Copyright © 2014 Tech Science Press CMES, vol.x, no.y, pp.1-27, 2014

A systematic review of algorithms with linear-time behaviour to generate Delaunay and Voronoi tessellations

Sanderson L. Gonzaga de Oliveira[1], Jéssica Renata Nogueira[1], and João Manuel R. S. Tavares[2]

Abstract: Triangulations and tetrahedrizations are important geometrical discretization procedures applied to several areas, such as the reconstruction of surfaces and data visualization. Delaunay and Voronoi tessellations are discretization structures of domains with desirable geometrical properties. In this work, a systematic review of algorithms with linear-time behaviour to generate 2D/3D Delaunay and/or Voronoi tessellations is presented.

Keywords: Mesh generation; computer-aided design, engineering, and manufacturing; computational geometry and topology.

1 Introduction

Meshes are used in a huge number of applications, and especially in finite element discretization, which is a central tool in scientific computing. Triangles are the simplest polygon in the Euclidean plane. In simple terms, triangles are closed polygons in planar geometry with the smallest number of sides. In the mesh generation context, there is an extensive use of triangular meshes, as shown by Gonzaga de Oliveira, Kischinhevsky, and Tavares (2013). In this field of study, the most common form of triangle meshes are Delaunay triangulations. The popularity of these meshes is mainly because they can be built quickly and have very attractive geometric characteristics; for example, Voronoi diagrams (a dual mesh of the Delaunay triangulation) may capture proximity. Moreover, Delaunay triangulations are used to represent parts of a continuous space in a way that allows numerical algorithms to compute characteristics of that space [Edelsbrunner (2001)]. Delaunay and Voronoi tessellations have been used in various applications in the fields

[1] Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras, MG, Brazil, sanderson@dcc.ufla.br, jes.r.nogueira@gmail.com

[2] Instituto de Engenharia Mecânica e Gestão Industrial, Departamento de Engenharia Mecânica, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal, tavares@fe.up.pt

of science and engineering, such as: *i*) computer graphics [e.g. Alliez, Meyer, and Desbrun (2002)]; *ii*) industrial design [e.g. Nordin, Hopf, Motte, Bjärnemo, and Eckhardt (2011)]; *iii*) medical applications [e.g. Puentes, Dhibi, Bressollette, Guias, and Solaiman (2009)]; *iv*) modelling of composite and porous materials [e.g. Dong and Atluri (2012)]; *v*) modelling of deformable objects [e.g. Busaryev, Dey, and Wang (2013)]; *vi*) molecular modelling [e.g. Lin, Wang, and Zeng (2014)]; *vii*) tessellation of solid shapes [e.g. Bishay and Atluri (2012)]; *viii*) terrain modeling [e.g. Tucker, Lancaster, Gasparini, Bras, and Rybarczyk (2001)]; *ix*) and video games [e.g. De Gyves, Toledo, and Rudomín (2013)]. Therefore, Delaunay and Voronoi tessellations have been extensively studied and different techniques have been used to build these structures.

For a set of 2D (or 3D) points, Delaunay tessellation is described as a triangulation (tetrahedrization in 3D) that the circumcircle (circumsphere) of each triangle (tetrahedron) does not have inner points. Delaunay triangulation (tetrahedrization in 3D) is unique in the case that there are not four (five) or more cocircular (cospherical) points in the point set. For example, this uniqueness of the structure does not occur when there are four points comprising a square. This square can be triangulated by inserting an edge in one of the two diagonals. Clearly, four points that form a square are cocircular.

An important step in numerical methods, such as in the finite element method or finite volume discretizations, is the generation of well-shaped meshes; a mesh is well-shaped if all of its polytopes have a small aspect ratio. Li (2000) noted that the smallest angle of a simplex, such as a triangle in 2D or a tetrahedron in 3D, is always bounded if this simplex has a bounded aspect ratio. This author also explained that the aspect ratio and the circumradius-to-shortest edge ratio of a triangle differ by only a constant factor. However, this is not true with a simplex in three or higher dimensions. A tetrahedron with a small circumradius-to-shortest edge ratio and a large aspect ratio is called a *sliver*. In other words, a sliver is a tetrahedron whose vertices are almost coplanar and whose circumradius is not much longer than its shortest edge length. Shewchuk (1998a) noted that a sliver can have a circumradius-to-shortest edge ratio as low as $\frac{1}{\sqrt{2}}$, yet it can be considered problematic in other measures due to its small volume and altitude, and its dihedral angles can also be small (close to 0°) or large (close to 180°). Slivers are the problematic polyhedra in 3D meshes. Thus, a simulation may be inaccurate, or it may not converge to the solution if the mesh used has slivers. Cavendish, Field, and Frey (1985) already perceived the ubiquity of slivers in 3D Delaunay triangulations. Talmor (1997) noted that Delaunay tessellations have slivers even from a well-spaced point set. Thus, one of the main difficulties of 3D mesh generation comes from the presence of slivers. Despite the slivers Delaunay methods are valuable for gener-

ating 3D meshes, and there are many methods available to improve 3D Delaunay meshes, such as: the final meshes of Liu, Li, and Chen (2008) and Liu, Chen, and Sun (2009) methods are good after a local transformation or reconnection in order to improve Delaunay tessellations.

A Voronoi diagram is a specific kind of spatial decomposition. Let $S \subseteq \mathbb{R}^d$ be a set of n *generating points* (designated also as *sites* or *generators*) p_i in the Euclidean space (in the simplest and most common case), for $1 \leq i \leq n$. The Voronoi polytope P_i of a generating point p_i is the set of all points in $\mathbb{R}^d$ that are at least as close to p_i as to any other generating point in S. Formally, for dimensions d equal to 2 or 3, each Voronoi polytope is a set of points $P_i = \{p \in \mathbb{R}^d : (\exists p_i \in S)(\forall p_j \in S)\ ||p - p_i|| \leq ||p - p_j||$, with $i \neq j \ \wedge \ 1 \leq i, j \leq n\}$.

If the diametric ball of every boundary simplex of the Delaunay tessellation is empty, then the tessellation is a *boundary conforming Delaunay mesh* of the simulation domain. Thus, all the vertices of dual Voronoi diagram lie inside the simulation domain. Hence, it is a mesh suitable to be used with finite volume discretization for solving many problems. In Figure 1, a Delaunay triangulation and the corresponding Voronoi diagram partition are shown.

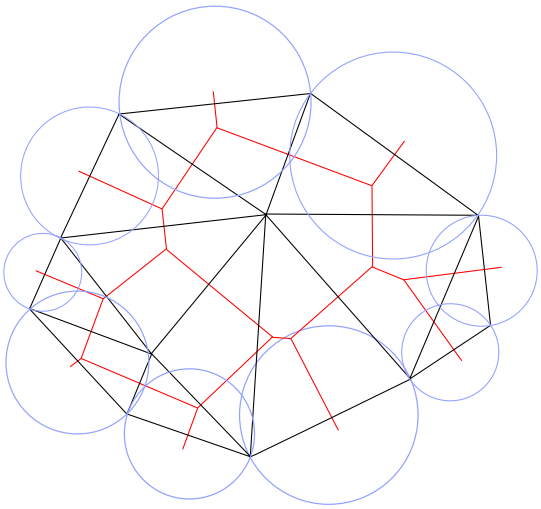

Figure 1: Delaunay triangulation in black, the corresponding Voronoi diagram partition in red, and diametric circles of boundary edges in blue.

Shamos and Hoey (1975) presented the first optimal divide-and-conquer algorithm for Voronoi diagrams. They showed that its $O(n \lg n)$[1] worst-case running time is

[1] Here, $O(\lg n)$ is used instead of $O(\log n)$, because $\log n$ implies base 10 and $\lg n$ means that the base

optimal with a real random access machine (RAM) computation model. Since the Delaunay triangulation of a point set is linearly reducible from the Voronoi diagram by duality, the building of the Delaunay triangulation could be carried out in $\Theta(n \lg n)$. The first direct worst-case optimal 2D Delaunay divide-and-conquer algorithm was published by Lee and Schachter (1980). Sibson and Green (1978) published a $O\left(n^{\frac{3}{2}}\right)$ average running time algorithm for 2D Delaunay triangulation.

The cost to build these structures is higher in cases with dimensions greater than two; however, with restrictions, the algorithms to generate Delaunay or Voronoi tessellations can be near linear computational time. These algorithms are reviewed in this work.

Section 2 gives details about the procedures used in this systematic review. The preliminary period from 1980 to 1988 is addressed in Section 3: the first algorithms developed for the two meshes under study are based on the incremental or the divide-and-conquer approach and were tested with up to 2^{16} points. A second period in the development of these algorithms was from 1989 to 2001. In 1989, an algorithm with linear-time behaviour was proposed, in which the points were inserted randomly. In the same year, an algorithm for 3D Voronoi tessellation was published. In 1992, the first robust algorithm for the generation of 2D Voronoi diagrams with linear-time behaviour and up to one million generating points was proposed. These and other algorithms are discussed in Section 4. From 2003 on, the proposed algorithms have been mainly based on techniques that use a biased randomized insertion order of the input points or insertion of the input points according to a specific order. There was a predominance of incremental algorithms for the generation of Delaunay tessellation in this period, and the authors tested their algorithms using sets with millions of points. These algorithms are reviewed in Section 5. Final remarks are presented in Section 6.

2 Systematic Review

This review, which began in November, 2013, concerns linear-time behaviour algorithms for Delaunay and/or Voronoi tessellations. We conducted this review using Scopus® and Google Scholar databases.

We searched Scopus® database using the terms: ((Topic = (“Delaunay”) AND Topic = (“linear”)) OR (Topic = (“Voronoi”) AND Topic = (“linear”))) refined by: Publication types = (ALL) AND Languages = (ENGLISH). These terms were searched in the title, abstract and keywords of the articles indexed in the database. This search resulted in 1048 articles.

does not matter: for constants a and c and sufficiently higher n values, one has $\log_c n = \log_c a^{\log_a n} = \log_c a \cdot \log_a n = O(\lg n)$ because $\log_c a$ is constant.

The titles and abstracts of the articles found were then read independently by two reviewers and as there were no disagreements in the selections made, a third reviewer was not needed. Besides the articles that met with the eligibility criteria, other articles were used in order to support some of the concepts involved in the algorithms identified. In addition, for the papers found that were presented at conferences, Google Scholar database was searched to find the possibly journal versions.

Then, to have a clear comparison of the studies selected, data were extracted according to the following headings: authors, year of publication, tessellation generated (Delaunay, Voronoi or both), experimental data (maximum number of points used in the tests as well as the type of point distribution), results and conclusions.

From among the 1048 articles retrieved, 34 algorithms were selected and are shown in Table 1. The algorithms were divided into three groups, according to the period in which they were published, and the designed techniques were divided into: divide-and-conquer algorithms, gift-wrapping algorithms, incremental algorithms, lifting-map algorithms, sweep-line algorithms, hybrid sweep-line and divide-and-conquer algorithms, and algorithms without any associated technique. In addition, the maximum number of points or segments tested in each algorithm is also shown.

Table 2 presents the number and corresponding percentage (%) of algorithms with linear behaviour to generate Delaunay tessellations, Voronoi diagrams or both, according to the technique used.

3 First period (1980-1988)

The first expected linear-time algorithm was proposed by Bentley, Weide, and Yao (1978, 1980), to generate 2D Voronoi diagrams. This divide-and-conquer-based algorithm runs in expected linear time and uses sets of input points uniformly distributed in a unit square. The basic idea is to search cells (see the division of the domain into squares as shown in Figure 2a) in a relatively small neighbourhood of each point in a spiral-like fashion until at least one point is found in each octant, as shown in Figure 2b. The *tentative* Voronoi polygon of the centre point is formed by considering just those points inside a circumsphere around the centre point. If the search takes more than $O(\lg n)$ cells, it switches to an optimal divide-and-conquer worst-case algorithm. The authors presented simulations with 10,000 points.

Maus (1984) proposed an incremental algorithm for Delaunay triangulation with expected linear computational time. This behaviour was reached with sets of points distributed somehow uniformly. In this approach, the initial domain is subdivided into buckets, and the radix sort is used to order the set of points according to their 2D coordinates. The author used the idea that numbers are represented in computers by K bits. Therefore, radix sort can order n points in $O(nK)$ time, in which

Period	Algorithm	Technique	Tessellation	N
1	Bentley, Weide, and Yao (1978)	divide and conquer	Voronoi	10000
	Maus (1984)	incremental	Delaunay	-
	Ohya, Iri, and Murota (1984)	incremental	Voronoi	32768
	Dwyer (1987)	divide and conquer	Delaunay	65536
	Katajainen and Koppinen (1988)	divide and conquer	Delaunay	32768
2	Aggarwal, Guibas, Saxe, and Shor (1989)	lifting map	Voronoi	-
	Dwyer (1989, 1991)	gift wrapping	Voronoi	-
	Chew (1990)	gift wrapping	Voronoi	-
	Sugihara and Iri (1992)	incremental	Voronoi	1000000
	Klein and Lingas (1992)	sweep line and divide and conquer	Voronoi	-
	Klein and Lingas (1993, 1996)	divide and conquer	Voronoi	-
	Tsai (1993)	incremental	Delaunay, Voronoi	50000
	Chin and Wang (1995, 1998)	-	CDT, CVD	-
	Djidjev and Lingas (1995)	lifting map	Voronoi	-
	Su and Drysdale (1997)	incremental	Delaunay	131072
	Su and Drysdale (1997)	sweep line	Delaunay	131072
	Shewchuk (1996)	incremental, sweep line or divide and conquer	Delaunay, CDT, Voronoi	1000000
	Held (1998)	incremental	Voronoi	8000 (s)
	Lemaire and Moreau (2000)	divide and conquer	Delaunay	10000000
	Held (2001)	incremental	Voronoi	524288 (s)
3	Amenta, Choi, and Rote (2003)	incremental	Delaunay	10000000
	Liu and Snoeyink (2005)	incremental	Delaunay	1024000
	Buchin (2005)	(two) incremental	Delaunay	-
	Boissonnat, Devillers, and Hornus (2009)	incremental	Delaunay	256000
	Buchin (2009)	incremental	Delaunay	-
	Buchin and Mulzer (2009, 2011)	incremental	Delaunay	-
	Yang and Choi (2010)	incremental	CDT	2000 (s)
	Ebeida, Mitchell, Davidson, Patney, Knupp, and Owens (2011)	-	CDT	8271560
	Yang, Choi, and Jung (2011)	divide and conquer	Delaunay	30000
	Löffler and Mulzer (2011, 2012)	incremental	Delaunay	-
	Schrijvers, van Bommel, and Buchin (2013)	incremental	Delaunay	4194304
	Liu, Yan, and Lo (2013)	incremental	Delaunay	5500000
	Lo (2013)	incremental	Delaunay	100000000

Table 1: The 34 linear-time algorithms found for either Delaunay, Constrained Delaunay (CDT), Voronoi or Constrained Voronoi (CVD) Tessellations, grouped into the period in which they were published; and N is the maximum number of points or segments tested.

K is a constant. It starts with a single Delaunay triangle and incrementally finds other valid Delaunay triangles. For sets of points distributed non-uniformly, this algorithm is quadratic.

Ohya, Iri, and Murota (1984) proposed an incremental algorithm to generate 2D Voronoi diagrams with average optimal linear computational time with points uni-

Technique	Period						*N*	*%*
	1		2			3		
	Tessellation		Tessellation			Tess.		
	VD	DT	VD	DT	B	DT		
Divide and conquer	1	2	1	1	1	1	7	20.6
Gift wrapping	0	0	2	0	0	0	2	5.9
Incremental	1	1	3	1	1	12	19	55.9
Lifting map	0	0	2	0	0	0	2	5.9
Both sweep line and divide and conquer	0	0	1	0	0	0	1	2.9
Sweep line	0	0	0	1	0	0	1	2.9
Without associated technique	0	0	0	0	1	1	2	5.9
Total	2	3	9	3	3	14	34	100

Table 2: Number (*N*) and corresponding percentage (*%*) of algorithms with linear behaviour to generate Delaunay Tessellations (DT), Voronoi Diagrams (VD) or both (B), according to the technique used.

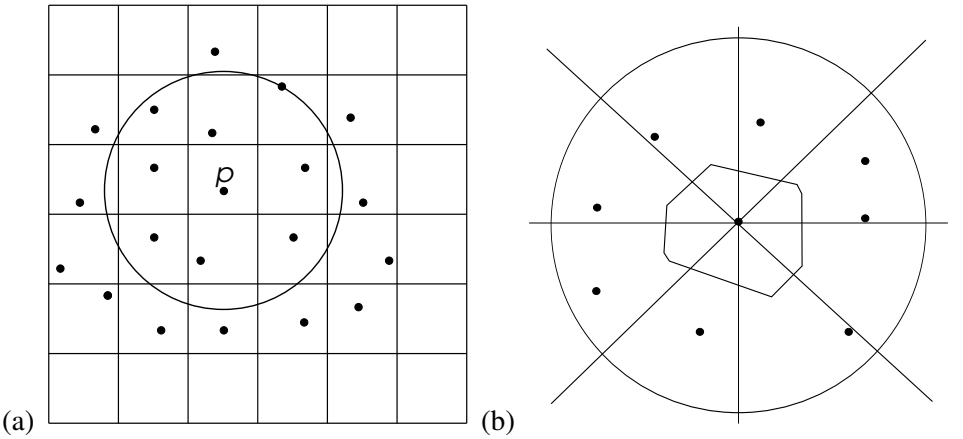

Figure 2: Construction of a Voronoi polygon by the Bentley, Weide, and Yao (1978, 1980) algorithm: (a) spiral search around point *p* using cells and (b) division of the plane into octants.

formly distributed, and quadratic in the worst case. It is possible that this algorithm was the first average linear algorithm with robust characteristics. Ohya, Iri, and Murota (1984) bucketed the points and processed the buckets to insert points taking into account a breadth-first traversal of a quadtree so that the next edge inserted is probably near the correct triangle. It generates Voronoi diagrams, even from highly non-uniform point distributions; however, it is unstable. The authors

showed simulations with up to 2^{15} points.

Dwyer (1987) proposed a divide-and-conquer algorithm for the generation of Delaunay triangulations. The domain is subdivided into $O(n/\lg n)$ square cells. The author built the Delaunay triangulation of the points within each cell using the Guibas and Stolfi (1985) algorithm. The triangulations within each row of cells are grouped in pairs until the triangulation of the row has been completed. Moreover, row triangulations are joined in pairs to complete the triangulation of the entire set of points. Dwyer (1987) algorithm is $O(n \lg n)$ in the worst case. However, according to Dwyer (1987), partitioning the mesh into squared polygons reduced it to $O(n \lg \lg n)$ in the average case for a large class of points distribution. This algorithm was tested on inputs with up to 2^{16} points. In relation to the computational time, this algorithm was competitive against linear-time algorithms.

Katajainen and Koppinen (1988) presented a modified Dwyer's divide-and-conquer algorithm to generate Delaunay triangulations with expected linear computational time for sets of points distributed almost uniformly, and with $O(n \lg n)$ time in the worst case. Inspired by the work of Ohya, Iri, and Murota (1984), Katajainen and Koppinen (1988) divided the 2D space into approximately n cells that were merged according to a quadtree-like order. Tests carried out by the authors showed that their algorithm performed similarly to the modified algorithm proposed by Dwyer (1987) in simulations with up to 2^{15} points.

The complexities and the conditions under which such complexities are reached by the 5 algorithms found for the first period are indicated in Table 3.

Algorithm	Complexity	Comment
Bentley, Weide, and Yao (1978, 1980)	expected linear time	set of points distributed uniformly in a unit square
	worst case: $O(n \lg n)$	analyzed by the authors
Maus (1984)	expected linear time	set of points distributed in a quasi-uniform manner
	worst-case: expected $O(n^2)$	in data distributions such as delta-shaped distributions, in which essentially all data points are centered around one point
Ohya, Iri, and Murota (1984)	average case: $O(n)$	n generating points distributed uniformly in the unit square
	worst case: $O(n^2)$	reported in Sibson and Green (1978)
Dwyer (1987)	average case: $O(n \lg \lg n)$	points drawn independently according to a large class of distributions
	worst case: $O(n \lg n)$	analyzed by the author
Katajainen and Koppinen (1988)	expected linear time	set of points distributed in a quasi-uniform manner
	worst case: $O(n \lg n)$	analyzed by the authors

Table 3: The 5 linear-time behaviour algorithms found for the first period, their complexities, and the conditions under which such complexities are reached.

4 Second period (1989-2001)

Aggarwal, Guibas, Saxe, and Shor (1989) proposed an algorithm for the generation of 2D Voronoi diagrams. Probably, this was the first algorithm with linear-time behaviour for sets of random points. In theory, this algorithm is $\Theta(n)$ in situations that the input set forms a convex polygon in a counter-clockwise order, and uses a lifting map for the construction of Voronoi diagrams.

Dwyer (1989, 1991) presented the first d-dimensional algorithm with expected linear computational time behaviour using gift-wrapping and bucketing techniques for the generation of Voronoi tessellations in dimensions higher or equal to two. This was probably the first algorithm with linear-time behaviour to generate 3D Delaunay or Voronoi tessellations. This algorithm has linear computational time for sets of almost uniformly distributed points in a unit d-ball by using a dictionary to represent a linear array of buckets and stores the facets in which only one adjacent polygon is known.

Chew (1990) proposed a gift-wrapping expected linear-time algorithm to generate 2D Voronoi diagrams. In this algorithm, a clockwise convex polygon is built from the input points and the gift-wrapping technique is applied. According to Chew (1990), this algorithm is easier to implement and probably faster than the Aggarwal, Guibas, Saxe, and Shor (1989) algorithm.

Sugihara and Iri (1992) presented an incremental algorithm for the generation of 2D Voronoi diagrams. This algorithm is numerically stable and its average complexity is linear in terms of the number of generators. In this topology oriented approach, there are guaranties of building Voronoi diagrams with up to one million generating points. In Figure 3, one can see an example of an initial Voronoi partition and the resulting Voronoi partition after inserting a generating point p according to this algorithm.

Klein and Lingas (1992) showed that a constrained Voronoi diagram of a simple polygon can be built with linear computational time by using Manhattan metric. Their algorithm uses sweep-line and divide-and-conquer techniques to generate 2D Voronoi diagrams. This approach was extended to Euclidean measure by Klein and Lingas (1993, 1996). These authors also presented an expected linear-time algorithm applying a divide-and-conquer approach, to compute constrained 2D Voronoi diagrams.

Tsai (1993) proposed a convex-hull incremental algorithm for Delaunay triangulations and 2D Voronoi diagrams. Using simulations with up to 50,000 points, Tsai showed that his algorithm is approximately linear with randomly spaced points, and quadratic in the worst case.

Chin and Wang (1995, 1998) presented an algorithm for the computation of con-

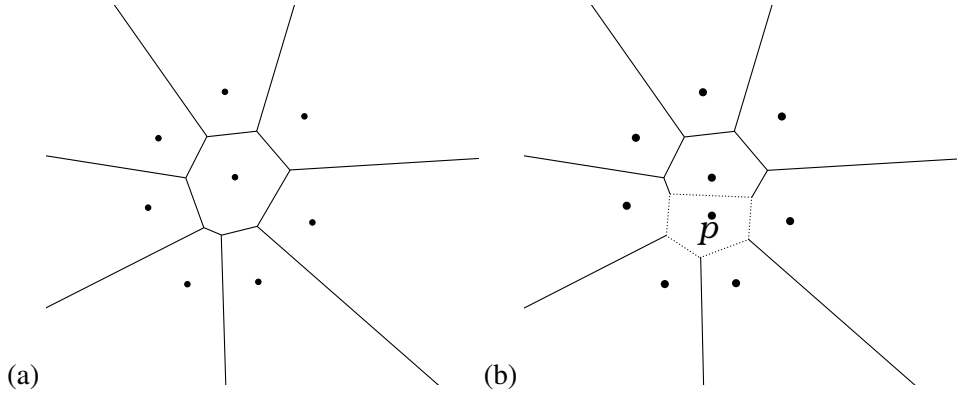

Figure 3: Incremental-type Sugihara and Iri (1992) method for building Voronoi diagrams: (a) initial Voronoi partition and (b) resulting Voronoi partition after inserting a generating point p.

strained Delaunay triangulations or Voronoi diagrams in a simple polygon. The authors showed that their algorithm presents linear time by using the potential method. It should be noted that such analysis guarantees the average performance of each operation in the worst case. Given a set of points sorted by their 2D coordinates, Djidjev and Lingas (1995) proved that Voronoi diagrams can also be built with linear computational time using lifting map, based on the work by Aggarwal, Guibas, Saxe, and Shor (1989).

Su and Drysdale (1995) carried out comparisons with nine algorithms to build Delaunay or Voronoi structures: *i*) Dwyer (1987) divide-and-conquer algorithm; *ii*) Fortune (1987) sweep-line algorithm; *iii*) Ohya, Iri, and Murota (1984) incremental algorithm; *iv*) the incremental and *v*) the gift-wrapping algorithms, proposed by Su and Drysdale (1995) for Delaunay triangulations. According to Su and Drysdale (1995), their incremental algorithm showed a linear-time behaviour in uniform point distribution in a unit square. In addition, Su and Drysdale (1997) carried out comparisons among these five algorithms and also with four other algorithms: *vi*) Guibas and Stolfi (1985) incremental algorithm; *vii*) Devillers' implementation of Delaunay Tree code; *viii*) Barber, Dobkin, and Huhdanpa (1996) convex-hull algorithm; *ix*) and the version of Delaunay Tree code implemented in the Library of Efficient Data types and Algorithms (LEDA). In Su and Drysdale (1997), the algorithm of Dwyer (1987) was modified to use Fortune's stable in-circle test. In addition, Su and Drysdale (1997) modified the data structure used in the algorithm of Ohya, Iri, and Murota (1984) and in the incremental algorithm proposed in Su and Drysdale (1995). The Su and Drysdale (1995, 1997) comparisons were based

on sets of up to 131,072 points according to several distributions. Based on the numerical tests performed, the best three algorithms were Dwyer's, Su and Drysdale (1997) incremental algorithm, and Fortune's algorithm using a heap. Moreover, Su and Drysdale (1997) reported that Dwyer (1987) divide-and-conquer algorithm was the fastest overall and was the most resistant to bad data point distribution with $O(n \lg n)$ computational time in the worst case. The runtime of Su and Drysdale (1997) incremental algorithm increased more quickly than the runtimes of both modified Fortune's and Dwyer's algorithms.

Shewchuk (1996, 2002) proposed an algorithm named *Triangle* for building 2D Delaunay triangulations, constrained Delaunay triangulations and Voronoi diagrams. The Shewchuk (1996) implemented versions of this algorithm were based on: the incremental insertion algorithm of Lawson (1977), sweep-line algorithm of Fortune (1987), and two divide-and-conquer algorithms of Dwyer (1987) with alternating or vertical cuts. According to Shewchuk (1996), the divide-and-conquer approach with alternating cuts was the fastest algorithm in almost all tests. Although *Triangle* may be slow in the case of triangulating uniformly distributed point sets, it exhibited fast running times on more complex inputs. Shewchuk (1996) performed tests using sets of up to one million points. Shewchuk (1996) used Delaunay refinement of Ruppert (1995) to generate Delaunay triangulations with guarantees of mesh quality. According to Shewchuk (1996), Delaunay refinement commonly is $O(n)$ in practice; but for using a heap, *Triangle* is $O(n \lg n)$, regardless of the distribution of points. This algorithm became popular mainly because the author made the code available and gave details of the implementation. Moreover, according to Ebeida, Mitchell, Davidson, Patney, Knupp, and Owens (2011), *Triangle* is nearly linear in practice.

Held (1998) proposed an incremental algorithm for the generation of 2D Voronoi diagrams. This algorithm uses wave propagation to compute the diagrams for curvilinear polygons, which are simple and closed polygons, and for areas delimited by straight lines. According to Held (1998), the computational cost of this algorithm seemed to grow linearly. Using inputs up to 8,000 line segments, Held (1998) compared his algorithm experimentally against an adapted version of the Lee (1982) divide-and-conquer algorithm. According to the author, the Lee (1982) algorithm was approximately 46%-47% slower in the tests than the Held (1998) algorithm.

Lemaire and Moreau (2000) proposed a divide-and-conquer algorithm for the generation of Delaunay tessellations. It is an expected linear-time algorithm when applied to a unit hypercube with point density of almost uniform probability. Lemaire and Moreau (2000) compared this algorithm against the Lee and Schachter (1980), Dwyer (1987), Katajainen and Koppinen (1988) and Adam, Elbaz, and Spehner (1996) algorithms. In experiments carried out using over 10 million points, the

Lemaire and Moreau (2000) algorithm was faster than the Lee and Schachter (1980) and Adam, Elbaz, and Spehner (1996) algorithms with uniform point distributions, and it was also the fastest algorithm for non-uniform point distributions in a unit square.

Held (2001) proposed an incremental algorithm for the generation of 2D Voronoi diagrams, with n input segments. According to Held (2001), this algorithm needed about $0.01 \cdot n \; log_2 \; n$ milliseconds to compute Voronoi diagrams of n line segments in tests carried out with up to 2^{19} segments in a Sun Ultra 30 computer with Solaris 2.6 as the operational system, a 296MHz processor and 384MB of main memory. According to Held (2001), this behaviour was valid for a wide variety of synthetic and real data.

The complexities and the conditions under which such complexities are reached by the 15 algorithms found for this period are indicated in Table 4.

5 Third period (2003-2013)

The generation of Delaunay or Voronoi tessellations of sets of points is independent of the order in which these points are processed. However, the computational cost of an algorithm for each of these meshes depends on the order in which the points are processed.

Amenta, Choi, and Rote (2003) observed that the hierarchy of modern computer memory and paging policies favour the locality of reference. Current computer memory systems cache recently used data on the assumption that those data will be probably used again very soon. The insertion of points in a *biased randomized insertion order* (BRIO) is an incremental algorithm, in which the order of point insertion is biased randomly. This means that this algorithm preserves enough randomness in the input points so that the performance of a randomized incremental algorithm is not changed, but orders the points by spatial locality to gain cache coherence and to retain optimality for generating 3D structures. The points are inserted in rounds, in which each point is chosen independently with 50% of probability to be inserted in the current round. Amenta, Choi, and Rote (2003) indicated that the expected running time is $O(n^2)$ in the worst case and $O(n \lg n)$ in the realistic case; moreover, it runs quickly for many point distributions. According to Amenta, Choi, and Rote (2003), BRIO's performance is nearly linear. The authors presented tests with up to 10 million points. However, according to other researchers [Liu and Snoeyink (2005); Zhou and Jones (2005)], the practical performance of BRIO is not promising. Indeed, Amenta, Choi, and Rote (2003) considered BRIO as a concept instead of a specific order, and that BRIO could be combined with other insertion schemes. This algorithm led to a conceptual change

Algorithm	Complexity	Comment
Aggarwal, Guibas, Saxe, and Shor (1989)	$\Theta(n)$	points form the vertices of a convex polygon
Dwyer (1989, 1991)	expected linear behaviour	set of points quasi-uniformly distributed
	worst case: $O(S_n n \lg n)$, in which S_n is the number of dual simplices in the result	if a balanced-tree implementation of priority queues is used
	worst case: $O(S_n n)$	if the use of buckets is abandoned on any point search that examines $\sqrt{n}$ buckets
Chew (1990)	expected linear time	set of points in the plane such that the points taken in order form the vertices of a convex polygon
	worst case: $O(n^2)$	e.g. considering points along one branch of a parabola
Sugihara and Iri (1992)	average case: linear	behaviour observed in the experimental tests
Klein and Lingas (1992)	linear	using Manhattan metric
Klein and Lingas (1993, 1996)	expected linear time	builds the bounded Voronoi diagram of a simple polygon
Tsai (1993)	expected linear time	randomly distributed points in the Euclidean plane
	worst case: $O(n^2)$	analyzed by Larkin (1991)
Chin and Wang (1995, 1998)	linear	using amortized analysis
Djidjev and Lingas (1995)	linear	points in the plane in sorted order with respect to two perpendicular directions
Shewchuk (1996)	nearly linear in practice	for difficult inputs; but it may be slow for uniformly distributed point sets
	worst case: $O(n \lg n)$	regardless of the points distribution
Su and Drysdale (1997)	incremental: expected linear time	points uniformly distributed
Su and Drysdale (1997)	modified Fortune's algorithm: linear behaviour	
Held (1998)	almost linear	results in tests
	worst cases: $O(n \lg n)$,	for a generalization of convex areas
	and $O(n^2 \lg n)$	analyzed by the author
Lemaire and Moreau (2000)	expected linear time	if merging two subsets is assumed to take time proportional to the number of the involved unfinished points
	worst case: $\Theta(n \lg n)$	analyzed by the authors
Held (2001)	$0,01n \log_2 n$ ms	wide variety of data in a 296 MHz Sun Ultra 30

Table 4: The 15 linear-time behaviour algorithms found in the second period, their complexities and the conditions under which such complexities are reached.

in the development of algorithms with linear-time behaviour for Delaunay tessellations.

Liu and Snoeyink (2005) proposed an incremental algorithm for 3D Delaunay tessellations based on space-filling curves. In an algorithm that uses a space-filling curve, the order of points to be inserted is defined by the order of the curve.

In the Liu and Snoeyink (2005) algorithm, the Hilbert curve (see Figure 4) was used to determine the order of the input points. Liu and Snoeyink (2005) compared the algorithm with the Hilbert curve against QHull [Barber, Dobkin, and Huhdanpa (1996)], CGAL [Devillers (1998); Boissonnat, Devillers, Pion, Teillaud, and Yvinec (2002)], Hull [Clarkson (1992)] and Pyramid [Shewchuk (1998b)] algorithms. According to Liu and Snoeyink (2005), in tests with up to 1,024,000 points, their algorithm was the fastest, particularly for uniform point distributions. The Liu and Snoeyink (2005) algorithm was also a conceptual change in the development of algorithms with linear-time behaviour for Delaunay tessellations. Since the publication of this algorithm, the algorithms with linear-time behaviour for Delaunay tessellations are mostly incremental and use the insertion of points in a pre-determined order. For example, the Hilbert curve, combined with concepts of BRIO, is employed in CGAL [Delage and Devillers (2013)].

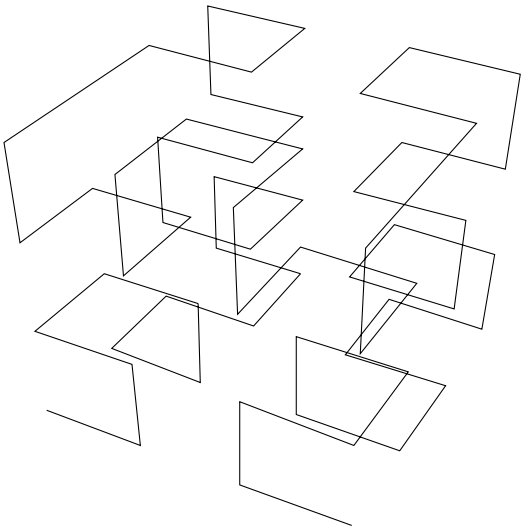

Figure 4: 3D Hilbert space-filling curve.

Buchin (2005) proposed an incremental algorithm for Delaunay triangulations based on BRIO and space-filling curves. According to Buchin (2005), this algorithm was expected to have linear time for uniform point distributions in a bounded convex region. Buchin (2005) also presented a second incremental algorithm for the generation of Delaunay tessellations with expected linear time in cases with input points distributed uniformly in a d-dimensional bounded convex open region.

Boissonnat, Devillers, and Hornus (2009) proposed an incremental algorithm for Delaunay tessellations and carried out tests with up to 6 dimensions. According to

the authors, tests carried out with up to 256,000 points in 6 dimensions showed that this algorithm surpassed in time and memory, the implementations that were available for the exact calculation of the Delaunay triangulation. In addition, in terms of performance, this algorithm could be compared with QHull [Barber, Dobkin, and Huhdanpa (1996)], which is a non-robust algorithm for the generation of Delaunay tessellations. In the Boissonnat, Devillers, and Hornus (2009) algorithm, the points are previously sorted along a d-dimensional Hilbert curve. According to these authors, the worst cases of this algorithm is $O(n \lg n)$ for 2D structures and $O(n^{\lceil \frac{d}{2} \rceil})$ for structures with higher dimensions. However, the authors explained that the worst case does not occur in practice. Also according to them, the main limitation of their algorithm is its memory usage. In order to overcome this problem, the authors proposed a variant of the algorithm, but it is 6 or 8 times slower than the original version.

The generation of Delaunay triangulations using space-filling curves was also studied by Buchin (2009). Buchin (2009) developed a theoretical analysis for the linear or near-linear running time in experiments using the *incremental construction con BRIO* [Amenta, Choi, and Rote (2003)] with space-filling curve orders; briefly, his algorithm incrementally builds the Delaunay triangulation using the order of the Hilbert curve previously computed. According to the author, the use of the two combined techniques results in quadratic algorithms in degenerated input cases. Also according to the author, the use of these two combined techniques results in algorithms that are, generally, $O(n \lg n)$ for point sets with a polynomially bounded spread. For Buchin (2009), spread is the quotient between the highest and the lowest point-to-point distance. Buchin (2009) showed that this algorithm has expected linear time for different random point distributions.

Buchin and Mulzer (2009, 2011) presented several results related to the generation of Delaunay triangulations. Among the most relevant results, one can point out: Delaunay triangulation can be computed in $O(\text{sort}(n))$ on a word RAM model, in which sort(n) is the necessary time to sort n numbers; if both axis for sorting a set of points is known, a Delaunay triangulation can be generated with a random algebraic computation tree with expected linear depth. Buchin and Mulzer (2011) used a variant of the Random Incremented Construction (RIC). For the generation of Delaunay triangulation, BRIO with Dependent Choices (BrioDC) was used for the insertion of points in the domain. In the BrioDC algorithm, the problem of building a Delaunay triangulation was reduced to the problem of building the nearest-neighbour graph based on BRIO concepts. According to these authors, if the nearest-neighbour graph can be computed in linear time, the reduction of a Delaunay triangulation to the nearest-neighbour graph will be proportional to the structural changes in RIC, which would always be in linear time for copla-

nar points. This algorithm is linear for small integers by sorting them using the radix-sort technique. Schrijvers, van Bommel, and Buchin (2012) stated that this algorithm presented a linear-time behaviour in experiments, but the constant factor in the running time was high. The high constant is due to the construction of the worst-case optimal nearest-neighbour graph.

Yang and Choi (2010) proposed a compact incremental algorithm for the generation of constrained 3D Delaunay tessellations. This algorithm is especially useful for 3D visualizations in mobile devices with low memory capacity and CPU speed. In general, mobile devices are provided with previously built tessellations models, because a large amount of memory is required to generate these structures. In particular, the algorithm of Yang and Choi (2010) uses a small amount of memory to generate Delaunay tessellations. According to the authors, this algorithm is $O(n \lg n)$ in the worst case, and presented a linear-time behaviour in experimental tests with sets of up to 2,000 segments.

Ebeida, Mitchell, Davidson, Patney, Knupp, and Owens (2011) proposed an algorithm to generate constrained Delaunay triangulations composed of triangles with angles from 30^o to 120^o. In this algorithm, vertices are added, step by step, in an empty disc and the probability of inserting a vertex in a disc was proportional to its area, except in a neighbourhood of the domain boundary. This algorithm is based on Delaunay refinement algorithms and, according to the authors, can be parallelized. Also, this algorithm requires $O(n \lg n)$ operations and $O(n)$ of memory. In tests with sets of up to 8,271,560 points, this algorithm presented a nearly linear performance in a squared uniform mesh, and had a performance similar to *Triangle* [Shewchuk (1996, 2002)]. Moreover, implementation of Ebeida, Mitchell, Davidson, Patney, Knupp, and Owens (2011) and *Triangle* required more or less the same time to triangulate. On the other hand, *Triangle* generated points much faster.

Yang, Choi, and Jung (2011) proposed a divide-and-conquer algorithm to build Delaunay triangulations in which the merge process is based on edge flip operations without deleting any of the existing edges or triangles. According to the authors, empirical results with sets of up to 30,000 points showed that this is an expected $O(n \lg n)$ algorithm. Nevertheless, the test results showed linear-time complexity for a quasi-uniformly distributed sites set. The maintenance of this algorithm is easily performed as it uses a compact data structure with easy access to the data.

Löffler and Mulzer (2011, 2012) proved that Delaunay triangulations and quadtrees are equivalent structures. In the proposed incremental algorithm, a compressed quadtree is built from a set of points in the plane and from this tree a Delaunay triangulation is generated. In a compressed quadtree, long paths with only one non-empty child node are changed to single edges. According to the authors, given a compressed quadtree, this algorithm has linear time for Delaunay triangulations

in a pointer machine model, and it is linear for small integers also by sorting them using radix sort.

Schrijvers, van Bommel, and Buchin (2012, 2013) proposed incremental algorithms to generate Delaunay triangulations with linear time for small integers. Schrijvers, van Bommel, and Buchin (2012) evaluated variants of BRIO and BrioDC techniques that showed linear-time behaviours in tests with sets of up to 2^{22} points. The variants evaluated were RIC and BRIO algorithms using Peano, Sierpiński, and Hilbert curves and their variants. According to the authors, the proposed variants tended to avoid the worst-case behaviour and the squarified versions of the curves allowed quick location of points. Also according to the authors, the fastest algorithm in the experiment was the one based on the BRIO technique. Schrijvers, van Bommel, and Buchin (2013) claimed that using the nearest-neighbour graphs of each round in BrioDC, the number of simplices visited is reduced by more than 25% compared to the various fast space-filling curves they had tested.

Liu, Yan, and Lo (2013) proposed an incremental algorithm for Delaunay triangulations that was generalized for structures with dimension higher than 2. With this algorithm, the authors proposed a sequence of point insertions based on a new order from breadth-first search on a kd-tree. A standard kd-tree is built splitting the point subset by the median point in an axis, and this point is stored in the root of the tree. The remaining two subset points are stored into left and right leaf nodes. In a 2D point set, each subset point is divided along the x-median (if the y-median was used to split the original point set), then the y-median is used and so forth, splitting the remaining point subsets until each leaf node contains only one point. However, Liu, Yan, and Lo (2013) used a slightly different splitting scheme to divide the input point subsets, and called it cutting-longest-edge rule. A bounding box is determined from a set of points, and their rule is to cut the longest edge of the bounding box in order to create regions of more homogeneous size along different dimensions. In Figure 5, one can see an example of this building process for a set of seven 2D points by the cutting-longest-edge rule.

According to Liu, Yan, and Lo (2013), their algorithm was faster than the algorithm that uses the Hilbert curve, than the algorithm that uses the BRIO and than a random algorithm. Still, according to these authors, tests with sets of up to 5.5 million points in non-uniform 3D point distributions in seven scholastic models (cube, plane, paraboloid, spiral, disc, cylinder, and axes) showed that the algorithm based on kd-tree was very stable, and the algorithm that used Hilbert curve and the algorithm that used BRIO technique had drastically deteriorated performances. Using a set of up to 5.5 million points, Liu, Yan, and Lo (2013) found that the algorithm used with the Hilbert curve failed, and that the algorithm with random insertion of points was aborted because the necessary time to generate the triangulation was

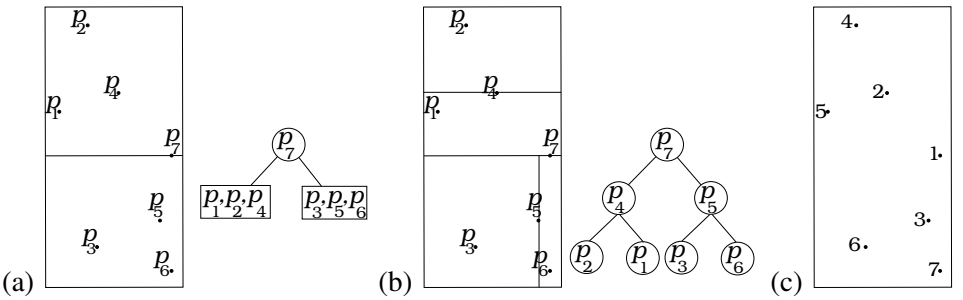

Figure 5: Steps to build a kd-tree following the cutting-longest-edge rule proposed by Liu, Yan, and Lo (2013): In (a), the point distribution is divided by the median point p_7 because the vertical side of the bounding box is longer than the horizontal side. Then, the bounding box is divided into two rectangles, and the process of assembling the kd-tree is started. (It should be noticed that with this partition, the horizontal side is longer than the vertical side of the bottom rectangle.) In (b), the two consequent point subsets are divided again, resulting in the new insertion sequence for this point set by the breadth-first travel across the kd-tree, that is, p_7, p_4, p_5, p_2, p_1, p_3, p_6 that are relabelled in a new insertion sequence as shown in (c).

superior to 30 minutes.

Lo (2013) proposed an incremental algorithm for Delaunay triangulations. The author presented a scheme of multi-grid insertions that showed better results in the 2D case than the Liu, Yan, and Lo (2013) algorithm. The scheme used resulted from the recursive application of a regular grid to each polygon. According to this author, it resulted in a scheme with an almost linear-time behaviour for sets of points that are locally uniformly distributed. Also, the scheme of multi-grid insertion was the most stable and the most efficient when compared to schemes of kd-tree insertion and of a regular grid. According to the same author, the scheme of the regular grid was very sensitive to the point distribution and the kd-tree scheme had a high computational cost for triangulations with a high number of elongated triangles. This author observed an almost linear-time behaviour when using a scheme of multi-grid insertion. The presented tests were performed with up to 100 million points in uniform, line, cross, spiral, circle, and cluster point distributions. It should be noticed that Lo's comparisons were limited to the 2D case, in which either the regular grid or the multi-grid scheme could be used with a low computational cost.

The complexities and the conditions under which such complexities are reached by

the 14 algorithms found for the third period are shown in Table 5.

6 Conclusions

A systematic review of algorithms with linear computational time behaviour for the generation of 2D/3D Delaunay and Voronoi tessellations was presented. The main strategies that have been employed for the development of algorithms with linear-time behaviour for the generation of these tessellations over the past 36 years were enumerated. From the search conducted it was possible to identify the algorithms that are the probable state-of-art solutions for the generation of Delaunay tessellations: Lo (2013) and Liu, Yan, and Lo (2013) algorithms for the generation of 2D and 3D Delaunay tessellations, respectively.

The divide-and-conquer strategy has been reasonably well employed for the development of algorithms with linear-time behaviour for the generation of Delaunay or Voronoi tessellations. On average at least one divide-and-conquer algorithm with linear-time behaviour for the generation of Delaunay or Voronoi tessellation has been published each 4 and a half years over the past 36 years.

On the other hand, the incremental approach has been employed even more than the divide-and-conquer approach for the development of algorithms with linear-time behaviour for these meshes. On average at least one incremental algorithm with linear-time behaviour for the generation of Delaunay or Voronoi tessellations has been published almost every year and a half over the last 30 years. More specifically, since the publications of BRIO [Amenta, Choi, and Rote (2003)] concepts and the algorithm of Liu and Snoeyink (2005), the tendency in the development of algorithms for Delaunay tessellations has been by the incremental approach and by a means of insertions in which the reference locality is preserved. Furthermore, since 2003, more than one new incremental algorithm has been proposed on average per year against one divide-and-conquer algorithm in the same period.

Acknowledgement: This work was developed with the support of CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico (National Council for Scientific and Technological Development, in Brazil), CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Coordination for Enhancement of Higher Education Personnel, in Brazil), and FAPEMIG - Fundação de Amparo à Pesquisa do Estado de Minas Gerais (Minas Gerais Research Support Foundation, in Brazil). This work was also partially done in the scope of the project with reference "PTDC/BBB-BMD/3088/2012" financially supported by Fundação para a Ciência e a Tecnologia (FCT) in Portugal.

Algorithm	Complexity	Comment
Amenta, Choi, and Rote (2003)	nearly linear	results obtained in tests
	worst cases: $O(n^2)$, and $O(n \lg n)$ in the "realistic" case	verified by the authors
Liu and Snoeyink (2005)	tess3 was faster than QHull, CGAL, Hull and Pyramid	results obtained in tests
Buchin (2005)	expected linear time	1st algorithm: points distributed independently and uniformly in a bounded convex area
Buchin (2005)		2nd algorithm: points distributed independently and uniformly in a d-dimensional bounded convex open region
Boissonnat, Devillers, and Hornus (2009)	worst case when $d > 2$: $O\left(n^{\lceil d/2 \rceil}\right)$	fast in practice, authors presented comparisons showing that their implementation outperformed available codes for Delaunay triangulations and can be used with large input sets in spaces of dimensions up to 6
	worst case (2D): $O(n \lg n)$	by generating an "onion-like" layered simplicial subdivision of the convex hull, which is also used as a locating data-structure
Buchin (2009)	expected linear time	many random point distributions
	worst cases: $O(n^2)$,	degenerate cases
	and $O(n \lg n)$	if the quotient between the highest and the lowest point to point distance is polynomially bounded
Buchin and Mulzer (2009, 2011)	linear	for small integers, but the constant factor is high
Yang and Choi (2010)	linear	experimental results
	worst case: $O(n \lg n)$	regardless of the site distribution
Ebeida, Mitchell, Davidson, Patney, Knupp, and Owens (2011)	nearly linear in practice	given sample points prelocated in a squared uniform grid
	$O(n \lg n)$ expected time	the $\lg n$ dependence is very mild
Yang, Choi, and Jung (2011)	linear behaviour	test results in the quasi-uniformly distributed points set
	$O(n \lg n)$ expected time	observed in the experiments
Löffler and Mulzer (2011, 2012)	linear	on a pointer machine and for small integers
Schrijvers et al. (2012, 2013)	linear	on small integers
	worst case: $O(n \lg n)$	
Liu, Yan, and Lo (2013)	it runs faster than the algorithms with the Hilbert curve order and with BRIO, and also faster than a random algorithm	especially for non-uniform point distributions over a wide range of benchmark examples
Lo (2013)	quasi-linear	for the triangulation of distribution sets with local characteristics similar to those of a uniform point distribution, such as line, circle and cluster

Table 5: The 14 linear-time behaviour algorithms found for the third period, their complexities, and the conditions under which such complexities are reached.

References

Adam, B.; Elbaz, M.; Spehner, J. C. (1996): Construction du diagramme de Delaunay dans le plan en utilisant les mélanges de tris. In *Actes 4èmes Journées de l'AFIG*, pp. 215–223, France.

Aggarwal, A.; Guibas, L.; Saxe, J.; Shor, P. (1989): A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete & Computational Geometry*, vol. 4, pp. 591–604.

Alliez, P.; Meyer, M.; Desbrun, M. (2002): Interactive geometry remeshing. *ACM Trans. Graph.*, vol. 21, no. 3, pp. 347–354.

Amenta, N.; Choi, S.; Rote, G. (2003): Incremental constructions con BRIO. In *Proc. 90th Annual Symposium on Computational Geometry*, pp. 211–219, USA. ACM.

Barber, C. B.; Dobkin, D. P.; Huhdanpa, H. (1996): The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, vol. 22, no. 4, pp. 469–483.

Bentley, J. L.; Weide, B. W.; Yao, A. C. (1978): Optimal expected-time algorithms for closest point problems. In *16th Proceedings - Annual Allerton Conference on Communication, Control, and Computing*, pp. 843–851, Monticello, IL, USA.

Bentley, J. L.; Weide, B. W.; Yao, A. C. (1980): Optimal expected-time algorithms for closest point problems. *ACM Transactions on Mathematical Software*, vol. 6, pp. 563–580.

Bishay, P. L.; Atluri, S. N. (2012): High-performance 3D hybrid/mixed, and simple 3D Voronoi cell finite elements, for macro- & micro-mechanical modeling of solids, without using multi-field variational principles. *CMES: Computer Modeling in Engineering & Sciences*, vol. 84, no. 1, pp. 41–97.

Boissonnat, J.-D.; Devillers, O.; Hornus, S. (2009): Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *Proc. 25th Annual Symposium on Computational Geometry*, pp. 208–216, USA. ACM.

Boissonnat, J.-D.; Devillers, O.; Pion, S.; Teillaud, M.; Yvinec, M. (2002): Triangulations in CGAL. *Computational Geometry: Theory and Applications*, vol. 22, pp. 5–19.

Buchin, K. (2005): Incremental construction along space-filling curves. In *21st European Workshop on Computational Geometry*, pp. 17–20, Eindhoven.

Buchin, K. (2009): Constructing Delaunay triangulations along space-filling curves. In Fiat, A.; Sanders, P.(Eds): *Algorithms-ESA 2009*, volume 5757 of *Lecture Notes in Computer Science*, pp. 119–130. Springer Berlin Heidelberg.

Buchin, K.; Mulzer, W. (2009): Delaunay triangulations in $O(\text{sort}(n))$ time and more. In *Proc. Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pp. 139–148.

Buchin, K.; Mulzer, W. (2011): Delaunay triangulations in $O(\text{sort}(n))$ time and more. *J. ACM*, vol. 58, no. 2, pp. 1–27.

Busaryev, O.; Dey, T. K.; Wang, H. (2013): Adaptive fracture simulation of multi-layered thin plates. *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 52:1–52:6.

Cavendish, J. C.; Field, D. A.; Frey, W. H. (1985): An approach to automatic three-dimensional nite element mesh generation. *International Journal of Numerical Methods in Engineering*, vol. 21, pp. 329–347.

Chew, L. P. (1990): Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Department of Mathematics and Computer Science, Dartmouth College, Hanover, USA, 1990.

Chin, F.; Wang, C. A. (1995): Finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple polygon in linear time. In Spirakis, P.(Ed): *Proceedings of the Third Annual European Symposium on Algorithms - ESA '95, Lecture Notes in Computer Science 979*, pp. 280–294, Corfu, Greece. Springer Verlag.

Chin, F.; Wang, C. A. (1998): Finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple polygon in linear time. *SIAM Journal On Computing*, vol. 28, no. 2, pp. 471–486.

Clarkson, K. L. (1992): Safe and effective determinant evaluation. In *Proc. 33rd Annual Symposium on Foundations of Computer Science*, pp. 387–395.

De Gyves, O.; Toledo, L.; Rudomín, I. (2013): Proximity queries for crowd simulation using truncated Voronoi diagrams. In *Proceedings of Motion on Games*, MIG '13, pp. 65:87–65:92, New York. ACM.

Delage, C.; Devillers, O. (2013): Spatial sorting. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition.

Devillers, O. (1998): Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annual Symposium on Computational Geometry*, pp. 106–115. ACM.

Djidjev, H. N.; Lingas, A. (1995): On computing the Voronoi diagram for sorted point sets. *International Journal of Computational Geometry & Applications*, vol. 5, no. 3, pp. 327–337.

Dong, L.; Atluri, S. N. (2012): T-Trefftz Voronoi cell finite elements with elastic/rigid inclusions or voids for micromechanical analysis of composite and porous materials. *CMES: Computer Modeling in Engineering & Sciences*, vol. 83, no. 2, pp. 183–219.

Dwyer, R. A. (1987): A faster divide-and-conquer algorithm for constructing Delaunay triangulations. *Algorithmica*, vol. 2, pp. 137–151.

Dwyer, R. A. (1989): Higher-dimensional Voronoi diagrams in linear expected time. In *Proc. 5th Annual Symposium on Computational Geometry*, pp. 326–333. ACM, USA.

Dwyer, R. A. (1991): Higher-dimensional Voronoi diagrams in linear expected time. *Discrete & Computational Geometry*, vol. 6, pp. 343–367.

Ebeida, M. S.; Mitchell, S. A.; Davidson, A. A.; Patney, A.; Knupp, P. M.; Owens, J. D. (2011): Efficient and good Delaunay meshes from random points. *Computer-Aided Design*, vol. 43, no. 11, pp. 1506–1515.

Edelsbrunner, H. (2001): *Geometry and topology for mesh generation*. Cambridge monographs on applied and computational mathematics. Cambridge University Press, New York.

Fortune, S. (1987): A sweepline algorithm for Voronoi diagrams. *Algorithmica*, vol. 2, pp. 153–174.

Gonzaga de Oliveira, S. L.; Kischinhevsky, M.; Tavares, J. M. R. S. (2013): Novel graph-based adaptive triangular mesh refinement for finite-volume discretizations. *CMES: Computer Modeling in Engineering & Sciences*, vol. 95, no. 2, pp. 119–141.

Guibas, L. J.; Stolfi, J. (1985): Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, vol. 4, no. 2, pp. 74–123.

Held, M. (1998): Voronoi diagrams and offset curves of curvilinear polygons. *Computer-Aided Design*, vol. 30, no. 4, pp. 287–300.

Held, M. (2001): Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Computational Geometry*, vol. 18, pp. 95–123.

Katajainen, J.; Koppinen, M. (1988): Constructing Delaunay triangulations by merging buckets in quadtree order. *Annales Societatis Mathematicae Polonae*, vol. 3, pp. 275–288.

Klein, R.; Lingas, A. (1992): Manhattonian proximity in a simple polygon. In *Proc. 8th Annual Symposium on Computational Geometry*, pp. 312–319, USA. ACM.

Klein, R.; Lingas, A. (1993): A linear-time randomized algorithm for the bounded Voronoi diagram of a simple polygon. In *Proceeding SCG'93 Proceedings of the ninth annual symposium on Computational geometry*, pp. 124–132, New York, NY. ACM.

Klein, R.; Lingas, A. (1996): A linear-time randomized algorithm for the bounded Voronoi diagram of a simple polygon. *International Journal of Computational Geometry and Applications*, vol. 6, no. 3, pp. 263–278.

Larkin, B. K. (1991): An ansi c program to determine in expected linear time the vertices of the convex hull of a set of planar points. *Computers & Geosciences*, vol. 17, no. 3, pp. 431–443.

Lawson, C. L. (1977): Software for C^1 surface interpolation. In Rice, J. R.(Ed): *Mathematical Software III*, pp. 161–194, New York. Academic Press.

Lee, D. T. (1982): Medial axis transformation of a planar shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, no. 4, pp. 363–369.

Lee, D. T.; Schachter, B. J. (1980): Two algorithms for constructing a Delaunay triangulation. *International Journal of Computer and Information Sciences*, vol. 9, no. 3, pp. 219–242.

Lemaire, C.; Moreau, J.-M. (2000): A probabilistic result on multi-dimensional Delaunay triangulations, and its application to the 2D case. *Computational Geometry*, vol. 17, no. 1-2, pp. 69–96.

Li, X. (2000): *Sliver-free three dimensional Delaunay mesh generation*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 2000.

Lin, L.; Wang, X.; Zeng, X. (2014): Geometrical modeling of cell division and cell remodeling based on Voronoi tessellation method. *CMES: Computer Modeling in Engineering & Sciences*, vol. 98, no. 2, pp. 203–220.

Liu, J.; Chen, Y. Q.; Sun, S. L. (2009): Small polyhedron reconnection for mesh improvement and its implementation based on advancing front technique. *International Journal for Numerical Methods in Engineering*, vol. 79, no. 8, pp. 1004–1018.

Liu, J.; Li, S.; Chen, Y. (2008): A fast and practical method to pack spheres for mesh generation. *Acta Mechanica Sinica*, vol. 24, no. 4, pp. 439–447.

Liu, J.-F.; Yan, J.-H.; Lo, S. H. (2013): A new insertion sequence for incremental Delaunay triangulation. *Acta Mechanica Sinica*, vol. 29, no. 1, pp. 99–109.

Liu, Y.; Snoeyink, J. (2005): A comparison of five implementations of 3D Delaunay Tessellation. *Combinatorial and Computational Geometry*, vol. 52, pp. 439–458.

Lo, S. H. (2013): Delaunay triangulation of non-uniform point distributions by means of multi-grid insertion. *Finite Elements in Analysis and Design*, vol. 63, pp. 8–22.

Löffler, M.; Mulzer, W. (2011): Triangulating the square and squaring the triangle: quadtrees and Delaunay triangulations are equivalent. In *Proc. 22th Annual ACM SIAM SODA*, pp. 1759–1777.

Löffler, M.; Mulzer, W. (2012): Triangulating the square and squaring the triangle: quadtrees and Delaunay triangulations are equivalent. *SIAM J. Comput.*, vol. 41, no. 4, pp. 941–974.

Maus, A. (1984): Delaunay triangulation and the convex hull of n points in expected linear time. *BIT*, vol. 24, no. 2, pp. 151–163.

Nordin, A.; Hopf, A.; Motte, D.; Bjärnemo, R.; Eckhardt, C.-C. (2011): An approach to constraint-based and mass-customizable product design. *Journal of Computing and Information Science in Engineering*, vol. 11, no. 011006, pp. 1–7.

Ohya, T.; Iri, M.; Murota, K. (1984): Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms. *Journal of the Operational Research Society of Japan*, vol. 27, pp. 306–337.

Puentes, J.; Dhibi, M.; Bressollette, L.; Guias, B.; Solaiman, B. (2009): Computer-assisted venous thrombosis volume quantification. *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 2, pp. 174–183.

Ruppert, J. (1995): A Delaunay refinement algorithm for quality 2-dimensional mesh generation. In *Selected papers 4th Annual ACM SIAM SODA*, pp. 548–585, USA. Academic Press, Inc.

Schrijvers, O.; van Bommel, F.; Buchin, K. (2012): Delaunay triangulations on the word RAM: Towards a practical worst-case optimal algorithm. In *28th European Workshop on Computational Geometry (EuroCG), Booklet of Abstracts*, pp. 13–16.

Schrijvers, O.; van Bommel, F.; Buchin, K. (2013): Delaunay triangulations on the word RAM: Towards a practical worst-case optimal algorithm. In *Proc. 10th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*, pp. 7–15.

Shamos, M. I.; Hoey, D. (1975): Closest-point problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, pp. 151–162, Washington, DC, USA. IEEE Computer Society.

Shewchuk, J. R. (1996): Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Selected papers from the Workshop on Computational Geometry*, pp. 203–222, UK. Springer-Verlag.

Shewchuk, J. R. (1998): Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, pp. 86–95, Minneapolis, Minnesota. Association for Computing Machinery.

Shewchuk, J. R. (1998): Tetrahedral mesh generation by Delaunay refinement. In *Proc. 40th Annual Symposium on Computational Geometry*, pp. 86–95, USA. ACM.

Shewchuk, J. R. (2002): Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, vol. 22, no. 1-3, pp. 21–74.

Sibson, R.; Green, P. J. (1978): Computing dirichlet tessellations in the plane. *Comput. J.*, vol. 21, pp. 168–173.

Su, P.; Drysdale, R. L. S. (1995): A comparison of sequential Delaunay triangulation algorithms. In *Proc. of 11th Annu. ACM Symposium on Computational Geometry*, 61–70.

Su, P.; Drysdale, R. L. S. (1997): A comparison of sequential Delaunay triangulation algorithms. *Computational Geometry*, vol. 7, pp. 361–385.

Sugihara, K.; Iri, M. (1992): Construction of the Voronoi diagram for one million generators in single-precision arithmetic. *Proc. IEEE*, vol. 80, no. 9, pp. 1471–1484.

Talmor, D. (1997): *Well-Spaced Points for Numerical Methods*. PhD thesis, Carnegie Mellon University, USA, 1997.

Tsai, V. J. D. (1993): Fast topological construction of Delaunay triangulations and Voronoi diagrams. *Computers & Geosciences*, vol. 19, no. 10, pp. 1463–1474.

Tucker, G. E.; Lancaster, S. T.; Gasparini, N. M.; Bras, R. L.; Rybarczyk, S. M. (2001): An object-oriented framework for distributed hydrologic and geomorphic modeling using triangulated irregular networks. *Computers & Geosciences*, vol. 27, no. 8, pp. 959 – 973.

Yang, S. W.; Choi, Y. (2010): Triangulation of CAD data for visualization using a compact array-based triangle data structure. *Computer and Graphics*, vol. 34, no. 4, pp. 424–429.

Yang, S.-W.; Choi, Y.; Jung, C.-K. (2011): A divide-and-conquer Delaunay triangulation algorithm with a vertex array and flip operations in two-dimensional space. *International Journal of Precision Engineering and Manufacturing*, vol. 12, no. 3, pp. 435–442.

Zhou, S.; Jones, C. B. (2005): HCPO: an efficient insertion order for incremental Delaunay triangulation. *Information Processing Letters*, vol. 93, no. 1, pp. 37–42.