay tessellations is profoundly influenced by the point-insertion sequence, as both the numbers of orientation operations and conflicting polytopes depend on the insertion order (e.g. see [9] and references therein). In addition, paging policies and modern hierarchical memory architecture benefit programs that consider locality of reference. In particular, cache coherence is achieved when a sequence of recent memory references is grouped locally rather than randomly in the memory address space. Therefore, cache coherence should be considered highly significant in the design of algorithms. Thus, an efficient incremental algorithm for Delaunay tessellations uses properly the cache hierarchy to obtain high cache hit rates.

In an important paper in this field, Amenta *et al.* [1] evaluated the sequence in which the points are added to the mesh with the Biased Randomized Insertion Order (BRIO) technique. In this approach, an adequate spatial location of points is assumed to produce a large amount of cache hits.

Liu and Snoeyink [10] presented an incremental algorithm for building Delaunay tessellations in which points are added to the mesh in the sequence provided by the Hilbert curve.

Liu and Snoeyink [10] and Schrijvers *et al.* [11] provide a complete description about the influence of selecting the sequential order and the proper quantity of randomness. Zhou and Jones[14], Buchin [5, 6], and Boissonnat *et al.* [4] also evaluated methods that integrate randomness with deterministic orders. Thus, currently, the incremental algorithm for Delaunay tessellations implemented in the latest version of the Computational Geometry Algorithms Library (CGAL) [12] employs the Hilbert space-filling curve order combined with the BRIO scheme [1]. Specifically, the incremental algorithm that uses the BRIO–Hilbert strategy with *spatial middle* policy employed in CGAL [12] splits each partition exactly at its center [12] (https://doc.cgal.org/latest/Spatial_sorting/index.html). Instead of subdividing each partition in a rigid way at its center, the incremental algorithm that uses the BRIO–Hilbert strategy with *spatial median* policy employed in CGAL [12] subdivides each partition considering the median point alternately in each coordinate. To be more specific, these incremental algorithms implemented in CGAL [12] organize the point set in random buckets of increasing sizes, and the Hilbert order is used only inside a bucket [12]. Thus, these geometric algorithms available in CGAL [12] combine randomness and cache coherence [1]. A number of works [1, 14, 5, 6, 4] have demonstrated that this approach yields sufficient randomness to incorporate the gains of both random and locality provided by a space-filling curve order when generating Delaunay tessellations.

Liu *et al.* [9] presented an incremental method for generating 3-D Delaunay tessellations in which points are added to the mesh conforming to a level-order traversal of the cut-longest-edge kd–tree. Liu *et al.* [9] exhibited extensive experiments in which this incremental algorithm with point-insertion sequence provided by the cut-longest-edge kd–tree order outpaced the preceding possible state-of-the-art method (an incremental algorithm with point-insertion sequence provided by the Hilbert curve [10]) in various 3-D point distributions. Recently [7], this incremental algorithm with point-insertion sequence provided by the cut-longest-edge kd–tree surpassed incremental algorithms with several point-insertion sequences. In this publication [7], the experiments focused on implementation characteristics of incremental algorithms employing deterministic orders (i.e. without the use of randomness) to build Delaunay tessellations in seven 3-D point distributions (i.e. the same 3-D point distributions used by Liu *et al.* [9]).

Liu *et al.* [9] compared their algorithm with the incremental algorithm implemented in the 4.0 version of CGAL (in 2013), which did not use the middle and median policies. In particular, the median policy employed in the incremental algorithm for Delaunay tessellations implemented in the latest version of CGAL [12] is similar to the idea of the kd–tree order introduced by Liu *et al.* [9] in their algorithm. A difference in these schemes is that Liu *et al.* [9] used a cut-longest-edge strategy instead of splitting the partition alternately in each coordinate, which is the original approach of the kd–tree order [3].

The purpose of this present paper is to conduct a comparison of three state-of-the-art incremental algorithms for generating 3-D Delaunay tessellations. Specifically, this work evaluates the algorithm with point-insertion sequence provided by the cut-longest-edge kd–tree against the BRIO–Hilbert order (i.e. with the use of randomness) using spatial middle and median policies in inexact predicates employed in the 4.11 version of CGAL, which was released in September of 2017 [12].

To evaluate the three incremental algorithms for 3-D Delaunay tessellations, this present computational experiment uses eight 3-D point distributions, with sets ranging from 1 to 40 million points. Specifically, the unit interval is used as domain in our experiments. In addition, four 3-D test models are used in the experiments.

The remainder of this paper is structured as follows. Section 2 presents and analyzes the results. Finally, Section 3 addresses the conclusions.

2 Results and analysis

The three incremental algorithms evaluated here were implemented in the C++ programming language. The g++ 4.6.3-1 compiler was used. The experiments were performed on an Intel® Xeon® E5620 CPU 2.40GHz (12MB cache, 24GB of main memory 1067MHz) (Intel; Santa Clara, CA, USA) workstation. The Ubuntu 16.04.3 64-bits operating system was used in this machine, with kernel 4.4.0-98-generic.

Table 1 and Figures 1–4 show the results of execution times in eight point distributions in the 3-D unit cube when using three point-insertion sequences in incremental algorithms for building Delaunay tessellations: random points, points on a cylinder, points around a disk, points around three planes, points along three axes, points around a paraboloid, points around a spiral, and points on a saddle. Three executions were carried out for each point set, ranging from 1 to 40 million points. Numbers in bold face in Table 1 are the best results.

Table 1: Execution times (in seconds) of incremental algorithms with point-insertion sequence provided by three orders [CGAL BRIO–Hilbert order with spatial middle (SMi) and median (SMe) policies, and cut-longest-edge kd–tree (KDt)] in eight 3-D point distributions ($N * 10^6$).

N	Axes			Cylinder			Disk			Paraboloid		
	SMi	KDt	SMe	SMi	KDt	SMe	SMi	KDt	SMe	SMi	KDt	SMe
1	10	10	12	10	10	10	10	10	10	10	10	10
10	**97**	101	119	**97**	100	98	100	100	101	**97**	99	99
20	**196**	201	233	**195**	200	198	199	199	202	**194**	199	198
30	**292**	301	354	**291**	301	297	**299**	300	304	**291**	297	297
35	**339**	350	412	**341**	352	347	353	353	357	**343**	349	349
40	**387**	399	471	**390**	402	397	397	397	404	**392**	402	401
N	Planes			Random points			Saddle			Spiral		
	SMi	KDt	SMe	SMi	KDt	SMe	SMi	KDt	SMe	SMi	KDt	SMe
1	**9**	10	10	10	10	10	10	**9**	10	9	9	10
10	**96**	99	102	**97**	99	99	**97**	98	99	**96**	98	98
20	**194**	199	207	**195**	198	197	**193**	197	197	**194**	199	197
30	**290**	295	311	**293**	297	299	**290**	294	298	305	271	**269**
35	344	344	367	**341**	347	346	**338**	342	348	361	315	**314**
40	**388**	395	425	**388**	397	395	**386**	392	396	**390**	393	395

Although the BRIO–Hilbert strategy together with the middle policy has obtained higher execution costs than the other two incremental algorithms evaluated here when applied to instances composed of 30 and 35 million points around a spiral, the BRIO–Hilbert order with middle policy obtained lower execution costs when applied to instances comprised of 40 million points in this 3-D point distribution. Thus, the trends remained consistent over the eight 3-D point distributions used. Although Table 1 shows that the execution times of the algorithm with point-insertion sequence provided by the BRIO–Hilbert order along with the spatial middle policy employed in the latest version of CGAL [12] are lower than the two other algorithms evaluated in the eight 3-D point distributions used, Figures 1–4 indicate that in most of the cases the differences between the algorithms are rather small.

Figures 5–8 show that the memory requirements of the three point-insertion sequences in incremental algorithms for building 3-D Delaunay tessellations analyzed in this computational

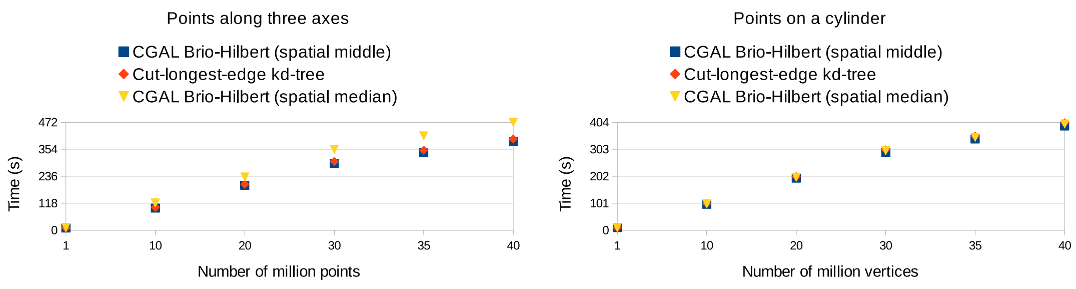

Figure 1: Execution times (in seconds) of incremental algorithms with three point-insertion sequences evaluated in two point distributions (points along three axes and points on a cylinder) on the 3-D unit cube.

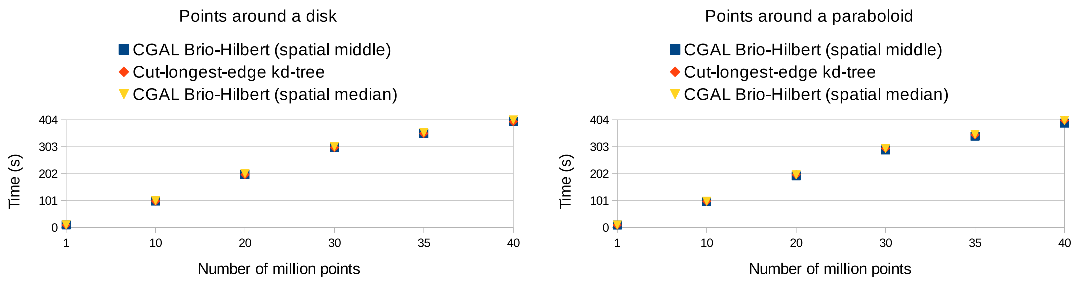

Figure 2: Execution times (in seconds) of incremental algorithms with three point-insertion sequences evaluated in two point distributions (points around a disk and points around a paraboloid) on the 3-D unit cube.

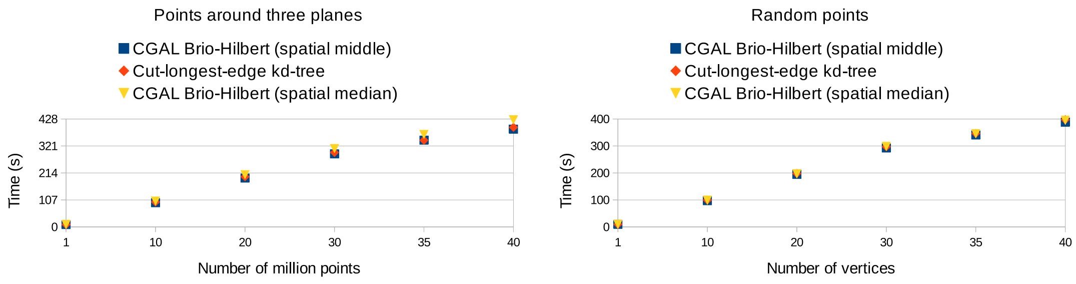

Figure 3: Execution times (in seconds) of incremental algorithms with three point-insertion sequences evaluated in two point distributions (points around three planes and random points) on the 3-D unit cube.

experiment are very similar when applied to instances arising from eight point distributions in the 3-D unit cube. In particular, we used the sysconf function to record the memory consumption. This computational experiment shows that the execution times and memory usage of the algorithm with point-insertion sequence provided by the BRIO–Hilbert order along with

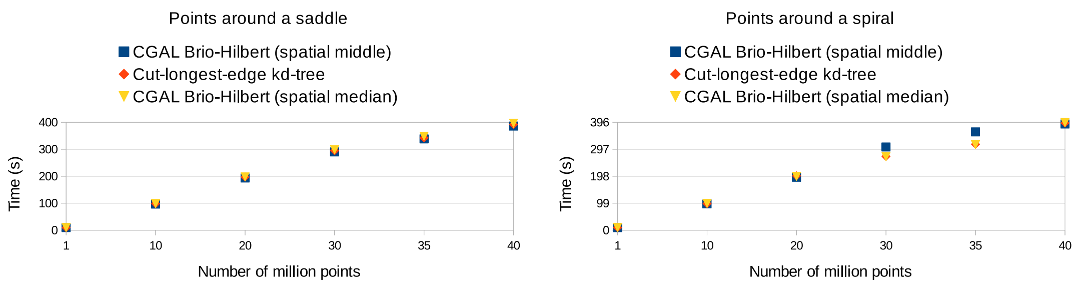

Figure 4: Execution times (in seconds) of incremental algorithms with three point-insertion sequences evaluated in two point distributions (points around a saddle and points around a spiral) on the 3-D unit cube.

the spatial middle policy used in the latest version of Computational Geometry Algorithms Library [12] are slightly lower than the two other algorithms evaluated in the eight 3-D point distributions used.

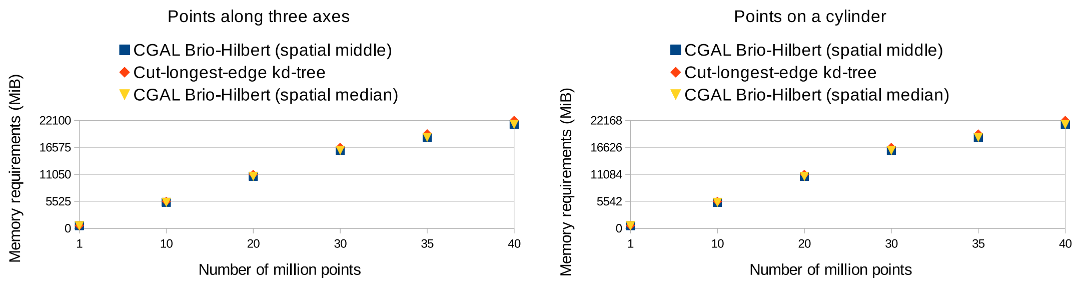

Figure 5: Memory requirements (MiB) of incremental algorithms with three point-insertion sequences evaluated in two point distributions (points along three axes and points on a cylinder) on the 3-D unit cube.

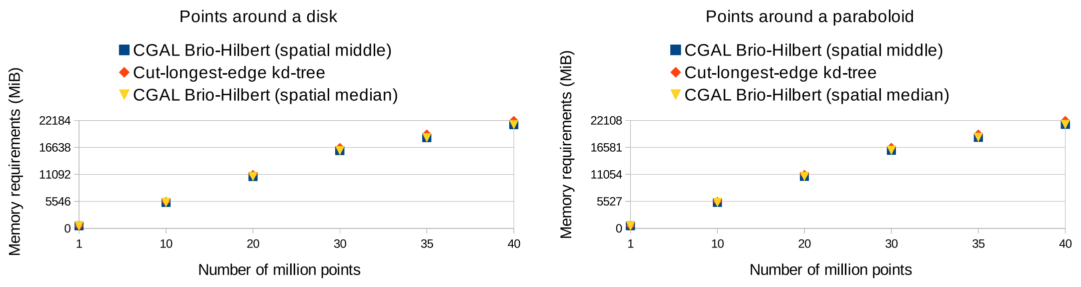

Figure 6: Memory requirements (MiB) of incremental algorithms with three point-insertion sequences evaluated in two point distributions (points around a disk and points around a paraboloid) on the 3-D unit cube.

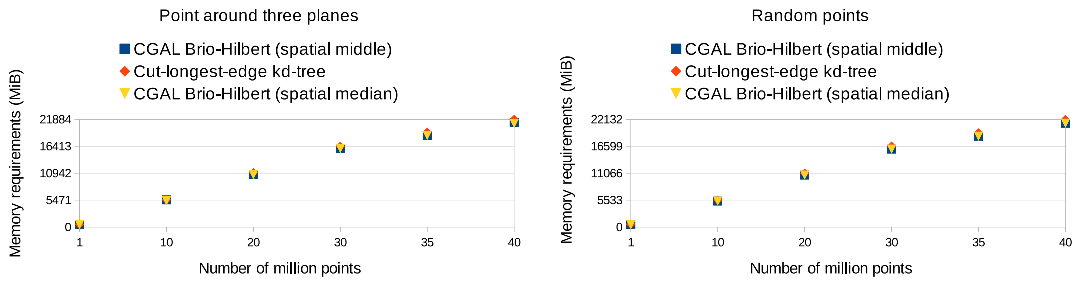

Figure 7: Memory requirements (MiB) of incremental algorithms with three point-insertion sequences evaluated in two point distributions (points around three planes and random points) on the 3-D unit cube.

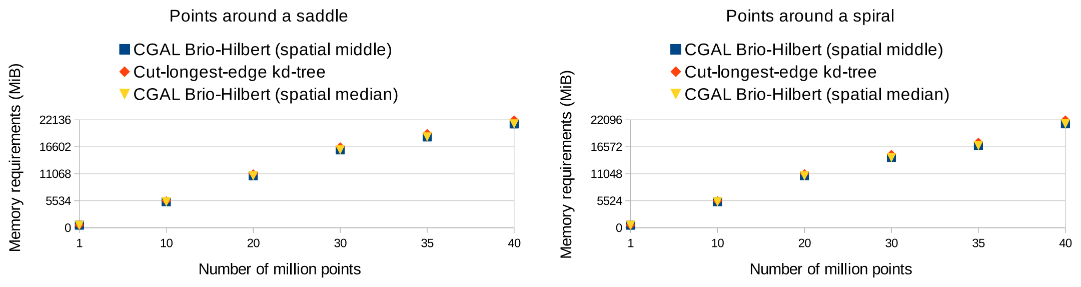

Figure 8: Memory requirements (MiB) of incremental algorithms with three point-insertion sequences evaluated in two point distributions (points around a saddle and points around a spiral) on the 3-D unit cube.

Exploratory investigations with both schemes employed in CGAL [12] using exact predicates showed that the spatial median policy dominated the spatial middle policy in seven 3-D point distributions. Specifically, the incremental algorithm with point-insertion sequence provided by the BRIO–Hilbert order with spatial middle policy achieved lower execution times than the spatial median policy only when applied to instances from points along three axes.

Four 3-D test models available on two different repositories [13, 2] were used in this computational experiment. Table 2 and Figure 9 show that the execution times of the incremental algorithm with point-insertion sequence provided by the cut-longest-edge kd–tree order were lower than the two other incremental algorithms evaluated in this computational experiment when applied to three (Vellum manuscript, Asian Dragon [13], and Napoleon [2]) 3-D test models used here. On the other hand, the incremental algorithm with point-insertion sequence provided by the BRIO–Hilbert order alongside spatial middle policy obtained lower execution times than the two other incremental algorithms evaluated in our experiments when applied to the Thai Statue 3-D test model [13] (see Figure 10). In addition, Figure 11 shows that the memory requirements of the incremental algorithm with point-insertion sequence provided by the cut-longest-edge kd–tree were slightly larger than the two other incremental algorithms evaluated here when applied to these four standard 3-D test models.

Table 2: Execution times (in seconds) of incremental algorithms with point-insertion sequence provided by three orders [CGAL BRIO–Hilbert order with spatial middle (SMi) and median (SMe) policies, and cut-longest-edge kd–tree (KDt)] applied to four 3-D test models (Vellum manuscript, Asian Dragon, Thai Statue [13], and Napoleon [2]).

3-D test model	No. of points	KDt	SMi	SMe
Vellum manuscript	2155617	19.1	19.8	20.4
Napoleon	3396797	19.9	23.4	24.0
Asian Dragon	3609600	32.8	34.6	34.3
Thai Statue	4999996	38.2	37.5	39.5

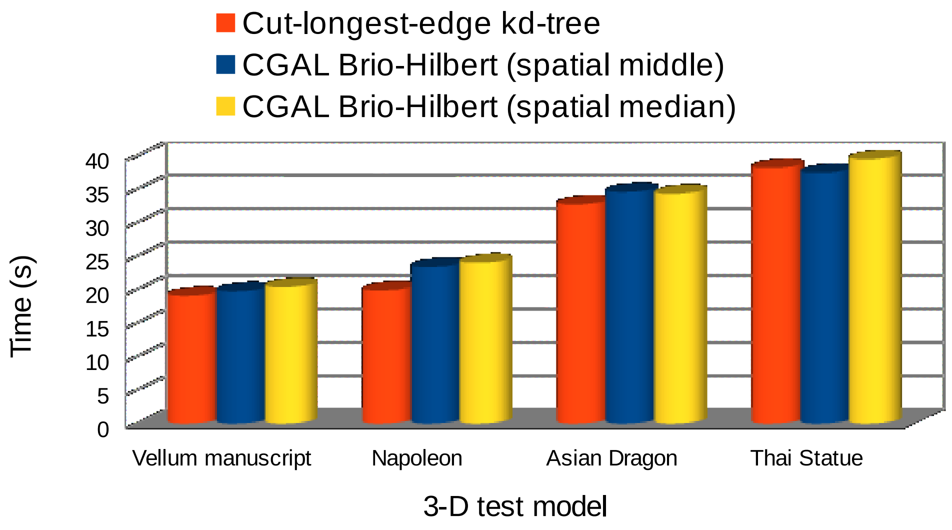

Figure 9: Execution times (in seconds) of incremental algorithms with three point-insertion sequences evaluated for four 3-D test models (see Figure 10).

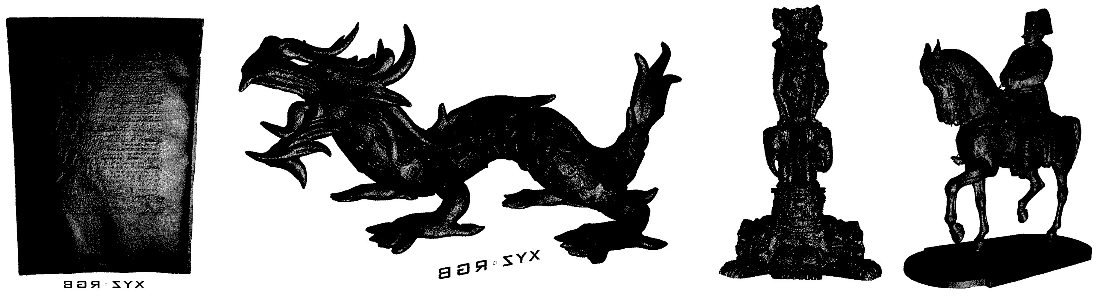

Figure 10: Four 3-D test models: Vellum manuscript, Asian Dragon, Thai Statue [13], and Napoleon [2]).

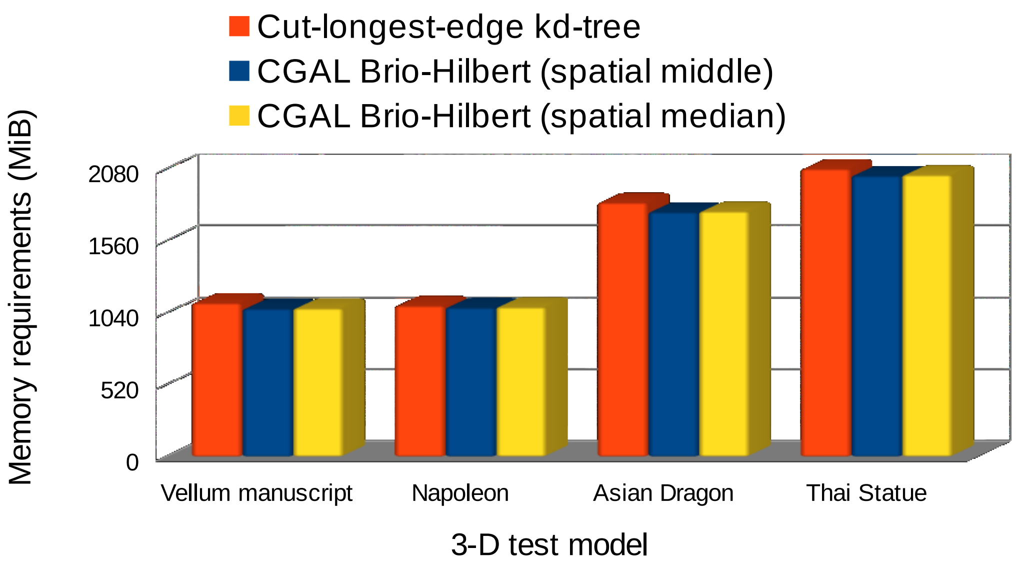

Figure 11: Memory requirements (MiB) of incremental algorithms with three point-insertion sequences evaluated for four 3-D test models (see Figure 10).

3 Conclusions

This work evaluated three point-insertion sequences in incremental algorithms for 3-D Delaunay tessellations. Experiments were performed in instances ranging from 1 to 40 million points.

The median policy implemented in the latest version of CGAL [12] is similar to the sequence provided by the kd–tree order. Despite this fact, the incremental algorithm with point-insertion sequence provided by the cut-longest-edge kd–tree order obtained, in a larger number of runs, lower execution costs than the BRIO–Hilbert order along with the median policy implemented in this version of CGAL [12]. Moreover, the incremental algorithm with point-insertion sequence provided by the cut-longest-edge kd–tree obtained overall lower execution costs than the two other incremental algorithms included in our experiments when applied to small 3-D test models (see Table 2). In spite of this and also despite the fact that the CGAL default constructor applies the median policy, the incremental algorithm with point-insertion sequence provided by the BRIO–Hilbert order combined with spatial middle policy employed in the latest version of CGAL [12] obtained slightly lower execution times and slightly smaller memory requirements than the two other algorithms evaluated in the eight 3-D point distributions and in the largest 3-D test model used. These results are consistent with the findings presented in the literature. Therefore, the incremental algorithm with point-insertion sequence provided by the BRIO–Hilbert order combined with spatial middle policy in inexact predicates employed in the latest version of CGAL [12] can be considered as the current state-of-the-art method for the building of Delaunay tessellations in the eight 3-D point distributions that were included in our experiments.

We plan to evaluate an incremental algorithm with point-insertion sequence provided by the cut-longest-edge kd–tree in tandem with the BRIO scheme against the other strategies analyzed in this appraisal. In addition, we intend to implement parallel versions of these methods.

Acknowledgements

We are grateful to Prof. Dr. Jianfei Liu, from the Department of Mechanics and Engineering Science, College of Engineering, Peking University, for sharing his program code and for his helpful comments.

References

[1] N. Amenta, S. Choi, and G. Rote. Incremental constructions con BRIO. In *Proceedings of the nineteenth annual symposium on Computational geometry*, SCG'03, pages 211–219, San Diego, CA, June 2003. ACM.

[2] Artec3D. Artec3D. https://www.artec3d.com/3d-models, 2018.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975.

[4] J. D. Boissonnat, O. Devillers, and H. Samuel. Incremental construction of the Delaunay graph in medium dimension. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, SCG'09, pages 208–216, Aarhus, Denmark, June 2009. ACM.

[5] K. Buchin. Constructing Delaunay triangulations along space-filling curves. In *Proceedings of the 2nd International Symposium Voronoi Diagrams (ISVD) in Science and Engineering*, pages 184–195, Seoul, Korea, 2005.

[6] K. Buchin. *Organizing point sets: Space-filling curves, Delaunay tessellations of random point sets, and flow complexes.* PhD thesis, Free University, Berlin, 2007.

[7] S. L. Gonzaga de Oliveira and J. R. Nogueira. An evaluation of point-insertion sequences for incremental Delaunay tessellations. *Computational & Applied Mathematics*, 37(1):641–674, 2018.

[8] S. L. Gonzaga de Oliveira, J. R. Nogueira, and J. M. R. S. Tavares. A systematic review of algorithms with linear-time behaviour to generate Delaunay and Voronoi tessellations. *CMES - Computer Modeling in Engineering and Sciences*, 100(1):31–57, 2014.

[9] J.-F. Liu, J.-H. Yan, and S. H. Lo. A new insertion sequence for incremental Delaunay triangulation. *Acta Mechanica Sinica*, 29(1):99–109, 2013.

[10] Y. Liu and J. Snoeyink. A comparison of five implementations of 3D Delaunay Tessellation. In J.E. Goodman, J. Pach, and E. Welzl, editors, *Combinatorial and Computational Geometry*, volume 52, pages 439–458, Cambridge, MA, 2005. MSRI Publications.

[11] O. Schrijvers, F. van Bommel, and K. Buchin. Delaunay triangulations on the Word RAM: Towards a practical worst-case optimal algorithm. In *Proceedings of the 10th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*, pages 7–15, Saint Petersburg, Russia, 2013.

[12] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.11 edition, 2017. http://doc.cgal.org/4.11/Manual/packages.html.

[13] The Stanford Models. The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3Dscanrep, 2014.

[14] S. Zhou and C. B. Jones. HCPO: an efficient insertion order for incremental Delaunay triangulation. *Information Processing Letters*, 93:37–42, 2005.