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

Sanderson L. Gonzaga de Oliveira¹, Jéssica Renata Nogueira¹, and João Manuel R. S. Tavares²

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

¹Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras, MG, Brazil, sanderson@dcc.ufla.br, jes.r.nogueira@gmail.com
²Instituto 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 Rudomin (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 $P_i = \{ p \in \mathbb{R}^d : (\exists p_i \in S)(\forall p_j \in S) ||p - p_i|| \leq ||p - p_j||, \text{ with } i \neq j \land 1 \leq i, j \leq n \}$.

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 $O(n \lg n)$\textsuperscript{1} worst-case running time is

\textsuperscript{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 $\Theta(n \log n)$. 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 $O(n^3/2)$ 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.

$\log_e a \cdot \log_a n = O(\log n)$ because $\log_e a$ 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 $O(\log n)$ 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
<table>
<thead>
<tr>
<th>Period</th>
<th>Algorithm</th>
<th>Technique</th>
<th>Tessellation</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bentley, Weide, and Yao (1978)</td>
<td>divide and conquer</td>
<td>Voronoi</td>
<td>10000</td>
</tr>
<tr>
<td></td>
<td>Maus (1984)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ohya, Iri, and Murota (1984)</td>
<td>incremental</td>
<td>Voronoi</td>
<td>32768</td>
</tr>
<tr>
<td></td>
<td>Dwyer (1987)</td>
<td>divide and conquer</td>
<td>Delaunay</td>
<td>65536</td>
</tr>
<tr>
<td></td>
<td>Katajainen and Koppinen (1988)</td>
<td>divide and conquer</td>
<td>Delaunay</td>
<td>32768</td>
</tr>
<tr>
<td></td>
<td>Aggarwal, Guibas, Saxe, and Shor (1989)</td>
<td>lifting map</td>
<td>Voronoi</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Dwyer (1989, 1991)</td>
<td>gift wrapping</td>
<td>Voronoi</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Chew (1990)</td>
<td>gift wrapping</td>
<td>Voronoi</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Sugihara and Iri (1992)</td>
<td>incremental</td>
<td>Voronoi</td>
<td>1000000</td>
</tr>
<tr>
<td></td>
<td>Klein and Lingas (1992)</td>
<td>sweep line and divide and conquer</td>
<td>Voronoi</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Klein and Lingas (1993, 1996)</td>
<td>divide and conquer</td>
<td>Voronoi</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Tsai (1993)</td>
<td>incremental</td>
<td>Delaunay, Voronoi</td>
<td>50000</td>
</tr>
<tr>
<td></td>
<td>Chin and Wang (1995, 1998)</td>
<td>-</td>
<td>CDT, CVD</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Djidjev and Lingas (1995)</td>
<td>lifting map</td>
<td>Voronoi</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Su and Drysdale (1997)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>131072</td>
</tr>
<tr>
<td></td>
<td>Su and Drysdale (1997)</td>
<td>sweep line</td>
<td>Delaunay</td>
<td>131072</td>
</tr>
<tr>
<td></td>
<td>Shewchuk (1996)</td>
<td>incremental, sweep line or divide and conquer</td>
<td>Delaunay, CDT, Voronoi</td>
<td>1000000</td>
</tr>
<tr>
<td></td>
<td>Held (1998)</td>
<td>incremental</td>
<td>Voronoi</td>
<td>8000 (s)</td>
</tr>
<tr>
<td></td>
<td>Lemaire and Moreau (2000)</td>
<td>divide and conquer</td>
<td>Delaunay</td>
<td>10000000</td>
</tr>
<tr>
<td></td>
<td>Held (2001)</td>
<td>incremental</td>
<td>Voronoi</td>
<td>524288 (s)</td>
</tr>
<tr>
<td></td>
<td>Amenta, Choi, and Rote (2003)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>10000000</td>
</tr>
<tr>
<td></td>
<td>Liu and Snoeyink (2005)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>1024000</td>
</tr>
<tr>
<td></td>
<td>Buchin (2005)</td>
<td>(two) incremental</td>
<td>Delaunay</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Boissonnat, Devillers, and Hornus (2009)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>256000</td>
</tr>
<tr>
<td></td>
<td>Buchin (2009)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Buchin and Mulzer (2009, 2011)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Yang and Choi (2010)</td>
<td>incremental</td>
<td>CDT</td>
<td>2000 (s)</td>
</tr>
<tr>
<td></td>
<td>Ebeida, Mitchell, Davidson, Patney, Knupp, and Owens (2011)</td>
<td>-</td>
<td>CDT</td>
<td>8271560</td>
</tr>
<tr>
<td></td>
<td>Yang, Choi, and Jung (2011)</td>
<td>divide and conquer</td>
<td>Delaunay</td>
<td>3000</td>
</tr>
<tr>
<td></td>
<td>Löffler and Mulzer (2011, 2012)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Schrijvers, van Bommel, and Buchin (2013)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>4194304</td>
</tr>
<tr>
<td></td>
<td>Liu, Yan, and Lo (2013)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>5500000</td>
</tr>
<tr>
<td></td>
<td>Lo (2013)</td>
<td>incremental</td>
<td>Delaunay</td>
<td>100000000</td>
</tr>
</tbody>
</table>

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-
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 algo-
rithm 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 in-
serted 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 $O(n \log n)$ 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 $O(n \log n)$ in the worst case. However, according to Dwyer (1987), partitioning the mesh into squared polygons reduced it to $O(n \log \log n)$ 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 $O(n \log n)$ 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.

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

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).

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 $O(n \log n)$ 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 $O(n \log n)$, 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
### Table 4: The 15 linear-time behaviour algorithms found in the second period, their complexities and the conditions under which such complexities are reached.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggarwal, Guibas, Saxe, and Shor (1989)</td>
<td>Θ(n)</td>
<td>points form the vertices of a convex polygon</td>
</tr>
<tr>
<td>Dwyer (1989, 1991)</td>
<td></td>
<td>expected linear behaviour set of points quasi-uniformly distributed</td>
</tr>
<tr>
<td></td>
<td>worst case: O(S₀ n lg n), in which S₀ is the number of dual simplices in the result</td>
<td>if a balanced-tree implementation of priority queues is used</td>
</tr>
<tr>
<td></td>
<td>worst case: O(S₀ n)</td>
<td>if the use of buckets is abandoned on any point search that examines  \sqrt{n} buckets</td>
</tr>
<tr>
<td>Chew (1990)</td>
<td></td>
<td>expected linear time set of points in the plane such that the points taken in order form the vertices of a convex polygon</td>
</tr>
<tr>
<td></td>
<td>worst case:</td>
<td>e.g. considering points along one branch of a parabola</td>
</tr>
<tr>
<td></td>
<td>O(n²)</td>
<td></td>
</tr>
<tr>
<td>Sugihara and Iri (1992)</td>
<td>average case:</td>
<td>behaviour observed in the experimental tests</td>
</tr>
<tr>
<td></td>
<td>linear</td>
<td></td>
</tr>
<tr>
<td>Klein and Lingas (1992)</td>
<td></td>
<td>linear</td>
</tr>
<tr>
<td>Klein and Lingas (1993, 1996)</td>
<td>expected linear time</td>
<td>builds the bounded Voronoi diagram of a simple polygon</td>
</tr>
<tr>
<td>Tsai (1993)</td>
<td></td>
<td>randomly distributed points in the Euclidean plane</td>
</tr>
<tr>
<td></td>
<td>worst case:</td>
<td>analyzed by Larkin (1991)</td>
</tr>
<tr>
<td></td>
<td>O(n²)</td>
<td></td>
</tr>
<tr>
<td>Djidjev and Lingas (1995)</td>
<td></td>
<td>points in the plane in sorted order with respect to two perpendicular directions</td>
</tr>
<tr>
<td>Shewchuk (1996)</td>
<td>nearly linear in practice</td>
<td>for difficult inputs; but it may be slow for uniformly distributed point sets</td>
</tr>
<tr>
<td></td>
<td>worst case: O(n lg n)</td>
<td>regardless of the points distribution</td>
</tr>
<tr>
<td>Su and Drysdale (1997)</td>
<td>modified Fortune's algorithm: linear behaviour</td>
<td>points uniformly distributed</td>
</tr>
<tr>
<td>Su and Drysdale (1997)</td>
<td>incremental:</td>
<td>results in tests</td>
</tr>
<tr>
<td>Held (1998)</td>
<td>expected linear time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>worst cases:</td>
<td>for a generalization of convex areas</td>
</tr>
<tr>
<td></td>
<td>Θ(n lg n), and O(n² lg n)</td>
<td>analyzed by the author</td>
</tr>
<tr>
<td>Held (1998)</td>
<td>expected linear time</td>
<td>if merging two subsets is assumed to take time proportional to the number of the involved unfinished points</td>
</tr>
<tr>
<td></td>
<td>worst case:</td>
<td>analyzed by the authors</td>
</tr>
<tr>
<td></td>
<td>Θ(n lg n)</td>
<td></td>
</tr>
<tr>
<td>Lemaire and Moreau (2000)</td>
<td></td>
<td>wide variety of data in a 296 MHz Sun Ultra 30</td>
</tr>
<tr>
<td>Held (2001)</td>
<td>0.01 n log₂ n ms</td>
<td></td>
</tr>
</tbody>
</table>

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 Huhdanpaa (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.](image)

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 Huhdanpaa (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 $30^\circ$ to $120^\circ$. 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
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.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amenta, Choi, and Rote (2003)</td>
<td>nearly linear, worst cases: $O(n^2)$, and $O(n \lg n)$ in the &quot;realistic&quot; case</td>
<td>results obtained in tests verified by the authors</td>
</tr>
<tr>
<td>Liu and Snoeyink (2005)</td>
<td>tess3 was faster than QHull, CGAL, Hull and Pyramid</td>
<td>results obtained in tests</td>
</tr>
<tr>
<td>Buchin (2005)</td>
<td>expected linear time</td>
<td>1st algorithm: points distributed independently and uniformly in a bounded convex area</td>
</tr>
<tr>
<td>Buchin (2005)</td>
<td>worst case when $d &gt; 2$: $O\left(n^{\lceil d/2 \rceil}\right)$</td>
<td>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</td>
</tr>
<tr>
<td>Boissonnat, Devillers, and Hornus (2009)</td>
<td>worst case (2D): $O(n \lg n)$</td>
<td>by generating an “onion-like” layered simplicial subdivision of the convex hull, which is also used as a locating data-structure</td>
</tr>
<tr>
<td>Buchin (2009)</td>
<td>expected linear time</td>
<td>many random point distributions</td>
</tr>
<tr>
<td>Buchin and Mulzer (2009, 2011)</td>
<td>linear</td>
<td>for small integers, but the constant factor is high</td>
</tr>
<tr>
<td>Yang and Choi (2010)</td>
<td>linear</td>
<td>experimental results</td>
</tr>
<tr>
<td>Ebeida, Mitchell, Davidson, Patney, Knupp, and Owens (2011)</td>
<td>nearly linear in practice $O(n \lg n)$ expected time</td>
<td>given sample points prelocated in a squared uniform grid the $\lg n$ dependence is very mild</td>
</tr>
<tr>
<td>Yang, Choi, and Jung (2011)</td>
<td>linear behaviour $O(n \lg n)$ expected time</td>
<td>test results in the quasi-uniformly distributed points set observed in the experiments</td>
</tr>
<tr>
<td>Löfler and Mulzer (2011, 2012)</td>
<td>linear</td>
<td>on a pointer machine and for small integers</td>
</tr>
<tr>
<td>Schrijvers et al. (2012, 2013)</td>
<td>linear</td>
<td>on small integers</td>
</tr>
<tr>
<td>Liu, Yan, and Lo (2013)</td>
<td>it runs faster than the algorithms with the Hilbert curve order and with BRIO, and also faster than a random algorithm</td>
<td>especially for non-uniform point distributions over a wide range of benchmark examples</td>
</tr>
<tr>
<td>Lo (2013)</td>
<td>quasi-linear</td>
<td>for the triangulation of distribution sets with local characteristics similar to those of a uniform point distribution, such as line, circle and cluster</td>
</tr>
</tbody>
</table>

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


