Game and Media Technology Master thesis

ICA-5956439

Local Order Types

Author:
Huck Nuchelmans BSc

Supervisors:
prof. dr. M. J. van Kreveld
dr. M. Löffler

Presented for the degree of Master of Science at

UTRECHT UNIVERSITY
Department of Information and Computing Sciences

August 31, 2018

ABSTRACT

This thesis considers past research on the order type of point configurations and introduces a variation on this concept: the *local* order type. The regular order type is a mapping that assigns to each ordered triple of points the orientation of these three points: positive, neutral or negative. The local order type is a similar mapping but only considers a subset of all triples. In order to decide which triples are included in the order type, a triple selection method must be chosen. We present six different methods and some of their properties. For one of these variations, the Delaunay order type, we give algorithms to compare and enumerate them. For the extended Delaunay order type we present similar but incomplete algorithms, that can be completed by solving two open problems.

CONTENTS

1 INTRODUCTION

The order type of a point configuration is a mapping that assigns to each ordered triple of points an orientation: positive, negative or neutral. In the plane a triple is positive if the points appear in counterclockwise order, negative if they appear in clockwise order and neutral if all three points lie on the same line. Instead of the absolute position of points in the form of coordinates, the order type describes the relative position and combinatorial properties of points in a configuration. While order types are defined for configurations in any number of dimensions, most attention goes towards the order type of point sets in the plane.

For a fixed number of points an infinite number of different point sets exists, but only a finite number of order types. Every point set has exactly one order type and many properties are shared between point sets with the same order type. This has proven to be useful for solving mathematical problems. Some problems are solved for every possible point set by solving them for every order type. Order types can also be applied to content generation, for example the automatic generation of levels in a video game. Using a different order type as the base for every level guarantees that levels are combinatorially different. If you randomise the coordinates, the values will be different but you have no control over the actual structure of the point set. So while two point sets may be different in their coordinates, a human could look at them and consider them the same. Order types get rid of this problem by being rotation, scale and x-scale invariant. If you rotate a point set, scale it or even scale it along a single axis, the order type remains the same, since none of these operations change the orientation of triples.

Order types as we know them were introduced in 1983 by Goodman and Pollack's paper on multidimensional sorting [17]. Later various works on order types have appeared by Aichholzer et al., most importantly introducing an algorithm to enumerate all order types for a given point set size n. As a result of this the order type database with point set representations of all order types up to $n = 11$ [1, 5] was created.

Research in the enumeration of order types has been motivated by the search of the rectilinear crossing number [2, 5, 13]. The rectilinear crossing number for a given n is defined as the minimal number of crossings in a complete graph with n nodes, where the edges between nodes are straight line segments. It is a variation on the crossing number, where edges are not necessarily straight.

Order types helped push this research forward by presenting a way to get the number of crossings for every set of a given n. This uses the fact that the order types group the infinite number of different configurations—when it comes to coordinate representations—into a fininite number of classes: the order types of the configurations. Since the crossing properties are the same between all configurations within this class, the number of crossings only need to be counted for one representation of each order type. With the use of order types Aichholzer et al. found the rectilinear crossing numbers for $n = 11$ and $n = 12$ [2], and later for all n up to 17 [5].

The crossing properties of a point set also give the possible ways to triangulate this set. This way order types can help with solving many problems regarding triangulations. An example is the search for an efficient algorithm to determine the number of triangulations a point set can have [1].

Puzzle games can use point configurations as a part of their level generation. An example is a game where the player is presented with a straight-line drawing of a graph where some edges intersect. They are then tasked to untangle it by swapping vertices, until edges no longer intersect each other. For such games, level diversity is desirable [20]. Order types can play a role in ensuring this diversity, by taking a different order type as the starting point of every level. In order to get a realisation of every order type, an enumeration algorithm or database such as the one described in Section 2.4 is needed. Note that the described algorithm takes a lot of time, making it unsuitable for online uses in games, however it can be used to generate levels offline once, before distribution.

Order types have applications in the field of pattern recognition too as mentioned by Goodman and Pollack [17]. In some cases it makes sense to reduce an image to a point configuration. We can then go even further and encode it as the λ-matrix of that configuration. This representation has the benefit that it can easily be compared, as described in Section 2.3. Most importantly, the order type of a point set is rotation and scale invariant. These properties are often desirable in the field of pattern recognition.

Some of the earliest applications of order types were in the field of chemistry, where they are usually referred to as chirotopes [12]. Order types can be used to compare *stereoisomers*, which consist of the same number of atoms, but in different orientations. Order types are a way to distinguish between right-handed and left-handed compounds.

An order type includes all triples. For every pair of points it has information on where they lie relatively to each other and all other points. For many applications, including the ones that were just discussed this is exactly what we want.

But imagine a content generation system that is to produce a large embedded graph. Do we really care about all triples? If edges are only ever going to appear between points that lie close to each other, the relative position of a point at the other side of the set will most likely not result in a different graph.

In this situation, when operating on very large point configurations, a more loose variation of the order type could be useful, one where not every triple's orientation matters. Only a subset of all triples must be taken into consideration. Which triples these should be, depends on the application.

This thesis introduces the generic notion of a local order type, which takes into account a subset of all possible triples. We introduce different triple selection rules, resulting in six different *local* order type variations. The properties and complexity of these variations are studied and compared to the regular *global* order type. For global order types, algorithms exist to encode, compare and enumerate order types. We aim to give such algorithms for local order types in general and a selection of specialised algorithms for specific local order type variations.

Section 2 gives an overview of previous work on order types. Knowledge of this field is necessary to reason about local order types, as almost any research on global order types can also be applied to local order types. Section 3 defines the local order type and discusses the properties of six different local order type variations. In Section 4 we take a close-up look at a selection of variations, namely the Delaunay-based local order types. One of the highlights from this section is the enumeration algorithm for these order type variations.

2 PRELIMINARIES ON ORDER TYPES

2.1 Definitions

Goodman and Pollack (1983) [17] describe order types as a way to implement multidimensional geometric sorting. The order type of a point configuration gives the extreme points of the set, but also describes the relative position of all the other points. Sorting in a single dimension is fairly straightforward and only one order type exists in this case. Given a set of n points on a line, it is always possible to label these points $1, \ldots, n$ in such a way that i lies to the left of j for every $i \leq j$. This set always has two extremes—a minimum and a maximum—and all internal points lie in order. In higher dimensions a more complex order appears, as points can not always be labelled to result in the same order type. Working with order types becomes easier if point sets are assumed to be in general position. In this case this means that no three points can lie on the same line.

Definition 2.1. *The* order type *is a description of the ordering of points $\{P_1, P_2, \ldots P_n\}$ in $\mathbb{R}^d$. It assigns to every ordered multiple $i_1, i_2, \ldots i_{d+1}$ the orientation of the point multiple $P_{i_1} P_{i_2} \ldots P_{i_{d+1}}$. If the points are in general position, these orientations are limited to positive (+) and negative (−). Otherwise they can be neutral (0) too.*

Instead of *order type*, this concept is sometimes referred to as the *chirotope* or the *multiplex* [12], and it is a special case of the *n-ordered sets* [23].

Sorting in one dimension can be seen in two different ways. One can find for each point P_i the *number* $\lambda(i)$ of points to its right side, or the *set* $\Lambda(i)$ of points on that side. Knowing one of these is enough to find the other. The value of $\lambda(i)$ can be found by counting the elements in $\Lambda(i)$, while $\Lambda(i)$ can be found by looking at the λ-value for the other points: If $\lambda(i) > \lambda(j)$, we can conclude that P_j lies to the right of P_i, thus $\Lambda(i)$ must contain P_j.

In the one-dimensional case, sorting is accomplished by looking at pairs of points. In general the atoms of the order pattern of a point configuration in d-dimensional space consist of $d+1$ points each. This means that in the plane we consider the orientation of triples. Again this can be thought of in two different ways. For each pair of points P_i, P_j, one can find the *number* $\lambda(i, j)$ of points P_k lying to the left of the directed line through P_i and P_j (the triple $P_i P_j P_k$ is counterclockwise), or the *set* $\Lambda(i, j)$ of all points P_k to the left of $P_i P_j$ can be found. Again, one of these follows from the other as proven by Goodman and Pollack. Note that $\Lambda(i, j)$ and $\lambda(i, j)$ are undefined if $i = j$. An undefined result of Λ is represented by the Ω symbol and an undefinded result of λ by ω.

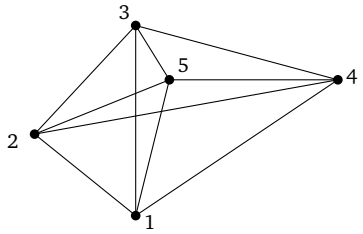
Definition 2.2 (Λ and λ)**.** *For a point configuration $\mathcal{C} = \{P_1, \ldots, P_n\}$ in $\mathbb{R}^d$, $\Lambda(i_1, \ldots, i_d)$ denotes the set of points on the positive side of the hyperplane affinely spanned by $P_{i_1}, \ldots, P_{i_d}$. $\lambda(i_1, \ldots, i_d)$ denotes the number of points on the positive side of that same hyperplane.*

If $P_1, \ldots, P_n$ are affinely independent:

$$\Lambda(i_1, \ldots, i_d) = \{P_j | P_{i_1} \ldots P_{i_d} P_j > 0\}$$
$$\lambda(i_1, \ldots, i_d) = |\Lambda(i_1, \ldots, i_d)|$$

Otherwise:

$$\Lambda(i_1, \ldots, i_d) = \Omega$$
$$\lambda(i_1, \ldots, i_d) = \omega$$



$$\lambda_{ij} = \begin{pmatrix} \omega & 0 & 1 & 3 & 2 \\ 3 & \omega & 0 & 2 & 1 \\ 2 & 3 & \omega & 0 & 1 \\ 0 & 1 & 3 & \omega & 2 \\ 1 & 2 & 2 & 1 & \omega \end{pmatrix}$$

Figure 2.1: $C = \{P_1, \ldots, P_5\}$ with its λ-matrix.

Λ and λ both fully encode the order type of a point configuration.

To find the order type of a point configuration algorithmically, a hyperplane rotation around a *face-flag* is performed. A face-flag is a sequence of faces on a polytope of increasing dimensionality. In algorithms regarding order types, this polytope is the convex hull of the point set. In d-dimensional space, a face-flag consists of a sequence of $d-1$ faces where every next face's dimensionality increases by one, and every face is a bounding face of the next one. This means that in the plane a face-flag is merely a point and the hyperplane rotating through it is a line. In three dimensions it is the sequence of a point σ_0 and a line segment σ_1, of which one endpoint corresponds to σ_0. Around this face-flag a plane is rotated.

Definition 2.3. *A* face-flag *of the polytope* Π *in* $\mathbb{R}^d$ *is a sequence* $\sigma_0, \ldots, \sigma_{d-2}$ *of faces of* Π*, where* σ_i *is a face of* σ_{i+1} *and has dimensionality* i*.*

The order type of a configuration of n points in the plane can be represented as a λ-matrix with dimensions $n \times n$. Figure 2.1 shows an example of such a matrix, along with the point configuration it belongs to. Take for example the result of $\lambda(1,5)$. We can read from the matrix that $\lambda(1,5) = 2$. If we look at the point set, this corresponds with the number of points to the left of the directed line P_1P_5, these points being P_2 and P_3.

Multiple λ-matrices can encode the same order type. This depends on the labelling of points. Additionally, if all triples are ordered in the opposite direction, the order type is usually considered to be the same. Depending on the labelling, different λ-matrices can represent the same point set, and thus the same order type. This is inconvenient when comparing order types, as all different labellings would have to be tested. A solution to this problem will be discussed in Section 2.3.

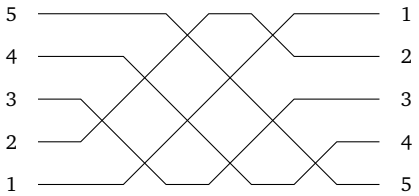


Figure 2.2: Example of a wiring diagram.

In some algorithms, sets of points are imagined in the projective plane instead of the Euclidean plane. This can make operations computionally easier. This concept is used in the process of enumerating order types [1]. In this algorithm, order types are grouped together into projective classes. The order types in the same class have the same realisability. For this purpose we need to distinguish between regular Euclidean order types and projective order types.

Definition 2.4. *A* projective order type *is the order type of a point configuration in the projective plane. Euclidean order types can be grouped together into classes of projective order types.*

Because point configurations are equivalent to line arrangements through duality, line arrangements are also studied as a part of research related to order types. More specifically, pseudoline arrangements are used in the process of enumerating order types [1]. This procedure will be described in detail in the following sections. Pseudoline arrangements are a more general version of regular line arrangements. Pseudolines do not need to be straight, but they preserve topological and combinatorial properties of straight lines [16]. A pseudoline does not cross itself and in a pseudoline arrangement, every pair intersects exactly once.

Definition 2.5. *A* pseudoline *is a simple closed curve whose removal does not disconnect the real projective plane.*

Definition 2.6. *A* pseudoline arrangement *is a set of labelled pseudolines, every pair crossing exactly once.*

A common representation of pseudoline arrangements is the wiring diagram. Lines are all drawn horizontally except in the locations where they cross and switch positions.

Not every arrangement of pseudolines is realisable as an arrangement of straight lines. In this case we say the arrangement is not stretchable. A λ-matrix can be constructed from each pseudoline arrangement. How this is done will be explained in Section 2.4. Not all of these λ-matrices correspond to a realisable point configuration. Order types that are realisible are simply referred to as order types. The complete set of order

types, including unrealisible ones are called pseudo order types.

Definition 2.7. *A* pseudo order type *is the order type associated with a λ-function that follows from a pseudo-line arrangement. The collection of pseudo order types is a superset of the collection of order types. Pseudo order types that are not an order type can not be realised as a point set.*

Order types are a special case of oriented matroids [24]. The oriented matroid is a widely used representation that generalises many different geometric objects, such as point configurations and arrangements of lines (and hyperplanes in general), but also convex polytopes and directed graphs. Oriented matroids map elements in a set to a direction. In the case of order types in the plane, these elements are triples, and the direction can be clockwise or counterclockwise.

2.2 Geometric sorting

The algorithm for geometric sorting in the plane is introduced in [17] along with a general algorithm for any number of dimensions. These algorithms take a labelled set of points and return the λ-function associated with it.

Intuitively the algorithm in the plane comes down to the following. For every ordered pair of points P_i and P_j a ray from P_i through P_j is cast and then rotated counterclockwise 180 degrees. During the rotation, the ray may encounter a number of points. As discussed before it is unnecessary to know which points they are as long as we know how many they are. The number of points encountered by the ray should be the result of $\lambda(i, j)$.

See Figure 2.3 for an illustration. As the ray rotates, it encounters P_4, P_5 and P_7, totalling a number of three points which lie on the left of directed line P_1P_2. This allows us to say $\lambda(1, 2) = 3$. To get the complete λ-matrix this is repeated for every pair of points.

Pseudo code for the GEOMETRICSORT2D algorithm can be found in Appendix A. The algorithm fills the λ matrix one row at a time. For every P_i it first walks through all points again and filters out the current point and other points that might have the same coordinates. All remaining points are translated in order to get their position relative to P_i. Additionally each point is mirrored in P_i and stored as a copy. The original and mirrored points are then grouped into rays and sorted into counterclockwise order as seen from P_i. Finally, for every j, we fill out $\lambda(i, j)$ by summing up the number of original points in each ray between the ray containing P_j and the ray containing P_j's mirrored copy.

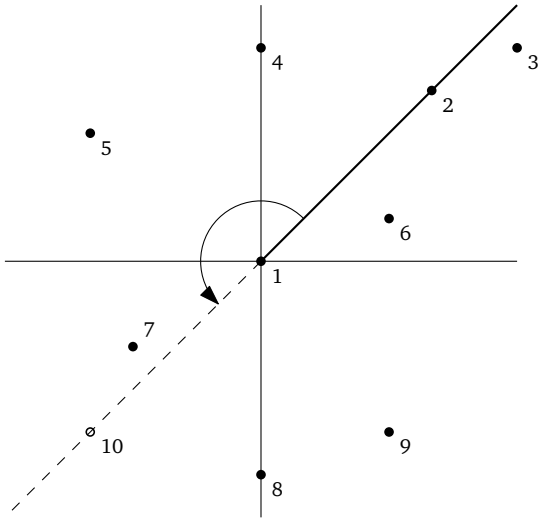


Figure 2.3: The ray originating from P_1 going through P_2 and rotating to P_{10}, where P_{10} is the result of P_2 getting mirrored in P_1.

The translation of all other points takes linear time. The sorting can be done in $O(n \log n)$ time. If the computations of each sum is done by reusing the sum for the previous j and correcting the λ value, it can be done in linear time. This results in an execution time of $O(n \log n)$ for each i, resulting in a total time of $O(n^2 \log n)$.

The radial ordering of points can be optimised by performing this task in the dual plane. The ordering can be derived from the dual line arrangement in linear time, resulting in a total execution time of $O(n^2)$.

An algorithm exists to find the order type of point configurations in higher dimensions. This algorithm is described by Goodman and Pollack [17] and takes $O(n^d(d! + \log n)/(d-1)!)$ time. Given a large value of n and considering that the number of dimensions is constant, this simplifies to $O(n^d \log n)$ time. This is in correspondence with the execution of the algorithm for points in the plane.

2.3 Comparing randomly numbered configurations

For many applications, it is required to compare the order types of two separate point configurations. Of course it is possible to compute the λ-matrix of both sets and compare the result, but it is very unlikely that the labelling of points is the same in both sets.

In order to solve this problem, Goodman and Pollack [17] introduce the concept of a canonical ordering. A canonical ordering of a point set in the plane exists for every point P_i on the convex hull. The ordering is found by casting a ray from this point, pointing outwards, away

from the hull, and then rotating it counterclockwise. Points are added to the canonical ordering as they are encountered by the ray. When two points lie on the same ray, they are handled in the order of increasing distance to P_i. In higher dimensions the canonical ordering is found by sweeping a hyperplane around a face-flag.

The algorithm to determine a canonical ordering of a point set in the plane has some similarity to the geometric sorting algorithm. To avoid unnecessary repetition, CANONICALORDERING2D reuses parts of GEOMETRICSORT2D. The algorithm expects a point configuration and a point on its convex hull as input. This point will be used as the starting point of the canonical ordering. It walks through all points and sorts all points into rays as seen from the starting point. It then walks through these rays in order to find the points in counterclockwise order. For the complete pseudo code refer to the appendix.

Similarly to the GEOMETRICSORT2D algorithm, the sorting of rays can be done in $O(n \log n)$ time. All other parts take linear time, resulting in an overall execution time of $O(n \log n)$. Goodman and Pollack [17] give an algorithm for an arbitrary number of dimensions with the same time-complexity.

In order to confirm that two labelled point configurations $\mathscr{C} = \{P_1, \ldots, P_n\}$ and $\mathscr{C}' = \{P'_1, \ldots, P'_n\}$ have the same order type, a mapping must be found that matches each point in $\mathscr{C}$ to a point in $\mathscr{C}'$. If every possible mapping must be tried, this results in $n!$ possibilities. Luckily there are some restrictions on which points can be matched. An extreme point can only correspond to another extreme point. Additionally, if two points are matched, their canonical orderings should agree on the labelling of the other points. This means we only need to try as many possibilites as there are points on the convex hull of the point sets, giving a maximum of n possibilities.

For configuration $\mathscr{C}$ we pick a random point on the convex hull and determine the canonical ordering induced by it. We now try to match this point to all the points on the convex hull of $\mathscr{C}'$. This is done by permuting the λ-matrix of $\mathscr{C}$ and comparing it to the λ-matrix of $\mathscr{C}'$. The algorithm returns a set of mappings that result in equal λ-matrices. If the set is empty, the point configurations do not have the same order type. Pseudo code for COMPARECONFIGURATIONS2D can be found in the appendix.

We know the execution time needed for GEOMETRICSORT2D and CANONICALORDERING2D. Finding the convex hull of a point set can be done in $O(n \log n)$ time [10]. At most n comparisons need to be done and every comparison takes $O(n^2)$ time, resulting in a total execution time of $O(n^3)$.

2.4 Enumeration of order types

An important project regarding order types is the database of order types in the plane by Aichholzer and Krasser [6]. The database contains realisations of all order types for points sets with 3 to 10 vertices. To create such a database, a method is needed to enumerate order types. This process was initially described by Aichholzer et al. in 2002 [1]. Additional work has been done to improve this method. Examples include the generalisation to higher dimensions [15] and complete point extension [5], adding order types for point sets of size 11 to the database.

The number of order types quickly increases with increasing n as seen in Table 2.1 and it becomes harder to find realisations of order types. Certain properties only hold for an n-point set if it holds for at least one of its $(n-1)$-point sets. This theory can be used to find new solutions to the rectilinear crossing number problem. A method called complete point extension is introduced [5] to generate all order types of size $n+1$ that contain a given n-point order type as a sub-order type.

n	Number of order types
4	2
5	3
6	16
7	135
8	3 315
9	158 817
10	14 309 547
11	2 334 512 907

Table 2.1: Number of different order types for point sets of size n.

The problem of the rectilinear crossing number of point sets has been a driving force in making improvements to the database of order types. Order types have helped to find the rectilinear crossing number for all n up to 17.

Aichholzer's order type database and enumeration algorithms are limited to point configurations in general position in the plane. An alternative method exists without these restrictions. Finschi and Fukuda [15] present an approach to generate order types, not just in the plane, but in any number of dimensions. For this they exploit the similarities of order types and oriented matroids.

The enumeration of order types as described by Aichholzer et al. [1] generates point set realisations for all order types for $n = 4, \ldots, 10$ and can be split into three

steps. First the complete set of pseudo order types must be generated. This means all realisable order types are also contained once in this set. Wiring diagrams are used for this purpose. Then these pseudo order types are grouped together in classes of projective order types. Order types that belong to the same class are guaranteed to have the same realisability. The last step is to find which order types are realisable and find a realisable point set if this is the case. The algorithm is limited to point configurations in general position in the plane.

In order to generate a candidate list, a collection of λ-matrices is generated. We learned previously that these matrices depend on the labelling of the point set and multiple λ-matrices can represent the same order type. For this reason, the points are assumed to be in *natural ordering*. This is defined as the order that results in the λ-matrix that is lexicographically the smallest, when read line by line. This means P_1 will always be a point on the convex hull and all other points will appear in clockwise order as seen from P_1.

The collection of λ-matrices is created by generating a collection of wiring diagrams. The crossing order of the pseudolines $g_1, \ldots, g_n$ determine the values in the matrix, using the following assumptions for wiring diagrams. No line intersections lie above g_1 and every other line crosses g_1 from top to bottom. The value of $\lambda(i,j)$ where $1 \leq i < j \leq n$ can be read from the arrangement by counting the number of lines that have the crossing of g_i and g_j above them, excluding g_i and g_j. Note that the crossing order of g_1 will always be $2, 3, \ldots, n$, making it possible to omit it from the arrangement. Because the point configuration is in general position the remaining fields of the matrix can be filled using the fact that $\lambda(i,j) + \lambda(j,i) = n-2$. See Figure 2.4 for an illustration of a wiring diagram and the order type that can be extracted from it. Note that if wire 1 would be omitted from the diagram,the remainder of the λ-matrix could still be read from it.

Euclidean order types belong to the same projective class if they can be *rotated* to each other. Rotating can be done around any extreme point in the order type and reverses the orientation of all triples containing that point. After a rotation the order type must be minimised again to be lexicographically the smallest representation. Because order types in the same projective class share realisability, realisability only needs to be tested once for every class.

What remains is the final and computationally the most complex step. The problem of determining realisability of an order type is NP-hard. Aichholzer et al. use a collection of heuristics to accomplish it. The number of projective classes that must be realisable were already known from literature. A scanning process is started using an insertion method, building n-point realisations

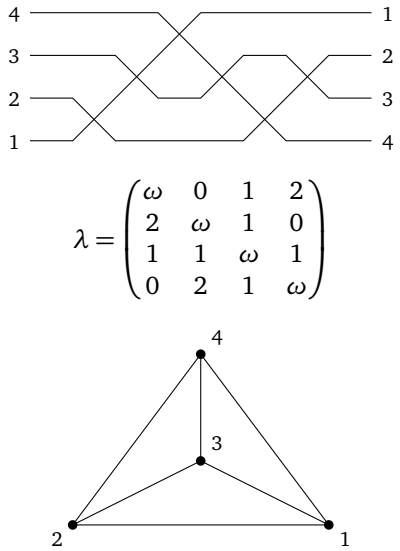


Figure 2.4: A wiring diagram, λ-matrix, and point realisation, all belonging to the same order type.

by adding a point to $(n-1)$-point realisations. This raises the problem that the specific geometric realisation of a set of size $n-1$ can limit the number of possible order types that can be realised by adding a point to it. For this reason the process was repeated, but with small random perturbations applied to the $(n-1)$-point sets. These perturbations change the geometry but preserve the order type.

Once a realisation of one of the order types in a projective class is found, the class can be marked as realisable. If the number of realisable classes has reached the expected number, the final step can be performed. If the number is not yet reached but the realisability of some classes remains undetermined, a second method is used to try realisability. This more exhaustive method aims to build a realisation from the ground up and was only used in a few cases and only for $n = 10$.

The final step is to find realisations for all remaining order types in realisable projective classes. These are derived from the existing realisations in that class, by mapping the points to a sphere, applying rotations and projecting back to the plane.

2.5 Geodesic order types

Geodesic order types are a variation on order types where the point configuration is embedded inside a simple polygon [4]. It is based on the geodesic distance between two points, which is the length of the shortest path between two points without moving outside the simple polygon boundary. In determining the order type of a point configuration, this boundary influences the radial ordering of points and thus the direction of triples.

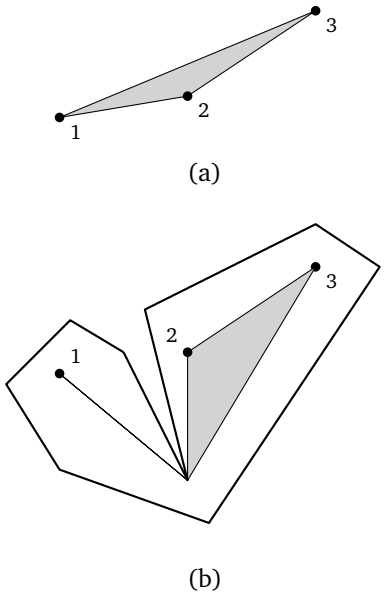


Figure 2.5: The difference in radial ordering between an unconstrained and geodesic setting.

In Figure 2.5 this is clearly visible. In the unconstrained situation (a) the counterclockwise radial ordering as seen from P_1 is P_2, P_3. When the boundary is introduced in (b), this changes to P_3, P_2 as the polygon pushes and bends the edges between the points. This also causes the direction of the triple $P_1P_2P_3$ to change its orientation from counterclockwise to clockwise.

Part of the order type in an unconstrained setting is the convex hull of the point set. The equivalent of the convex hull in a geodesic setting is the geodesic hull. The geodesic hull of point set S is the region that contains the shortest paths in geodesic setting between all pairs of points in S.

By changing the polygon boundary, the order type of a point set S can be changed. From this follows that the subset of points that make up the geodesic hull can change as well. Aichholzer et al. [4] prove that for any subset $\mathscr{B}$ of at least four points a polygon P can be found such that $\mathscr{B}$ forms the geodesic hull of S induced by P.

2.6 Cross-sections of line arrangements

The cross-section of a line arrangement $\mathscr{L}$ in general position in $\mathbb{R}^3$ with a plane not parallel with any line in $\mathscr{L}$ results in a point configuration. If we consider this plane to be horizontal and sweep it vertically with a constant speed, this models a point configuration where the points move in the plane with a constant speed. Research has been done on the number of order types these sets of moving points can have [8]. The number of different order types cross-sections of one line arrangement can have is found to have a tight bound of $O(n^9)$.

2.7 Radial orderings

It is possible to reconstruct the order type of a point set from partial information on the position of points. Aichholzer et al. [3] prove that the order type can be determined when given the radial orderings of other points as seen from each point in the set. This holds when the convex hull of the set contains at least four points. Otherwise, there can be $n-1$ candidates, that could all be the order type of this point set.

The reconstruction algorithm runs in polynomial time and it works by repeatedly solving the problem for subsets of five points.

An improved version of the algorithm was later introduced [7] that runs in linear time. First the convex hull of the point set is constructed from the radial orderings. Then the remaining orientations are found. The nature of this problem makes it impossible to solve it in less than linear time, making this algorithm optimal.

2.8 Storage of order types and point set representations

This section contains some notes on the storage that is necessary for algorithms and databases regarding order types. This concerns both the encoding of order types themselves—for example in the form of a λ-matrix—and the storage required for coordinate representations of order type realisations.

For the storage of coordinate representations a grid of integer values can be used. To be able to differentiate between different order types, a certain grid size is required. When more points are added to the configuration, one can imagine that it becomes hard to maintain the general position of the set and prevent any points from being colinear. Goodman et al. found that a grid to store coordinate representations of order types of size n requires exponential storage [18].

Aichholzer et al. [1] note that they aim to present *nice* representations of order types. Their coordinate representations follow the following rules:

- No x or y coordinate is allowed to be the same. This makes it easier to sort along one of the axes and prevents the presence of parallel lines as the result of a duality transformation.
- There is a minimum distance of 4 between each pair of points.

- The configurations are in *very* general position. Besides being non-colinear, there must be a distance of 1 between every line through two points and any other point.

- No four points lie on a common circle.

Despite these rules, they manage to store all realisations for $n \leq 8$ with coordinates as one byte integers. Realisations for $n = 9$ and $n = 10$ require two byte integers.

We are not only interested in storing point representations but also in storing the order type itself. Since the λ-matrix is a representation of an order type, storing this matrix is enough, requiring storage of $O(n^d)$ values for an n-point set in d dimensions, and thus $O(n^2)$ in the plane. However, as n increases, the values that are being stored also increase in size, requiring more bits for each value. Felsner [14] found a more efficient way of storing pseudoline arrangements, where the number of values is still $O(n^2)$ but every value is a single bit. Because a pseudo order type follows directly from an arrangement of pseudolines and every order type is also a pseudo order type, the same method can be applied to the storage of order types.

A wiring diagram can be encoded as the list of orderings in which each pseudoline crosses the other lines. Each ordering $\sigma_i = (\sigma_1^i, \ldots, \sigma_{n-1}^i)$ is a permutation of the numbers $1, \ldots, n$, excluding i. The complete encoding is then contained in the vector $(\sigma_i, \ldots, \sigma_n)$.

For the example in Figure 2.4 this would result in the vector

$$((2,3,4),(1,4,3),(1,4,2),(1,3,2)).$$

Felsner states that the storage of this arrangement can be reduced by defining

$$\tau_j^i = \begin{cases} 1 & \text{if } \sigma_j^i > i \\ 0 & \text{otherwise} \end{cases}$$

$$\tau_i = (\tau_1^i, \ldots, \tau_{n-1}^i).$$

Then the vector $(\tau_i, \ldots, \tau_n)$ can also serve as a complete encoding of the arrangement. In our case this would reduce to

$$((1,1,1),(0,1,1),(0,1,0),(0,0,0)).$$

Note that a one also corresponds to an upwards crossing and a zero to a downwards crossing in the wiring diagram. We now have a representation of an order type that has the same number of values as the λ-matrix, but where every value is either one or zero, requiring a single bit per value.

3 LOCAL ORDER TYPE DEFINITIONS

For the order type of a point configuration the orientation of every triple of points matters. We introduce the local order type, a variation on the order type that only takes into account a subset of all triples. We define the local order type for an arbitrary number of dimensions but in the following we will only consider the case of points in the plane.

Definition 3.1. *The* local order type *is a description of the ordering of points $\{P_1, P_2 \ldots, P_n\}$ in $\mathbb{R}^d$. To a subset of all ordered tuples $i_1, i_2, \ldots i_{d+1}$ it assigns the orientation of the point tuples $P_{i_1} P_{i_2} \ldots P_{i_{d+1}}$: positive, negative or neutral. Which tuples are included in the subset is determined by the specified tuple selection method.*

The aim of the local order type is to create classes of order types that more closely resemble the human perception of similarity of point configurations. Note that the similarity of point sets depends on how that point set is used. For example, is it used to create a graph or is it presented as just a point set?

3.1 Triple selection

There are countless ways to determine which triples must be included. We introduce the following specifications of local order types:

1. The *Delaunay order type* includes a triple only if the circle through the three points contains no other points.

2. The *extended Delaunay order type* includes a triple if at least one point is connected to the other two through a line segment of the Delaunay triangulation of the point configuration.

3. The *k-nearest order type*. Every point wants to form triples with its k nearest points. For each point, the order type includes every combination of that point and any two of its k nearest points.

4. The *maximum distance order type*. Every point wants to form a triple with all other points that are within a given distance. For each point, the order type includes every combination of that point and any two of its candidate points.

5a. The *maximum angle order type*. We consider the

triangle formed by the three points. A triple is included if the angle opposite to the longest side of the triangle does not exceed a maximum angle.

5b. The *nonobtuse order type* includes a triple only if the three points form a nonobtuse triangle: every angle of the triangle is at most ninety degrees.

6. The *circumscribed circle order type* includes a triple if the circle through the three points has a radius that does not exceed a given value.

Some relations between these different order types are immediately clear. The nonobtuse order type is a case of the maximum angle order type where the maximum is set to ninety degrees. The circumscribed circle order type combines the maximum angle and maximum distance order type, as both the distance and the angle between points influence its triple selection. The k-nearest and maximum distance order type have a similar process of creating triples: every point nominates candidate points. The extended Delaunay order type contains at least the triples that the regular Delaunay order type includes.

It is possible to have two point sets with a different global order type, but the same local order type, but it also works the other way around. Because of the use of a triple selection procedure, point sets with the same global order type can have a different local order type by including different subsets of triples. Figure 3.1 and 3.2 show examples of both cases.

For the k-nearest order type points are considered in general position when the distance between each pair of points is unique in addition to no three points being colinear. The Delaunay-based order types expect point configurations where no four points lie on the same empty circle.

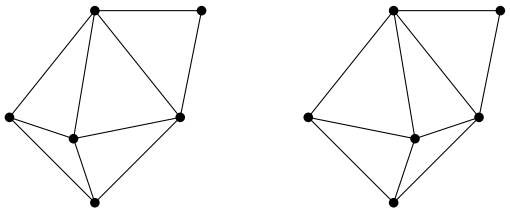


Figure 3.1: Two point sets and their Delaunay triangulation. The point sets have a different global order type but the same Delaunay order type.

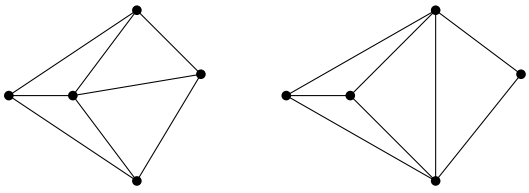


Figure 3.2: Two point sets and their Delaunay triangulation. The point sets have the same global order type but a different Delaunay order type.

3.2 Basic properties of local order type variations

We start by looking at basic properties of the defined local order types.

- First we want to know if the selection of triples takes into account only the relations between the three considered points or if it also takes into account other points in the configuration. We will call this property **context dependence**. If a local order type is context independent, this means a repetition of the same triple in a different location will always be included as well. This is not true for context dependent order types as the surrounding points need to be taken into account as well.

- Additionally, the local order type can be **scale invariant**. This is the case if the order type stays the same when the entire set is scaled up or down. A variation on scale invariance is x-scale invariance. In this case the order type remains the same even if the set is scaled along one axis. The global order type of a point set is x-scale invariant and, naturally following from this, scale invariant. We exclude x-scale invariance from our list, because none of the presented local order types are x-scale invariant.

- Do all points **participate** in the order type? A point participates if it is part of at least one triple that is included by the order type variation.

- Is the point selection for triple forming **symmetrical**? This is only relevant for variations where each point selects a set of other points that it wants to form triples with. These are the extended Delaunay, k-nearest and the maximum distance order types. The selection process is considered symmetrical if the following holds: point P_i selects P_j as a candidate, if and only if P_j selects P_i as a candidate.

- Finally, we want to know if a point configuration with a different number of **extreme points** always results in a different order type. For the global order type this is the case, but not necessarily for all local order type variations, as not all triples are considered.

Question	1	2	3	4	5	6
Context dependence	yes	yes	yes	no	no	no
Scale invariance	yes	yes	yes	no	yes	no
Participation	yes	yes	yes	no	no	no
Symmetry	yes	yes	no	yes	yes	yes
Extreme points	yes	yes	no	no	no	no

Table 3.1: Basic properties of local order type variations. Refer to Section 3.1 for the order type descriptions.

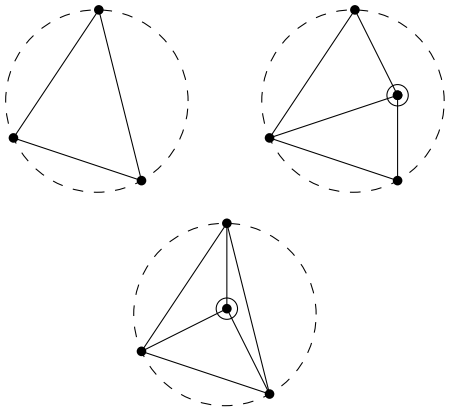


Figure 3.3: Nearby points affect the inclusion of triples in the Delaunay triangulation.

Table 3.1 contains the answers to these questions for all local order types that were defined in the previous section.

In distance- and angle-based order types the selection of triples only depends on the relative position of the points that would make up the triple. In Delaunay based and k-nearest order types, the addition or removal of other points can change the participation of a triple. For this reason repeated triples are included in the former, but not necessarily in the latter.

If a point is placed inside the circle through the points of an included triple, that triple will no longer be included by the Delaunay order type. Figure 3.3 shows two examples of such points. Both are added inside the circle, but one is outside the triangle and the other inside.

Figure 3.4 shows how the addition of a point can change the selection of the k nearest points of point P_0. If we consider the circle with P_0 as its center, going through its k-th nearest point, any point added inside this circle becomes one of the k nearest points. The point that was previously the k-th nearest will no longer be selected.

The only local order types that are not scale invariant are the ones that have explicitely encoded distances. These are the maximum distance and the circumscribed circle order types. For the maximum distance order type, a value is chosen for the maximum distance. For the circumscribed circle order type the diameter of the circle must be fixed. We can construct a point set where a pair of points has exactly the maximum distance between them. If this point set is now scaled up, their distance will exceed the maximum and they will no longer be included as a pair. Similarly, we can construct a triple with a circumscribed circle that has exactly the maximum radius. If this triple is scaled up, so is the radius of the circle and the triple will no longer qualify.

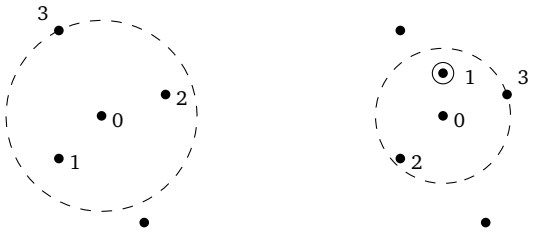


Figure 3.4: Adding a point changes the selection of k-nearest points. Selected points are labelled in ascending order regarding their distance to P_0 with $k = 3$.

In Delaunay-based order types and the k-nearest order type, all points contribute to at least one triple. For the k-nearest order type every point selects k points and thus participates in at least $\binom{k}{2}$ triples. Distance- and angle-based order types do not have any guarantee on participation. If a point is too far away from all other points, or if all angles are too big, it can be left out. It is even possible to construct a point set such that not a single triple is included.

The selection process of the maximum distance order type is symmetrical, because it is based on the absolute distance between two points. It is impossible for the two points to disagree on this. For the k-nearest order type, the selection is influenced by other nearby points, making it asymmetrical. An example of this can be seen in Figure 3.5. The extended Delaunay order type can also be seen as having a symmetrical selection process, because a point selects another point if the two share an edge in the Delaunay graph. All other variations, while not having a clear per point selection strategy, could also be considered symmetrical.

Only for Delaunay-based order types there exists a guarantee that a different number of points on the convex

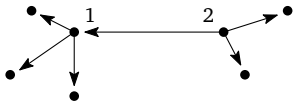


Figure 3.5: Point p_2 selects p_1, but not the other way around.

hull of the point set results in a different order type. The number of points on the convex hull determines the number of triangles in the Delaunay triangulation and thus the number of triples in the Delaunay order type. Two order types with a different number of triples can never be the same. For the extended Delaunay order type, we will show in Section 4.2 that the convex hull can be reconstructed from the order type, even if the coordinates of the points are not known. Since the convex hull follows from the order type, a different number of points on the hull must also result in a different order type.

For the other order types this is not the case. For distance- and angle-based order types, we can construct a point set such that zero triples are selected. This can be done for any number of points on the convex hull. More on the number of selected triples for each order type variation can be read in the next section.

We cover another property of local order types. For some order type variations, we can make a statement about the points in the area of a given triple. In particular, we will consider the points that lie inside the triangle formed by the points of a triple. The distance between a point inside a triple and the line through two of the points forming the triple is smaller than the distance between this line and the third point. A desirable property of a local order type could be that these points also form a triple with each pair.

For the Delauny order type, such points do not exist by definition. For the extended Delaunay order type, the only guarantee is that at least one of the points inside the triple has an edge connecting to the middle point of the triple. The middle point is the point that has edges to both of the other points forming the triple. We cannot make a generic statement on all points inside the triple.

For the k-nearest order type we can state that points inside a triangle must be a candidate point of at least one of the points that forms the triangle. This is the point that formed the triangle with two of its candidate points. Any point inside the triangle is closer to this point than the farthest of its two candidates, so it must be a candidate point as well. Similarly, for the maximum distance order type this means that a point inside a triangle forms triples with all possible pairs of the points that form the triangle.

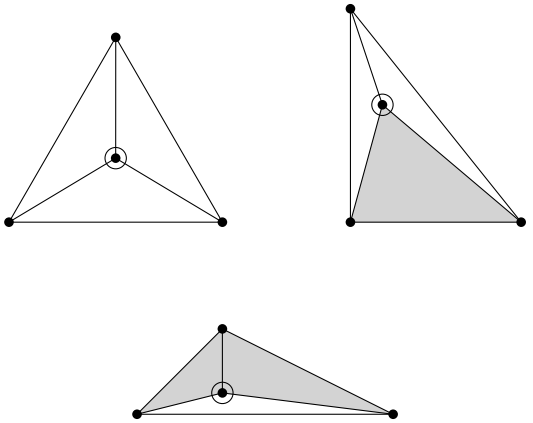


Figure 3.6: Examples of triangles and added points that result in 0, 1 and 2 *better* triangles. Filled triangles are better triangles.

When selecting a point inside a triangle to form a new triangle with, the angle in that point will be larger than the angle in the point that is being replaced. Depending on the point that is being replaced this can make the new triangle either more or less suitable for the maximum angle order type. If it replaces the point that already had the largest angle, the new triangle is guaranteed to have a larger maximum angle. Figure 3.6 shows that there are cases where zero, one or two of the newly formed triangles are more suitable. No case exists where all three triangles are better. This follows from the fact that at least one of the triangles is formed by replacing the angle that was already the largest, making it even larger.

For the circumscribed circle order type, we can reason about the radius of the circumscribed circle when taking a point inside the triangle. The radius r of the circumscribed circle of a triangle ABC with angles α, β, γ and opposite edges a, b, c can be found using the following formula.

$$2r = \frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}$$

When forming a triangle with a point inside the triangle, the length of one of the sides a, b, c remains the same, but the angle α, β or γ opposite to that side is replaced by a larger angle. The angles are always somewhere between 0 and π radians. $\sin\alpha$ returns the highest value at $\alpha = \pi/2$ and becomes smaller as α gets bigger or smaller. This means that the circumscribed circle gets larger as α moves away from $\pi/2$ radians and gets smaller as it moves towards it.

3.3 Complexity of local order type variations

We aim to give upper and lower bounds for the number of **triples** that an order type can include. We also look at the complexity of the **graph** that is formed by drawing an edge between every pair of points that participates in the same triple. The values given are bounds on the number of edges. Finally, we give bounds on the number of order types that can belong to the same local order type. We will call these local order types order type **classes** and list the maximum known class size.

For some local order type variations we need to give bounds for both the best and worst case. The best case is defined as the minimal case and the worst case as the maximal case. The results are shown in Table 3.2, where n is the number of points in the point configuration and h represents the number of points on the convex hull. For the class size, we define a function $f(x)$ that returns the number of global order types that exist for point configurations of size x.

The exact number of triangles in a Delaunay triangulation, or any point set triangulation, is known to be $2n-2-h$ [10]. The number of edges is equal to $3n-3-h$, as follows from the Euler characteristic. It can be derived from the number of triangles given that each triangle has three edges and every edge is shared by two triangles, except for those edges that make up the convex hull.

For the extended Delaunay order type it is possible to have a point with a linear degree, resulting in a quadratic number of triples. An example is the situation where $n-1$ points lie on a circle and the remaining point lies in the middle of this circle with an edge to every other point. The middle point then initiates triples with all possible combinations of other points. For the order type graph we now add edges between all these combinations too, resulting in the complete graph. The upper bound on the number of triples is also quadratic, since the Delaunay graph has $O(n)$ edges, resulting in $O(n^2)$ pairs of edges. Triples in the extended Delaunay order type are formed by pairs of edges, resulting in $O(n^2)$ triples.

For the k-nearest order type all of the n points select exactly k points. An edge will appear in the graph for each of these k points and $\binom{k}{2}$ triples can be formed. Overall this results in $\Theta(nk)$ edges and $\Theta(nk^2)$ triples. Note that if $k = n$ (or $n-1$ to be exact) this will result in the complete graph and the inclusion of all possible triples. This is reflected by the resulting bounds of $\Theta(n^3)$ and $\Theta(n^2)$.

For all distance- and angle-based order type variations, the minimal case occurs when all distances or angles exceed the allowed value. For distance based order types such a configuration can be obtained by scaling up any configuration until all distances are too large. For angle-based order types an example is a point set where all points lie (almost) on the same line. This can be accomplished by scaling any configuration along a single axis. The maximal case occurs when all points form triples with every other point.

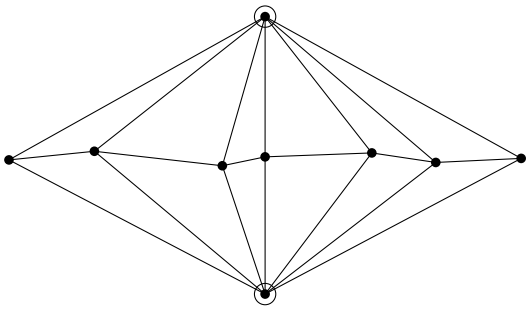


Figure 3.7: Two points are added to a point configuration such that all triangles of the Delaunay triangulation include one of these points.

For the Delaunay order type, we have a construction that shows a large number of different global order types that can map to the same local order type. Take a point configuration of n points with any global order type and scale it down along one axis until the points almost lie on a line. Now add two points, one on either side of this line. Refer to Figure 3.7.

It is always possible to add these two points, such that the Delaunay triangulation of the $n+2$ point configuration has the same topology regardless of the order types of the initial n points. This is the case because the point set can always be scaled down further along one axis, increasing the diameters of the circumscribed circles of any three points until one of the two added points lies inside this circle.

For a point configuration of n points with a fixed Delaunay order type, the number of different global order types that can be formed is the same as the number of global order types that can be formed by $n-2$ points.

This construction cannot be reused for extended Delaunay order type as is, since triples consisting of three points of the original set are still included, possibly resulting in a smaller class size. It could be possible that the $n-2$ points in the middle can be transformed such that they always lie in the same pattern, but this is still an open problem. For now all we can say is that the class size of the extended Delaunay order type cannot be greater than the class size of the Delaunay order type.

To determine a possible class size for the k-nearest order

Metric	1	2	3	4, 5, 6
Triples	$2n-2-h$	$O(n^2)$	$\Theta(nk^2)$	0, $\Theta(n^3)$
Graph	$3n-3-h$	$O(n^2)$	$\Theta(nk)$	0, $\Theta(n^2)$
Class size	$f(n-2)$	?	$f(n/(k+1))$	$f(n)$

Table 3.2: Complexity of local order type variations. Refer to Section 3.1 for the order type descriptions.

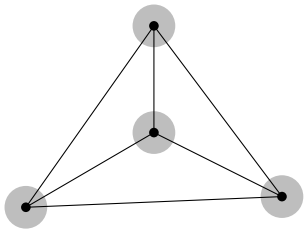


Figure 3.8: Points are replaced by clusters of points.

type, we can also create a construction. This works by taking any point set and replacing each point by a cluster of $k+1$ points. The distance between any two clusters must exceed the diameter of each cluster to make sure that no point selects a point from another cluster as one of its k nearest neighbours. Additionally the clusters must be in general position. Any line through two points from different clusters should never cross a third cluster.

The clusters can be arranged in any way without changing the local order type. The number of different global order types that can be formed depends on the number of clusters. For a point configuration of n points, the number of different order types that can be formed is the same as the number of order types that can be formed by $n/(k+1)$ points.

For distance- and angle-based local order types, point sets for all regular order types exists that all have the same local order type. This occurs for the local order type that includes no triples at all, for which point sets are described above. Any point set and thus any order type can be transformed into such a set by scaling along both axes or a single axis. Both of these transformations maintain the global order type.

3.4 Robustness of local order type variations

Finally, we will assess the robustness of local order types, by looking at the number of changes to the order type that happen as a result of the addition or removal of a point to or from the point configuration. In order to count the number of changes, a clear definition of a change is necessary. We define it as the addition or deletion of an included triple. Inclusion of triples is the only aspect of the local order type that can change as the result of insertion or deletion of points. The orientation of existing triples always remains the same. Table 3.3 shows the results for all of our local order type variations.

The minimal case for the Delaunay and extended Delaunay order type occurs when a point is added without causing any edge flips. For points that are added outside the convex hull it is possible to only form one new triangle. For a point inside the convex hull, one triangle is removed and three new ones are added. These are two cases where a constant number of changes occur.

The maximal case happens when linearly many edge flips are performed. For example by adding a point inside a point configuration where every point lies on the convex hull. The new point can be placed such that all existing triangles are removed and new ones are added. Since the number of triangles in Delaunay triangulation is linear, the change is also linear. For the extended Delaunay order type this results in a quadratic change to the number of triples.

When a point is added to a set, for the k-nearest order type it selects exactly k candidate points to form $\binom{k}{2}$ new triples with. All these triples are guaranteed to be new, since they contain the point that was previously not in the set. In addition to this, a number of existing points can swap out one of their candidate points for the new point. If this happens, k triples are deleted and k new ones are added. The number of points for which this can happen is $O(k)$, as follows from Lemma 3.1, keeping the number of changes at $\Theta(k^2)$.

Lemma 3.1. *A point can be one of the k-nearest points of at most $6k$ other points.*

Proof. Consider the nearest point graph, a directed graph where there is a connection from every point to its k nearest points. The outdegree of each point is exactly k. We are looking for the maximum indegree of a point, which is equal to the number of points that select this point as one of their k nearest neighbours.

Now imagine a point p that has an indegree $> 6k$. There will always be a wedge with p as its apex with an angle < 60 degrees that contains $> k$ points that selected p as one if their neighbours. Let q be the farthest of these points. All other points ($\geq k$) lie closer to q than p does. Since p is one of the nearest points of q, these $\geq k$

Action	1	2	3	4, 5, 6
Add/remove	$O(1)$, $O(n)$	$O(1)$, $O(n^2)$	$\Theta(k^2)$	0, $\Theta(n^2)$

Table 3.3: Robustness of local order type variations. Refer to Section 3.1 for the order type descriptions.

points must be as well. This results in a contradiction where q has marked $> k$ points as its nearest points. This means the indegree cannot exceed $6k$. ■

Note that a deleted triple can still be in the order type because another point still forms that triple with its candidate points and an added triple could previously be formed by the new point and its candidate points.

For distance- and angle-based local order types, adding or removing a point only creates or destroys triples that contain the point in question. As discussed in the previous section, it is possible for a point to form triples with all other pairs of points, while it is also possible that the point does not participate in any triples. In the maximal case, adding a point creates $\Theta(n^2)$ new points. In the minimal case, no new triples are formed.

3.5 Local order type encoding

For order types, both global and local, there are countless ways to represent them. It is a hard requirement for an encoding to be specific, in that it represents exactly one order type. Ideally an encoding would also be unique and easy to compare. If the representation is unique, the represented order type can only be encoded by that single representation. Having a unique representation makes comparing the order type of two point sets a trivial task.

It is always possible to represent an order type by simply listing the coordinates of a point set representation, requiring linear storage in the number of points. A point set only has one order type, so it is a correct representation of the order type. However, it contains much more information beyond the combinatorial properties of the point set, making it a non-unique representation. This encoding takes up little storage but requires much computation in order to be compared.

The other extreme approach is to list the orientation of all ordered triples, requiring $O(n^3)$ storage for a point set of n points. Equivalent to storing a list of triples is storing the Λ function, containing the set of points on the positive side of each ordered pair of points.

For the global order type the λ matrix can be derived from this representation, requiring only quadratic storage. As described in Section 2.1 the λ matrix only encodes one order type because not only does the number of points on the positive side of a pair follow from the set, but also the other way around. This only works because the global order type includes all triples, so for local order types the λ function is no longer a correct representation.

For local order types, the most straight-forward approach is to list all ordered triples and mark them clockwise, counterclockwise, neutral—if non-general position is allowed—or omitted. All triples occur three times this way. If we only consider point configurations in general position, some storage can be saved by listing every included triple once in some clockwise order. Triples that are not included by the local order type are not stored.

A more familiar way to do roughly the same thing is through an altered version of the Λ function. For every ordered pair P_i, P_j of points, a point is included in $\Lambda(i, j)$ if it forms a triple with P_i and P_j and lies to the left of the directed line P_iP_j. Note that this is again only sufficient for point sets in general position. If a triple $P_iP_jP_k$ is omitted or neutral, both scenarios result in P_k not being included in $\Lambda(P_i, P_j)$.

3.6 Comparing the local order type of point sets

In order to compare local order types of configurations, the lists of triples or Λ functions must be compared. Just like with global order types, the problem arises that the labelling of points can be different. Sadly we cannot reuse the properties of canonical orderings of points since for local order types there is no guarantee that a point on the convex hull in configuration $\mathscr{C}$ always maps to a point on the convex hull in configuration $\mathscr{C}'$.

Any optimisations we can make, depend on the triple selection method that is used. If the triple selection method is unknown, many different mappings of points must be considered. Not all possible mappings need to be checked though, as a point can only be matched to a point that occurs in the same number of triples.

In Section 3.2 we saw that the convex hull does have a direct impact on the Delaunay-based order types. This allows us to create optimised comparison algorithms for these variations. More on these algorithms will appear in the next section.

4 DELAUNAY-BASED ORDER TYPES

The previous section contained some information on local order types in general but also highlighted the difference between triple selection methods. This section focuses on the two Delaunay-based local order types and aims to present encodings and algorithms tailored to these variations. This includes algorithms for comparing the order types of two point sets and a way to enumerate all order types for any set of a given size n.

We selected the Delaunay-based order types because as seen in the previous section they are scale invariant and context dependent. These are properties that would be suitable for the application of content generation. Additionally, Delaunay triangulations have been researched extensively in the past.

In the following we will only consider point sets in general position. In addition to no three points being colinear, we require that no four points lie on the same circle. This ensures a point set only has a single Delaunay triangulation.

4.1 The Delaunay order type and graph isomorphism

For the Delaunay order type, the topology of the Delaunay triangulation that resulted in that order type can serve as an encoding for the order type. This follows from the fact that the triples follow directly from the triangles of the triangulation and that the triangulation can be reconstructed from the list of triples. The triples provide a list of triangles that can only be glued together in one way. Knowing that all inner faces must be triangles, it is even sufficient if only the edges are known, except for the case when the configuration has a convex hull of size three. In this case the outer face is also a triangle, making it impossible to recognise it from a list of edges.

For configurations with more than three points on the convex hull, this means that their Delaunay order types can be compared by creating Delaunay triangulations for both configurations and checking them for edge isomorphism. For configurations with exactly three points on the convex hull, this is insufficient. Additionally, given the mapping that results from the isomorphism check, we must verify that the outer face is formed using the same three vertices.

Checking for isomorphism of two triangulations T and T', both of n points, can be done by taking any edge on the convex hull in T and fitting it on every edge on the convex hull in set T'. Matching one pair of edges uniquely maps the remaining edges. In linear time we check if the mapping is complete. Doing this for each edge in the convex hull of set T' means that it requires $O(n^2)$ time to check isomorphism of two triangulations.

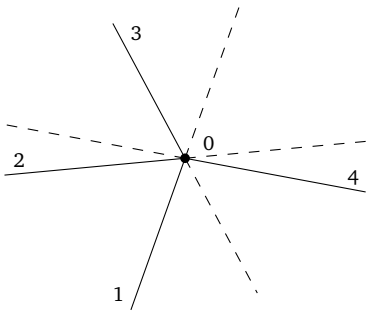


Figure 4.1: One vertex of a Delaunay triangulation with its incident edges and mirrored versions of those edges.

Since the Delaunay triangulation of a point configuration can also be computed in $O(n^2)$ time [21], the overall time required to compare the Delaunay order type of two configurations is $O(n^2)$.

4.2 Encoding and comparing the extended Delaunay order type

For the extended Delaunay order type, an alternative encoding exists too. Every vertex P_i of the Delaunay graph of n points forms triples with the adjacent vertices. The orientation of these triples depends on where P_k lies relative to the directed line P_jP_i. This information can be stored for triples with P_i at their center by creating a radial ordering of points. Let P_{n+k} be the result of mirroring P_k in P_i. The radial ordering must include all adjacent vertices plus the mirrored versions of these points.

Take the example in Figure 4.1. The points are not shown as their distance is not important, so the labels appear at the edges incident to each point. Dashed edges go towards mirrored points. Let $n = 5$. A radial ordering for P_0 is $1, 2, 9, 3, 6, 7, 4, 8$. To determine the direction of a triple $P_jP_0P_k$, start at j in the radial ordering and walk through it in clockwise order. If k appears before $n + j$ the triple is counterclockwise. If it appears after $n + j$, the triple is in clockwise orientation. Creating a radial ordering for every P_i fully encodes the orientation of every triple in the extended Delaunay order type and requires $O(n)$ storage. This is the case because for every point we store a list of its neighbours and the sum of the degrees of all points is $O(n)$. Whether it is also a unique encoding remains to be answered.

This encoding can only be used for comparing the extended Delaunay order type of two configurations if each extended Delaunay order type can only appear on a single unlabeled Delaunay triangulation topology. In this case the Delaunay triangulations of two configu-

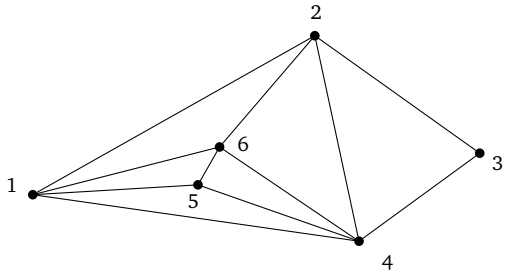


Figure 4.2: A Delaunay triangulation where P_1, P_2, P_4 and P_6 consistently appear in triples of the extended Delaunay order type with each other and P_5 even though there is no edge between P_2 and P_5.

rations must first be checked for isomorphism. If this check succeeds, proceed by comparing the radial orderings of matched points. No counterexample is known and there are steps that suggest that the following conjecture holds but no definite proof has been found.

Conjecture 4.1. *The topology of the Delaunay triangulation of a point configuration follows directly from its extended Delaunay order type.*

One of the steps towards proving this conjecture is that the convex hull of the triangulation follows directly from the extended Delaunay order type. If $\Lambda(i,j) = \emptyset$ then edge P_iP_j is part of the clockwise convex hull or it does not exist at all. Given that consecutive edges on the convex hull share a point and that three consecutive points on the hull form a clockwise triple, only one possible convex hull arises from these pairs. This could lead to an incremental approach where one point on the hull is removed along with its edges and triples and the new convex hull is detected. However this does not work in its current state as the removal leaves it unclear how the point was connected to the inner points and the remaining graph is not guaranteed to be a valid Delaunay triangulation with a convex hull anymore.

Another approach can involve finding neighbouring points of each point through the list of triples or Λ function. For a point P_i and every other point P_j compute $M(i,j) = \Lambda(i,j) \cup \Lambda(j,i)$. This set contains every point P_k for which a triple exists consisting of P_i, P_j and P_k.

By comparing these sets, we can find the subset of points that consistently appear in triples with each other and P_i, suggesting that these could be the neighbours of P_i. A counterexample exist, shown in Figure 4.2, where P_2 appears in this subset for $i = 5$ even though there is no edge between P_2 and P_5. However it is still possible to find that there is no edge between P_2 and P_5 by looking at $\Lambda(3,2)$ which is guaranteed to contain every point that shares an edge with P_2 or P_3 since P_2P_3 is an edge on the convex hull. P_5 is not in $\Lambda(3,2)$ guaranteeing that edge P_2P_5 does not exist.

Additionally, taking $i = 2$ for the triangulation in Figure 4.2 and comparing $M(2,j)$ for each $j \neq 2$, results in two possible solutions. P_1, P_4 and P_6 appear in all other M values, but P_3 in all others except $M(2,5)$ and P_5 appears in all others except $M(2,3)$. This results in two sets of points that consistently appear in triples with each other and P_2: $\{P_1, P_3, P_4, P_6\}$ and $\{P_1, P_4, P_5, P_6\}$. Only the former corresponds to the neighbours of P_2.

So far these steps are insufficient to prove Conjecture 4.1, but they seem like a step in the right direction. As long as the conjecture is unproven, the slower generic algorithm for comparing local order types can be used. If the conjecture holds, we can make the following analysis of the runtime of the algorithm to compare the extended Delaunay order type of two point sets. First, the Delaunay order types of both sets are compared by checking their Delaunay triangulations for isomorphism in $O(n^2)$ time. If they do not match, it is guaranteed the extended Delaunay order types also do not match. If they do, we continue by checking the radial orderings for each point. For each point, the equivalent point is found in the other configuration, the neighbours are mirrored and the radial orderings are compared, all in linear time. Doing this for every point takes $O(n^2)$ time, which is also the runtime of all steps combined.

4.3 Enumeration of the Delaunay order type

In general there are two possible approaches to enumerate a set of order types for a given point set size n. For the global order types we saw that the possible permutations of the λ-matrix were generated, before checking if this resulted in realisable point sets. For local order types this is not as feasible, as the encodings discussed in the previous sections are less suitable for direct enumeration. Instead we will present a method that uses point sets as a starting point and then finds an encoding of the local order type based on each point set. Theoretically this method works for any value of n, but we are limited by the size of the order type database, which only contains order type realisations for $n \leq 11$.

For the Delaunay-based order types, it makes sense to generate all possible Delaunay triangulations. This is possible with the use of the order type database, triangulation enumeration and Delaunay realisability algorithms. The following steps must be followed to enumerate all Delaunay order types for n points for a given value of n. Let h be the number of points in the triangulation that lie on the convex hull.

1. Use the order type database to get a point set for every order type with n points. For each of these point sets, enumerate all possible triangulations.

2. When triangulations are equivalent through isomorphism, discard all except one of them. For configurations with $h = 3$, it is necessary to also check if the outer face matches, as described in Section 4.1.

3. Check the Delaunay realisability of the remaining triangulations and discard those that are not realisable as a Delaunay triangulation.

Taking the triangles of the remaining triangulation as the triples of their Delaunay order type results in a different Delaunay order type for every triangulation and the set includes all possible Delaunay order types. The inclusion of all Delaunay order types is proven by the fact that the ways to triangulate a point set follow from its global order type and we enumerated all triangulations for each order type. While these three steps are sufficient to enumerate the order types, note that they do not yield realising point sets. The resulting configurations are not yet accurate realisations of their associated Delaunay order types. To accomplish this, their points must be repositioned such that the triangulation becomes a valid Delaunay triangulation.

For the first step, we need to enumerate all possible triangulations of a given point configuration. Let k be the number of possible triangulations of a configuration, then all these triangulations can be found in $O(nk)$ time using a reverse search algorithm [9]. A more efficient version of this algorithm was later found by using a search tree, to solve the problem in $O(k \log\log n)$ time [11].

For step two we must efficiently check isomorphism of two triangulations. This can be done using the algorithm described earlier. Note that it is only necessary to check for isomorphism of triangulations that have the same h, as the number of edges and triangles fully depends on this number. This allows us to apply the entire algorithm in batches by first splitting up the order type database into collections of point sets with the same h and running the algorithm separately on each batch.

Step three requires us to check the Delaunay realisability of a triangulation graph. Determining realisability can be done by solving a linear system in polynomial time [19]. Actually finding the solution requires extra steps. For certain classes of triangulations this still takes polynomial time [22]. A Delaunay realisation must satisfy the following five constraints.

(C1) For each inner face, the sum of the three angle variables associated with it equals 180 degrees.

(C2) For each inner vertex, the sum of all angle variables associated with it equals 360 degrees.

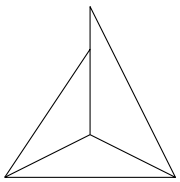


Figure 4.3: A result of constraints (C1)–(C5) that is not a valid Delaunay triangulation.

(C3) For each outer vertex, the sum of all angle variables associated with it is less than or equal to 180 degrees.

(C4) For each inner edge, the sum of both its *facing angle variables* is less than or equal to 180 degrees. A facing angle of an edge is the angle between the other two edges incident to the same face.

(C5) Each angle variable is positive.

Using linear programming to find valid angle variables is not sufficient yet to find a proper Delaunay realisation because the triangles might not be able to be glued together correctly as seen in Figure 4.3. But if a solution is found, a realisation is guaranteed to exist.

A sixth constraint (C6) exists, which is defined as follows. Let v be an inner vertex of the triangulation and $w_1, w_2 \ldots, w_k$ its neighbours in counterclockwise order. The *counterclockwise angles* $\phi_1, \phi_2, \ldots \phi_k$ around v are defined as

$$(v, w_1, w_2), (v, w_2, w_3), \ldots, (v, w_k, w_1)$$

and the *clockwise angles* $\theta_1, \theta_2, \ldots, \theta_k$ around v as

$$(v, w_2, w_1), (v, w_3, w_2), \ldots, (v, w_1, w_k).$$

Now if we define for a given vertex v

$$F(v) = \frac{\sin(\phi_1)\sin(\phi_2)\cdots\sin(\phi_k)}{\sin(\theta_1)\sin(\theta_2)\cdots\sin(\theta_k)}$$

then as defined by [19] the sixth constraint is

(C6) For each inner vertex v, $F(v) = 1$.

Solving system (C1)–(C6) is sufficient to create a Delaunay realisation of a triangulation. However no efficient algorithm is known to solve this system. An alternative algorithm is presented in [22], which reuses the system (C1)–(C5). Note again that this system is already sufficient for determining whether a Delaunay realisation exists, without yielding a valid realisation.

4.4 Enumeration of the extended Delaunay order type

For the enumeration of extended Delaunay order types we might be able to use a similar method. This method depends on Conjecture 4.1. In this case the second step must be extended so it also compares the radial orderings of adjacent vertices and mirrored adjacent vertices. This difference arises from the fact that the extended Delaunay order type not only depends on the topology of the triangulation but also on the orientation of points that are connected to the same point, without taking part in the same triangle. Because the order type database is used as a starting point, we do have the guarantee that all possible orientations of such triples are included.

The first and third steps are identical to the ones described in the previous section. The second step uses the comparison algorithm described in Section 4.2. For the extended order type we cannot use the algorithm described in [22] to create a point set representation. This algorithm does not respect the global order type and thus the extended triples do not necessarily have the same orientation. This makes the algorithm insufficient.

It is also possible that the result of step three might contain some false positives. When we check Delaunay realisability of a triangulation it does not mean the realisation has the same order type. This could mean that some of the enumerated order types are actually not realisable. The algorithm only works if it can be proven that whenever a triangulation is Delaunay realisable, there also exists a realisation that maintains the radial orderings of the triangulations. So far we have found no proof in favour or against this statement. If this is not the case, instead of checking for Delaunay realisability, the step must be replaced and specifically test if the triangulation can be realised as a Delaunay triangulation without changing the orientation of triples that would be selected by the extended Delaunay order type. Such an algorithm is thus far unknown.

4.5 Extending the Delaunay order type

The extended Delaunay order type is an extension of the Delaunay order type by containing at least the triples of the Delaunay order type plus additional triples. This means that multiple extended Delaunay order types belong to the same Delaunay order type.

One step towards creating point set realisations for extended Delaunay order types would be to take a Delaunay triangulation as the starting point and move a vertex to create all possible orientations of the extended triples through that vertex.

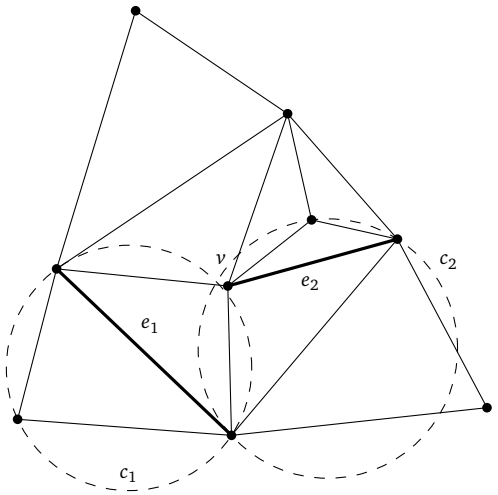


Figure 4.4: Partial construction to determine the safe area of v. For edges e_1 and e_2, respectively circles c_1 and c_2 denote areas that v must stay outside or inside of.

Consider a point configuration. It has a Delaunay order type and an extended Delaunay order type. It is possible to move a single point around and keep the Delaunay order type the same while changing the extended Delaunay order type, as long as this alteration does not change the Delaunay triangulation by making any edge invalid.

The vertex v has an area around it where it can move freely without altering the Delaunay triangulation. This area can be constructed using a set of circles through nearby points. Edge flips must be prevented for every triangle v is a part of. These flips can happen in two ways, resulting in two rules for constructing the safe area.

1. For each edge (v, w) incident to v, if the two faces it is associated with together form a convex shape, a circle must be drawn through the points on the corners of this quadrangle, excluding v. Vertex v needs to stay inside this circle to prevent the edge from flipping.

2. For each edge (w_1, w_2) of which v is a facing angle, if the two faces it is associated with together form a convex shape, draw the triangle through w_1, w_2 and the other facing angle of the edge. Vertex v must stay outside this circle to prevent the edge from flipping.

Refer to Figure 4.4 for an example of both rules. Figure 4.5 shows the end result of applying both rules.

In Figure 4.5 we can already see that v is not allowed to create all possible orientations for all of its extended triples. There are two triples that will never change their

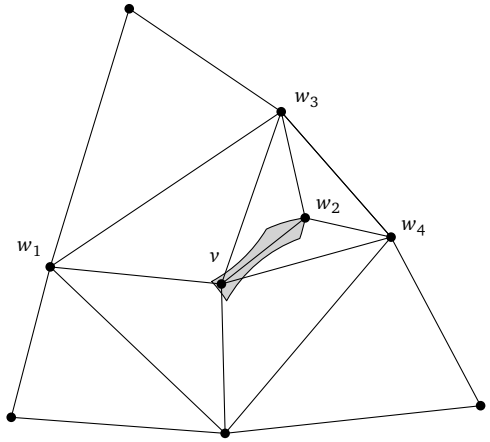


Figure 4.5: Vertex v in a Delaunay triangulation and its safe area.

orientation. Triple w_1vw_2 will always be counterclockwise. It might still be possible to change its orientation by moving other vertices. Triple w_3vw_4 is guaranteed to always be counterclockwise, because it is impossible to change the orientation of $w_3w_2w_4$ without changing the triangulation and w_3vw_4 must always match that orientation.

This method of finding the area in which a point can move freely, could be a part of an alternative enumeration algorithm, may the one described in the previous section be proven to be insufficient.

5 DISCUSSION

We defined the local order type and found a suitable encoding in the form of a modified version of the Λ function. Several triple selection methods and their properties were presented. Of these the Delaunay-based variations showed favourable characteristics. We discussed the possibilities to encode, compare and enumerate these order type variations.

We presented a collection of properties of six different triple selection methods, which served as a first step towards selecting a suitable method. Keep in mind that while some variations may exhibit properties that seem favourable, the final choice will always depend on the application. For the Delaunay order type, we successfully found algorithms for comparing and enumerating the different ones for a given number of points. Past research on global order types proved to be useful as we were able to use the original order type database in the enumeration process of Delaunay order types.

We presented a possible encoding and enumeration algorithm for the extended Delaunay order types, but there are still two open problems that must be solved before we can conclude if these algorithms are valid. This includes Conjecture 4.1 and the question whether we can find a Delaunay realisation of a triangulation that leaves the extended Delaunay order type of the underlying point configuration unchanged.

So far we have not looked at how the local order types hold up in different applications. We have shown through small examples that point sets with the same global order type can have a different local order type and vice versa, but we have not performed any research into the human perception of point sets and how well local order types represent the differences humans spot between point sets.

Since the algorithms for the extended Delaunay order type are incomplete, further research in this area is necessary. Conjecture 4.1 must be proven or shown to be false and based on this the presented algorithm must be finished or discarded. In addition to that all presented algorithms are as of yet unimplemented. The steps we took for Delaunay-based order types can also be repeated for the remaining triple selection methods, possibly resulting in algorithms for these cases.

We presented the theory around local order types, but its applications must still be explored and compared to human perception. This can also give a better insight into the possible applications of each individual triple selection method.

REFERENCES

[1] Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. “Enumerating order types for small point sets with applications”. In: *Order* 19.3 (2002), pp. 265–281.

[2] Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. “On the crossing number of complete graphs”. In: *Proceedings of the eighteenth annual symposium on Computational geometry*. ACM. 2002, pp. 19–24.

[3] Oswin Aichholzer, Jean Cardinal, Vincent Kusters, Stefan Langerman, and Pavel Valtr. “Reconstructing Point Set Order Typesfrom Radial Orderings”. In: *International Symposium on Algorithms and Computation*. Springer. 2014, pp. 15–26.

[4] Oswin Aichholzer, Matias Korman, Alexander Pilz, and Birgit Vogtenhuber. “Geodesic order types”. In: *Algorithmica* 70.1 (2014), pp. 112–128.

[5] Oswin Aichholzer and Hannes Krasser. “Abstract order type extension and new results on the rectilinear crossing number”. In: *Computational Geometry* 36.1 (2007), pp. 2–15.

[6] Oswin Aichholzer and Hannes Krasser. “The point set order type data base: A collection of

applications and results." In: *CCCG*. Vol. 1. 2001, pp. 17–20.

[7] Oswin Aichholzer, Vincent Kusters, Wolfgang Mulzer, Alexander Pilz, and Manuel Wettstein. "An optimal algorithm for reconstructing point set order types from radial orderings". In: *International Symposium on Algorithms and Computation*. Springer. 2015, pp. 505–516.

[8] Oswin Aichholzer, Ruy Fabila Monroy, Ferran Hurtado, Pablo Pérez-Lantero, Andres J Ruiz-Vargas, Jorge Urrutia, and Birgit Vogtenhuber. "Order types and cross-sections of line arrangements in $\mathbb{R}^3$." In: *CCCG*. 2014.

[9] David Avis and Komei Fukuda. "Reverse search for enumeration". In: *Discrete Applied Mathematics* 65.1-3 (1996), pp. 21–46.

[10] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry*. Springer, 2008.

[11] Sergei Bespamyatnikh. "An efficient algorithm for enumeration of triangulations". In: *Computational Geometry* 23.3 (2002), pp. 271–279.

[12] A Dreiding and Karl Wirth. "The multiplex. A classification of finite ordered point sets in oriented d-dimensional space, Math". In: *Chemistry* 8 (1980), pp. 341–352.

[13] Paul Erdös and Richard K Guy. "Crossing number problems". In: *The American Mathematical Monthly* 80.1 (1973), pp. 52–58.

[14] Stefan Felsner. "On the number of arrangements of pseudolines". In: *Proceedings of the twelfth annual Symposium on Computational Geometry*. ACM. 1996, pp. 30–37.

[15] Lukas Finschi and Komei Fukuda. "Combinatorial generation of small point configurations and hyperplane arrangements". In: *Discrete and Computational Geometry–The Goodman-Pollack Festschrift* 25 (2003), pp. 425–440.

[16] Jacob E Goodman. "Pseudoline arrangements". In: *Handbook of Discrete and Computational Geometry*. CRC Press, Inc. 1997, pp. 83–109.

[17] Jacob E Goodman and Richard Pollack. "Multidimensional sorting". In: *SIAM Journal on Computing* 12.3 (1983), pp. 484–507.

[18] Jacob E Goodman, Richard Pollack, and Bernd Sturmfels. "Coordinate representation of order types requires exponential storage". In: *Proceedings of the twenty-first annual ACM Symposium on Theory of Computing*. ACM. 1989, pp. 405–410.

[19] Tetsuya Hiroshima, Yuichiro Miyamoto, and Kokichi Sugihara. "Another proof of polynomial-time recognizability of Delaunay graphs". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 83.4 (2000), pp. 627–638.

[20] Rutger Kraaijer, Marc van Kreveld, Wouter Meulemans, and André van Renssen. "Geometry and generation of a new graph planarity game". In: *Proceedings of the IEEE Conference on Computational Intelligence and Games*. 2018.

[21] Der-Tsai Lee and Bruce J Schachter. "Two algorithms for constructing a Delaunay triangulation". In: *International Journal of Computer & Information Sciences* 9.3 (1980), pp. 219–242.

[22] Kevin M Lillis and Sriram V Pemmaraju. "On the efficiency of a local iterative algorithm to compute Delaunay realizations". In: *International Workshop on Experimental and Efficient Algorithms*. Springer. 2008, pp. 69–86.

[23] L Novoa. "On n-ordered sets and order completeness". In: *Pacific Journal of Mathematics* 15.4 (1965), pp. 1337–1345.

[24] Jürgen Richter-Gebert and Günter M Ziegler. "Oriented matroids". In: *Handbook of Discrete and Computational Geometry*. CRC Press, Inc., 1997, pp. 111–132.

Appendices

A ALGORITHMS

Algorithm 1: GEOMETRICSORT2D

Input. $(P_i = (x_i, y_i)), 1 \le i \le n$.

Output. $(\lambda(i,j)), 1 \le i \le n, 1 \le j \le n$.

1. **for** $i = 1$ **to** n **do**
2. **for** $j = 1$ **to** n **do**
3. $u_j \leftarrow x_j - x_i$; $v_j \leftarrow y_j - y_i$.
4. **if** $(u_j, v_j) = (0,0)$ **then**
5. $\lambda(i,j) \leftarrow \omega$
6. **else**
7. Call j good.
8. $u_{n+j} \leftarrow -u_j$;$v_{n+j} \leftarrow -v_j$.
9. $m_j \leftarrow m_{n+j} \leftarrow v_j/u_j$ **if** $u_j \neq 0$
10. **end if**
11. **end for**
12. Sort the indices $\{j|j \text{ is good}\} \cup \{n+j|j \text{ is good}\}$ into subsets (rays) in counterclockwise order, as follows:
    a) first those for which $u_j > 0$, using m_j as key;
    b) next those for which $u_j = 0$ and $v_j > 0$;
    c) next those for which $u_j < 0$, using m_j as key;
    d) finally those for which $u_j = 0$ and $v_j = 0$; Let the indices be represented as

$$J_{11}, \ldots, J_{1s_1}, \cdots, J_{r1}, \ldots J_r s_r$$

    where $J_{k1}, \ldots J_{ks_k}$ belong to the same ray.
    (Note: if the point set is in general position, this results in $s_k = 1$ and every subset will contain only one point.)
13. **for** $k = 1$ **to** r **do**
14. $n_k \leftarrow |\{m|1 \le m \le s_k, J_m \le n\}|$
15. **end for**
16. **for all** good $j = 1, \ldots, n$ **do**
17. (Note: $k(j)$ is the number of the subset that holds j.)
18. **if** $k(j+n) > k(j)$ **then**
19. $\lambda(i,j) \leftarrow \sum_{k=k(j)+1}^{k(j+n)-1} n_k$
20. **else**
21. $\lambda(i,j) \leftarrow \sum_{k=k(j)+1}^{r} n_k + \sum_{k=1}^{k=k(j)-1} n_k$
22. **end if**
23. **end for**
24. **end for**
25. **return** λ.

Algorithm 2: CANONICALORDERING2D

Input. $(P_i = (x_i, y_i)), 1 \leq i \leq n$ and $j \in \{1, \ldots, n\}$ and P_h, a point on the convex hull of $\{P_1, \ldots, P_n\}$.

Output. $\pi(1), \ldots, \pi(n)$, the canonical ordering associated with P_h.

1. Perform step 2 to 12 of GEOMETRICSORT2D with $i \leftarrow h$. This results in a sorted list of all points around P_h. If a subset of rays contains multiple points, these must be sorted by their distance to P_h. Let $J_i, \ldots, J_n$ be a flat list of indices.
2. Find the point P_i that should come first in the canonical ordering. This is the neighbouring point of P_i on the convex hull that the sweep will meet first.
3. $\pi(1) \leftarrow h$.
4. **for** $j \leftarrow 1$ **to** $n-1$ **do**
5. (Note: $l(i)$ is the index of i in J.)
6. **if** $l(i) + j \leq n$ **then**
7. $\pi(j+1) \leftarrow J_{l(i)+j}$
8. **else**
9. $\pi(j+1) \leftarrow J_{l(i)+j-n}$
10. **end if**
11. **end for**
12. **return** π.

Algorithm 3: COMPARECONFIGURATIONS2D

Input. $\mathscr{C} = (P_i(x_i, y_i)), 1 \leq i \leq n$ and $\mathscr{C}' = (P_i'(x_i', y_i')), 1 \leq i \leq n'$.

Output. $\{\pi \in S_n | \mathscr{C}^{(\pi)} \sim \mathscr{C}'\}$.

1. $\lambda_{\mathscr{C}} \leftarrow$ GEOMETRICSORT2D($\mathscr{C}$).
2. $\lambda_{\mathscr{C}'} \leftarrow$ GEOMETRICSORT2D($\mathscr{C}'$).
3. Pick a point σ on the convex hull of $\mathscr{C}$.
4. $\pi \leftarrow$ CANONICALORDERING2D($\mathscr{C}, \sigma$).
5. **for all** points σ' on the convex hull of $\mathscr{C}'$ **do**
6. $\pi' \leftarrow$ CANONICALORDERING2D($\mathscr{C}', \sigma'$).
7. $\pi'' \leftarrow \pi^{-1}\pi'$.
8. **for** $i \leftarrow 1$ **to** n **do**
9. **for** $j \leftarrow 1$ **to** n **do**
10. Compare $\lambda_{\mathscr{C}}(i, j)$ and $\lambda_{\mathscr{C}'}(\pi''(i), \pi''(j))$
11. **end for**
12. **end for**
13. **if** all entries matched, record π''.
14. **end for**
15. **return** all recorded instances of π''.