UNIVERSITÉ LIBRE DE BRUXELLES
Faculté des Sciences

Regions, Distances and Graphs

Thèse présentée en vue de l'obtention
du grade de Docteur en Sciences

Sébastien COLLETTE
Année académique 2006–2007

Avoir deux directeurs de thèse, c'est avant tout une richesse et une opportunité d'apprendre deux fois plus. Cela permet d'avoir deux points de vue, et cela force à faire ses propres choix. J'ai eu ce privilège.

Aussi je voudrais remercier Jean et Stefan. D'abord pour m'avoir guidé, conseillé, motivé; et pour avoir passé tout ce temps à relire et corriger mes travaux. Mais aussi, et surtout, pour m'avoir fait découvrir un travail collaboratif passionnant. Ils m'ont prouvé qu'il suffit d'un set de table et quelques crayons pour progresser sur les problèmes les plus ardus.

I would like to thank the Carleton Computational Geometry Group for inviting me twice. It was a pleasure for me to be there, and I had the opportunity to work with great people.

Un grand merci à Raymond Devillers, pour sa relecture attentive et ses nombreux conseils avisés.

Merci aussi à tout ceux qui ont contribué à rendre le travail au Département d'Informatique agréable et intéressant. En particulier le groupe Algo, et les collègues du "mardi" midi qui se reconnaîtront.

Enfin, je remercie Sophie et ma famille qui m'ont toujours incité à aller jusqu'au bout de mes rêves.

Contents

Chapter 1

Introduction

In many problems in pattern recognition, such as clustering and computational approaches to perception, one is given a set of points on the plane and it is desired to find some structure among the points in the form of edges connecting a subset of the pairs of points.

– In [Tou80], ***Godfried T. Toussaint****, 1980*

This citation, dating back to the early times of computational geometry, expresses a fundamental but intentionally vague question: beginning with a set of points in $\mathbb{R}^2$, construct a graph using the points as vertices and defining the edges in a satisfying manner. In the same work, Toussaint wrote that one would like to find an algorithm such that the graph obtained is meaningful in some sense. Obviously, for each potential application and user, another set of characteristics will be of particular interest, leading to different ways of giving structure to the point set.

This problem is central to different fields: pattern recognition as stated above, but also geometric network and VLSI design, motion planning, medical imaging and telecommunications [She93, Cla87, RL73, Zah71]. A desirable property for communication and transportation networks, is to have transmission channels, or roads, minimizing the transmission distance. Therefore one could directly connect every pair of sites, resulting in a quadratic number of edges. But this solution is often rejected because it is too expensive or not realistic: imagine a country with straight roads between every possible pair of cities.

Chew [Che86] discovered in 1986 that even planar networks (i.e. without crossings) could ensure small transmission distances. It was the beginning of a long standing study of the so-called *spanner networks*, in which the detour – which corresponds to the ratio between the actual path and the straight line distance – when moving in the network from one site to another is at most a constant.

The first spanners Chew introduced were based on L_1 metric Delaunay triangulations, where the detour – or *spanning ratio* – is bounded by $\sqrt{10}$. Chew [Che89], Dobkin *et al.* [DFS90], and Keil and Gutwin [KG89] all studied the spanning ratio of Euclidean Delaunay triangulations, and concluded that it is upper bounded by $\frac{2\pi}{3\cos\pi/6} \approx 2.42$.

Later works by Das *et al.* [DJ89, ADD$^+$93] showed that many different graphs and triangulations were spanners, and gave two conditions – the *good polygon* and the *diamond* property – which, if satisfied by some graph, ensure a bounded spanning ratio.

These spanners were unpractical to construct, and their spanning ratio was bounded by a fixed constant. This motivated the introduction of the Θ-graph by Keil and Gutwin [KG92]: they gave a method leading to an easy construction of a spanner graph given any set of points, and where the tradeoff between the spanning ratio and the size of the graph could be fine-tuned by the user. Then followed a long series of new spanners: variants of the Θ-graph, such as skip list spanner [AMS94], sink spanner [ADM$^+$95] and ordered Θ-graph [BGM04], various triangulations and graphs based on the Delaunay constraints, for instance when using convex distances defined by equilateral triangles [Che89].

But the study of graphs generated by sets of points has not been restricted to the study of the spanning ratio and the resemblances between spanners and the complete graphs. In parallel with the work on spanners, another kind of graphs appeared, and other properties were studied. The relative neighborhood graphs introduced by Toussaint in his seminal 1980 paper as a solution to the problem of generating graphs were extended and generalized to form a family which is now called the *proximity graphs*. The oldest examples of proximity graphs include the Delaunay triangulations and the minimum spanning trees.

The proximity graphs define the relation between vertices in terms of proximity: two vertices form an edge if they are close in some sense (e.g. small Euclidean distance, nearest neighbor, empty region around them). Dozens [IS85, GS69, KR85, CCJ91, JT92, AM92, Vel92] of graphs are encompassed in that family. The study of these graphs consisted in the analysis of plenty of properties (size of the graph generated, connectivity,

spanning ratio, diameter, maximum degree of the vertices, possibilities of cycles, number of connected components, *etc*).

After more than two decades, we propose in this thesis to reconsider the original question. And as noted above, we do not want to propose a unique solution, as every application has specific requirements, but rather frameworks to define and analyze graphs which correspond better to our needs.

The first aspect that we would like to improve is the fact that the properties of these graphs are not significant for most purposes because they are too generic. For instance, if the graph we consider represents a network in which we want to do efficient routing, the known bounds on the size of paths in the corresponding graph family gives us an upper bound on the number of steps to go from any point to any other. But what if we just need to reach points a few steps away? Or if we are trying to find a path in a dense region of our network, with high connectivity? Can't we bound the query time more tightly? We look for upper bounds which would be dependent on the queries we perform.

The other aspect considered in this work is that it might be difficult to choose which family of graphs to consider given a set of properties. If we want a spanner graph, we could use the Θ-graph algorithm to construct the graph, but what if we also need the graph to be triangle-free? Which families of proximity graphs are cycle-free and connected? Ideally, one would give a list of desirable properties, and get in return the corresponding algorithm to construct a graph such that, whatever the set of points, the result would satisfy the user's needs.

In their paper on proximate point searching, Demaine, Iacono and Langerman introduced a way to define neighborhood of points with anchored regions and defined region-counting distances [DIL04]. While it was not their original purpose, we apply these results to the study of geometric graphs generated by sets of points, which leads us to propose solutions for the problems stated above: we define a new family of properties for geometric graphs, the *local properties*, which give better bounds when we consider close points in the data set; and we introduce *empty region graphs*, a family of proximity graphs in which, given a list of desirable properties, we can easily determine how to construct graphs for any set of points. This thesis is devoted to the detailed presentation of these results and their background.

Outline

Chapter 2 – Background

We begin by defining the context in which the results of the following chapters take place. We describe the *proximity graphs* and the *t-spanner graphs*, which are two families of geometric graphs. We illustrate these definitions with different kinds of graphs belonging to these families such as the Θ-graphs, the nearest neighbor graphs and the β-skeletons.

Then, we define queries that can be performed when using these graphs as data structures. We first present the *dictionary problem* in one dimension, show generalizations to the plane, and define properties of data structures supporting these queries. In particular we present *distribution-sensitive* data structures, and the *dynamic finger property*, which ensures that consecutive queries for semantically close items are performed efficiently. The model that we will use all along this thesis for queries on geometric graphs is described.

Chapter 3 – Anchored Regions and Region-Counting Distances

The *region-counting distances*, introduced by Demaine, Iacono and Langerman [DIL04], associate to any pair of points (p, q) the number of items of a dataset S contained in a region $R(p, q)$ surrounding p and q. In this chapter, we present these distances, their associated regions and their properties.

We define region-counting disks and circles, and study the combinatorial complexity of these objects. We give algorithms to construct them, and show that the complexity of a region-counting circle of radius k can be as large as the complexity of the k^{th} level in an arrangement of lines. We then present algorithms to compute ϵ-approximations of region-counting distances and approximations of region-counting circles whose complexity is constant.

We present another family of parameterized distances, the *speed distances*, and show the relation between this family and the region-counting distances.

Chapter 4 – Local Properties

In this chapter, we propose a definition of locality for properties of geometric graphs. We measure the local density of graphs using the region-counting distances between pairs of vertices, and we use this density to define local properties of classes of graphs.

We illustrate locality by introducing the *local diameter* of geometric graphs: we define it as the upper bound on the size of the shortest path between any pair of vertices, expressed as a function of the density of the graph around these vertices. We determine the local diameter of several well-studied graphs such as the Θ-graph, the ordered Θ-graph and the skip list spanner. We also show that various operations, such as path and point queries using geometric graphs as data structures, have complexities which can be expressed as local properties.

Chapter 5 – Empty Region Graphs

A family of proximity graphs, called *empty region graphs* (ERG) is presented. The vertices of an ERG are points in the plane, and two points are connected if their neighborhood, defined by a region, does not contain any other point. The region defining the neighborhood of two points is a parameter of the graph. This family of graphs includes several known proximity graphs such as nearest neighbor graphs, β-skeletons or Θ-graphs.

This chapter concentrates on ERGs that are invariant under translations, rotations and uniform scaling of the set of vertices. We give conditions on the region defining an ERG to ensure a number of properties that might be desirable in applications, such as planarity, connectivity, triangle-freeness, cycle-freeness, bipartiteness and bounded degree. These conditions take the form of what we call *tight regions*: maximal or minimal regions that a region must contain or be contained in to make the graph satisfy a given property.

We show that every monotone property has at least one corresponding tight region; we discuss possibilities and limitations of this general model for constructing a graph from a point set.

We end this chapter with a generic framework to construct ERG using the construction algorithms for Θ-graphs and Gabriel graphs as a subroutine.

Chapter 6 – Sigma-Local Graphs

We introduce and analyze the *σ-local graphs*, a subfamily of ERG based on a definition of σ-locality by Jeff Erickson [Eri03]. We present two *ad-hoc* algorithms to construct such graphs given a fixed real value σ and a set of points S. We also present a preprocessing method to find the graph corresponding to any σ in linear time using $O(n \log^{O(1)} n)$ space and preprocessing time.

We also consider the following question: given a set of points, what is the extremal value of σ such that the corresponding graph has properties such as connectivity, planarity, completeness and absence of isolated vertices. We give the corresponding algorithms, which can also be used to efficiently construct the graphs.

Contributions and Acknowledgement

This work was supported by the *Fonds National de la Recherche Scientifique*. Except for results of Chapter 2 and other results whose authors are cited where first described, all contributions of this work are original results to our knowledge.

Results on *region-counting circles*, presented in Chapter 3, were proposed in collaboration with Jean Cardinal and Stefan Langerman. They have been presented at the *17th Canadian Conference on Computational Geometry* [CCL05a]. Other results in that chapter have been published in [CCL04, CCLar, CCL05b].

The contributions on *local properties of geometric graphs* were originally presented at the *16th Canadian Conference on Computational Geometry* [CCL04] and published in a special issue of the journal *Computational Geometry: Theory and Applications* [CCLar] dedicated to the best papers of the conference. These are resulting from our work with Jean Cardinal and Stefan Langerman; they can be found in Chapter 4.

Work on *empty region graphs*, which is presented in Chapter 5, has been done in collaboration with Jean Cardinal and Stefan Langerman. We thank Samuel Fiorini for interesting discussions on this topic. Preliminary results have been presented at the *21st European Workshop on Computational Geometry* under the title *region-counting graphs* [CCL05b].

Definitions and algorithms for the construction of *σ-local graphs* are presented in Chapter 6. My coauthors for this work are Prosenjit Bose, Stefan Langerman, Anil Maheshwari, Pat Morin and Michiel Smid.

Chapter 2

Background

2.1 Introduction

This chapter introduces the context in which our results take place. Most of the results presented here were previously known or are corollaries of previous contributions. Our aim is not to provide an extensive survey, but rather to make this document self-contained.

All along this document, we suppose that we are given a set S of points in the plane. We will not analyze S in itself, but concentrate on the associated graphs and data structures. The objectives of the next sections are first to present graphs generated by sets of points that we use as data structures, and secondly to define queries that these data structures should support.

For the first aspect, we consider here ***geometric graphs***. These are graphs with vertices in the plane; we use the items in the set S as the vertices of our graphs. Two families of geometric graphs will be of particular interest: ***t-spanner graphs*** [Che86] and ***proximity graphs*** [JT92]. These are presented in Section 2.2.

In every t-spanner graph, there exists between any pair of vertices a path whose Euclidean length is less than t times the Euclidean distance between these vertices. Subsection 2.2.1 presents examples of t-spanners such as the Θ-graph and its variants and the proximate point searching graph.

The proximity graphs are defined on a finite set of vertices in the plane and two vertices form an edge if they are close in some sense. The proximity can be measured for instance by the Euclidean distance between these vertices, the distance to other vertices of the graph, or the number of other vertices in a given neighborhood. Subsection 2.2.2

describes the unit disk graphs, the nearest neighbor graphs and the β-skeletons as examples of proximity graphs.

Geometric graphs are used in numerous fields: motion planning, VLSI design, Geographic Information Systems [Cla87, She93]. They are often considered as data structures e.g. for a set of points. We may want to know if a point is in the data structure (***point query***), find the closest point in the data structure to a given point (***nearest neighbor query***), or see how we can, in a network, go from one point to another (***path query***).

These queries and their complexities are only meaningful in the context of a given ***computational model***, which is considered in Subsection 2.3.3. We propose to use a slightly adapted pointer model to correspond tightly to the specificities of our data structures.

The point query problem in $\mathbb{R}^2$ is a 2-dimensional extension of the ***dictionary problem***, which consists in retrieving the information associated with a key in a totally ordered dataset. In Subsection 2.3.2 we detail known results for that problem in one and two dimensions. We also detail properties of data structures supporting these queries such as the ***dynamic finger property*** which ensures that successive queries on lexically close items are performed efficiently (for example consecutive words in a dictionnary or adjacent cities on a map).

The path query problem in a graph is presented in Subsection 2.3.4. We present routing algorithms which let us, following the edges of a given graph, go from one vertex to any other.

Contents

2.2 Geometric Graphs

Let us begin by defining the geometric graphs:

Definition 2.1 *A **geometric graph** $G = (V, E)$ consists of a set V of elements of $\mathbb{R}^2$ called vertices and a set E of pairs of members of V called edges.*

As stated above, the graphs on which we work are generated by sets of points, which means that they are uniquely defined by their set of vertices:

Definition 2.2 *A family of **graphs generated by a set of points** parameterized by a function f is a set of graphs $G = (V, E)$ defined by finite sets of points $P \subseteq \mathbb{R}^2$, with $V = P$ and such that the set of edges $E = f(P)$.*

We say that a graph is ***undirected*** if the edges are unordered pairs, and ***directed*** otherwise. The ***length of an edge*** (p, q) is the Euclidean distance $\mathrm{d}_2(p, q)$ between its endpoints p and q. The ***degree*** of a vertex p is the cardinality of the set of edges having p as endpoint.

Definition 2.3 *A **path** $P = (u_1, u_2, \ldots, u_k)$ in a graph G between the vertices u_1 and u_k is a sequence of distinct vertices of V such that*

$$\forall 1 \leq i < k \ : \ (u_i, u_{i+1}) \in E$$

Sometimes we consider undirected path even if the graph itself is directed:

Definition 2.4 *An **undirected path** $P = (u_1, u_2, \ldots, u_k)$ in a directed graph G between the vertices u_1 and u_k is a sequence of distinct vertices of V such that*

$$\forall 1 \leq i < k \ : \ (u_i, u_{i+1}) \in E \quad or \quad (u_{i+1}, u_i) \in E$$

Note that even if we have defined paths as sequences of k vertices, we can equivalently define them as sequences of $k - 1$ edges. A ***cycle*** is a closed path, with the same vertex at both ends.

A graph is ***connected*** when there exists a path (directed or not) between any pair of vertices. We say that a directed graph is ***strongly connected*** when there exists a directed path in both directions between any pair of vertices. The notion of length can be extended to paths, for which it is defined as the sum of the Euclidean length of the consecutive edges:

Definition 2.5 *The **length of a path** $P = (u_1, u_2, \ldots, u_k)$ in the graph G is*

$$\mathrm{d}_G(P) = \sum_{i=1}^{k-1} \mathrm{d}_2(u_i, u_{i+1})$$

Definition 2.6 *The **length of the shortest path** between the vertices u and v in the graph G is*

$$\mathrm{d}_G(u, v) = \min_{P \in P_{u,v}^G} \mathrm{d}_G(P)$$

where $P_{u,v}^G$ is the set of paths between u and v in G.

The size of a path is the cardinality of its set of vertices:

Definition 2.7 *The **size of a path** $P = (u_1, u_2, \ldots, u_k)$ in the graph G is*

$$|P| = |(u_1, u_2, \ldots, u_k)| = k$$

Definition 2.8 *The **shortest size path** between the vertices u and v is*

$$\mathrm{ssp}(u, v) = \left\{ P \in P_{u,v}^G \;\middle|\; |P| = \min_{P' \in P_{u,v}^G} |P'| \right\}$$

where $P_{u,v}^G$ is the set of paths between u and v in G.

This allows us to define the notion of diameter of a graph. Intuitively, it is the minimal number of steps to go from any vertex to another while following the edges of the graph.

Definition 2.9 *The **diameter** of a graph G is $\max_{u,v \in V} |ssp(u, v)|$.*

2.2.1 t-Spanner Graphs

The family of t-spanner graphs [Che86], or equivalently t-spanners, form a subclass of geometric graphs in which the length of the shortest path $\mathrm{d}_G(u, v)$ joining two vertices u and v is within a factor of the Euclidean distance between them. This subsection is dedicated to the definition of these graphs and to the description of some well-known t-spanners.

Definition 2.10 *For $t \in [1, \infty)$, a graph in the plane is a* ***t-spanner*** *if and only if*

$$\forall u, v \in V \ : \ \frac{\mathrm{d}_G(u, v)}{\mathrm{d}_2(u, v)} \leq t,$$

where t is called the ***spanning ratio*** *or* ***dilation*** *of the graph.*

A path between a pair of vertices (p, q) satisfying the spanning condition is called a ***t-spanner path***. Note that it is not necessarily the shortest length path, nor the shortest size path between p and q.

Θ-Graphs

Chew [Che86] defined t-spanners with the aim of approximating the complete Euclidean graph. Keil and Gutwin [KG92] introduced the Θ-graph to provide a way to construct t-spanners efficiently. Note that these graphs are similar to the Yao graphs, introduced independently [Yao82].

Let Θ and k be constants, with $k = (2\pi)/\Theta$. We restrict k to be an integer, such that Θ divides 2π. A **Θ-*graph*** G is a directed graph defined by a point set V. Each vertex has up to k outgoing edges connected to the closest vertex in k different cones. The i^{th} ***cone*** associated to a vertex p in a Θ-graph is the subspace containing all the points with absolute angle from p between $i\Theta$ included and $(i+1)\Theta$ excluded. As there are $k = (2\pi)/\Theta$ cones for each vertex, the k cones cover the plane.

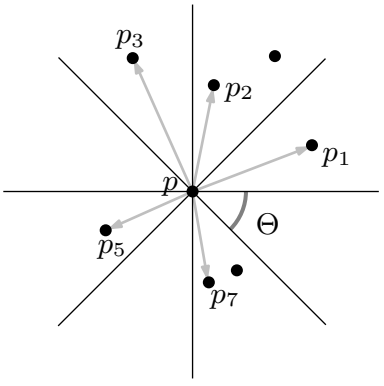


Figure 2.1: Plane subdivision using cones, as used in Θ-graphs and variants.

Figure 2.1 shows the cones associated with one vertex. In each cone i of the vertex p, an outgoing edge is connected to the closest vertex, denoted by p_i, if such a vertex

exists. Various definitions of the closest vertex can be found in the literature; while the Euclidean distance (Figure 2.2a) seems to be the most natural, the closest vertex in a cone is defined in [KG92, Yao82] as the one having the closest orthogonal projection onto the border of the cone (Figure 2.2b). Other authors define it as the vertex with the closest orthogonal projection onto the bisector of the cone (Figure 2.2c) e.g. [RS91].

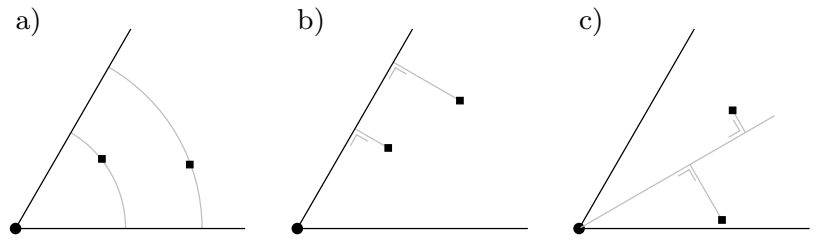


Figure 2.2: Different ways to see which is the closest vertex in a cone.

Keil and Gutwin [KG92] proved that the Θ-graph with $\Theta < (\pi/4)$ is a t-spanner graph with $t = (1/\cos\Theta)(1/(1-\tan\Theta))$, using the orthogonal projection on the border of the cone to choose the closest vertex. Ruppert and Seidel [RS91] also proved that the Θ-graph with $\Theta < (\pi/4)$ is a t-spanner graph, with $t = (1-2\sin(\Theta/2))^{-1}$, when using the orthogonal projection onto the bisector to choose the closest vertex in a cone. Θ-graphs have at most kn edges, at most k out-going edges per vertex and at most $n-1$ in-going edges. The diameter of the graph is $\Theta(n)$ for instance when the vertices of the graph are placed on a line (here Θ indicates an asymptotically tight bound, and not the parameter of the graph).

Ordered Θ-Graphs

The ***ordered Θ-graph*** $G = (V, \pi)$ [BGM04] is mainly based on the Θ-graph, and uses the same cone subdivision of the plane. It has been proposed by Bose, Gudmundsson and Morin, to address some weaknesses of the original Θ-graph (such as its diameter which can be $\Theta(n)$). The difference is that in the ordered Θ-graph the order π of insertion of the vertices matters: each vertex is connected to up to k edges, which are the closest (orthogonal projection on the cone boundary) among the previously inserted vertices in the k cones.

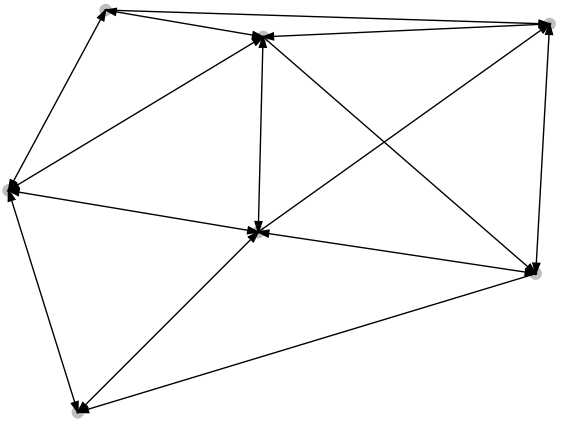


Figure 2.3: Example of a Θ-graph, with $\Theta = \pi/3$.

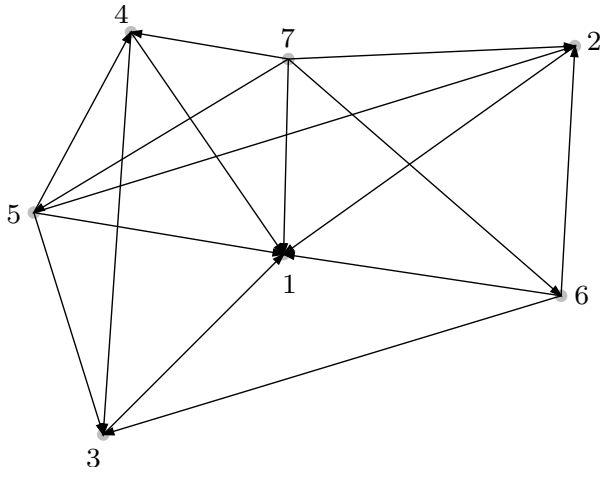


Figure 2.4: Example of an ordered Θ-graph, with $\Theta = \pi/3$. The numbers represent the order of insertion.

The graph is not strongly connected: from one vertex we cannot find an out-going path to any vertex which has been inserted afterwards. This is why, in what follows, the ordered Θ-graphs are seen as undirected graphs.

If the order π is a random permutation of the set of vertices, the corresponding graph has a small diameter ($O(\log n)$ expected). As we rely on this property in this work, when we refer to ordered Θ-graphs we refer to random ordered Θ-graphs where vertices are inserted in random order. Based on the properties of the Θ-graph, the ordered variant also has a linear number of edges, a linear in-degree and an out-degree bounded by k. Its spanning ratio is $t \leq \frac{1}{\cos(\Theta)-\sin(\Theta)}$.

Skip List Spanners

The ***skip list*** is a data structure introduced by Pugh [Pug90] to create lists with low access cost, by adding multiple levels to the list. The main idea is to use $\log n$ levels containing sorted items. Each element is present in all levels up to level l with probability 2^{-l}, so that the l^{th} level contains on average $n/2^l$ items. To access an item, we do linear search in the highest level, and as soon as we know that the item is not in that level (i.e. we went one step too far), we go one step back and continue to search in the level below. As every item in level i is also present in all levels i' with $i' < i$, we continue the search from where we stopped. Using this structure, any item in the list can be reached in $O(\log n)$ (expected) time.

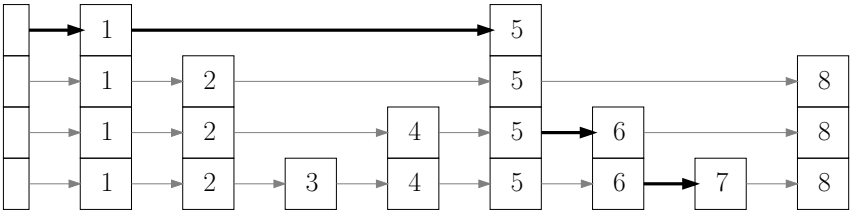


Figure 2.5: Example of skip list. The highlighted path corresponds to access to item 7.

The ***skip list spanner*** $G = (V, \omega)$ [AMS94] is an extension of the Θ-graph, inspired by the skip lists, introduced by Arya, Mount and Smid, where every vertex is assigned to a level. Their goal was to find a spanner with a low diameter. It is a randomized structure; the level assignment ω is made by throwing a fair coin for every vertex. As the coin flips are independent, on average half of the vertices will get a head, and will

be assigned to the first level. For the other half, a second independent coin flip assigns, on average, one quarter of the points to the second level. For the last quarter, a third coin is thrown, and so on. This will lead typically to $\log n$ levels, level l containing $n/2^l$ vertices on average.

For each level, a Θ-graph is constructed, containing only the vertices present in the current level and all the higher levels. This means that a vertex at level l is present in l different Θ-graphs. From there, we know that the out-degree is in $O(\log n)$ with high probability, as Θ is a constant. The spanning ratio is upper bounded by that of the Θ-graph, as the skip list spanner contains it. Its diameter is in $O(\log n)$ with high probability.

Proximate Point Searching Graphs

In [DIL04], Demaine, Iacono and Langerman defined a data structure to support point queries. This structure can be seen as a graph associated to the set of points stored in the data structure.

The ***proximate point searching graph***, as defined in [DIL04], is a directed graph where each vertex has $O(\log n)$ outgoing edges. This graph is, by multiple aspects, very similar to the skip list spanner [AMS94]: a vertex is surrounded by $k = 2\pi/\Theta$ non-overlapping cones; the edges in cone k handle travel in a direction with absolute angle between $i\Theta$ and $(i+1)\Theta$.

In what follows the analysis is done using k equal to 6. If k is not 6, Demaine *et al.* [DIL04] suggest to do a scaling, by a factor of $2\sin(\pi/k)$ along the line that bisects the cone to do the construction of the triangles defined below: this ensures that we can always consider equilateral triangles.

In the proximate point searching graph, each vertex p is in $\lceil \log_{3/2} n \rceil$ levels. At level r and in each cone, p is associated with the largest equilateral triangle τ containing at most $(3/2)^r$ points including p. Note that, as we consider the case where $k = 6$, the boundaries of the cones corresponds to the borders of the triangle. Figure 2.6 shows the triangles present at level two.

The triangle τ contains three overlapping sub-triangles τ_1, τ_2 and τ_3, each of them containing up to $(3/2)^{r-1}$ points which are the closest points to each corner of the triangle τ. The sub-triangle containing the apex p of the cone is τ_1. Figure 2.7 shows three views of the triangle τ at level four, and thus containing at most five points. For each cone

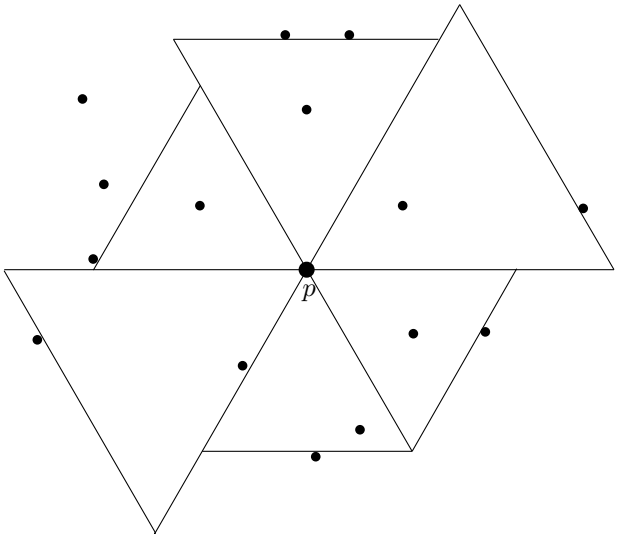


Figure 2.6: Triangles in each cone at level 2.

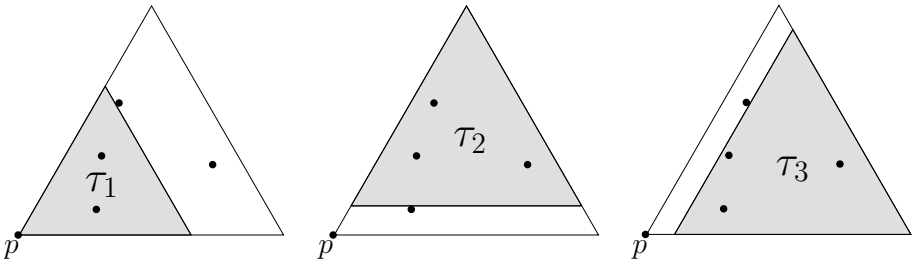
and level, p has six outgoing edges connected to the vertices closest to each border edge of τ_2 and τ_3.

We know that we can reach a vertex in each sub-triangle in one step, because there is an edge to the closest vertex to each edge in τ_2 and τ_3, and that p_i itself is contained in τ_1. Moreover it has been shown in [DIL04] that $\tau_1 \cup \tau_2 \cup \tau_3 \supseteq \tau$.

Lemma 2.1 *The proximate point searching graph contains a Θ-graph as its subgraph with $\Theta = 2\pi/k$.*

Proof We know that for each vertex p, in each cone, at level two the triangle τ contains up to two vertices (in fact at most 9/4 vertices): p, and the closest vertex in that cone. We also know that τ_1, τ_2 and τ_3 contain at most one vertex. τ_1 contains p, and thus the other vertex is contained in τ_2 or τ_3.

As we know that there is an edge to at least one vertex in τ_2 and τ_3, if a vertex is present, and as we know that the closest vertex in the cone is the only vertex of τ_2 or τ_3, we can say that there is an edge between the apex and the closest vertex in the cone, which is precisely the definition of the Θ-graph. As we extend a triangle, whose extended edge is perpendicular to the bisector of the cone, we must consider the closest orthogonal projection on the bisector (Figure 2.2c). □



$$r = 4\ ;\ \left\lfloor \left(\tfrac{3}{2}\right)^r \right\rfloor = \left\lfloor \tfrac{81}{16} \right\rfloor = 5\ ;\ \left\lfloor \left(\tfrac{3}{2}\right)^{r-1} \right\rfloor = \left\lfloor \tfrac{27}{8} \right\rfloor = 3$$

Figure 2.7: Subdivision of τ into three overlapping sub-triangles.

The Θ-graph is a t-spanner, and is a subgraph of the proximate point searching graph. This shows that the latter is a t-spanner. Other properties of this graph can be found in [DIL04]: it has $O(n \log n)$ edges, at most $O(\log n)$ out-going edges and $O(n \log n)$ in-going edges per vertex, and a diameter of $O(\log n)$.

2.2.2 Proximity Graphs

The ***proximity graphs*** [JT92], also called ***neighborhood graphs***, are a family of graphs where edges are present between close pairs of vertices. There is no formal definition of that proximity constraint, and thus many graphs belong to that family.

These graphs are well-studied; a survey of Jaromczyk and Toussaint [JT92] discusses many of them, such as ***Delaunay triangulations***, ***relative neighborhood graphs*** [JT92, AM92], ***Gabriel graphs*** [GS69], ***β-skeletons*** [KR85], ***rectangular influence graphs*** [IS85]. Other examples of proximity graphs include the ***Θ-graphs*** [KG92, Yao82] described in Subsection 2.2.1, and the ***γ-neighborhood graphs*** [Vel92].

In what follows, we will describe the ***unit disk graphs*** [CCJ91], the ***nearest neighbor graphs*** [EPY97] and the ***β-skeletons*** [KR85], as these graphs will be useful in the next chapters.

Unit Disk Graphs

The ***unit disk graphs*** [CCJ91] use what is probably the most natural definition of proximity: two vertices p and q are close if their Euclidean distance $\mathrm{d}_2(p, q)$ is small. They form an edge if the distance is less than 1.

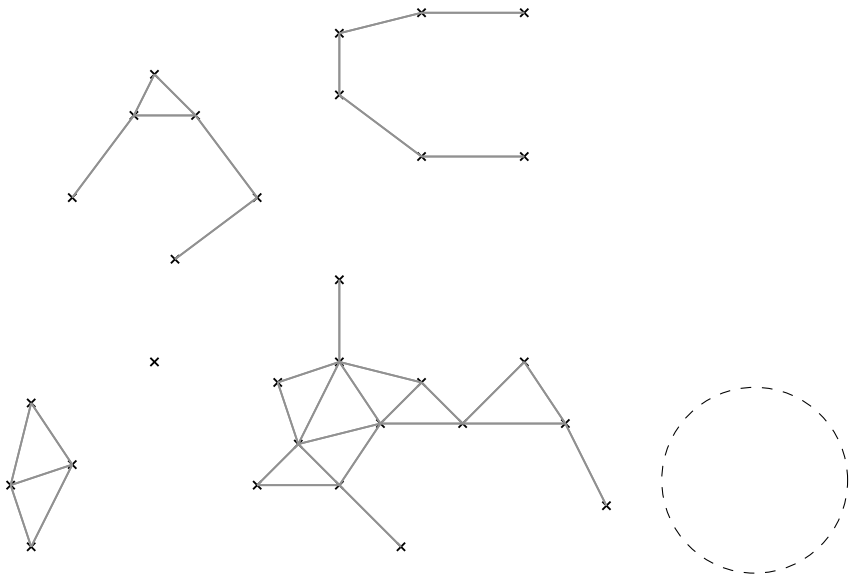


Figure 2.8: Example of a unit disk graph. The dashed circle has radius 1.

These graphs are useful in the context of cellular networks and of ad-hoc wireless communications: informations can be exchanged between nodes in the transmitting range of each other. These unit disk graphs are undirected as the proximity relation is symmetric. They are not always connected, and can have $\Omega(n^2)$ edges.

Nearest Neighbor Graphs

The ***nearest neighbor graphs*** [EPY97] are proximity graphs where each vertex is connected to its nearest neighbor with respect to the Euclidean distance. As the relation is not symmetric, the graph is directed.

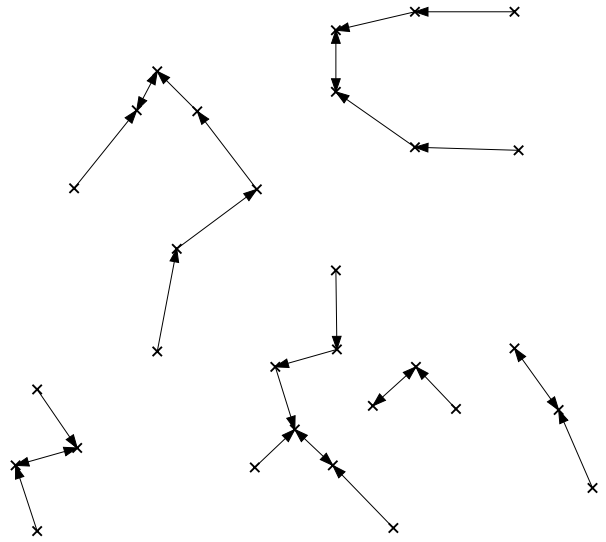


Figure 2.9: Example of a nearest neighbor graph.

These graphs have n edges, the out-degree is 1 and the in-degree is at most 6 for every vertex. They contain no cycle of length greater than 2 for sets in general position (i.e. the distance between any couple of pairs of vertices is distinct), and not always connected.

β-Skeletons

Let β be a fixed constant. The β-skeletons are a parameterized family of proximity graphs introduced by Kirkpatrick and Radke [KR85]. In the ***β-skeletons***, two vertices

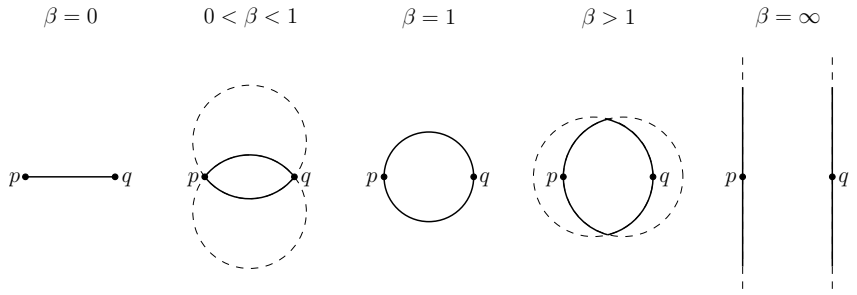


Figure 2.10: Regions $I_\beta(p, q)$ for different values of β

p and q are close if the region $I_\beta(p, q)$ does not contain any other point of the set, with $I_\beta(p, q)$ defined as follows:

- For $\beta = 0$, $I_\beta(p, q)$ is the line segment pq.
- For $0 < \beta < 1$, $I_\beta(p, q)$ is the intersection of the two disks of radius $\mathrm{d}_2(p, q)/(2\beta)$ with p and q on their boundaries, where d_2 is the Euclidean distance.
- For $1 \leq \beta < \infty$, $I_\beta(p, q)$ is the intersection of the two disks of radius $\beta\ \mathrm{d}_2(p, q)/2$ and centered at the points $(1-\beta/2)p+(\beta/2)q$ and $(\beta/2)p+(1-\beta/2)q$, respectively.
- For $\beta = \infty$, $I_\beta(p, q)$ is the infinite strip perpendicular to the line segment from p to q.

An ***open*** β-skeleton has edges between pairs of vertices (p, q) even if there are other items on the outer boundary of $I_\beta(p, q)$, while a ***closed*** β-skeleton has edges only if the region is totally empty. A ***Gabriel graph*** is a closed 1-skeleton; a ***relative neighborhood graph*** is an open 2-skeleton.

Every open β-skeleton with $\beta \leq 2$ and every closed β-skeleton with $\beta < 2$ is connected. Note that the graphs are monotone with respect to the parameter β, which means that for a given set of points, as β shrinks edges are added and never removed. The β-skeletons for $\beta \geq 1$ can be constructed in $O(n \log n)$ time, as show in [KR85]. For β-skeletons with $\beta < 1$, some sets of points yield graphs containing $\Omega(n^2)$ edges and requiring $\Omega(n^2)$ time to be constructed.

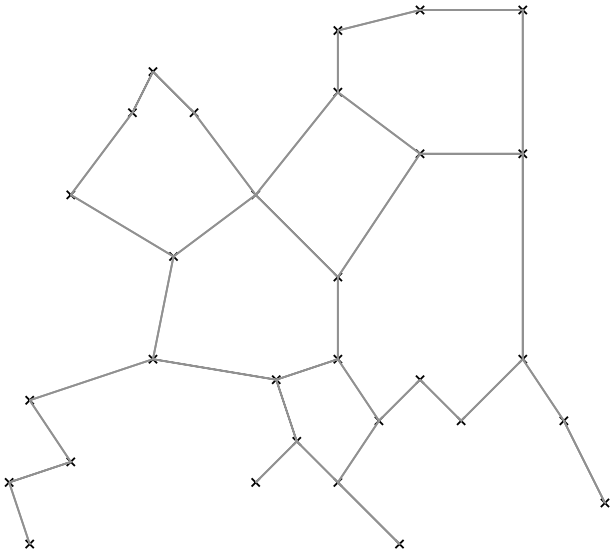


Figure 2.11: Example of an open 2-skeleton, also called relative neighborhood graph.

2.3 Queries and Computational Model

As stated above, we can consider graphs as data structures. A simple example could be to have a graph representing the transportation network between cities. Then, natural questions arise such as how to reach one city starting from another or what is the shortest path between one city and the other. These questions are then simple queries in the data structure. In this section, we will detail point and path queries, and describe the model in which we will perform these queries.

2.3.1 Queries

We begin by formally defining the queries that we will consider in the remainder of this document:

Definition 2.11 *The **point query** consists of, given a data set S and a query point q, finding the point q in S if such an item exists.*

When we use a graph as the underlying data structure, the point query corresponds to finding a vertex with a given label in a graph. This is an extension of the dictionary problem, that we detail in the next subsection.

Definition 2.12 *The **path query** consists of, given a graph G and a pair of query vertices (p, q), finding a path between those vertices, if such a path exists.*

For the point query, we would like to answer the query as efficiently as possible, meaning that we would like to take a path in the graph with short size, as the complexity of the queries will depend on the number of steps in the graph.

For path queries, however, this is not always the case. Here we have to consider the quality of the paths produced: one could prefer a t-spanner path or the shortest length path, instead of the shortest size path. In what follows, we consider the case where the paths returned are t-spanners, unless specified otherwise.

2.3.2 Dictionary Problem and Dynamic Finger

The ***dictionary problem*** consists in retrieving the information associated with a key in a totally ordered dataset. For keys in $\mathbb{R}$, many approaches have been proposed for that problem, from binary search trees and skip lists to hash tables. We will not describe

these methods here, as this would be out of the scope of this work; a survey can be found in [CLR90].

Some of the existing data structures for the dictionary problem have an interesting property, when we consider a sequence of queries:

Definition 2.13 *The **rank difference** between two items p and q in a totally ordered set, denoted by $|r(q) - r(p)|$, is*

$$|\{x \in S : \min(p, q) \leq x \leq \max(p, q)\}|$$

which is the number of items contained in the interval between them.

Definition 2.14 *A dictionary data structure for a finite set S has the **dynamic finger property** for the dictionary problem if and only if it achieves $O(\log |r(q_i) - r(q_{i-1})|)$ query time, where q_i is the i^{th} query.*

These structures where the running time is a function of the sequence of queries are called ***distribution-sensitive*** data structures. For instance, ***level-linked trees*** of Brown and Tarjan [BT80] and ***splay trees*** of Tarjan and Sleator [ST85] have the dynamic finger property.

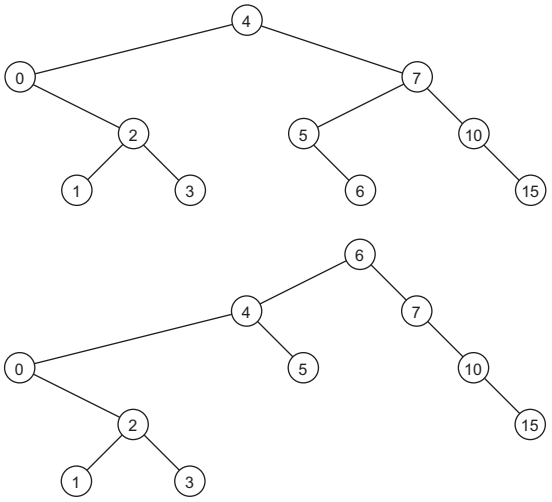


Figure 2.12: Splay tree example: before and after access to item 6.

The splay tree is a good example to illustrate the dynamic finger property: it is a binary search tree, in which accesses restructure the tree. In splay trees, the structure is modified each time a key is accessed (using the so-called zig-zig and zig-zag rotations), so that the queried point is at the root of the structure at the end of the query. If this point has been accessed recently, the query will be answered faster. A bottom-up method is used for rotations. Starting with the accessed node, rotations are applied until the node is at the root. At each node, we consider the parent and grand-parent nodes to know which kind of rotation is needed. As nodes which have a small rank difference with the previous key are also brought up by the rotations, this ensures that splay trees have the dynamic finger property. The first proof for this fact was published by Cole in 2000 in a series of two large articles and is quite complicated [CMSS00, Col00].

The rank difference and the dynamic finger property are not defined in $\mathbb{R}^2$, as there is no implicit total order. Demaine, Iacono and Langerman introduced ***region-counting distances*** [DIL04] as a generalization of the rank difference: the rank difference between two items of a set in $\mathbb{R}^2$ is defined as the number of items of the set in their neighborhood. More detailed explanations and discussions on the definition of the neighborhood and distances are presented in Chapter 3.

They also generalized the dynamic finger property in $\mathbb{R}^2$ and defined it like in the dictionary problem but using the rank difference generalization.

2.3.3 Model used for Queries on Graphs

In the ***pointer model*** [vEB90], algorithms must use nothing else than a graph as data structure, where every vertex has a constant number of out-going labeled edges. One of the vertices is the ***center*** or ***root***, by which we begin to visit the graph. The allowed operations in that model consist of following the edges, comparing two vertices to determine if they are the same, creating or modifying existing edges, creating or modifying existing vertices.

We use a slightly different model, which corresponds more precisely to the graphs we study. Our model uses a graph as unique data structure. Unless otherwise specified, these operations are the only ones allowed: create or delete edges and vertices, follow an edge, compare two vertices and determine if they are the same, list all the vertices of the graph (in time $O(|V|)$); list all the edges of the graph (in time $O(|E|)$); list the in-going and out-going edges of a given vertex.

As the degree of our vertices can be large, we include in our model an efficient way of selecting an edge from a given vertex: each vertex contains an access structure, in the pointer model, to access adjacent edges individually. Such a structure is used for example in skip list spanners (see Subsection 2.2.1), in which edges are partitioned into levels, and every operation can be restricted to a given level. For instance, we can list all the out-going edges in level two for a given vertex. This model enforces the use of the graph structure: we can follow edges and paths (while staying in the same level); we can access the level above and below in $O(1)$ time. But we cannot reach directly an arbitrary vertex or level in the graph.

2.3.4 Queries and Routing

For the path query problem, we want to find a path in a graph. This corresponds to the concept of routing, first defined in the context of transportation and telecommunication networks:

Definition 2.15 *Given a geometric graph G, **routing** consists in, given a pair of vertices (p, q), finding a path from p to q.*

For the point query problem, our goal in the next chapters will be to find distribution-sensitive structures. A point query will not consist in finding an item starting from the root of the structure, but rather from the vertex corresponding to the previous query. In that case, a point query corresponds to finding a route from one vertex to the other in the graph. In other words, we are looking for efficient online routing algorithms:

Definition 2.16 *Given a geometric graph G, **online routing** consists in, starting with a given vertex, finding a path to the query point while taking only local decisions, which consist in finding the next step of the path based on the position of the current vertex, of its neighbors, of the query point, and of a constant number of state variables.*

Routing in graphs has been studied extensively. In Morin's thesis [Mor01] different algorithms for online routing in geometric graphs and triangulations were proposed. Morin showed that even in planar geometric graphs, online routing is a hard problem: even if there are routing algorithms guaranteeing that one will eventually reach the destinations, the lengths and size of the paths can be large. There are however algorithms producing t-spanner paths for the Θ-graphs and its variants.

For point queries, the adapted online routing algorithms for the corresponding geometric graphs can be directly used. For path queries, however, some of the restrictions

induced by online routing can be relaxed. We want to find the best route in the graph; this can sometimes be achieved by constructing a path from both ends.

In what follows, we describe known (online or not) routing algorithms for some of the t-spanners detailed before. We indicate, for every graph, if the paths that are constructed are t-spanners and we give bounds on the size of these paths.

Θ-Graphs

```
Let p be the current vertex;
Let q be the destination vertex;
Let P be an empty path;
while p ≠ q do
    Compute α the absolute angle between p and q;
    Let i = ⌊α/Θ⌋;
    Add the edge (p, p_i) to P;
    Let p = p_i;
end
```

Figure 2.13: Θ-path algorithm [KG92].

We denote by p_i the closest vertex in the i^{th} cone centered at vertex p. A simple online routing algorithm, given by Figure 2.13, can be used to reach any vertex from any other following edges of the graph. The algorithm always follows the edge in the cone containing the other endpoint of the path, until it is reached. The paths produced are t-spanners (with $t = (1/\cos\Theta)(1/(1-\tan\Theta))$); they can be composed of up to $n-1$ edges, for example when the vertices are nearly on a line.

Ordered Θ-Graphs

We recall that an ordered Θ-graph is a variant of the Θ-graph in which the construction is done incrementally by adding the vertices one after the other, with an insertion sequence π. We denote by π_p the position of the vertex p in the insertion sequence. There is an edge from p to q only if $\pi_p > \pi_q$.

In an ordered Θ-graph we must consider undirected paths, as the graph is not strongly connected. An undirected t-spanner path can be constructed using the algorithm given by Figure 2.14. Both ends of the path are considered at each step of the algorithm; this

```
Let p be the origin;
Let q be the destination;
Let P be an empty path;
while p ≠ q do
    if π_p > π_q then
        Compute α the absolute angle between p and q;
        Let i = ⌊α/Θ⌋;
        Add the edge (p, p_i) to P;
        Let p = p_i;
    else
        Compute α the absolute angle between q and p;
        Let i = ⌊α/Θ⌋;
        Add the edge (q, q_i) to P;
        Let q = q_i;
    end
end
```

Figure 2.14: Ordered Θ-path algorithm [BGM04].

implies that this algorithm cannot be used for online routing and thus for point queries. The end whose vertex has the highest order (the one inserted the latest) is used to extend the path. At each step, the algorithm selects the cone containing the other end, which is the target. The edge contained in this cone is added to the path. This either completes the path, or reduces it to find a sub-path recursively. The expected path size is $O(\log n)$ as, on average, every two steps the order of one of the endpoints of the path will increase by at least a constant.

Skip List Spanners

We denote by $p_{(l,i)}$ the edge from the vertex p to its closest neighbor at level l in the i^{th} cone.

Even if the skip list spanner is strongly connected, the routing algorithm (given by Figure 2.15) constructs the path by both ends like in the ordered Θ-graph case. At each step, one edge will be added to the end whose level l is the lowest. This will be achieved by finding the cone in which the other end lies, and using the edge associated to the level l in this cone.

```
Let p be the origin;
Let q be the destination;
Let P be an empty path;
while p ≠ q do
    if L_p < L_q then
        Compute α the absolute angle between p and q;
        Let i = ⌊α/Θ⌋;
        Add the edge (p, p_(L_p,i)) to P;
        Set p = p_(L_p,i);
    else
        Compute α the absolute angle between q and p;
        Let i = ⌊α/Θ⌋;
        Add the edge (q, q_(L_q,i)) to P;
        Set q = q_(L_q,i);
    end
end
```

Figure 2.15: Skip list spanner path algorithm [AMS94].

This edge either completes the path, or leads to a new vertex with level at least l, and the complete path is found recursively. The constructed path is a t-spanner path with expected size $O(\log n)$.

Proximate Point Searching Graphs

The routing algorithm given by Figure 2.16 takes advantage of the proximate point searching graph to reach the target in a small number of steps. However, the path is not a t-spanner. Figure 2.17 illustrates such a situation where the total length of the path is large compared to the distance to the query point. This comes from the fact that the size of the triangles and sub-triangles is related to the position of the vertices, and is not bounded. Thus the triangle at the lowest level containing the target vertex can contain other vertices which can be arbitrarily far, and which will be used as stepping stones to the target.

But as this graph contains the Θ-graph, the routing algorithm of the latter can be adapted to get t-spanner paths of larger size than the path obtained with the algorithm of Figure 2.16.

```
Let p be the origin;
Let q be the destination;
Let P be an empty path;
Let r be the current level (initialized later);
while p ≠ q do
    Compute α the absolute angle between p and q;
    Let i = ⌊α/Θ⌋;
    Set r = 1;
    while q ∉ τ(p,r,i) do
        Set r to r+1;
    end
    Find vertex v, adjacent to p or p itself, and cone i such that
    τ(v,r−1,i) contains q;
    Add the edge (p, v) to P;
    Set p = v;
end
```

Figure 2.16: Proximate point searching graph routing algorithm.

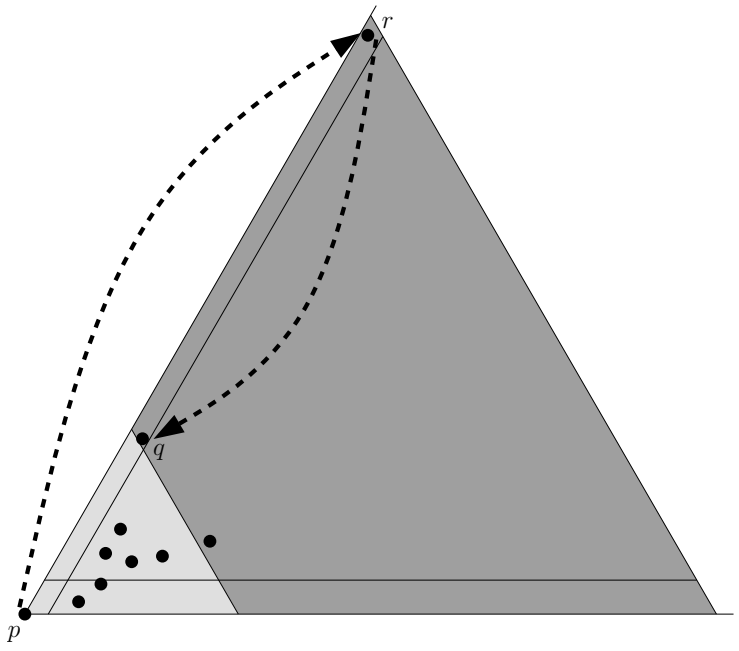


Figure 2.17: Proximate point searching path obtained by the routing algorithm.

Chapter 3

Anchored Regions and Region-Counting Distances

3.1 Introduction

In Chapter 2, we gave a definition of the ***rank difference*** between two items in a totally ordered set. If we have a set S of points in $\mathbb{R}$, we can equivalently define the rank difference between any two points p and q as the size of the intersection of S and the ball (p, q). As sets of points in $\mathbb{R}^2$ have no implicit total order, the rank difference is not defined.

The region-counting distances were introduced by Demaine, Iacono and Langerman [DIL04] as a two-dimensional equivalent for the rank difference. Given a set of points S in $\mathbb{R}^2$, and a pair of points p and q, the region-counting distance between them is the number of items of S in their neighborhood. They defined the ***neighborhood*** of a pair of points in $\mathbb{R}^2$ in terms of ***anchored region***: given a template region D with two anchor points a and b, the neighborhood of any pair (p, q) is the region obtained by translating, rotating and scaling D such that (a, b) coincides with (p, q).

The objective of this chapter is to analyze the anchored regions and the region-counting distances: give their properties, compare the region-counting distances to other similar distances, give computational and combinatorial bounds to compute disks and circles generated by inequalities using these distances.

In Section 3.2, we define the ***influence regions***, which are generalizations of anchored regions, and give a formal definition of the region-counting distances. We describe

different families of regions which will be useful all along this work. We present different properties of the regions and of the distances, and show how these properties are related.

Section 3.3 is about disks and circles defined using the region-counting distances. We describe the model in which we define the anchored regions and the corresponding region-counting circles.

These results might prove useful in many contexts. For example, efficient algorithms for computing region-counting circles could be used to speed up the constuction of proximate point location data structures described in [IL03]. Furthermore, in the proximate point searching structure [DIL04], given a current item p in the structure we can reach any other item q in time logarithmic in the region-counting distance between p and q. A region-counting disk of radius k centered at p gives all the items which can be reached in $O(\log k)$ time. This is of great interest if there are real-time constraints, as it gives a bound on what can be done without missing a deadline.

In most cases, a description of the disk itself cannot be stored efficiently for each item in the data structure as its space complexity grows at least linearly with n, but we show how to construct approximations of region-counting disks. An approach based on ϵ-approximations is proposed at the end of this chapter.

We also describe another parameterized distance family, the ***speed distances***, and we analyze the relations between both families.

Contents

3.2 Definitions and Properties

The region-counting distance between two points is defined as the number of items in their neighborhood defined by an influence region.

Definition 3.1 *An **influence region** R is a function mapping a pair (p, q) of points in $\mathbb{R}^2$ to a subset of $\mathbb{R}^2$.*

Definition 3.2 *An **anchored region** R is an influence region parameterized by a triple (a, b, D), where a and b are points in $\mathbb{R}^2$ and D is a subset of $\mathbb{R}^2$. The set $R(p, q)$ is the subset of $\mathbb{R}^2$ obtained by translating, rotating and uniformly scaling D so that a maps to p and b maps to q.*

The distance between p and q is then the cardinality of the intersection of the dataset and the neighborhood $R(p, q)$. The point set S is an implicit parameter of these distance functions.

Definition 3.3 *The **region-counting distance** [DIL04] $\mathrm{d}_R = \mathrm{d}_R^S$ parameterized by a finite point set $S \subseteq \mathbb{R}^2$ and an influence region R, is defined by $\mathrm{d}_R(p, q) = |S \cap R(p, q)|$.*

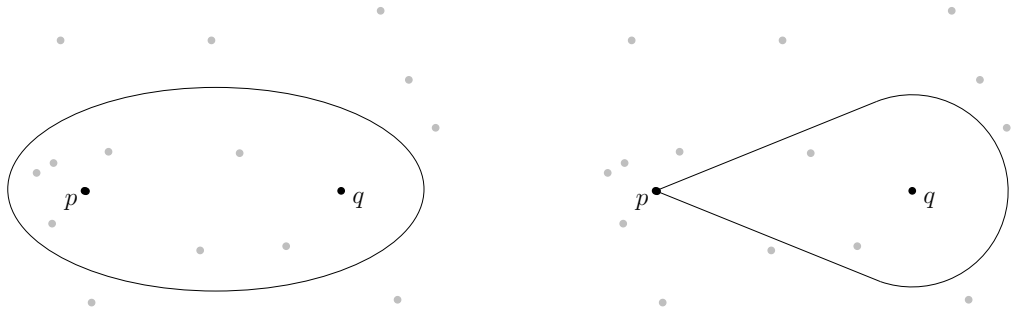


Figure 3.1: Ellipse region-counting distance $\mathrm{d}_{E_t}(p, q) = 9$, and ice-cream cone region-counting distance $\mathrm{d}_{Y_t}(p, q) = 4$.

Figure 3.1 shows two different template regions which can be used to compute the region-counting distance between two vertices p and q.

Definition 3.4 *A region-counting distance is **symmetric** if and only if it satisfies $\mathrm{d}_R^S(p, q) = \mathrm{d}_R^S(q, p)$, for any set S and any pair of points p, q.*

Definition 3.5 *A distance function is **monotone** [DIL04] if and only if $\mathrm{d}(p, q) \leq \mathrm{d}(p, r)$ for any three points p, q and r appearing in that order on a line.*

Definition 3.6 *A distance function parameterized by a data set S is **sane** [DIL04] if and only if*

$$\forall p, |\{q \in S \mid \mathrm{d}(p,q) < k\}| \leq k^{O(1)}$$

These last two properties are restrictions on region-counting distances used in [DIL04]. This last property – sanity – has been introduced to ensure that there can be a data structure with $O(\log \mathrm{d}(p,q))$ query time. In the pointer model, this can only happen if the neighborhood has polynomial size.

The properties of the distance depend on characteristics of the corresponding template region. We restrict our study to distances using anchored influence regions with p and q inside the region $R(p,q)$. In some parts of this work, we also restrict the study to star-shaped regions.

Definition 3.7 *A region R in $\mathbb{R}^2$ is **star-shaped** with respect to the point $p \in R$ if and only if*

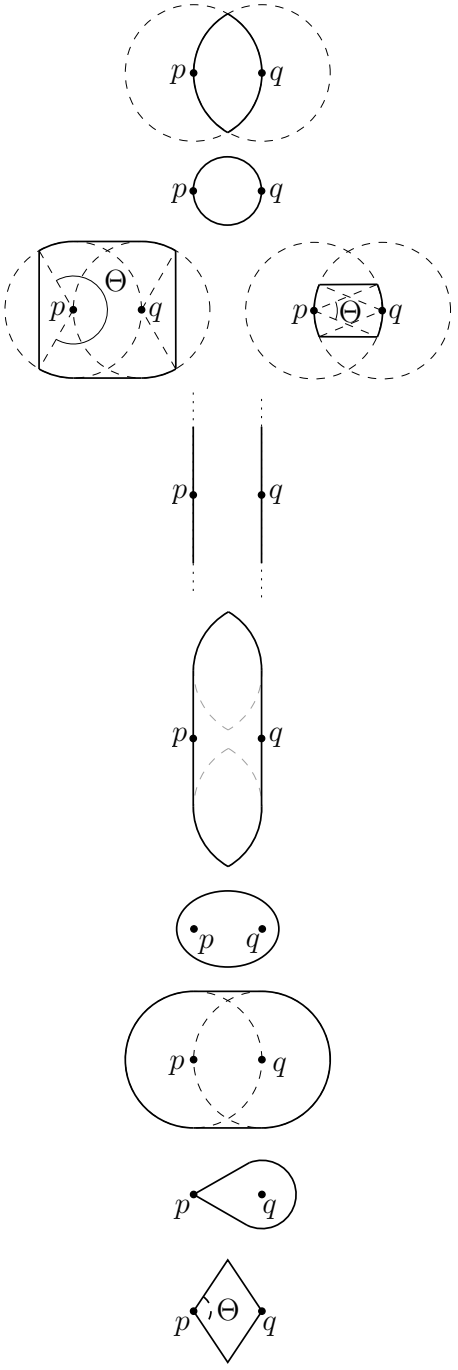
$$\forall\, q \in R, \mathrm{segment}(p,q) \subseteq R$$

These restrictions are motivated by the following direct observations:

- A region-counting distance is ***invariant under rotation, translation and uniform scaling*** if and only if it can be defined by an anchored region.
- A region-counting distance is ***symmetric*** if and only if its influence region $R(p,q)$ is symmetric with respect to the center of the line segment pq.
- A region-counting distance is ***monotone*** if and only if its anchored region is ***star-shaped*** with respect to p.

Figure 3.2 present example regions used all along this document. We denote by R_t a region which depends on a parameter t; in what follows d_2 is the Euclidean distance.

All regions of Figure 3.2 are star-shaped and yield monotone distances. The ice-cream cone is the only one for which the distance is not symmetric.



The ***lune*** $L(p, q)$ is the intersection of two disks of radius $\mathrm{d}_2(p, q)$ centered at p and q.

The ***ball*** $B(p, q)$ is the disk of diameter pq.

The ***pacman*** $P_\Theta(p, q)$ is bounded by the convex hull of two pie-wedges of angle Θ, with apex in p and in q facing each other.

The ***slab*** $S(p, q)$ is the infinite strip perpendicular to the line segment pq with p and q on the boundary.

The ***truncated slab*** $T(p, q)$ is the convex region containing two lunes on top of each other. The distance between their centers is $2 \cdot d_2(p, q)$.

Given $t \geq 1$, the ***ellipse*** $E_t(p, q)$ is the locus of the points r such that $\mathrm{d}_2(p, r) + \mathrm{d}_2(r, q) \leq t \cdot \mathrm{d}_2(p, q)$.

The ***mastercard*** $M(p, q)$ is bounded by the convex hull of two disks of radius $\mathrm{d}_2(p, q)$ centered at p and q.

Given $t < 1$, the ***ice-cream cone*** [DIL04] $Y_t(p, q)$ is bounded by the convex hull of p and a disk of radius $t \cdot \mathrm{d}_2(p, q)$ centered at q.

The ***diamond*** $D_\Theta(p, q)$ is a diamond with interior angle Θ at p and q.

Figure 3.2: Examples of regions

3.3 Distance Inequalities: Region-Counting Disks

In this section we will study the region-counting disks, which are the regions of the plane at region-counting distance less than a given k from a center. We focus on monotone region-counting distances.

Definition 3.8 *The **region-counting disk** of radius k centered at x, denoted by $D^k_{x,R}$, is the locus of the points of $\mathbb{R}^2$ at a region-counting distance less than or equal to k of the center x using the anchored region $R(p,q)$. Formally, $D^k_{x,R} = \{y \in \mathbb{R}^2 | \mathrm{d}_R(x,y) \leq k\}$.*

Definition 3.9 *The **boundary** of a region R, denoted by ∂R, is $\{x \in \mathbb{R}^2 | \forall \epsilon > 0 \in \mathbb{R}, \exists u \in R, v \notin R : \mathrm{d}_2(x,u) < \epsilon, \mathrm{d}_2(x,v) < \epsilon\}$.*

Definition 3.10 *The **region-counting circle** of radius k centered at x is $C^k_{x,R} = \partial D^k_{x,R}$.*

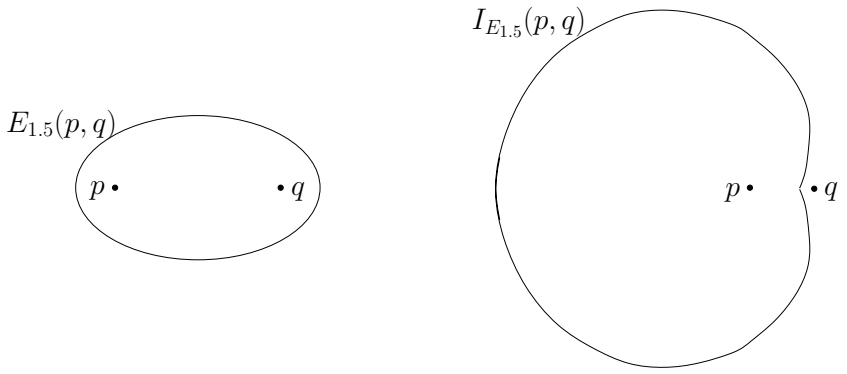


Figure 3.3: The ellipse $E_{1.5}(p,q)$ anchored region and its corresponding inverse region.

Definition 3.11 *We define the **inverse region** $I_R(p,q)$ corresponding to the region $R(p,q)$ to be $I_R(p,q) = \{x \in \mathbb{R}^2 | q \notin R(p,x)\}$.*

The inverse region $I_R(p,q)$ is the region-counting disk of radius 0 and centered at p with a singleton q as dataset, as shown on Figure 3.3. That region is an influence region parameterized by p and q, and is an anchored region if R itself is an anchored region. The ***inverse curve*** is the boundary $\partial I_R(p,q)$ of the inverse region $I_R(p,q)$.

We now recall the standard notion of level in an arrangement of x-monotone curves:

Definition 3.12 *A curve is* ***x-monotone*** *if and only if any vertical line intersects it at most once.*

Definition 3.13 *Given a fixed finite set L of x-monotone curves in the plane, the* ***level of a point*** *$x \in \mathbb{R}^2$ is the number of curves of L lying strictly below x.*

Definition 3.14 *Given a fixed finite set L of star-shaped curves with p in their kernel, the* ***polar-level*** *of a point $x \in \mathbb{R}^2$ is the number of curves crossed by the open segment px.*

An ***edge in an arrangement*** of curves is a maximal curve segment not intersected by any other curve. Since all points on an edge have the same level in the case of x-monotone curves, and have the same polar-level in the case of star-shaped curves, the ***level of an edge*** is the level of the points on that edge.

Definition 3.15 *The k^{th}* ***level in an arrangement*** *of a set of curves L is the set of all edges with level exactly k.*

Note that this definition also holds for polar-levels.

If a region-counting distance $\mathrm{d}_R(p, q)$ is monotone, it is defined by a star-shaped region $R(p, q)$. The corresponding inverse region $I_R(p, q)$ is also star-shaped because the inverse region represents the locus of the points at a distance 0 of p: as the distance $\mathrm{d}_R(p, q)$ is monotone, it does not decrease when q is moving away from p on the line pq, which implies that the region at distance 0 is star-shaped.

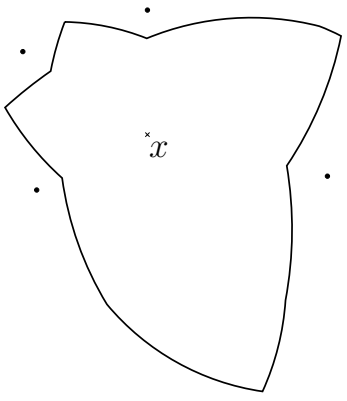


Figure 3.4: Ellipse region-counting circle $C^0_{x,E_{1.5}}$ in a set of 4 points.

Lemma 3.1 *The region-counting circle $C^k_{x,R}$ is the k^{th} polar-level in the arrangement of inverse curves $\{I_R(x,p)|p \in S\}$.*

Proof We claim that for a given set S and a center x, a point p is at distance $\mathrm{d}_R(x,p) \leq k$ if and only if $|\{q \in S|p \notin I_R(x,q)\}| \leq k$. This can be derived as follows:

$$\begin{aligned} & p \notin I_R(x,q) \\ \iff \quad & p \notin \{y \in \mathbb{R}^2 | q \notin R(x,y)\} \\ \iff \quad & q \in R(x,p) \end{aligned}$$

If $|\{q \in S|p \notin I_R(x,q)\}| > k$, there would be more than k points contained in the influence region $R(x,p)$. This means that the point p is outside of the inverse region of at most k items of the set.

As the regions and inverse regions are star-shaped, the point x is contained in the inverse region of all the n points of the set. A point q is outside of the inverse region of at most k items if the number of inverse curves crossed by the segment xq is less than or equal to k. This corresponds to the points at polar-level less than or equal to k, and the circle is thus the k^{th} polar-level in an arrangement of curves. □

Theorem 3.2 *Given two anchored regions R and R', and their corresponding inverse regions I_R and $I_{R'}$, the inverse region $I_{R\cup R'}$ of the anchored region $R \cup R'$ is $I_R \cap I_{R'}$ and the inverse region $I_{R\cap R'}$ of the anchored region $R \cap R'$ is $I_R \cup I_{R'}$.*

Proof By definition,

$$I_{R\cup R'}(p,q) = \{x \in \mathbb{R}^2 | q \notin R(p,x) \cup R'(p,x)\}$$

which is equivalent to

$$I_{R\cup R'}(p,q) = \{x \in \mathbb{R}^2 | q \notin R(p,x)\} \cap \{x \in \mathbb{R}^2 | q \notin R'(p,x)\}$$

Similarly,

$$I_{R\cap R'}(p,q) = \{x \in \mathbb{R}^2 | q \notin R(p,x) \cap R'(p,x)\}$$

Proof (Continued) which is equivalent to

$$I_{R\cap R'}(p,q) = \{x \in \mathbb{R}^2 | q \notin R(p,x)\} \cup \{x \in \mathbb{R}^2 | q \notin R'(p,x)\}$$

□

3.3.1 Model

As we want to bound the complexity of the region-counting circles, and as this complexity depends on the considered region, we describe in this section the model used to encode the regions.

In our model, a region is defined by a function

$$f_{R(p,q)}(x) = f_R(p,q,x) : \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$$

such that

$$R(p,q) = \{x \in \mathbb{R}^2 | f_{R(p,q)}(x) \geq 0\}$$

and

$$\partial R(p,q) = \{x \in \mathbb{R}^2 | f_{R(p,q)}(x) = 0\}$$

Then, the corresponding inverse region is $I_{R(p,q)} = \{x \in \mathbb{R}^2 | f_{R(p,x)}(q) < 0\}$, and we can also define the function associated with $I_{R(p,q)}$ by $f_{I_R(p,q)}(x) = -f_{R(p,x)}(q)$.

We restrict $f_{R(p,q)}(x)$ to be a polynomial function of bounded degree. We also consider regions and inverse regions resulting of a constant number of Boolean operations on two or more (possibly) different regions defined by functions. Note that all regions considered in this thesis satisfy this restriction.

3.3.2 Complexity of Region-Counting Circles

Definition 3.16 *The **complexity of a path or cycle** in an arrangement of curves is the number of edges in the path or cycle. By extension, the complexity of a disc or circle is the complexity of the cycle that forms its boundary.*

To study the complexity of the region-counting circles, we proceed in two steps. We first bound the maximum number of intersections between the inverse curves that can

occur in our model, then we bound the complexity of the levels in the arrangements of the regions composing the region-counting circles.

Definition 3.17 *A set of curves $\mathcal{C} = \{c_1, c_2, \ldots\}$ is **s-intersecting** if and only if*

$$\forall i \neq j, c_i, c_j \in \mathcal{C}, |\{c_i \cap c_j\}| \leq s$$

In what follows, $\mathcal{C}_R$ is $\{\partial R(p,q) | p, q \in S\}$. In our model, the regions either are defined by a function, or are Boolean compositions of such regions. We must thus consider both cases to bound the number of intersections of any region. As the function $f_{R(p,q)}(x)$ is a polynomial function of bounded degree, we know that the curve $\partial R(p,q)$ is continuously differentiable and that any pair of distinct curves in $\mathcal{C}_R$ intersects a constant number of times. This is due to the fact that the intersection of any pair of curves is the solution of a polynomial equation of bounded degree. The same observation holds for $\mathcal{C}_{I_R}$ as the function $f_{I_R(p,q)}(x)$ is also a polynomial function of bounded degree.

A bound on the number of intersections between regions defined as Boolean compositions is given in the next lemma.

Lemma 3.3 *If a set of curves $\mathcal{C} = \{c_1, c_2, \ldots\}$ is s-intersecting, then every set of curves $\mathcal{C}' = \{F_1(\mathcal{C}), F_2(\mathcal{C}), \ldots\}$ is st^2-intersecting, where the F_i are curves obtained by a Boolean formula of size bounded by t on items of $\mathcal{C}$.*

Proof For all pair $F_i(\mathcal{C}), F_j(\mathcal{C}) \in \mathcal{C}'$, the intersections occur on segments of the curves contained in $\mathcal{C}$. As $\mathcal{C}$ is s-intersecting, and that the Boolean formula is applied to at most t items of $\mathcal{C}$, there are st^2 possible intersections. □

Complexity of all region-counting circles with radius less than k

As previously mentioned, $C^k_{x,R}$ is the k^{th} polar level in an arrangement of translated and rotated copies of a star-shaped inverse curve, with p in its kernel.

We can thus express our problem in polar coordinates, where every inverse curve is described by a function $\rho = \tau_{I_R}(\theta)$ giving, for every angle θ, the distance between p and the curve ∂I_R in that direction.

The functions are polynomials of bounded degree, which implies that they are s-intersecting; therefore we can apply the following bound:

Theorem 3.4 (by Sharir [Sha91]) *The complexity of the first k-levels in an arrangement of general s-intersecting curves is in $O(\lambda_s(n/k)k^2)$, where $\lambda_s(x)$ is the maximum length of (m, s)-Davenport-Schinzel sequences. This bound is tight in the worst case.*

Simple examples, such as a dataset of points on a circle using the disk with diameter p, q as region, exhibit a $\Omega(kn)$ complexity.

Due to the behavior of λ_s, which is extremely slow-growing, we get that the complexity of the first k-levels in an arrangement of s-intersecting curves is in $O(kn \cdot 2^{\alpha^s(n/k)\log\alpha(n/k)})$ [AG86], where α is the inverse Ackermann function, which is lower than 4 for all reasonable value of n [SA96].

Complexity of region-counting circles with radius k

In this subsection, we prove that under some conditions on the region R, the complexity of a region-counting circle of radius k is at least as large as the complexity of the k^{th} level in a line arrangement.

The exact complexity of these levels is a long-standing open problem. It was originally analyzed by Lovász and Erdös *et al.* [Lov71, ELSS73] as the ***k-sets problem***: given a set S of n points in the plane, how many subsets of size k can be defined by the intersection of a half-plane with S. This is equivalent to finding the worst-case complexity of the k^{th} level in an arrangement of n lines [Ede87]. The best bounds we know are $n2^{\Omega(\sqrt{\log k})}$ by Tóth [TÓ0] and $O(nk^{1/3})$ by Dey [Dey98].

An upper bound for our problem was given by Chan [Cha05] who proved that in any planar arrangement of s-intersecting curves, the k^{th} level has $O(n^{2-1/(2s)})$ complexity. This bound can be applied directly if the problem is expressed in polar coordinates.

A randomized algorithm to efficiently construct a level has been presented by Har-Peled [HP00]; it has an expected running time of $O(\lambda_{s+2}(m+n)\log n)$, where m is the complexity of the level being constructed. It can be applied here to construct the disk of radius k.

Theorem 3.5 *Let $\rho = \tau_{I_R}(\theta)$ be the polar function describing an inverse curve $\partial I_R(p, q)$. If there exists an angle θ such that $\tau'_{I_R}(\theta)$, the derivative of $\tau_{I_R}(\theta)$, is defined and nonzero, and $\tau_{I_R}(\theta)$ is continuous and differentiable in the neighborhood of θ, then for any set L of lines, there exists a set S of points such that the complexity of the region-counting circle of radius k is at least the complexity of the k^{th} level in the arrangement of L.*

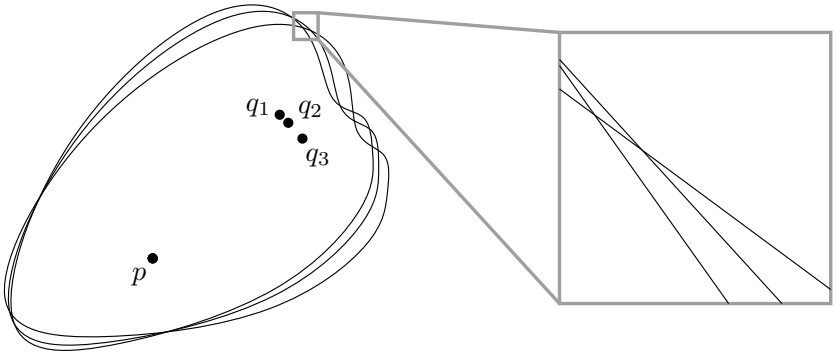
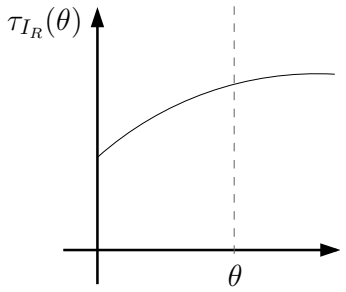


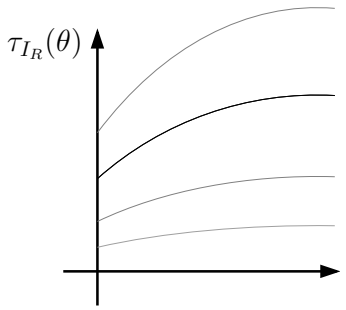
Figure 3.5: Simulation of a line arrangement with continuously differentiable curves.

Proof We simulate a line arrangement L by using sufficiently small curve segments of the boundary of the region-counting disk. Figure 3.5 illustrates the process.

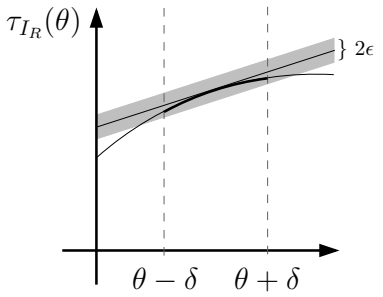
We consider the polar function $\tau_{I_R}(\theta)$. We know that in the neighborhood of the angle θ, the function is continuous.



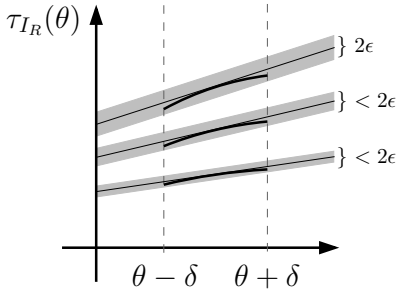
Given a region $I_R(p, q)$, if we move the point q along a line, the region I_R is scaled around the center p. In the polar representation, this corresponds to scaling the polar function along the vertical axis.



Proof (Continued) Suppose that the function is continuous on the interval $[\theta-\delta, \theta+\delta]$. For all ϵ, there exists a vertical scaling $c_{\max}$ such that the polar function $c_{\max} \cdot \tau_{I_R}(\theta)$ is at a distance at most ϵ from its tangent at θ, for every angle in the interval $[\theta-\delta, \theta+\delta]$.



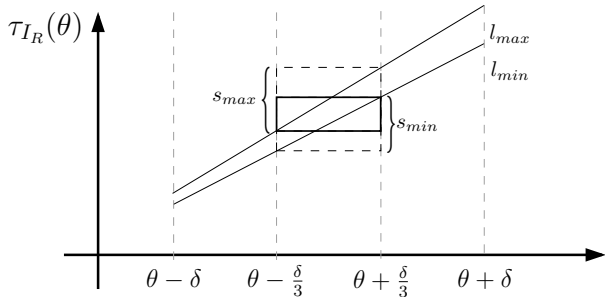
The distance to that tangent decreases when scaling down the function, i.e. if $c_i < c_{\max}$, the distance of the polar function $c_i \cdot \tau_{I_R}(\theta)$ to its tangent is less than ϵ over the interval.



In the original setting, when we rotate the point q around the point p, the region $I_R(p, q)$ is rotated by the same angle. In the polar representation, this corresponds to an horizontal translation.

Let $L_{\max}$ be the line tangent to $c_{\max} \cdot \tau_{I_R}$ at θ. Let $l_{\max}$ be the segment of $L_{\max}$ in the interval $[\theta - \delta, \theta + \delta]$. Because $\tau'_{I_R}(\theta) \neq 0$, $l_{\max}$ is not horizontal. In the interval $[\theta - \delta/3, \theta + \delta/3]$, there is a horizontal strip $s_{\max}$ in which for every point u, we can translate $l_{\max}$ horizontally such that it passes through u. We can scale down the polar function while ensuring that the tangent is still passing through that strip, to obtain another line segment that we call $l_{\min}$.

Proof (Continued) There is also a strip $s_{\min}$ in the interval for which every point can be intersected by $l_{\min}$ while allowing only horizontal translations.



Since $l_{\min}$ passes through $s_{\max}$, the two strips intersect, and there is a rectangle B in which every point is incident to some horizontal translation of $l_{\min}$ and of $l_{\max}$.

Thus by moving the point q in our original setting, we can simulate with tangents to our polar function any line segment with a slope in the interval $[\alpha_{\min}, \alpha_{\max}]$, where $\alpha_{\min}$ and $\alpha_{\max}$ are the slopes of $l_{\min}$ and $l_{\max}$, respectively.

Now, let L be an arrangement of n lines in general position. We can scale it and rotate it, so that the lines have slopes in the range $[\alpha_{\min}, \alpha_{\max}]$ and all intersections lie inside box B. We can apply these transformations without changing the complexity of the levels of the arrangement. For each line $\ell \in L$, there is a translation and scaling of $\tau_{I_R}(\theta)$ whose tangent at θ coincides with ℓ. This translation and scaling corresponds to some point q in the original setting. Furthermore by the argument above, the translated and scaled curve τ_{I_R} is at distance at most ϵ from ℓ in the box B. By choosing ϵ smaller than half the distance between any intersection point and any other line in the arrangement L, we can ensure that the combinatorial structure of L is identical to that of the arrangement of curves. Thus we can simulate the arrangement by carefully choosing n points in the original setting.

Since the region-counting disk contain a level in the curve arrangement inside B, this completes the proof. □

The condition on ∂I_R in this theorem can be expressed as a condition on the outer limit of the region defining the region-counting distance. If ∂I_R contains a continuously differentiable segment then ∂R also contains such a part.

Theorem 3.6 *Let $\partial I_R(p, q)$ be a curve whose polar function is continuously differentiable except for a finite set of r points. If the derivative of the polar function is zero for every differentiable point of the curve, then the complexity of the region-counting circle of radius k is $O(nr)$.*

In our model, the regions and curves defined by functions on $\mathbb{R}^2$ are continuously differentiable. However, the Boolean operations on these curves can add non-differentiable points at every intersection between two curves. The number of such points is thus bounded by the maximal number of intersections between the curves in our model, which is constant as proved before.

Proof The regions whose curves are described in the theorem are composed of unions of pie-wedges, as a zero derivative on a curve segment means that the segment is a circular arc. The region can be described by at most r pie-wedges.

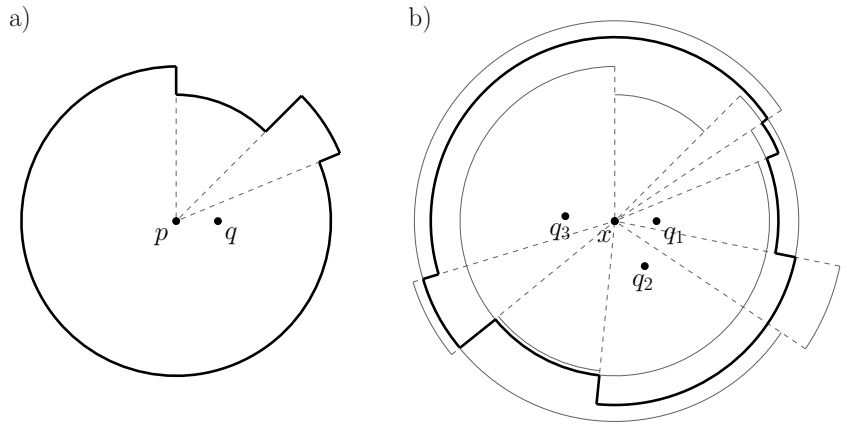


Figure 3.6: a) Inverse region $I_R(p, q)$ corresponding to Theorem 3.6. b) Region-counting circle $D^2_{x,R}$ corresponding to the set $\{q_1, q_2, q_3\}$.

Proof (Continued)

A pair of such curves using the same center x can only intersect in directions where one of the curves is non-differentiable, that is on the boundary of a pie-wedge. These are thus $2r$-intersecting curves.

Given a set of n points, a level in the arrangement of inverse curves is composed of circular arcs. There can be up to $2rn$ discrete directions corresponding to non-differentiable points of polar functions, and thus there are at most $2rn$ circular arcs. □

A simple algorithm can be used to construct the region-counting disks corresponding to the hypotheses of Theorem 3.6. We sort all the directions $\{\alpha_1, \alpha_2, \ldots\}$ corresponding to non-differentiable points. We store the curves in a balanced binary search tree sorted by their distance to the center in any direction in the range $[\alpha_1, \alpha_2]$. The k^{th} curve in that sorted order for the range $[\alpha_1, \alpha_2]$ is the boundary of the disk in that direction.

Then we go on with the next range $[\alpha_2, \alpha_3]$. We know that the only curves which are not sorted correctly are the ones associated to the non-differentiable point corresponding to α_2; we update the tree by reinserting the curves which are not correctly sorted (at a cost of $O(\log n)$), we select the k^{th} curve and keep going with the next range. The total complexity is $O(nr \log nr)$ and $O(n \log n)$ for initial sorting and construction of the binary tree, and $O(nr \log n)$ in total to update the tree (nr ranges at a cost of $O(\log n)$ each) and to determine the k^{th} curve, leading to a $O(nr \log nr)$ algorithm.

To conclude, depending if we are in Case 1 (see Theorem 3.5) or Case 2 (see Theorem 3.6), we get the following results:

	Case 1	Case 2
Upper bound	$O(n^{2-1/(2s)})$	$O(nr)$
Lower bound	$n2^{\Omega\left(\sqrt{\log k}\right)}$	$\Omega(nr)$
Construction	$O(\lambda_{s+2}(m+n) \log n)$	$O(nr \log nr)$

for a region-counting circle of radius k, and if the inverse region are s-intersecting.

3.3.3 Approximation of Region-Counting Distances and Disks

A natural question is to find an algorithm to compute the distance, and to find the bounds on the complexity of this operation. Computing the region-counting distance is equivalent to performing a range counting query, which is a well-studied problem [AE99]. As range counting queries are expensive to solve exactly, we propose here a method to approximate the distances, disks and circles.

We first bound the VC-dimension [VC71] corresponding to the regions. We consider the range space (S, R), where R represents $R(p, q)$ for all p and q in $\mathbb{R}^2$, with regions representable in the model defined above.

The VC-dimension [VC71] of a range space (S, R) is the cardinality of the largest set of points which can be shattered by the regions in R, i.e. the largest set of points such that any subset $s \subseteq S$ is equal to $r \cap S$ for some $r \in R$.

We know that every region defined by a polynomial function of bounded degree has a constant VC-dimension [VC71] and that every region defined by a Boolean function of constant size on regions with constant VC-dimension has a constant VC-dimension. This ensures that the VC-dimension of any region in our model is a constant, say d.

Vapnik and Chervonenkis [VC71] defined the following:

Definition 3.18 *An* ϵ***-approximation*** *of a set* S *is a subset* s *of* S *such that, for any range* R*,*

$$\left| \frac{|s \cap R|}{|s|} - \frac{|S \cap R|}{|S|} \right| \leq \epsilon$$

They proved that for all range spaces of VC-dimension d, there exist ϵ-approximations s of size $O(d/\epsilon^2 \log 1/\epsilon)$. We apply this result: let $d_R^\epsilon(p,q)$ be the region-counting distance defined on the ϵ-approximation $s \subset S$ of size $O(d/\epsilon^2 \log 1/\epsilon)$. By [VC71], we have that for every region $R(p,q)$

$$\left| \frac{|s \cap R(p,q)|}{|s|} - \frac{|S \cap R(p,q)|}{|S|} \right| \leq \epsilon$$

Thus, $d_R^\epsilon(p,q)|S|/|s|$ is an approximation of the region-counting distance $d_R(p,q)$. Deterministic algorithms to construct ϵ-approximations exists [Mat99]. Alternatively we can select $O(d/\epsilon^2 \log 1/\epsilon)$ points at random in the original set and it will be an ϵ-approximation with high probability, as shown in [VC71, HW87].

Definition 3.19 *An* ϵ***-approximated region-counting disk*** $D_{x,R}^{k,\epsilon}$ *is a region satisfying the following constraints:*

$$y \in D_{x,R}^{k,\epsilon} \Rightarrow \mathrm{d}_R(x,y) \leq k + n\epsilon$$

and

$$y \notin D_{x,R}^{k,\epsilon} \Rightarrow \mathrm{d}_R(x,y) \geq k - n\epsilon$$

Theorem 3.7 *The disk* $D_{x,R}^{k|s|/|S|}$ *using an* ϵ*-approximation* $s \subset S$ *is an* ϵ*-approximated region-counting disk* $D_{x,R}^{k,\epsilon}$ *for the full set* S*.*

In other words, to have an approximated disk of radius k, we only need to take a random sample of the points in S with fixed cardinality, and construct a disk of radius $k|s|/|S|$. The precision, however, depends on the size of the set.

Proof By definition of the region-counting circles, every point p on the boundary of the disk $\partial D_{x,R}^{k|s|/|S|}$ using the set s is at a region-counting distance equal to $k|s|/|S|$ from x. By definition of the region-counting distances, this means that there are exactly $k|s|/|S|$ points of s in $R(x,p)$. As R is a range space of bounded VC-dimension, and that s is an ϵ-approximation, we know that :

$$\left|\frac{|s \cap R(x,p)|}{|s|} - \frac{|S \cap R(x,p)|}{|S|}\right| \leq \epsilon$$

which is equivalent to

$$\left|\frac{k|s|}{|S|}\frac{1}{|s|} - \frac{|S \cap R(x,p)|}{|S|}\right| \leq \epsilon$$

$$\left|\frac{k}{|S|} - \frac{|S \cap R(x,p)|}{|S|}\right| \leq \epsilon$$

$$|k - |S \cap R(x,p)|| \leq \epsilon|S|$$

$$d_R(x,p) \leq k \pm \epsilon|S|$$

which is the definition of an ϵ-approximated region-counting disk.

We still have to prove that points inside the disk $D_{x,R}^{k|s|/|S|}$ satisfy the definition. As the boundary is at distance exactly $k|s|/|S|$ and that the distance is monotone in s and S, we know that all points inside the disk are at a distance at most $k|s|/|S|$ in s and at most $k \pm \epsilon|S|$ in S. This completes the proof. □

While we just proved that ϵ-approximated region-counting disks exist and that they have constant size, we need to put our result into perspective:

The constant here is huge, and these ϵ-approximated region-counting disks are not really useful in practice. As an example, if we have a set of 10000 points, and we would like to compute our distances with an additive error less than or equal to 500 (which corresponds to $\epsilon = 0.05$), we get that our set s should be of size 520 times the VC-dimension (which is here a small constant, but even if the VC-dimension is 1, the size of the disks is still important). In other words, we proved that such approximations exist, but our solution is probably far from optimal. An interesting problem would be to derive upper and lower bounds on the quantity of information required to define a region-counting circle with a given precision. Another open problem is to determine whether we could get approximated region-counting disks where the error is relative to the radius of the disk k, as opposed to the additive error presented here.

3.4 Speed Distances

3.4.1 Definition

The speed distances are another class of parameterized distance functions, depending on a function $f(p)$ that represents the inverse of the instantaneous speed at point p.

Definition 3.20 *A **speed distance*** $\mathrm{d}_f(p,q)$ *is parameterized by a function* $f : \mathbb{R}^2 \mapsto \mathbb{R}^+$*, and is defined as* $\mathrm{d}_f(p,q) = \int_0^{\mathrm{d}_2(p,q)} f(\sigma(t))dt$ *where* $\sigma(t) = p + t\frac{q-p}{\mathrm{d}_2(p,q)}$.

In other words, the speed distances are line integrals on the segment pq. We further insist that $k = \int_{\mathbb{R}} f(x,y)dxdy$ is finite. If f is bounded and integrable, this is ensured by setting $f(x,y)$ to 0 outside of some region.

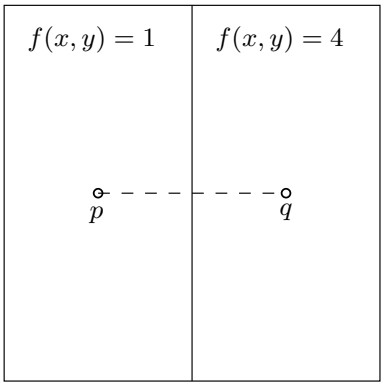


Figure 3.7: Example speed function.

Figure 3.7 shows an example where the speed function has been defined on a square with surface equal to one. On one half of the square, the speed function is equal to one, while it is equal to four on the other half. The points p and q are at Euclidean distance $1/2$. Then, their speed distance is:

$$\begin{aligned} \mathrm{d}_f(p,q) &= \frac{1}{4} + \frac{4}{4} \\ &= \frac{5}{4} \end{aligned}$$

3.4.2 Relations between RCD and Speed Distances

We introduce the ***area factor*** of the region-counting distances needed to prove a relation between both distance families.

Definition 3.21 *The **area factor** C of a region-counting distance based on the anchored region $R(p,q)$ is defined as*

$$C = \frac{\text{area}(R(p,q))}{(\text{d}_2(p,q))^2}$$

where $\text{d}_2(p,q)$ is the Euclidean distance.

We know that C is a constant because the anchored region $R(p,q)$ is obtained by a combination of translations, rotations and uniform scalings on an initial anchored region, and all of these transformations preserve the aspect ratio of the original anchored region.

The following theorem shows a relation between the region counting distances and the speed distances.

Theorem 3.8 *Let d_f be a speed distance with $k = \int_{\mathbb{R}^2} f(x,y)dxdy < \infty$; let $\text{d}^S_{R_t}$ be a region-counting distance, defined by $R_t = (a,b,D_t)$ with D_t strictly containing the closed segment from a to b and depending on parameter t. If the set S of n points is generated randomly using the density function $g(x,y) = f(x,y)/k$ and if $\lim_{t\to 0}\ C_t = 0$, then*

$$\forall p,q, \lim_{t\to 0} \frac{1}{n\, C_t\ \text{d}_2(p,q)}\ \text{E}\left[\text{d}^S_{R_t}(p,q)\right] = \frac{1}{k}\text{d}_f(p,q)$$

where C_t is the area factor of $\text{d}_{R_t}(p,q)$.

This theorem shows that for a given speed distance, we can generate a dataset S defining a region-counting distance which will approximate the speed distance as n tends to infinity, and as the considered neighborhood region shrinks.

Figure 3.8 shows at typical anchored region which can be used for our problem. We must define the ellipse parameter t differently here, as the area factor of the ellipse is 0 when $t = 1$, which is not precisely what we describe here. This can be easily solved by using $t' = t - 1$.

The following proves that we can approximate the line integral over our speed probability field by sampling the field over a thin region. As the sample size goes to infinity, and the region gets thinner, the approximation approaches the line integral.

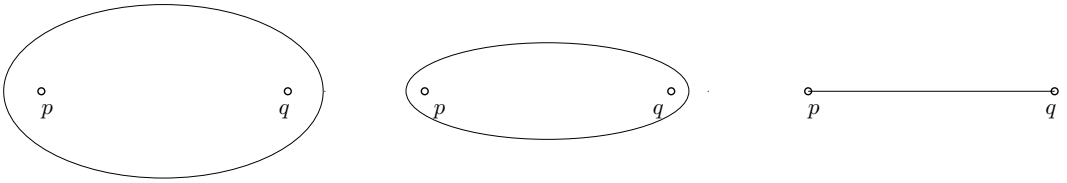


Figure 3.8: Region containing the segment from p to q, with surface decreasing to 0 when its parameter decreases.

Proof Let X_i be the i^{th} random point drawn according to the probability density function g. Let us denote by I_i the following indicator random variable:

$$I_i = \begin{cases} 1 & \text{if } X_i \in R_t(p,q), \\ 0 & \text{otherwise.} \end{cases}$$

We have $E[I_i] = \int_{R_t(p,q)} g(x,y)dxdy$. Then $\mathrm{d}_{R_t}(p,q)$ is a random variable equal to $\sum_i I_i$. The expectation is

$$\begin{aligned} E\left[d^S_{R_t}(p,q)\right] &= E\left[\sum_{i=1}^{n} I_i\right] \\ &= \sum_{i=1}^{n} E[I_i] \\ &= nE[I_i] \\ &= n\int_{R_t(p,q)} g(x,y)dxdy \end{aligned}$$

Now the limit simplifies to

$$\begin{aligned} \lim_{t\to 0} \frac{1}{n} \frac{\mathrm{E}[\mathrm{d}_{R_t}(p,q)]}{C_t\ \mathrm{d}_2(p,q)} &= \lim_{t\to 0} \frac{1}{n} \frac{\mathrm{E}[\mathrm{d}_{R_t}(p,q)]\ \mathrm{d}_2(p,q)}{\mathrm{area}(R_t(p,q))} \\ &= \lim_{t\to 0} \frac{\int_{R_t(p,q)} g(x,y)dxdy\ \mathrm{d}_2(p,q)}{\mathrm{area}(R_t(p,q))} \\ &= \lim_{t\to 0} \frac{\int_{R_t(p,q)} g(x,y)dxdy\ \mathrm{d}_2(p,q)}{\mathrm{area}(R_t(p,q))} \\ &= \lim_{t\to 0} \frac{1}{k} \frac{\int_{R_t(p,q)} f(x,y)dxdy\ \mathrm{d}_2(p,q)}{\mathrm{area}(R_t(p,q))} \end{aligned}$$

Proof (Continued)

$\frac{\int_{R_t(p,q)} f(x,y)dxdy}{\text{area}(R_t(p,q))}$ gives the mean value of f on the region R_t.

$\frac{\int_0^{\mathrm{d}_2(p,q)} f(\sigma(t))dt}{\mathrm{d}_2(p,q)}$ gives the mean value of f on the path $\sigma(t)$.

The integral on the surface tends to the integral on the path if the following conditions are satisfied: f must be continuous, derivable, integrable, and the anchored region R_t must tend continuously to the segment (p, q) when t tends to 0. Then,

$$\begin{aligned} \lim_{t\to 0} \frac{1}{n} \frac{\mathrm{E}[\mathrm{d}_{R_t}(p,q)]}{C_t\ \mathrm{d}_2(p,q)} &= \frac{1}{k}\int_0^{\mathrm{d}_2(p,q)} f(\sigma(t))dt \\ &= \frac{\mathrm{d}_f(p,q)}{k} \end{aligned}$$

□

3.5 Applications and Perspectives

Region-counting distances and anchored regions are useful tools to analyze and solve various geometric problems. In Chapter 2, we presented geometric graphs, as well as queries which can be applied to these graphs. Region-counting distances and anchored regions have been used to define efficient point searching [DIL04] and point location [IL03] structures and algorithms. In the remainder of this thesis, we complete that previous work by using these regions and distances in two different ways.

First, as the rank difference in $\mathbb{R}$, region-counting distances give information on the structure of the data set. They do not only give proximity measures, but also indicate the density of the set around a pair of points. This is very interesting in light of what has been done for distribution-sensitive data structures.

We said in Subsection 2.3.2 that these structures had running times expressed as functions of the sequence of queries. We can reasonably think that some data structures have running times depending on the density or of the proximity of items: for instance, it seems logical that path queries on maps in dense cities are more complex than in small villages. This is what we propose to study in the next chapter.

Secondly, we presented in Subsection 2.2.2 proximity graphs. Some of the graphs described use the emptiness of some region around a pair of vertices as their proximity condition to have an edge. We propose to generalize this approach in Chapter 5: we study the generic case of proximity graphs defined by empty anchored regions.

Chapter 4

Local Properties

4.1 Introduction

We presented in Section 2.2 ***geometric graphs***, where the vertices are points in the plane, and in particular the graphs generated by sets of points. The properties defined on such graphs are most of the time expressed as a function of n, the number of vertices of the graph. For instance, nearest neighbor graphs have n edges. When we consider a family of graphs, a property can also be expressed as an upper bound on some graph quantity: for instance the in-degree of the vertices in a Θ-graph is $O(n)$. In this context, this is what we call ***global properties***.

These global properties do not always reflect what we need: we are not always concerned by the whole graph, and for instance, an upper bound on the size of a path expressed as a function of n gives no information if we perform a query on close points in a huge and spread-out data set.

We say that a property is ***local*** if it is expressed as a function of the number of vertices geometrically close to the studied ones. To formalize this, we must define which vertices are close. Therefore we use the ***region-counting distances***, as described in Chapter 3: these are distance functions parameterized by a finite point set (the vertices of our graph) in which the distance between two points is the number of items contained in a region surrounding these points. We illustrate locality with the study of the ***local diameter***. The ***diameter*** of a graph is the maximum size (number of edges) of the shortest path between any pair of vertices. We define the corresponding ***local diameter*** as the same quantity, but as a function of the region-counting distance between the pair of points. Informally, a local property is a property that holds in all subgraphs induced by the neighborhoods around pairs of vertices of the graph.

The study of local properties is essential because the queries that we detailed in Section 2.3 have running times which can sometimes be expressed as local properties. The use of region-counting distances allows us to bound the time needed for these queries in a more flexible way than just ensuring that it has, or not, the dynamic finger property generalized to the plane.

At the end of this chapter, we summarize the results of our analysis of local properties for some geometric graphs. From there, we directly deduce improved bounds on the query time for close queries.

The reader might be familiar with another definition of locality in graph theory, such as in ***locally connected graphs***. In general, for a property $\mathcal{P}$, a graph is locally $\mathcal{P}$ if and only if the property $\mathcal{P}$ holds for the subgraph induced by any vertex and its neighbors.

While this is different from what is done in this chapter, the definitions are similar. The main difference is that we use template regions to define the neighborhood of vertices, while in the graph-theoretic sense the neighborhood is composed of the vertices adjacent to the current one.

Contents

4.2 Definition

Let $\mathcal{G}$ be the set of all possible geometric graphs. A graph property $\mathcal{P}$ is a subset of $\mathcal{G}$. We say that a graph G satisfies $\mathcal{P}$ if $G \in \mathcal{P}$.

Definition 4.1 *A **local property** $\mathcal{P}$ for a graph $G = (V, E)$ is an inequality between a function $f(G, p, q)$ and the region-counting distance $\mathrm{d}_R(p, q)$ between p and q in V.*

Definition 4.2 *A geometric graph G is said to **have the local property** $\mathcal{P}$ if this relation holds for all pairs (p, q) in V.*

Intuitively, a property is local if it depends only on vertices close to the ones considered. For instance, the ***local diameter*** of a graph is an upper bound on the size of the shortest path between any pair of vertices, expressed as a function of a region-counting distance between these vertices; the ***local complexity*** of path queries is the time required to perform a path query expressed as a function of the region-counting distance between its endpoints, *etc.*

The next subsection shows why t-spanner graphs are good candidates to study locality. We then analyze three local quantities: the local diameter, the point query complexity and the path query complexity. A typical desirable local property is to have a local diameter in $O(\log \mathrm{d}_R(p, q))$, we will see that this bound is satisfied by some of the graphs considered here.

4.3 Analysis of Local Properties

4.3.1 A Property of t-Spanner Paths

We first recall the notion of t-spanner path, first introduced in Subsection 2.2.1.

Definition 4.3 *A **t-spanner path** P between the vertices u and v satisfies*

$$\frac{\mathrm{d}_G(P)}{\mathrm{d}_2(u,v)} \leq t$$

A ***t-spanner graph*** is a graph in which there exists a t-spanner path between every pair of vertices.

Lemma 4.1 *Let G be a t-spanner graph; let p and q be two vertices of this graph. Any t-spanner path between p and q is composed of edges and vertices contained in an ellipse $E_t(p,q)$ whose foci are p and q and whose parameter is t.*

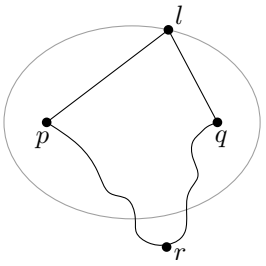


Figure 4.1: Ellipse with foci p and q and parameter $t = 1.5$.

Proof An ellipse with foci p and q and parameter t is the locus of the points l whose sum of the distance to the two foci is equal to t times the Euclidean distance between the foci (see Figure 4.1). As any path from p to q going through point r outside the ellipse has length at least $\mathrm{d}_2(p,r) + \mathrm{d}_2(r,q)$ which is more than $t \cdot \mathrm{d}_2(p,q)$ by definition of the ellipse, we know that it cannot be the t-spanner path, which has length at most $t \cdot \mathrm{d}_2(p,q)$. The ellipse is convex and all the vertices of the path are in the ellipse, thus the path edges are contained in the ellipse. □

4.3.2 Local Diameter

The property presented in the previous subsection allows to bound the local diameter for *every* t-spanner graph.

Theorem 4.2 *The local diameter is* $O(\mathrm{d}_{E_t}(p,q))$ *for every* t*-spanner.*

Proof We know that there is a t-spanner path in this ellipse, and the size of the shortest path is smaller than or equal to that of the t-spanner path. □

Note however that this bound is not tight, and that the use of *ad-hoc* approaches for different t-spanners lets us improve this result. For the ordered Θ-graph and the skip list spanner, we can show the following:

Theorem 4.3 *There exists a path of expected size* $O(\log \mathrm{d}_{E_t}(p,q))$ *between any pair of points* (p,q) *in an ordered* Θ*-graph or a skip list spanner, where* t *is the spanning ratio of the corresponding graph.*

Proof Let $G = (V, E)$ be an ordered Θ-graph and $G' = (V', E')$ where $V' = V \cap E_t(p,q)$ be the ordered Θ-graph generated by the vertices in the ellipse and the same ordering. We consider a path from p to q in G and G'. We will simulate the use of the path algorithm described in Subsection 2.3.4.

Beginning from both ends p and q, the path algorithm adds an edge and a vertex to the path at each step. By Lemma 4.1 and the fact that the underlying Θ-graph is a t-spanner, we know that this vertex is in the ellipse: the edge we use is part of the Θ-path, which is a t-spanner path.

To choose the vertex by which the path will be extended, the ordering is used. As the relative order is the same, the same choice will be made in V and V'. To choose which edge is added, the algorithm finds the cone in which the target item lies. Given a common origin and destination in V and V', the chosen cone will be the same, as this choice depends only on the coordinates of these two vertices. There is no vertex in V' which is not in V, and every vertex in the ellipse is in V and V'. The path algorithm selects the cone containing the target: the edge present in this cone is the same in G as in G'. If it was not the case, either a closer vertex would be present in the cone in V', or the closest vertex in V would not be in V' which is not possible because this vertex is in the ellipse.

Proof (Continued) This proves that every step of the algorithm will give the same result in both graphs: same cones and same edges will be chosen at the first step, leading to the same recursive approach.

It has been shown in [BGM04] that the diameter of an ordered Θ-graph with random permutation of the vertices as ordering is $c \log n$ with probability $1 - n^{-\Omega(c)}$. Since the graph G' is a valid ordered Θ-graph, and its ordering π' is induced by π and is thus a random permutation, the diameter of G' is $O(\log \mathrm{d}_{E_t}(p, q))$ with high probability. The path has thus a length of $O(\log \mathrm{d}_{E_t}(p, q))$.

The proof for the skip list spanner is similar: we consider the path in the subgraph, which is the same as in the graph if the common vertices are at the same level in the graph and the subgraph. The bound on the diameter is given by [AMS94]. □

4.3.3 Path Query

The ***path query*** consists in, given a graph G and a pair of query vertices (p, q), finding a t-spanner path between these vertices, if such a path exists. We present here bounds on the complexity of finding such paths.

For the Θ-graph and for the proximate point searching graph, a simple argument can be used: by Lemma 4.1, we know that the path is contained in an ellipse. The path size is thus at most the cardinality of the set of vertices in the ellipse, which is the ellipse region-counting distance. As every step of the path takes constant time using the Θ-path algorithm, the complexity of the path query is $O(\mathrm{d}_{E_t}(p, q))$.

For the ordered Θ-graph and the skip list spanner, the path query algorithm consists in beginning with the two ends of the path. The ordered Θ-graph path algorithm extends the path by the end with the highest order (the last vertex added), following the edge in the cone containing the other endpoint of the path, until it is reached. For the skip list spanner, recall that we allowed in our model (see subsection 2.3.3) to define at each vertex an access structure in the pointer model, and to regroup outgoing edges in classes that we can access individually. A simple doubly-linked list to access the different levels of the skip list spanner is enough: at each step we will only get to the next or the previous level. The skip list spanner path algorithm considers the highest level in which both ends are present. The path is extended by the end with the lowest level, following the edge in the cone containing the other endpoint of the path, until it is reached.

Theorem 4.4 *Path queries are answered in time $O(\log \mathrm{d}_{E_t}(p,q))$ with high probability in an ordered Θ-graph or a skip list spanner.*

Proof By Theorem 4.3, we know that for each pair of vertices (p,q), the length of the path between them is $O(\log \mathrm{d}_{E_t}(p,q))$. Each step of the path construction algorithms (see Figures 2.14 and 2.15) can be achieved in constant time, resulting in an $O(\log \mathrm{d}_{E_t}(p,q))$ bound for path queries in skip list spanners and ordered Θ-graphs. □

4.3.4 Point Query

The point query (see Section 2.3) complexity can be expressed as a local property, i.e. as a function of $\mathrm{d}_R(p,q)$. We recall that the point query consists of starting from the previous query point and reaching the query point as quickly as possible while following edges of the graph.

Typically, this method can be used when we know that close queries occur frequently. Close vertices influence the query time, while distant vertices are not taken into account to answer the query. This is the approach used in the proximate point searching data structure [DIL04], whose point query time is $O(\log \mathrm{d}_{Y_t}(p,q))$ where d_{Y_t} is the ice-cream cone region-counting distance [DIL04]. The ice-cream cone distance is in some sense better than the ellipse distance, because $\forall t_0, \exists t_1 : Y_{t_1}(p,q) \subseteq E_{t_0}(p,q)$ while the converse is not true (see Figure 4.2).

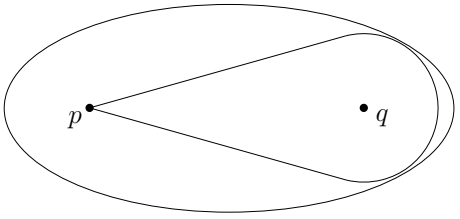


Figure 4.2: There is an ice-cream cone in every ellipse.

For the other graphs, we know that the number of nodes visited to reach the query is at most the length of a shortest path between the previous query and the current one. An upper bound on the point query complexity is the product of the length of the path

and the complexity of the routing algorithm at each node. This gives an $O(\mathrm{d}_{E_t}(p,q))$ complexity for the Θ-graph and skip list spanner.

Theorem 4.5 *The time complexity of point queries is bounded by the time to perform a path query between the previous and the current query in Θ-graphs and proximate point searching graphs.*

Proof The path query algorithm can be used to construct a path. We just need to know one of its endpoints (the previous query) and the direction to the current query point. □

For the skip list spanner, we cannot use the path algorithm of Figure 2.15 to answer point queries, because the algorithm constructs the path from both ends. This does not make sense for the point query, as we do not have a pointer to the target (we only have its position), and thus we cannot construct the path from that end. This is why the point query complexity is upper bounded by the one for the Θ-graph. It is open whether a better search algorithm exists for the skip list spanner.

The ordered Θ-graph is not considered here because it is not strongly connected; as our model is a variant of the pointer model, we have to follow the pointers corresponding to the edges, which is not possible here. A solution would be to double the pointers, and to consider the graph as undirected. However, this would potentially lead to $n-1$ pointers going out of the same vertex, resulting in inefficient algorithms.

4.4 Results

	Θ-graph	Ordered Θ-graph
Edges	$O(n)$	$O(n)$
In-degree	$O(n)$	$O(n)$
Out-degree	$\leq \frac{2\pi}{\theta}$	$\leq \frac{2\pi}{\theta}$
Spanning Ratio	$t \leq \frac{1}{1-2\sin(\frac{\Theta}{2})}$	$t \leq \frac{1}{\cos(\Theta)-\sin(\Theta)}$
Path query	$\mathbf{O(d)}$	$\mathbf{O(\log d)}$ w.h.p.
Point query	$\mathbf{O(d)}$	Not always possible
Diameter	$O(n)$	$O(\log n)$ w.h.p.
Local diameter	$\mathbf{O(d)}$	$\mathbf{O(\log d)}$ w.h.p.
Region Used	Ellipse E_t	Ellipse E_t

	Skip List Spanner	Proximate Point Search.
Edges	$O(n)$ w.h.p.	$O(n \log n)$
In-degree	$O(n)$	$O(n \log n)$
Out-degree	$O(\log n)$ w.h.p. ($\frac{2\pi}{\theta}$ per level)	$O(\log n)$
Spanning Ratio	$t \leq \frac{1}{1-2\sin(\frac{\Theta}{2})}$	Contains Θ-graph
Path query	$\mathbf{O(\log d)}$ w.h.p.	$\mathbf{O(d)}$
Point query	$\mathbf{O(d)}$	$\mathbf{O(\log d)}$
Diameter	$O(\log n)$ w.h.p.	$O(\log n)$
Local diameter	$\mathbf{O(\log d)}$ w.h.p.	$\mathbf{O(\log d)}$
Region Used	Ellipse E_t	Ice-Cream Cone Y_t

Figure 4.3: Comparison of properties of geometric graphs. d is the region-counting distance using the given region. Bounds are worst cases, or hold with high probability when mentioned (w.h.p.). Bold font indicates our results

The table given by Figure 4.3 shows properties of various graphs where n is the number of vertices and d is the region-counting distance between the two considered items using the corresponding region: the ellipse or the ice-cream cone [DIL04]. Other results summarized in that table come from the original papers describing these

graphs [DIL04, KG92, AMS94, BGM04], and are detailed in Subsection 2.2.1. For the point query, the two items are the previous and the current query, for the path query these are the path ends.

There is not only one path query algorithm. Here, we consider algorithms constructing t-spanner paths: we must ensure that the path computation will not use vertices and edges outside a shape. For the point query, the algorithm begins with the previous query and searches for a path to the current query. It is not always possible to use path query algorithms to perform point queries, as some of these algorithms construct the path from both ends. As one can see, a small local diameter is not sufficient to have a small path query or point query complexity. But it is a necessary condition: we cannot find a path containing k edges in less than k steps.

The proximate point searching graph combines small diameters and good point query performance, but at the expense of a larger number of edges.

Chapter 5

Empty Region Graphs

5.1 Introduction

We consider here ***proximity graphs*** (see Subsection 2.2.2 for definitions and examples), where two vertices form an edge if they are close, according to various definitions of proximity.

Previous work on these graphs traditionally consisted in the introduction of one or more graph families, followed by different contributions analyzing their properties and applications. For instance, in [MS80] the properties of the Gabriel graphs were used to analyze geographical data sets; in [BDEK06] authors analyze the spanning ratio of some proximity graphs. Surprisingly, the natural opposite approach does not seem to have been considered: to try to find a practical method which, given a set of desired graph properties, returns a corresponding proximity graph family. For this, we have to define a class of proximity graphs general enough to encompass many useful graphs, but simple enough to be analyzed.

We define the ***empty region graphs*** (or ***ERG***) as the general class of proximity graphs using the following definition of proximity: two vertices are close, and thus form an edge, if and only if some neighborhood around them is empty. The region corresponding to the neighborhood is a parameter defining the exact family of graphs in the class.

One motivation of our work was to design proximity graphs that are invariant under translations, rotations and uniform scaling. It can be shown that this property is satisfied if and only if the region defining the neighborhood around two points is obtained by

translating, rotating and uniformly scaling a template region. We further focus on convex and symmetric regions, hence on undirected proximity graphs.

The properties of these graphs are determined by the choice of the template region. More specifically, we say that a given template region ***satisfies*** a property if *for all* point sets V the graph generated using that region satisfies the property. Graph properties that are monotone with respect to either edge removal or addition are good candidates for investigation, as monotone properties that are satisfied by a template region are satisfied by all template regions included in it in the case of edge addition, or containing it in the case of edge removal.

This naturally raises the issue of template regions that are extremal with respect to the inclusion partial order among regions. We show for instance that the lune, defined as the intersection of two disks of radius $\mathrm{d}_2(p, q)$ and respective centers p and q, is the unique maximal region ensuring the connectivity of the graph. We call these extremal regions ***tight regions***. However, because the inclusion relation is not a total order, tight regions need not be unique. Tight regions can somehow be seen as a deterministic geometric analogue to thresholds for monotone properties studied in random graph theory [FK96]. We prove that these tight regions are defined for every monotone graph property.

The main contribution of this chapter is to provide a generalized approach to the problem of finding proximity graphs corresponding to properties. Also, even though some results were previously known, the analysis of the uniqueness of the template regions has not been considered before. And this strengthens the characterization: not only can we find template regions satisfying the properties, but we can also, given any template, tell if some properties (for which we know all tight regions) are guaranteed.

We end this chapter by proposing a simple method to construct empty region graphs using all the template regions presented in Chapter 3 and corresponding to the different properties studied all along this document.

Contents

5.2 Related Works

The study of graph-theoretic and combinatorial properties of proximity graphs is not new. Numerous references [Cim92, JT92, AM92, GS69, KR85, IS85] present extensive analyses of what properties can be satisfied with which proximity graph.

Restricted families of template regions have also been studied before. The β-skeletons are constructed using a template region based on a family of regions depending on a parameter β, and were extensively analyzed in that context [KR85]: given a value of β, we know directly if the graph is guaranteed to be connected, planar, acyclic, etc. and we can thus use that approach to achieve our primary goal: determine the proximity relation satisfying a property for every set of points. But the class of regions considered is very restrictive in β-skeletons, as it is always the intersection of two disks, while we propose an approach for every convex and symmetric region.

Theorems 5.5, 5.10 and 5.16 of this work are extensions of the work on β-skeletons, where we consider the uniqueness and the tightness of the template region among *all* convex symmetric regions.

Numerous works studied drawability problems for proximity graphs. For instance, Bose *et al.* [BDLL95, BLL96] studied extensively the following graph drawing question: given a tree T, does there exist a set of points in the plane such that a proximity graph induced by these points is isomorphic to T. They were able to provide a complete characterization for trees if the proximity graph is a relative neighborhood graph, relatively closest graph or a Gabriel graph, they also propose a partial characterization for β-skeletons. Lubiw and Sleumer [LS93] showed that all maximal outerplanar graphs can be represented as a Gabriel graph and a relative neighborhood graph; they show what geometric properties must be satisfied to layout a set of vertices such that the corresponding proximity graph is maximal outerplanar.

Other similar drawability problems were surveyed in [DLL95]. These studies are fundamentally different from what is done here: we work on a set of vertices in the plane on which we find proximity relations, while these papers propose a way to layout the vertices of a given graph G so that a proximity graph for those vertices is G.

Thus, given a set of points, we analyze the properties resulting from our choice of proximity relation, while these studies analyze, given a proximity relation, the properties required for the set of points to be drawable.

5.3 Definition

We define the empty region graphs as follows:

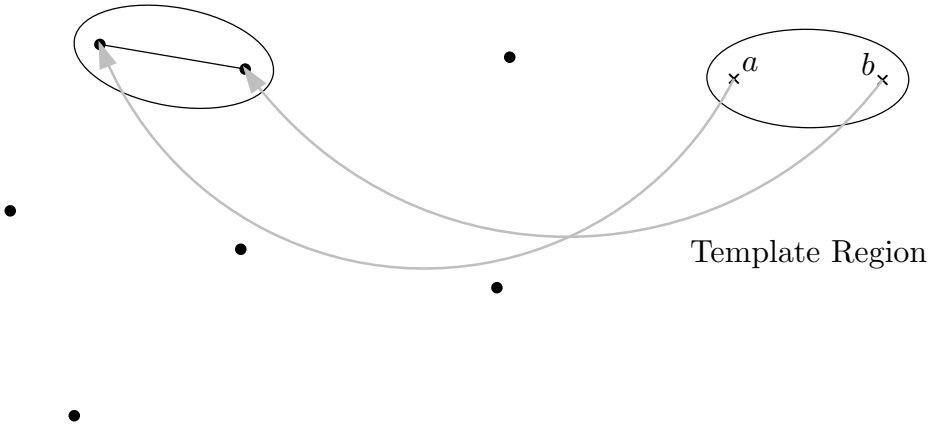
Definition 5.1 *An **empty region graph** $ERG_R(V) = (V, E)$ parameterized by an influence region R is a graph where V is a finite subset of $\mathbb{R}^2$ and*

$$\forall p, q \in V : (p, q) \in E \Leftrightarrow R(p, q) \cap S \setminus \{p, q\} = \emptyset.$$

This definition encompasses Θ-graphs, β-skeletons, nearest neighbor graphs, and more. However in what follows, we restrict ourselves to using anchored regions parameterized by triples (a, b, D) where D is convex and symmetric with respect to segment ab, and with a and b contained in D. The region is closed if not specified otherwise. Using only anchored regions is necessary and sufficient to guarantee the invariance of the graph structure under translation, rotation and uniform scaling of the set of points. We further restrict ourselves to regions symmetric with respect to the center $(a + b)/2$, hence undirected graphs.

Figure 5.1 shows the construction of an empty region graph with a template region satisfying our restrictions.

a)



b)

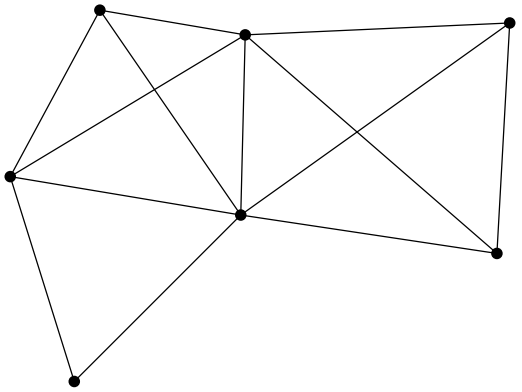


Figure 5.1: a) Mapping of the anchored region on a pair of vertices. As it is empty, the edge exists. b) Resulting empty region graph, after checking for every pair of vertices.

5.4 Monotone Properties and Tight Regions

In this section and the following we analyze which anchored regions must be used if we want to make sure that the resulting graph has some desirable graph or geometric properties. Let us first define formally the properties we want to consider:

Definition 5.2 *A **graph property** $\mathcal{P}$ on a family of graphs $\mathcal{G}$ is a set $\mathcal{P} \subseteq \mathcal{G}$. A graph $G \in \mathcal{G}$ **has property** $\mathcal{P}$ if $G \in \mathcal{P}$.*

Definition 5.3 *A graph property $\mathcal{P}$ is **monotone with respect to edge addition** (or **to edge removal** respectively) if and only if $\forall\, G = (V, E), G' = (V, E'), E \subseteq E'$ (or $E \supseteq E'$ respectively): $G \in \mathcal{P} \Rightarrow G' \in \mathcal{P}$.*

Our definition of monotonicity is slightly different from the one commonly used in graph theory. Usually, this is stated as follows: a property is monotone if and only if it is closed upon taking subgraphs. We add the symmetric definition with properties monotone upon taking supergraphs.

Definition 5.4 *An anchored region R **satisfies** a graph property $\mathcal{P}$ if and only if for all $V \subset \mathbb{R}^2$ finite, $ERG_R(V) \in \mathcal{P}$.*

Figure 5.4 shows empty region graphs defined by various anchored regions. Note that the graphs in Figure 5.4a, 5.4b and 5.4c are planar, while the graphs in Figure 5.4b, 5.4c and 5.4d are connected. Of course these are just examples, what we want is to prove that for *every* possible set of vertices, some properties are satisfied by some anchored regions. Moreover, we want to find the extremal – or tight – regions corresponding to some property:

Definition 5.5 *A convex and symmetric anchored region R is **tight** for a graph property $\mathcal{P}$ monotone with respect to edge addition (or to edge removal respectively) if and only if R satisfies $\mathcal{P}$ and for all convex and symmetric anchored region $R' \supset R$ (or $R' \subset R$ respectively), R' does not satisfy $\mathcal{P}$.*

Note that the monotonicity of the property with respect to edge removal implies that any region containing a tight region as a subset satisfies the property as well. On the other hand, for regions that are strictly contained in a tight region, one can always find a set of points generating a graph that does not have the property. A similar observation holds the other way round for properties that are monotone with respect to edge addition. Another definition of a tight region for a monotone property is a convex and symmetric

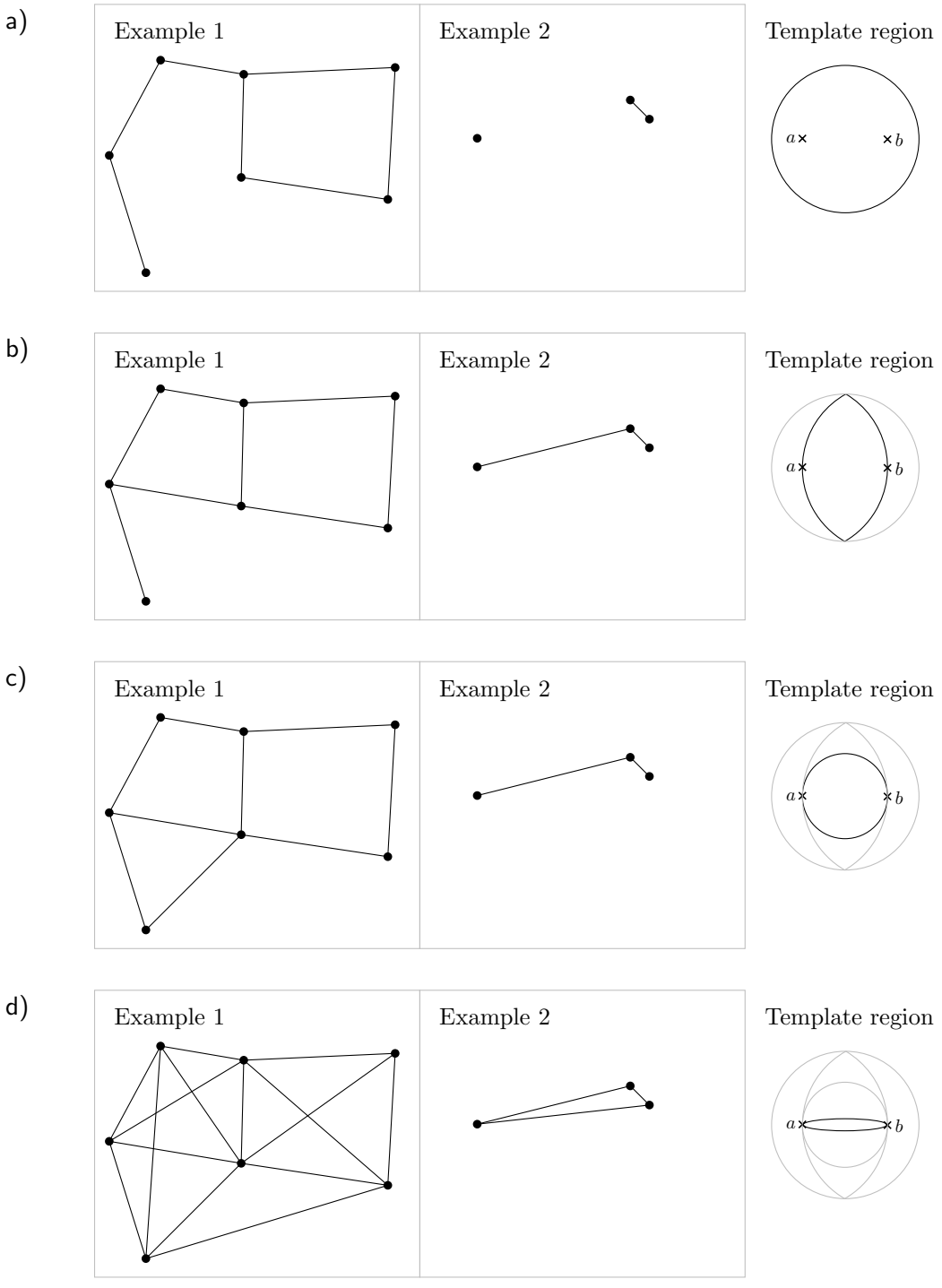


Figure 5.2: Examples of empty region graphs on two sets of points, using various anchored regions.

region that satisfies the property and is extremal for the inclusion partial order among the considered type of regions. We now show that tight regions always exist.

Lemma 5.1 *For every graph property $\mathcal{P}$ monotone with respect to edge removal (or to edge addition respectively) and region R satisfying $\mathcal{P}$, either R is tight, or there exists at least one region $R' \subset R$ (or $R' \supset R$ respectively) which is tight.*

Proof We first consider properties monotone with respect to edge removal. Let R be an anchored, convex and symmetric region satisfying property $\mathcal{P}$, and let $\mathcal{C}$ be the set of anchored, convex and symmetric regions $\{R' \subseteq R \mid R' \text{ satisfies } \mathcal{P}\}$.

A ***chain*** in that partially ordered set, with the relation of inclusion, is a sequence $(R_0, R_1, \ldots)$ such that for all i, $R_i \supseteq R_{i+1}$. A *lower bound* of that sequence is

$$R_\infty = \bigcap_i R_i$$

which by definition is also convex and symmetric.

We first show that every chain in $\mathcal{C}$ has a lower bound in $\mathcal{C}$. The proof is obtained by contradiction. Let R be a region in $\mathcal{C}$ and $(R = R_0, R_1, \ldots)$ be any chain in $\mathcal{C}$ beginning with R. We assume that R_∞ does not belong to $\mathcal{C}$ and derive a contradiction.

If R_∞ does not belong to $\mathcal{C}$, it implies that it does not satisfy $\mathcal{P}$. Thus there exists a set of vertices V such that $ERG_{R_\infty}(V) \notin \mathcal{P}$.

Let m_i be the size (number of edges) of the graph $ERG_{R_i}(V)$. We have $m_0 \leq m_1 \leq \ldots < m_\infty$. The last relation is strict, because otherwise there is a k such that $ERG_{R_k}(V) = ERG_{R_\infty}(V)$ and thus R_k does not satisfy $\mathcal{P}$ while it is in the chain.

As R_∞ does not satisfy the property, while it was the case for every item in the chain, the problem has to come from an edge present in $ERG_{R_\infty}(V)$ but not in $ERG_{R_i}(V)$ for any i.

Then, there exist three points p, q and r in V such that for all i, it holds that $r \in R_i(p, q)$ and $r \notin R_\infty(p, q)$. But as R_∞ is the intersection of R_i over all i, r is in $R_\infty(p, q)$. This proves that R_∞ satisfies $\mathcal{P}$ and is thus in $\mathcal{C}$.

By applying Zorn's Lemma [Cie97] to the set of anchored convex and symmetric regions, we know that if every chain in $\mathcal{C}$ has a lower bound *in* the set, then there is at least one minimal region R' in $\mathcal{C}$, contained in any region R, and satisfying the property. A region R' is minimal with respect to $\mathcal{C}$ if and only if no strict subset of R' is in $\mathcal{C}$. Thus $R' \subseteq R$ is a tight region for $\mathcal{P}$.

Proof (Continued) For properties monotone with respect to edge addition, the same reasoning applies. We consider upper bounds in chains and maximums regions. We can apply Zorn's Lemma, and we have to show that the chain $(\ldots, R_{-1}, R_0 = R)$ has always an upper bound in the set. The upper bound is $R_{-\infty} = \bigcup_i R_{-i}$, which is convex and symmetric. Then we show by contradiction that some point has to be included in $R_{-\infty}$ but excluded from R_i for every i while it is the union of these regions. □

Knowing tight regions for useful properties is important in practice, because it allows to check quickly if the properties we wish to obtain are satisfied or not. The uniqueness of a tight region is even more important, because knowing a single tight region R does not, in general, give any information on the properties guaranteed by regions that simultaneously do not contain R and are not contained in R. If the tight region R is unique, any region that does not contain R does not satisfy the property, even if it is not strictly included in R.

Lemma 5.2 *Let $\mathcal{P}$ be a monotone property with respect to edge removal (edge addition resp.) and R be the unique tight region satisfying that property. Every region $R' \not\supseteq R$ ($R' \not\subseteq R$ resp.) does not satisfy $\mathcal{P}$.*

Proof We consider properties monotone with respect to edge removal, the other case being symmetric. By contradiction, we suppose that R' satisfies $\mathcal{P}$. By Lemma 5.1 either R' is tight, which is impossible as R is the unique tight region, or there exists a tight region $R'' \subset R'$ which satisfies the property. In that case, we know that $R'' \not\subset R$, because otherwise R would not be tight. Thus, R'' and R are two tight regions, contradicting the uniqueness of R. □

Now given a set of compatible properties we wish the graph to satisfy, we can easily construct a region guaranteeing these properties. In the case of convex and symmetric regions and properties that are monotone with respect to edge removal, this is achieved by taking the convex hull of the union of the tight regions for each property. In some cases, this region can be proved to be extremal with respect to the inclusion partial order among the considered type of regions.

Lemma 5.3 *Let $\mathcal{P}$ and $\mathcal{P}'$ be two monotone properties with respect to edge removal, and R and R' two regions satisfying $\mathcal{P}$ and $\mathcal{P}'$ respectively. Then the convex hull of $R \cup R'$ satisfies $\mathcal{P} \cap \mathcal{P}'$. Furthermore, if R and R' are the unique tight regions for $\mathcal{P}$ and $\mathcal{P}'$, then the convex hull of $R \cup R'$ is tight and unique for $\mathcal{P} \cap \mathcal{P}'$.*

Proof The first statement derives directly from the definition. If R and R' are the unique tight regions, then from Lemma 5.2 any subset of $R \cup R'$ fails to satisfy either $\mathcal{P}$ or $\mathcal{P}'$. Hence $R \cup R'$ is tight for $\mathcal{P} \cap \mathcal{P}'$.

For every other convex region R'', such that $R'' \supseteq R$ and $R'' \supseteq R'$, we know that R'' contains the convex hull of $R \cup R'$. If R'' does not contain the convex hull of $R \cup R'$, either $R'' \not\supseteq R$ and R'' does not satisfy $\mathcal{P}$, or $R'' \not\supseteq R'$ and R'' does not satisfy $\mathcal{P}'$, and thus R'' does not satisfy $\mathcal{P} \cap \mathcal{P}'$. This shows that R is unique. □

A similar Lemma holds for properties that are monotone with respect to edge addition, the proof is omitted as it uses the same arguments as the previous one:

Lemma 5.4 *Let $\mathcal{P}$ and $\mathcal{P}'$ be two monotone properties with respect to edge addition, and R and R' two regions satisfying $\mathcal{P}$ and $\mathcal{P}'$ respectively. Then $R \cap R'$ satisfies $\mathcal{P} \cap \mathcal{P}'$. Furthermore, if R and R' are the unique tight regions for $\mathcal{P}$ and $\mathcal{P}'$, then $R \cap R'$ is tight and unique for $\mathcal{P} \cap \mathcal{P}'$.*

5.5 Geometric Properties

Here we consider geometric properties, i.e. properties depending on the position of the vertices.

5.5.1 Planar Embedding

The following theorem shows that the graph embedding obtained by drawing edges as straight line segments is always planar if and only if the region contains the ball $B(p,q)$ of diameter $\mathrm{d}_2(p,q)$. It strengthens a result of [Cim92], as it shows the uniqueness of the tight region.

Theorem 5.5 *The closed ball B is the unique tight region ensuring a planar embedding.*

Proof If any region $R(p,q)$ contains the ball $B(p,q)$, then the graph is a subset of the Delaunay graph, which is planar. Otherwise, let a be a point on the circle of diameter $\mathrm{d}_2(p,q)$ that is not in $R(p,q)$. By the symmetry of center $(p+q)/2$, we know that there exists a point b which is also not in $R(p,q)$. Using the set $\{p,q,a,b\}$, there is an edge between p and q as $R(p,q)$ does not contain any other point. There is also an edge between a and b as $R(a,b)$ does not contain p and q by symmetry of axis a,b. Thus, crossing edges exist and the embedding of the graph is not planar. □

5.5.2 Spanning Ratio

Recall that $\mathrm{d}_G(P)$ denotes the length of the path $P = (u_1, u_2, \ldots, u_k)$ in graph G, i.e. $\mathrm{d}_G(P) = \sum_{i=1}^{k-1} \mathrm{d}_2(u_i, u_{i+1})$. The distance in graph G between any vertex pair (u,v), denoted by $\mathrm{d}_G(u,v)$, is the minimal length among all paths between u and v. The spanning ratio of graph G is the maximum value of $\mathrm{d}_G(u,v)/\mathrm{d}_2(u,v)$ over all vertex pairs (u,v) in G.

We say that the spanning ratio of a graph generated by a set of n points is unbounded whenever it cannot be bounded by a constant independent of n, i.e. it can be made arbitrarily large for sufficiently large n.

Theorem 5.6 *For all anchored regions R with non-empty interior, there exists a real number $\alpha > 0$ such that for all n, there exists a set $V \subset \mathbb{R}^2$, with $|V| = n$, so that the spanning ratio of $ERG_R(V)$ is $\Omega(n^\alpha)$.*

Proof Let us consider a convex and symmetric region $R(p, q)$ with non-empty interior. Any such region contains a point r that is not on the segment pq. Every $R(p, q)$ contains a triangle with the points p, q and r as vertices. Symmetrically, a triangle pqr' is also contained in $R(p, q)$, where r' is a point symmetric to r with respect to the line pq.

By definition of the β-skeleton, for $0 < \beta < 1$, $I_\beta(p, q)$ is the intersection of the two disks of radius $\mathrm{d}_2(p, q)/(2\beta)$ passing through both p and q. The parameter β can be chosen such that $I_\beta(p, q)$ is contained in the union of the two triangles. Thus $R(p, q) \supseteq I_\beta(p, q)$ and $ERG_R(V)$ is a subgraph of the β-skeleton of V.

It has been shown in [BDEK06, WLM+03] that the spanning ratio of every β-skeleton with $\beta > 0$ is $\Omega(n^\alpha)$, with n the number of vertices in the graph and α a function of β. □

When we introduced the ERG, one of our motivations was to find a class of rotation-invariant proximity graphs that would produce t-spanners for any $t > 1$. The well-known Θ-graph, which is not invariant to rotations, exhibits a constant spanning ratio. The theorem above shows that it is not possible to find any convex and symmetric anchored region other than the segment that yields ERGs with a constant spanning ratio.

5.6 Forbidden Subgraphs

A property $\mathcal{P}$ defined by a forbidden subgraph F is the set of all graphs not having any subgraph isomorphic to F. Such a property $\mathcal{P}$ is monotone with respect to edge deletions. We denote by $F \subseteq F'$ the fact that F' has a subgraph isomorphic to F. The union $F \cup F'$ of two graphs $F = (V, E)$ and $F' = (V', E')$ with $V \cap V' = \emptyset$ is $(V \cup V', E \cup E')$. We say that region R forbids a subgraph F if for all set V, $ERG_R(V)$ has no subgraph isomorphic to F.

Lemma 5.7 *If the region R is tight for a forbidden subgraph F and for a forbidden subgraph $F' \supseteq F$, then it is tight for all forbidden subgraphs F'' with $F \subseteq F'' \subseteq F'$.*

Proof The region R forbids F'', as its subgraph F is forbidden by R. Suppose that R is not tight for forbidding F''. Then by Lemma 5.1 there exists a region $R' \subset R$ forbidding F''. As F'' is a subgraph of F', R' forbids F', leading to a contradiction as this means that R is not tight for forbidding F'. □

Theorem 5.8 *If $\mathcal{R}$ is the set of tight regions for a forbidden subgraph F and if $\mathcal{R}'$ is the set of tight regions for a forbidden subgraph F', then the set of tight regions for the forbidden subgraph $F \cup F'$ is $\{R \in \mathcal{R} \cup \mathcal{R}' | \forall R' \neq R \in \mathcal{R} \cup \mathcal{R}' : R \not\supseteq R'\}$.*

Proof We know that every region R in the set defined by the theorem satisfies the property, as it forbids either F or F', which ensures that $F \cup F'$ is forbidden. To derive a contradiction, we assume without loss of generality that some $R \in \mathcal{R}$ which belongs to the set is not tight for forbidding $F \cup F'$, the case with $R \in \mathcal{R}'$ not tight being symmetric.

By Lemma 5.1, if R is not tight, it implies that there exists a region $R' \subset R$ forbidding $F \cup F'$. We first note that R' does not forbid F since R is tight for forbidding F. We also know that R' does not forbid F' otherwise there would be a tight region R'' contained in R' forbidding F'. In that case, since $R'' \subseteq R' \subset R$, R would not have been taken in the set of regions for forbidding $F \cup F'$.

So R' forbids neither F nor F'. Thus there exist two sets of points V and V' such that $ERG_{R'}(V) \supseteq F$ and $ERG_{R'}(V') \supseteq F'$. Now we consider the union C of the regions $R'(p, q)$ for all pairs (p, q) in V. If R' is bounded, then $\mathbb{R}^2 \setminus C$ is not empty, and has compact subsets of arbitrary large surfaces.

Proof (Continued) If rotated and placed far enough, the other set V' of points can be embedded in such a subset, yielding a subgraph of the form $F \cup F'$, contradicting the fact that R' forbids $F \cup F'$. By our convexity and symmetry assumptions, if R' is not bounded and different from the whole plane, then it can be either a horizontal or a vertical slab. In both cases, the conclusion holds, for a union of a finite number of slabs always leaves uncovered angles.

To prove that the list is complete, we can use the same contradiction. The existence of a tight region for forbidding $F \cup F'$ that would not forbid F nor F' allows a similar construction. □

5.6.1 k-Star

Theorem 5.9 *The closed pacman $P_{4\pi/k}$ is a tight region forbidding a k-Star, for all $k \geq 2$.*

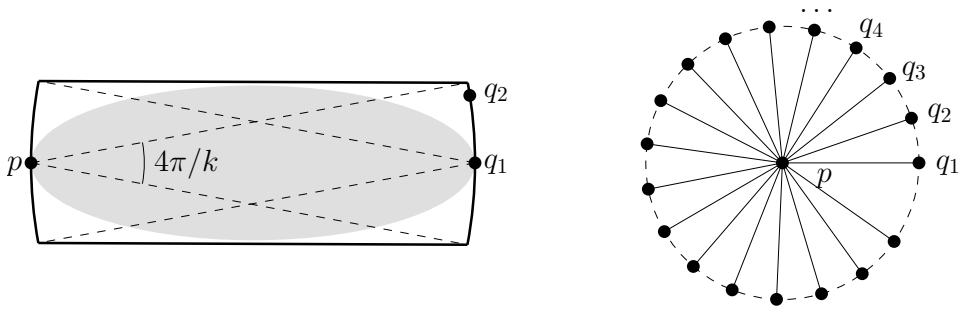


Figure 5.3: Region and set of points where a vertex has degree more than k.

Proof We show by construction that if we take any convex and symmetric region R smaller and included in the pacman $P_{4\pi/k}$, then we can construct a set V such that $ERG_R(V)$ contains a k-Star.

Let $R(p, q_1)$ be a region contained in $P_{4\pi/k}(p, q_1)$ as shown in Figure 5.3. We place a point q_2 on the boundary of $P_{4\pi/k}(p, q_1)$ and out of $R(p, q_1)$. Note that the angle $\angle q_1 p q_2$ is less than $2\pi/k$, say $2\pi/k - \delta$.

Now, we add the points $q_3, q_4, \ldots q_k$ with $\mathrm{d}_2(p, q_i) = \mathrm{d}_2(p, q_1)$, such that the angle $\angle q_i p q_{i+1}$ is equal to $2\pi/k + \delta/k$ for all $1 < i \leq k$. Thus $q_1 \notin R(p, q_i)$ for all $1 < i < k$.

Proof (Continued) Every edge (p, q_i) is present in the proximity graph, because the points were placed such that the template region $R(p, q_i)$ is empty for every i; thus we have a k-Star.

Now we must prove that if $R(p, q)$ contains $P_{4\pi/k}(p, q)$ the graph does not contain a k-Star. We can use a simple argument: we consider two consecutive edges (p, q_i) and (p, q_{i+1}). The angle $\angle q_i p q_{i+1}$ is greater than $2\pi/k$, otherwise either q_i is in $P_{4\pi/k}(p, q_{i+1})$ or q_{i+1} is in $P_{4\pi/k}(p, q_i)$. Thus there are less than k edges. □

5.6.2 k-Cycle

The regions corresponding to forbidding a k-cycle depend on the parameter k. The tight regions for forbidding a 3-cycle (also called triangle freeness) and 4-cycle are unique, while there are at least two regions for $k = 5$.

Theorem 5.10 *The closed lune L is the unique tight region for triangle freeness.*

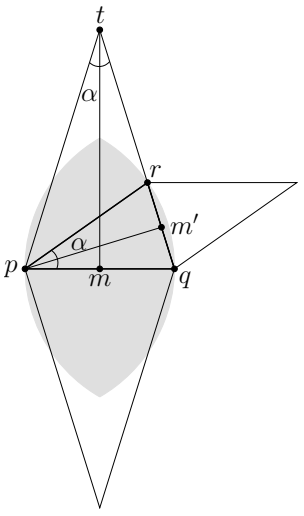


Figure 5.4: Construction of a triangle if the region is smaller than the lune.

Proof For every three vertices p, q and r of an ERG_R we consider the longest edge of any triangle in the graph. Suppose it is the edge (p, q), the other cases being symmetric.

If $R(p, q)$ contains the lune $L(p, q)$, this edge does not appear because $L(p, q)$ contains r, as the lune is the locus of the points at a distance less than or equal to $\mathrm{d}_2(p, q)$ from p and from q. This proves that the lune ensures triangle freeness.

Now, to prove that it is tight and unique we show that if a point of $L(p, q)$ is not in $R(p, q)$, there exists a point set which leads to a graph containing a triangle.

Figure 5.4 shows how to construct such a set: let $r \in L(p, q) \setminus R(p, q)$ be a point on the boundary of the lune. We show that pqr is a triangle in the graph if the data set is $\{p, q, r\}$.

The edge (p, q) exists, because r is not in $R(p, q)$. The edge (p, r) exists, because the lengths of (p, r) and (p, q) are equal and the region is symmetric. To prove that p is not in $R(q, r)$, we consider the lowest point t on the bisector of pq such that r is in the triangle pqt (see Figure 5.4). Note that this point t cannot be in $R(p, q)$, otherwise r would also be in $R(p, q)$. By construction, the triangles ptq and qpr are similar. Hence p is not in $R(q, r)$ and the edge (q, r) exists.

This also proves the uniqueness of the lune, as any point of the boundary of $L(p, q)$ has to be in $R(p, q)$, meaning that any region R contains L. □

Theorem 5.11 *The closed pacman P_π is the unique tight region forbidding a 4-cycle.*

Proof Let p, q, r, s be the four vertices of a 4-cycle, with (p, q) the longest edge. Either the angle $\angle pqr$ is smaller or equal to $\pi/2$, or the angle $\angle spq$ is smaller or equal to $\pi/2$. As (p, q) is the longest edge, r or s is in $P_\pi(p, q)$, and thus the edge (p, q) is not present.

For every other region $R(p, q) \not\supseteq P_\pi(p, q)$, we can create a 4-cycle: as we consider convex and symmetric regions, we know that if $P_\pi(p, q) \setminus R(p, q)$ is not empty, then we can place 4 points on the vertices of a square, and every edge of the square will be present.

This also proves the uniqueness of the pacman, as we showed that *any* incomparable region leads to a 4-cycle. □

Theorem 5.12 *The closed $P_{6\pi/5}$ is tight for forbidding 5-cycles.*

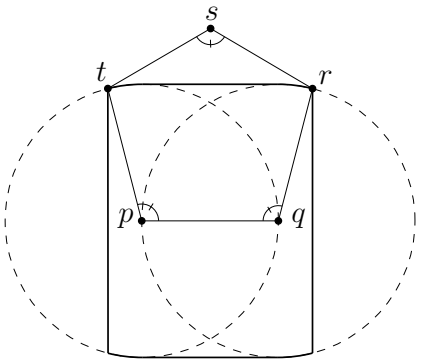


Figure 5.5: Angles if 5-cycles.

Proof In every 5-cycle, there is at least one angle which is not greater than $3\pi/5$ (see Figure 5.5). Let us say that it is the angle $\angle pqr$, with (p, q) at least as long as (q, r) (the other case being symmetric). As $\angle pqr$ is at most $3\pi/5$, r is in $P_{6\pi/5}(p, q)$, and the 5-cycle is not present. For every convex and symmetric region $R(p, q) \subset P_{6\pi/5}$, there exists a set of points corresponding to a 5-cycle. This can be obtained as follows: as $R(p, q)$ is convex and symmetric, and $R(p, q)$ is strictly contained in $P_{6\pi/5}(p, q)$, there are points of the boundary of $P_{6\pi/5}(p, q)$ on the dashed circles which are not in $R(p, q)$. We place, given a pair of points p and q, a point r at distance $\mathrm{d}_2(p, q)$ of q, out of $R(p, q)$ and on the outer limit of $P_{6\pi/5}$ (see Figure 5.5). We place t symmetrically at distance $\mathrm{d}_2(p, q)$ from p, in the missing region. We have $\angle pqr = \angle tpq = 3\pi/5 - \alpha$. We add a point s such that $\angle rst$ is $3\pi/5 - \alpha$, and that (r, s) and (s, t) have the same length. This ensures that the edges (r, s) and (s, t) are present, because $\angle pts$ and $\angle qrs$ are greater than $3\pi/5$. By construction, s is not in $R(t, p)$ nor in $R(q, r)$, and thus the edges (s, t) and (q, r) are present. □

Lemma 5.13 *The truncated slab S_5 satisfies 5-cycle freeness.*

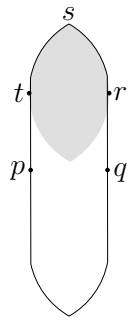


Figure 5.6: Truncated slab for 5-cycle freeness.

Proof Let us consider the largest edge of a 5-cycle, (p, q), and the slab $S(p, q)$. The slab always contains at least one vertex of the cycle: as p and q are at each side of $S(p, q)$ and that it is a cycle, another edge has to go through the slab. As (p, q) is the longest edge, no other edge can go through without having one of its endpoints in the region. This proves that the slab is sufficient.

However, the slab is not necessary: if (p, q) is the largest edge of a cycle (p, q, r, s, t), the point t is at a distance at most $\mathrm{d}_2(p, q)$ from p, as r from q. Now, the point s is contained in the intersection of a disk centered at t and a disk centered at r and with radius $\mathrm{d}_2(p, q)$. So there is only a part of the slab which is useful to ensure the 5-cycle freeness: the locus of the points of the slab which could be a vertex of the 5-cycle if all the other points were out of the slab. This region is $S_5(p, q)$, as shown in Figure 5.6, which is tight as removing a single point leads to a 5-cycle. □

This proves that there are at least two tight regions for 5-cycle freeness, as there is a tight region contained in S_5, and that S_5 and $P_{6\pi/5}$ are incomparable. We show in the next sections that cycle freeness, which corresponds to forbidding a k-cycle for every k, has also at least two tight regions.

5.7 Other properties

We grouped in this section other monotone properties which are neither geometric, nor are expressed as forbidden graphs.

5.7.1 Cycle Freeness

The two following lemmas will help us to characterize all the tight regions for cycle freeness.

Lemma 5.14 *The slab S is a tight region for the cycle freeness property.*

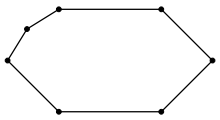


Figure 5.7: 7-cycle not forbidden by an open slab.

Proof Let us consider the largest edge (p, q) of a cycle. Since (p, q) is the largest edge, there must be another edge in the cycle going through $S(p, q)$ with at least one of its endpoints in $S(p, q)$. Hence (p, q) cannot be an edge of the empty-region graph.

If S is not tight, by Lemma 5.1 there exists a convex and symmetric region $R(p, q) \subset S(p, q)$; this region is either bounded or points on the boundary of the slab are missing (infinite continuous pieces of the border are missing, the region is neither closed nor open) and contained in the slab.

In either cases, placing a large even number of points equidistantly on a circle yields a cycle in the ERG, as the points will be properly aligned. □

We can even create an odd cycle if we take care of the intersections of the slabs. An example of a 7-cycle not forbidden by an open slab is shown in Figure 5.7; if the region is convex and the slab not closed we can always create a cycle.

Lemma 5.15 *Let M be the closed mastercard. If a region R not containing the slab S satisfies the cycle freeness property, $M(p, q) \setminus R(p, q)$ does not contain a connected portion of the boundary $\partial M(p, q)$ of positive length. In particular, R contains the open mastercard.*

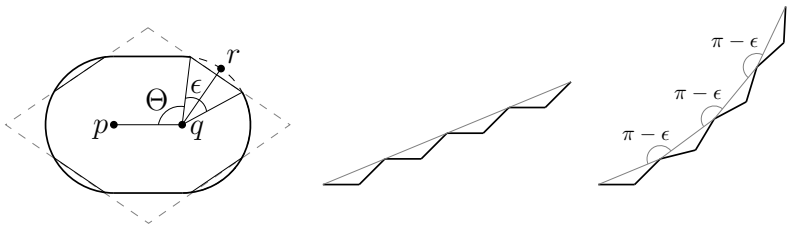


Figure 5.8: Chain construction when one point is missing.

Proof We first note that the closed mastercard M satisfies the property as it forbids a chain of length two: the mastercard of the largest of two edges would contain the endpoint of the other.

Let $R(p, q)$ be a region such that there is a range of adjacent points on the boundary of $M(p, q)$ not contained in $R(p, q)$. There exists a set of points generating an ERG containing a cycle that we can construct as follows. Let p and q be two points (see Figure 5.8). There is a point r in $M(p, q) \setminus R(p, q)$ and a neighborhood of its boundary around r which is not in $R(p, q)$. We denote by Θ the smallest angle of that neighborhood from p, centered at q; $\angle pqr$ is in the interval $(\Theta, \Theta + \epsilon)$.

To construct the cycle, we create a very long chain, as shown in Figure 5.8; this is possible because of the symmetry of the region. The ϵ freedom allows to bend it: every two segments, we can turn with an angle of up to ϵ degree in the same direction (see Figure 5.8).

Thus, we create a cycle of $2\lceil 2\pi/\epsilon \rceil$ segments, with turning angles $(\Theta, 2\pi - (\Theta + \delta), \Theta, 2\pi - (\Theta + \delta), \ldots)$, with $\delta = 2\pi/\lceil 2\pi/\epsilon \rceil$.

This proves that no adjacent points on the boundary of the closed mastercard are missing. By convexity, this implies that R contains the open mastercard. □

Theorem 5.16 *There is an infinite number of tight regions for cycle freeness: the closed slab S and an infinite set of regions M' contained in the closed mastercard and containing the open mastercard.*

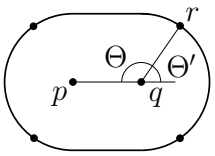


Figure 5.9: Angles allowed by the region.

Proof By Lemma 5.14, we know that the slab S is a tight region. We note that the open mastercard does not forbid cycles whose edges have the same length, and thus does *not* satisfy the property. However, by Lemma 5.15 any region satisfying the cycle freeness property and not containing the slab S contains the open mastercard M.

Let $R(p, q)$ be a convex and symmetric region such that there are exactly 4 points on its boundary missing compared to $M(p, q)$. Then, for some of these configurations we can create a cycle. Let Θ and $\Theta' = \pi - \Theta$ be then angles associated to these points, as shown in Figure 5.9. We can create a chain, in which we can only choose to rotate by Θ' clock- or counterclock-wise. We define the absolute angle of the first edge in the chain as 0. If a chain contains k edges, and that we chose i times to turn counterclockwise then the absolute direction of the k^{th} edge is $i\Theta' - (k - i)\Theta'$.

The considered chain forms a k-cycle when the $(k + 1)^{\text{th}}$ edge and the first one are the same i.e. begin at the same position and have the same absolute direction. For every set of vertices V, $ERG_R(V)$ is a planar graph as R contains the ball B. Thus the sum of the rotations is exactly equal to $\pm 2\pi$.

This means that if, for any k, the 4 points on the boundary of M whose corresponding Θ' is exactly equal to $2\pi/k$ are not in R, we can create a cyclic chain which is a regular k-gon; these points must be contained in any tight region not containing the slab.

By Lemma 5.15, we know that every tight region not containing S cannot have a connected portion of $\partial M(p, q)$ of positive length missing. We also know that these missing points in $M \setminus R$ cannot allow regular k-gons. Now, for any region R with exactly 4 points missing such that $\Theta' \neq 2\pi/k, \forall k \in \mathbb{N}$, the resulting graph contains no cycle. However, combinations of these missing points could lead to cycles.

Proof (Continued) If points at angles $\Theta_1, \Theta_2, \dots \Theta_m$ (with supplementary angles $\Theta'_1, \Theta'_2, \dots \Theta'_m$) are in $M(p,q) \setminus R(p,q)$, we can create a k-cycle only if

$$\exists k \in \mathbb{N}_0, \exists \alpha_1, \alpha_2, \dots \alpha_m \in \mathbb{N}$$

such that

$$\sum_{i=1}^{m} \alpha_i = k$$

and

$$\sum_{i=1}^{m} \Theta'_i \alpha_i = 2\pi$$

as linear combinations of these angles would lead to cycles. This is only a necessary condition, some of these sets cannot produce cycles (i.e. the corresponding chain would be self intersecting or the last edge of the chain is not adjacent to the first one).

On the other hand, we know that the following condition is sufficient to lead to a lk-cycle

$$\exists k, l \geq 3 \in \mathbb{N}_0, \exists \alpha_1, \alpha_2, \dots \alpha_m \in \mathbb{N}$$

such that

$$\sum_{i=1}^{m} \alpha_i = k$$

and

$$\sum_{i=1}^{m} \Theta'_i \alpha_i = 2\pi/l$$

We can construct these kl-cycles as follows: with k edges we can construct a chain such that the absolute angle between the first and last edge is exactly $2\pi/l$. By concatenating l such chains we ensure that the cumulative rotation is exactly 2π and the cycle is closed. Note that for this to work, the angles must be reasonably small i.e. the sequence of k edges must be planar and consecutive blocks of k edges cannot intersect. But there is still an infinity of such configurations.

In conclusion, we know that there is an infinity of subsets of the boundary of M which cannot be missing, as their lack would lead to cycles. We can restrict the subsets to consider to the extremal ones, where we cannot add a single point without enabling the creation of cycles, and there is still an infinity of subsets (any pair of missing points leading to a cycle generate at least two extremal subsets). They correspond to tight regions for cycle-freeness.

Proof (Continued) The last step consists in proving that these tight regions are contained in the closed mastercard M. We proceed by contradiction: if a region R is tight, contains the open mastercard and is not contained in the closed mastercard, there exists at least one point, $r \in R(p, q)$ out of the mastercard. r is at a distance bigger than $\mathrm{d}_2(p, q)$ from p and from q. Thus this point forbids an edge in a cycle where every edge has not the same length. But that cycle is forbidden by the open mastercard, which contradicts the fact that R is tight.

We end with an incomplete characterization of the regions M' in M which are tight, but this proves that there are infinitely many of these. □

5.7.2 Planarity

Here we consider the planarity of the graph, and not of its embedding which was analyzed in section 5.5.

Theorem 5.17 *The closed ball B is a tight region for planarity.*

Kuratowski's theorem says that planarity corresponds to forbidding $K_{3,3}$ and K_5 as minors; the following shows that these are forbidden if the template region contains the closed ball B.

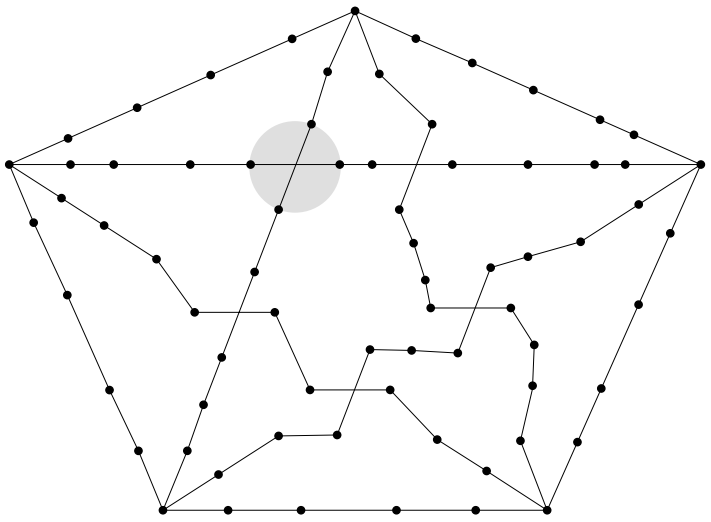


Figure 5.10: Construction of a K_5 if the template region does not contain the ball.

Proof As shown previously, the embedding of any ERG_B is planar. For every region $R(p,q) \subset B(p,q)$, we can create a set of edges having a K_5 as minor, as shown in Figure 5.10. The two gadgets used in this construction are crossing edges, which are constructed using the fact that the region is strictly contained in the ball, and that the induced $K_{1,4}$ is not even forbidden by the ball. □

Note that the gadgets presented here could be used to construct any minor. We conjecture that for every monotone property expressed as forbidden minors, at least one of the tight regions contains the closed ball B.

5.7.3 Connectivity

Theorem 5.18 *The open lune L is the unique tight region for connectivity.*

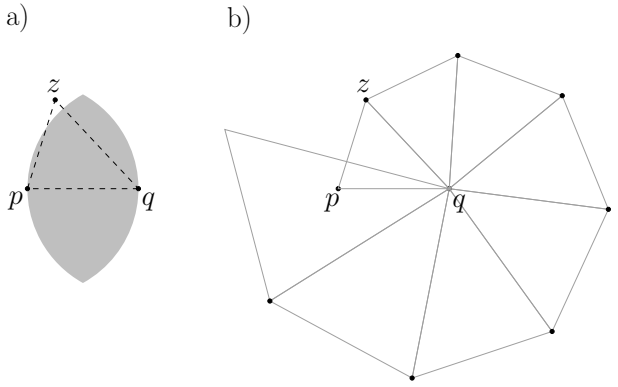


Figure 5.11: Construction of a disconnected graph.

Proof The open lune defines a connected graph as it is a relative neighborhood graph [JT92]. Connectivity is monotone with respect to edge addition. We cannot add a single point to the lune without losing the graph property, because otherwise, there are sets of vertices for which the ERG is not connected.

Proof (Continued) We can obtain such a set of vertices in the following way: $R(p, q)$ strictly contains a triangle joining the points p and q and a point z out of the open $L(p, q)$ (Figure 5.11a). We first consider the case where the edge (p, q) is smaller than the edge (q, z), the case with (p, q) greater than (q, z) being symmetric; then we analyze the case where (p, q) and (q, z) are equal.

We construct our set of vertices as shown in Figure 5.11b. Given a pair of points (p, q) we define a third point z which is contained in the region $R(p, q) \setminus L(p, q)$. We repeat the process using the points z and q while the triangle defined by the three points does not contain a previously defined point. This set of points defines an unconnected ERG: q is not connected to any other point, because there is always an item contained in $R(p, q)$.

For the case where (p, q) and (q, z) have the same length, we consider only three points, as shown in Figure 5.11a. The edge (p, q) is not in the graph as z is in $R(p, q)$. Symmetrically, the edge (q, z) is not in the graph as p is in $R(p, q)$. Thus the point q is isolated. □

5.7.4 Bipartiteness

Theorem 5.19 *There are an infinite number of tight regions for bipartiteness among which the closed slab S and an infinite set of mastercards M' contained in the closed mastercard and containing the open mastercard.*

Proof Cycle freeness implies bipartiteness. The tight regions for cycle freeness are tight for bipartiteness, because our counter-examples presented in the cycle freeness proof can be used to create odd cycle: even by adding a restriction on the parity of the cycles to create, we can still prove the existence of an infinite number of tight regions using exactly the same arguments. □

5.8 Combining properties

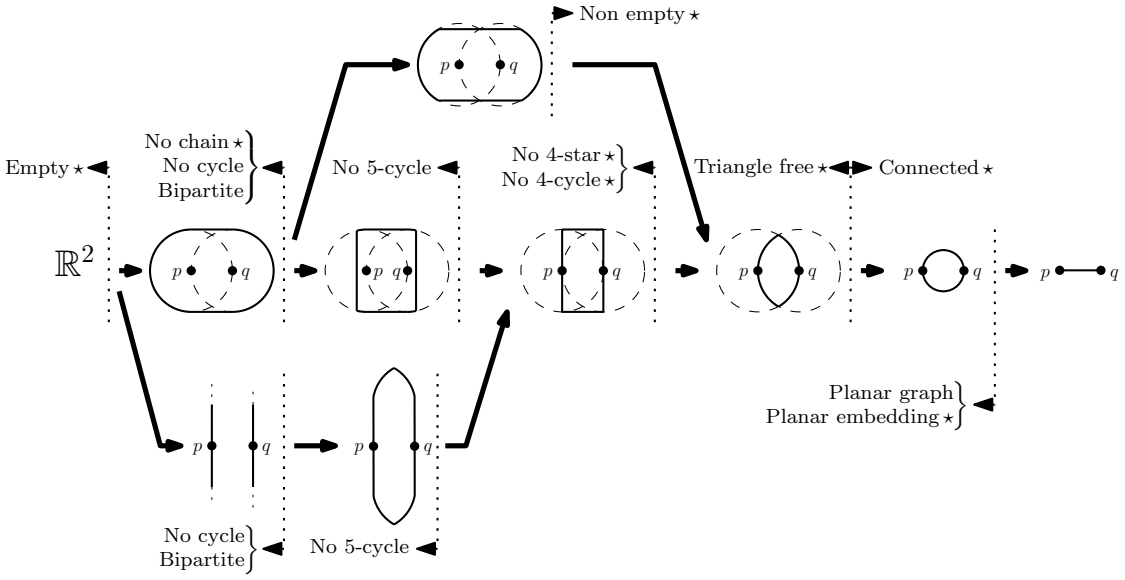


Figure 5.12: Partial order of inclusion among regions and corresponding monotone properties.

Figure 5.12 summarizes some of the results. Properties for which we proved that the tight region is unique are marked with ⋆; the vertical dotted lines represent the threshold at which some property is ensured. For instance every region at least as big as the lune ensures triangle freeness, while every region smaller than the lune leads to a connected graph.

The ERG model has limitations: not all sets of properties can be satisfied by a region. For instance, in our model, we cannot combine cycle freeness and connectivity, as the region cannot be smaller than the lune and bigger than the mastercard or than the slab at the same time. Some other properties always come together, such as forbidding a 4-star and a 4-cycle.

The figure can help in studying the properties of existing classes of graphs. For instance, for β-skeletons the parameter β determines the position of the influence region in the partial order from which one can deduce the set of properties that are satisfied.

ERGs constitute a convenient way to construct graphs from sets of points. Other definitions of proximity graphs using for instance non-convex, non-anchored, or non-

symmetric influence regions, could lead to a more fine-grained way of ensuring properties.

5.9 Construction of ERGs

If we want to construct empty region graphs for various template regions, we could develop *ad-hoc* methods corresponding to every template region that we want to consider. However, as a first step, we propose here a more generic approach, which, using the algorithm for the construction of Θ-graphs and of Gabriel graphs as subroutines, will let us construct a large class of empty region graphs.

The Θ-graph can be constructed in $O(n \log n)$ time whatever the proximity definition used (closest Euclidean neighbor or closest projection) and contains at most $\frac{2\pi}{\Theta} n$ edges [KG92]. It cannot be defined using anchored regions. However, the following lemma shows that the Θ-graphs contain ERGs as subgraphs:

Lemma 5.20 *Given a set of points S, the corresponding Θ-graph G using the closest Euclidean neighbor contains $ERG_{P_{2\Theta}}(S)$ as a subgraph, where $P_{2\Theta}$ is the pacman with angle 2Θ.*

Proof For every edge (p, q) in $ERG_{P_{2\Theta}}(S)$, q is the closest vertex of p in the cone in which it is contained, otherwise there exists another vertex contained in $P_{2\Theta}(p, q)$. □

The Gabriel graph (an ERG using the ball B), as well as any β-skeleton with $\beta \geq 1$, can be constructed in $O(n \log n)$ time [KR85]. Based on results in Subsection 5.6 on forbidden k-stars, we know that every ERG based on a region containing a pacman $P_{4\pi/k}$ has at most $O(kn)$ edges.

A large class of anchored regions yields ERG which can have a quadratic number of edges:

Lemma 5.21 *Let R be an anchored region contained in a diamond D_Θ for some Θ. Then, there exists sets of points S such that $ERG_R(S)$ has $\Omega(n^2)$ edges.*

Proof These regions yield graphs with $\Omega(n^2)$ edges if the set is, for instance, composed of points on two parallel lines far enough from each other. □

Given a set of points S and a template region R, an edge-validation algorithm for an edge (p, q) is an algorithm deciding whether (p, q) is contained in $ERG_R(S)$.

Theorem 5.22 *Given any fixed template region R, and the corresponding edge-validation algorithm with complexity $F(n)$, we can construct the corresponding ERG_R*

- *in $O(n \log n + n\,F(n))$ time if the region R contains the pacman $P_{2\Theta}$ for some Θ, or contains the ball B. The empty region graphs defined by that template region R have $O(n)$ edges.*
- *in $O(n^2\,F(n))$ time if the region R is contained in a D_Θ for some Θ. The empty region graphs defined by that template region R can have $\Omega(n^2)$ edges.*

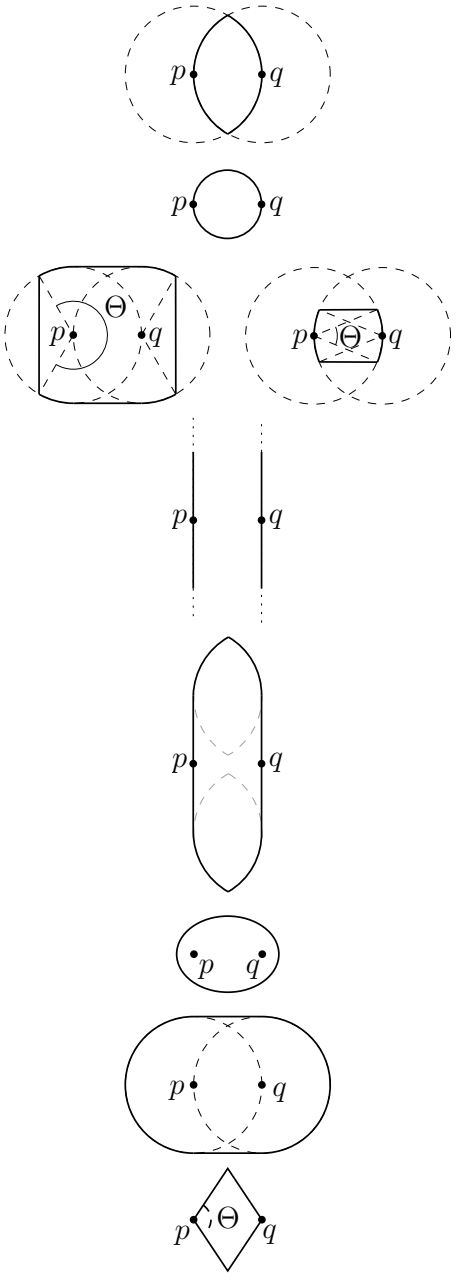
Proof The size of the graphs follows directly from the proofs of Lemma 5.20, Lemma 5.21, and the fact that the ball B ensures a planar graph. To construct ERG_R, for a region R for which we do not have an *ad-hoc* construction algorithm, we can proceed as follows:

We first determine the largest constant Θ such that $R(p,q) \supseteq P_{2\Theta}(p,q)$. If there is no such constant greater than 0, we check whether $R(p,q)$ contains the ball $B(p,q)$.

If one of these regions is contained in $R(p,q)$, we apply the corresponding graph construction algorithm (either the Θ-graph algorithm if the region contains a pacman, or the Gabriel graph algorithm if the region contains the ball). In this case, the resulting graph will have $O(n)$ edges and is constructed in $O(n \log n)$ time. We apply the edge-validation algorithm to each edge of the Θ-graph or the Gabriel graph, and we remove the edges which are not in ERG_R. This takes $O(n \log n + n\,F(n))$ time.

If R neither contains any pacman nor the ball, we apply the edge-validation algorithm to every pair of edges in S, resulting in a running time of $O(n^2\,F(n))$ to construct all the edges. □

Note that there is a gap between the regions yielding graphs with $O(n)$ edges and regions yielding graphs which *can* have $\Omega(n^2)$ edges. However this covers all convex regions we considered here, as shown by Figure 5.13. For some regions we do not use our generic framework but rather apply specific algorithms adapted to that region. We still rely on efficient edge-validation algorithms, for the cases where we do not have an ad-hoc algorithm.



The lune $L(p, q)$ corresponds to a 2-skeleton; ERG_L can thus be constructed in $O(n \log n)$ time.

The ball $B(p, q)$ corresponds to a 1-skeleton; ERG_B can thus be constructed in $O(n \log n)$ time.

By Theorem 5.22, ERG_{P_Θ} can be constructed in $O(n \log n + n\, F(n))$ time.

The slab $S(p, q)$ corresponds to a ∞-skeleton; ERG_S can thus be constructed in $O(n \log n)$ time.

The truncated slab $T(p, q)$ contains the ball; by Theorem 5.22, ERG_T can be constructed in $O(n \log n + n\, F(n))$ time.

Every ellipse contains a pacman, thus by Theorem 5.22, ERG_{E_t} can be constructed in $O(n \log n + n\, F(n))$ time.

The mastercard $M(p, q)$ contains the ball; by Theorem 5.22, ERG_T can be constructed in $O(n \log n + n\, F(n))$ time.

By Theorem 5.22, ERG_D can be constructed in $O(n^2\, F(n))$ time.

Figure 5.13: Construction of ERGs

Chapter 6

Sigma-Local Graphs

6.1 Introduction

In Chapter 5, we introduced a new family of proximity graphs called the empty region graphs. Given a set of desirable nice properties, we can find a template region satisfying all the required properties. We gave a framework to construct easily empty region graphs parameterized by a wide class of anchored regions.

In this chapter, we study the problem of constructing ERGs as efficiently as possible. This is more difficult as we have to develop *ad-hoc* algorithms for every anchored region we consider. We illustrate this with an example motivated by the notion of σ-locality introduced by Erickson [Eri03].

One of the constraints for a graph to be ***σ-local*** [Eri03] is that for every edge, there must exist a ball around both of its endpoints not containing any other vertex. If the distance between the endpoints of an edge is denoted by d, the balls must have radius at least d/σ. Given a set of points S, we define the ***σ-local graph*** of S as the maximal graphs satisfying this constraint. Those graphs belong to the family of ***proximity graphs***, and are ***empty region graphs*** according to the definition given in the previous chapter.

We study algorithms to construct σ-local graphs for a given set of points. Different approaches are proposed depending on the value of σ. We also present a data-structure, the ***σ-spectrum*** which, given σ as a query, returns the corresponding graph as efficiently as possible.

In Chapter 5, we characterized the template regions which ensure properties for every graph. In this chapter, we propose to determine the minimum or maximum value of σ such that a property is satisfied for a given set of points. This requires more work, as

we have to develop specific algorithms for σ-local graphs, but also for every property we want to consider.

The algorithms we propose allow us to test whether, given a set of points and a value of σ, the corresponding σ-local graph is complete, it contains isolated vertices, it is connected, and it is plane if the edges are embedded as straight line segments.

Contents

6.2 σ-Locality

The following definition was proposed by Jeff Erickson [Eri03]. Let $N(v)$ denote the set of neighbors of a vertex v in the graph G.

Definition 6.1 *A geometric graph is* σ***-local*** *if its local stretch $\sigma(G)$ is less than a fixed constant σ and its global stretch $\Sigma(G)$ is bounded by a fixed polynomial in the number of vertices, where*

$$\sigma(q) = \frac{\max\limits_{p \in N(q)} \mathrm{d}_2(p,q)}{\min\limits_{p \in V \setminus \{q\}} \mathrm{d}_2(p,q)}$$

$$\sigma(G) = \max_{q \in V} \sigma(q)$$

$$\Sigma(G) = \frac{\max\limits_{(p,q) \in E} \mathrm{d}_2(p,q)}{\min\limits_{(p,q) \in E} \mathrm{d}_2(p,q)}$$

This means that in every σ-local graph, there exists an edge between a pair of vertices (p,q) *only if* the two balls centered at p and q with radius $\mathrm{d}_2(p,q)/\sigma$ are empty, and the maximum ratio between the length of any pair of edges is bounded. This implies that every edge (p,q) satisfy $\mathrm{d}_2(p,q) \leq \sigma \cdot \min(W_p, W_q)$, where the weight W_p of a point p is the distance to its nearest neighbor.

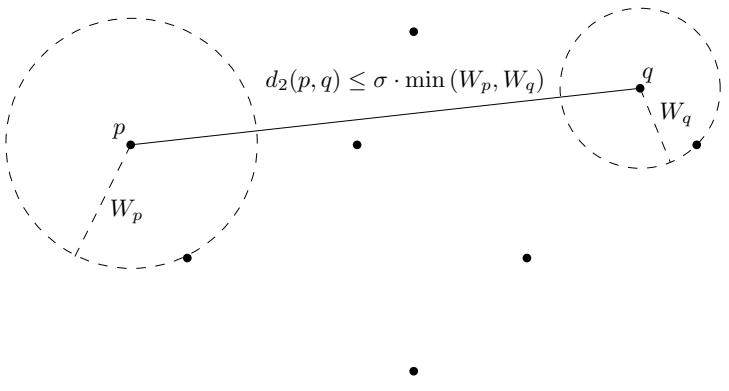


Figure 6.1: Restriction induced by σ-locality.

Erickson [Eri03] used this definition of σ-locality as a realistic input model for nonconvex polyhedra, and showed that Boolean combinations of two σ-local polyhedra defined by n vertices in $\mathbb{R}^d$ can be performed in $O(n \log n)$ time. The third condition (the poly-

nomial bound on $\Sigma(S)$) is needed for the analysis of complexities in [Eri03]. In what follows we will not consider that last constraint.

6.3 σ-Local Graphs

We define the σ-local graphs as follows:

Definition 6.2 *Let S be a set of n points in $\mathbb{R}^d$, and let $\sigma \geq 1$ be a real number. The **σ-local graph** of S, denoted by $G_\sigma(S)$, is defined to be the graph with vertex set S and edge set E, where (p, q) is in E if and only if*

1. *the ball with center p and radius $\mathrm{d}_2(p, q)/\sigma$ does not contain any point of $S \setminus \{p\}$ in its interior, and*
2. *the ball with center q and radius $\mathrm{d}_2(p, q)/\sigma$ does not contain any point of $S \setminus \{q\}$ in its interior.*

This corresponds to an empty region graph (see Chapter 5) using the pair of balls centered at p and q with radius $\mathrm{d}_2(p, q)/\sigma$ as template region.

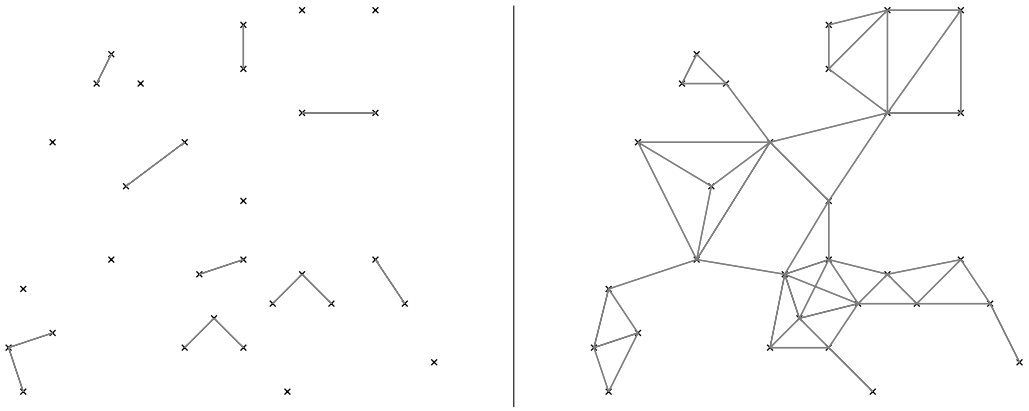


Figure 6.2: σ-local graphs for parameters 1 and 2.

6.3.1 Construction of $G_\sigma(S)$

In this subsection we show how to efficiently construct the σ-local graph of a given set of points for a value σ. We present two algorithms: one whose complexity depends on σ, and a another one whose complexity depends on the size of the graph.

Fixed σ

The first algorithm we present is valid for any real number $\sigma > 1$ and in any (constant) dimension d. However the running time is interesting only if σ^d is much smaller than n. We propose a more efficient solution for the other case in the next subsection.

Definition 6.3 *Given a separation ratio α, two point sets are **well-separated** if they can be enclosed in two spheres of radius ρ such that the distance between the two spheres is at least $\rho\alpha$.*

Definition 6.4 *Given a set S of n points in $\mathbb{R}^d$, a **well-separated pair decomposition** [CK95] is a set of $m = O(\alpha^d n)$ pairs of sets of points $\{A_i, B_i\}$ such that:*

- *For all p and q in S, there exists a unique value of $1 \leq i \leq m$ for which $p \in A_i$ and $q \in B_i$.*
- *For $1 \leq i \leq m$, A_i and B_i are well-separated (and in particular, A_i and B_i are disjoint sets for every i).*

Let $\sigma > 1$ and consider a well-separated pair decomposition $\{A_i, B_i\}$ of S with separation ratio 2σ. We obtain the following lemma:

Lemma 6.1 *Let (p, q) be an edge in $G_\sigma(S)$, and let i be the index such that $p \in A_i$ and $q \in B_i$. Then both A_i and B_i are singleton sets.*

Proof Assume that A_i contains a point r of S with $r \neq p$. Since the sets A_i and B_i are well-separated, there exist balls C and C' having the same radius ρ, such that $A_i \subseteq C$, $B_i \subseteq C'$, and the distance between C and C' is at least $2\sigma\rho$. Since p and r are both contained in C, we have $\mathrm{d}_2(p, r) \leq 2\rho$. On the other hand, since $p \in C$ and $q \in C'$, we have $\mathrm{d}_2(p, q) \geq 2\sigma\rho$. By combining these inequalities, it follows that $\mathrm{d}_2(p, r) \leq \mathrm{d}_2(p, q)/\sigma$, contradicting the fact that (p, q) is an edge in the σ-local graph $G_\sigma(S)$. □

Note that the converse of this lemma is, in general, *not* true.

Theorem 6.2 *The σ-local graph $G_\sigma(S)$ for a point set S in $\mathbb{R}^d$ containing n points and a given σ can be constructed in $O(\sigma^d n + n \log n)$ time.*

Proof Using Vaidya's algorithm [Vai89], we compute, for each point p in S, its weight W_p (the distance to its nearest neighbor). This can be done in $O(n \log n)$ time in any constant dimension.

Using Callahan and Kosaraju's algorithm [CK95], we compute a well-separated pair decomposition $\{A_i, B_i\}$, $1 \leq i \leq m$, for S, with separation ratio 2σ, where $m = O(\sigma^d n)$. This is achieved in $O(n \log n + \sigma^d n)$ time.

Then, we compute the index set I consisting of all indices i such that both A_i and B_i are singleton sets, in $O(\sigma^d n)$ time.

We initialize an empty edge set E. For each $i \in I$, let p_i and q_i be the points of S such that $A_i = \{p_i\}$ and $B_i = \{q_i\}$. We add the edge (p_i, q_i) to E if and only if both W_{p_i} and W_{q_i} are at most $\mathrm{d}_2(p_i, q_i)/\sigma$. This step takes $O(\sigma^d n)$ time.

The algorithm finishes by returning the graph $G = (S, E)$, which, by Lemma 6.1 is the σ-local graph $G_\sigma(S)$. □

This also shows that the σ-local graph of S contains $O(\sigma^d n)$ edges. This upper bound is only meaningful if $\sigma^d = o(n)$.

Large Fixed σ and $d = 2$

We present here an algorithm which is efficient for point sets in the plane.

Theorem 6.3 *The σ-local graph $G_\sigma(S)$ of a point set S in $\mathbb{R}^2$ containing n points and a given σ can be constructed in $O(n \log^3 n \log\log n + k)$ time, where k is the number of edges in the resulting graph.*

Proof For each point $p \in S$, we want to compute all $q \in S$, such that

1. $\sigma W_p \geq \mathrm{d}_2(p, q)$, and
2. $\sigma W_q \geq \mathrm{d}_2(p, q)$.

In this way, we obtain all edges (p, q) that are incident on p. Let $p = (p_1, p_2)$ and $q = (q_1, q_2)$.

Proof (Continued) We can rewrite the two conditions as:

1. $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_p)^2$, and
2. $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_q)^2$.

We denote by $S_{p,<}$ and $S_{p,>}$ the sets of points $\{q \in S | W_q \leq W_p\}$ and $\{q \in S | W_q > W_p\}$, respectively. We have to analyze two subproblems:

1. Given p, find the points q in $S_{p,>}$ such that $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_p)^2$.
2. Given p, find the points q in $S_{p,<}$ such that $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_q)^2$.

However, since the graph is undirected we do not have to consider both cases. For every edge (p, q) in $G_\sigma(S)$, $W_p \geq W_q$ or $W_p \leq W_q$. Thus, an edge will be constructed when we consider its endpoint with lower weight.

To solve the above subproblem, we store the points of S at the leaves of a balanced binary search tree, sorted by their weights. At each node u of this tree, we store a secondary data structure as outlined below.

Given a query point p, we want to find all $q \in S_{p,>}$ such that $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_p)^2$. We search in the tree for p. The search path partitions the set of all q with $W_p \leq W_q$ into $O(\log n)$ canonical nodes. (These are the right children of nodes on the path in which the path moves to the left child, see Figure 6.3.)

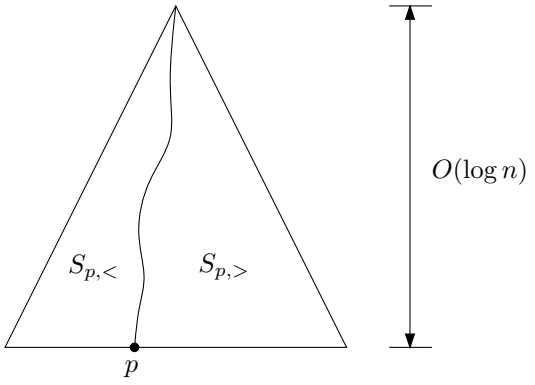


Figure 6.3: Binary tree associated to the point set.

Proof (Continued) For each canonical node u, we want to find all points q in the subtree of u for which $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_p)^2$. These are all points q that are in the circle with center p and radius σW_p. So we store at u a data structure that supports these queries. Thus the time to construct the σ-local graph is $O(n \log n)$ times the time for a circular range reporting query.

Aggarwal, Hansen, and Leighton [AHL90] provide a data-structure that supports circular range reporting queries in $O(\log n + k)$ time where k is the number of items reported. The structure can be built in $O(n \log^2 n \log\log n)$ time and uses $O(n \log n)$ space. Since the primary tree has $O(\log n)$ levels and each level encompasses a linear number of nodes, the total time complexity to construct $G_\sigma(S)$ is thus $O(n \log^3 n \log\log n + k)$. □

σ-Spectrum

Let us begin by describing the ***parametric search technique***, invented by Megiddo [Meg83].

This technique is used to solve certain optimization problems, when we have an efficient algorithm in the PRAM model for the corresponding decision problem. It computes the real value $\lambda - \lambda^*$ which optimizes a monotone function $f : \mathbb{R} \mapsto \mathbb{R}$ using an algorithm $A(\lambda)$ which solves the decision problem "is λ greater than λ^* ?".

To achieve this, Megiddo's key idea is to run the algorithm A on the *unknown* value λ^*. As the algorithm runs in the PRAM model, it will eventually need to compare λ^* to some other values x. The answer to these comparisons can be found easily by running $A(x)$, which will answer if x is greater than λ^*.

We denote by T_s the running time of the algorithm A. If A_p is an equivalent parallel algorithm running in T_p parallel steps using P processors, Megiddo proved that the total running time of the optimization algorithm using his technique is $O(PT_p + T_pT_s \log P)$.

We will apply this technique to prove the following lemma.

Lemma 6.4 *Given a point set S in $\mathbb{R}^2$ and a constant K, we can find the smallest real number σ such that $G_\sigma(S)$ has K edges in $O((n + K) \log^{O(1)} n)$ time.*

Proof First we consider the following decision problem: Given the set S, and given σ, decide if $G_\sigma(S)$ contains at least K edges. For this, we could simply execute the algorithm used for Theorem 6.3 and stop as soon as we have enough edges. But, since we will later use the parametric search technique [Meg83], we want to make sure that the algorithm can be run in the PRAM model, so we propose the following alternative approach.

The circular range reporting query in two dimensions corresponds to a half-space query in 3D [Aga04]: we lift the points from the xy plane onto a paraboloid $z = x^2 + y^2$; the intersection of a plane with the paraboloid projects down to a circle on the xy plane, and the intersection of a half-space projects down to a disk. Let S' be the 3D set corresponding to S; the procedure described hereunder allows us to perform half-space queries on S'.

Let T be a balanced tree that stores the points of S' at its leaves, in an arbitrary order. With each node u of T, we store the Dobkin-Kirkpatrick hierarchy [DK83] of the convex hull of all points in the subtree of u.

To answer a query, we are given a plane P in $\mathbb{R}^3$ and want all points of S' that are below P. We start at the root of T. Using the Dobkin-Kirkpatrick hierarchy, we can decide in $O(\log n)$ time if the convex polyhedron stored at the root

1. is completely above P: in this case, the query algorithm terminates,
2. is completely below P: in this case, we report all points stored in the tree and terminate,
3. intersects P: in this case, we recursively query both subtrees of the root.

Let k be the number of points that are below P. The number of nodes of T that are visited by the query algorithm is $O(k \log n)$. At each node, the algorithm spends $O(\log n)$ time. Of course, the algorithm can terminate as soon as K edges have been reported.

These queries can be solved in parallel: in case 3 of the query algorithm, we use two processors. In other words, each time we branch in the tree, we use an additional processor.

The decision problem (does $G_\sigma(S)$ have at least K edges?) can be solved sequentially, in $O((n+K)\log^2 n)$ time; it can be solved in parallel in $O(\log^{O(1)} n)$ time, using $n+K$ processors and only algebraic predicates.

Proof (Continued) This immediately allows us to use the parametric search technique of Megiddo [Meg83] to find the smallest σ such that $G_\sigma(S)$ contains at least K edges in time $O((n+K)\log^{O(1)} n)$. □

Theorem 6.5 *Using $O(n\log^{O(1)} n)$ space and preprocessing time, we can compute $G_\sigma(S)$ for any set S of n points in $\mathbb{R}^2$ and any σ in $O(k)$ time, where k is the number of edges in the resulting graph.*

Proof We can preprocess efficiently σ-local graphs because the size of the graph is a positive monotone function of σ: if σ grows, edges are added and never removed.

We first note that if the output has more than $n\log^{O(1)} n$ edges, we can construct the graph using the algorithm given in Theorem 6.3.

The preprocessing phase consists in constructing a graph with $n\log^3 n\log\log n$ edges, with the method given by Lemma 6.4. We sort the edges of the resulting graph with respect to σ and we store the edges in a list. All of this can be achieved in $O(n\log^{O(1)} n)$ time.

To answer a query, we simply output every edge in the list up to the corresponding value of σ. □

6.3.2 Testing Properties of $G_\sigma(S)$

In this section, we test properties of σ-local graphs for a given value of σ and point set S. We also propose algorithms to find the threshold value of σ to satisfy the given property on that particular set.

Completeness

A first simple question when we deal with different values of σ is to check whether or not the resulting graph is complete.

Let $\mathrm{NN}(p)$ and $\mathrm{FN}(p)$ denote the nearest and the farthest Euclidean neighbor of a vertex p in the set S.

Lemma 6.6 *The graph $G_\sigma(S)$ is not the complete graph on S if and only if there is a point p in S such that*

$$\frac{\mathrm{d}_2(p,\mathrm{FN}(p))}{\mathrm{d}_2(p,\mathrm{NN}(p))} \geq \sigma$$

Proof Assume that p is a point in S such that

$$\frac{\mathrm{d}_2(p, \mathrm{FN}(p))}{\mathrm{d}_2(p, \mathrm{NN}(p))} \geq \sigma$$

Let $q = \mathrm{FN}(p)$. Then $\mathrm{NN}(p)$ is contained in the ball with center p and radius $\mathrm{d}_2(p, q)/\sigma$. Thus, (p, q) is not an edge in $G_\sigma(S)$ and $G_\sigma(S)$ is not the complete graph.

To prove the converse, assume that (p, q) is not an edge in $G_\sigma(S)$. Then

$$\mathrm{d}_2(p, \mathrm{NN}(p)) \leq \mathrm{d}_2(p, q)/\sigma \leq \mathrm{d}_2(p, \mathrm{FN}(p))/\sigma,$$

or

$$\mathrm{d}_2(q, \mathrm{NN}(q)) \leq \mathrm{d}_2(p, q)/\sigma \leq \mathrm{d}_2(q, \mathrm{FN}(q))/\sigma.$$

Hence, p or q satisfies the condition in the lemma. □

This lemma implies that the largest value of σ for which $G_\sigma(S)$ is not the complete graph on S is given by

$$\sigma = \max_{p \in S} \frac{\mathrm{d}_2(p, \mathrm{FN}(p))}{\mathrm{d}_2(p, \mathrm{NN}(p))}.$$

Theorem 6.7 *Given a set of n points in $\mathbb{R}^2$, the maximum σ such that the corresponding σ-local graph is not complete can be found in $O(n \log n)$ time.*

Proof The nearest and farthest neighbor graph of the set of points can be constructed in $O(n \log n)$ time [OBSC00]. We find the maximum of the ratio $\frac{\mathrm{d}_2(p,\mathrm{FN}(p))}{\mathrm{d}_2(p,\mathrm{NN}(p))}$ for every point of S in linear time, and using Lemma 6.6 we know that this gives us the largest value of σ such that $G_\sigma(S)$ is not complete. □

No Isolated Vertex

A natural question when we want to analyze a graph is to determine if vertices are always part of connected components, or if they are isolated. Given a set of vertices, another view of the problem is to find the smallest σ such that the corresponding graph contains no isolated vertex.

Theorem 6.8 *Given a set S of points in $\mathbb{R}^2$, the σ-local graph $G_\sigma(S)$ with lowest σ such that $G_\sigma(S)$ contains no isolated vertex can be constructed in $O(n \log^5 n)$ time.*

Proof To find the smallest value of σ, we look at all the vertices and determine which value of σ connects them to another vertex in the graph. As we want to check if some vertex is isolated, we must consider both ends of each edge and not only the endpoint with lowest weight as we did in Theorem 6.3.

We describe the solution in three phases. First we show the preprocessing phase, in which we construct a data structure which will let us check if a point is isolated for a given σ. Second, we give a parallel algorithm using that data structure and answering queries for a fixed σ, and third we show how to derive an algorithm finding the optimal σ.

Preprocessing phase, in $O(n \log^2 n)$ time. As in Theorem 6.3, we use a balanced binary search tree T to store the vertices according to their weights. We associate to each internal node the Voronoi diagram of the set it represents. The depth of the tree is $O(\log n)$, and at each level, the canonical sets represent exactly n vertices divided in disjoint sets.

As the Voronoi diagram of n points can be computed in $O(n \log n)$ time, the total complexity to construct the n Voronoi diagrams associated to the internal nodes is thus $O(n \log^2 n)$: for each level in the tree, the complexity is bounded by $O(n \log n)$, and there are $O(\log n)$ levels.

Using the data structure created in this preprocessing phase, we can easily check if each point p connects to a point with higher weight: for the vertices in $S_{p,>}$, the vertex q leading to the lowest σ is the closest from p, because σ only depends on the distance $|pq|$ and W_p. Given a Voronoi diagram of the point set $S_{p,>}$, finding the point q leading to lowest σ is a point location problem: by definition it is the point whose Voronoi cell contains p.

For every point p, we follow the path from the root to the leaf of the tree representing that point. The union of all the sets corresponding to the nodes on that path constitutes the set $S_{>,p}$. Thus we need to perform a query in $O(\log n)$ Voronoi diagrams, and each query costs $O(\log n)$ time. This gives us $O(\log n)$ candidates for the closest point, and a linear search gives us the minimum σ.

So at the end of this phase, we have, for every point p, the minimum value of σ for which it would connect to a point in $S_{>,p}$.

Algorithm for a fixed σ, in $O(n \log^3 n)$ time. To check if there is no isolated vertex when the graph is built using a given σ, we start by the preprocessing phase. For the vertices p that are not connected to a point in $S_{>,p}$ using that value of σ, we still have to check that they are not connected to a point in $S_{<,p}$.

Proof (Continued) Therefore, we use the following approach: we look for a vertex $q \in S_{<,p}$ such that the edge (p, q) is compatible with the value of σ. The vertex p is connected to the vertex q if and only if

$$\frac{|pq|}{W_q} \leq \sigma$$

We lift the problem to $\mathbb{R}^3$ by mapping each vertex $q \in S_{p,<}$ to a plane P_q given by

$$P_q \equiv 2\,q_1\,x + 2\,q_2\,y + z = q_1^2 + q_2^2 - \sigma^2\,W_q^2$$

where q_1 and q_2 are respectively the first and the second coordinate of the point q. Thus the edge will appear for the value σ if the point $(p_1, p_2, -p_1^2 - p_2^2)$ is below P_q:

$$\begin{aligned} \frac{|pq|}{W_q} \leq \sigma \quad &\Longleftrightarrow \quad 2\,q_1 p_1 + 2\,q_2 p_2 + (-p_1^2 - p_2^2) \geq q_1^2 + q_2^2 - \sigma^2\,W_q^2 \\ &\Longleftrightarrow \quad (p_1, p_2, -p_1^2 - p_2^2) \text{ is below } P_q \end{aligned}$$

By computing the upper envelope of all the planes corresponding to the set $S_{p,<}$, we can check if p is connected to any point of that set. We proceed as follows:

1. For each node v of T in parallel, we use the parallel 3D convex hull algorithm of Amato and Chow [AP95], to compute the upper envelope of the planes defined by points in the subtree rooted at v. This can be achieved using $O(n \log n)$ processors in $O(\log^2 n)$ time.

2. For each node v of T in parallel, we use the parallel algorithm of Reif and Sen [RS92] to build a point location structure for the upper envelope computed in Step 1 - achieved with $O(n \log n)$ processors in $O(\log n)$ time.

3. For each point p in parallel, we locate p in each of its $\log(n)$ relevant point location structures to determine whether, for the current value of σ, the point p is connected to a point of lower weight. We use $O(n \log n)$ processors, and $O(\log n)$ time.

4. For each point p, it is not isolated if it is connected to a point of lower weight, or if σ is greater than the value for p computed in the preprocessing phase. Using $O(n)$ processors, it takes $O(\log n)$ time.

Proof (Continued) In conclusion, we have a parallel algorithm running in $O(n \log^3 n)$ total time which checks if there is no isolated vertex for a fixed σ.

Algorithm finding the lowest σ, in $O(n \log^5 n)$ time. To find the lowest σ, we use the parametric search technique of Megiddo [Meg83] which transforms a decision algorithm into an optimization algorithm efficiently.

We denote by T_s the running time of a decision algorithm A. If A_p is an equivalent parallel algorithm running in T_p parallel steps using P processors, Megiddo proved that the total running time of the optimization algorithm using his technique is $O(PT_p + T_pT_s \log P)$. By applying this to our decision algorithm for a fixed σ described above, we get, if we run it on $P = n$ processors, that $O(n \log^3 n + \log^2 n \cdot n \log^2 n \cdot \log n)$, which evaluates to $O(n \log^5 n)$ total time. □

Connectedness

Next we consider the problem of finding the minimum value of σ such that $G_\sigma(S)$ is connected. For this, we apply ideas used in Boruvka's algorithm [Bor26, MR95] for finding the minimum spanning tree.

Theorem 6.9 *Given a set S of n points in $\mathbb{R}^2$, the connected σ-local graph with minimum σ can be constructed in $O(n \log^7 n)$ time.*

Proof The algorithm consists of maintaining a list of all the components already identified and to update σ such as to connect progressively every component to others. We begin by applying the algorithm of Theorem 6.8, while labeling all the components created. We know that every vertex will be at least connected to one other vertex; thus we have an initial value for σ and a list of at most $n/2$ different components.

Then we proceed as follows: we put every vertex in the leaves of a binary tree, sorted by component number; the internal nodes of that tree represents (sub-)sets of components. Each internal node represents a set of vertices for which we create a secondary data structure as described in Theorem 6.8 with Voronoi diagrams and support for circular range queries.

Proof (Continued) In the binary tree, we can easily find sets covering every vertex in all components but one: from the root of the binary tree, there is a left-most and a right-most path (see Figure 6.4), leading respectively to the first and the last leaf of a component. The left (right resp.) children of the internal nodes on the left- (right-resp.) most path represents covering sets for all the other components. Then we find the smallest σ which connects one component to any other component by running the no-isolated algorithm for every point in that component on the points in the other components.

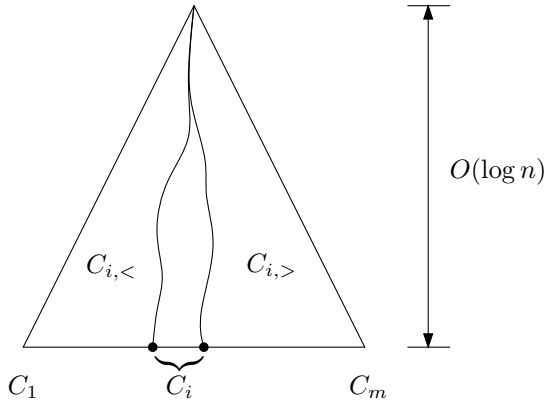


Figure 6.4: Binary tree associated with the components.

Proof (Continued) This reduces the number of components, and we continue up to the point where there is only one component; we can then return the lowest σ.

As the number of components is reduced each time by a factor of two, we repeat the no-isolated vertex algorithm $O(\log n)$ times, on $O(\log n)$ sets covering every component but the current one.

The time to initially construct and maintain the structure is dominated by the search of the optimum value; the complexity is thus $O(n \log^7 n)$. □

Planar Embedding

The last property we analyze is whether the σ-local graph has a planar graph embedding when drawing edges as straight line segments. The following algorithm allows us to find the largest value of σ for which this is the case, and as a by-product constructs the corresponding graph.

Theorem 6.10 *Given a set S of n points in $\mathbb{R}^2$, the planar σ-local graph with maximum σ can be constructed in $O(n \log^{O(1)} n)$ time.*

Proof Using Lemma 6.4, we can construct in $O(n \log^{O(1)} n)$ time a graph containing more than $3n - 6$ edges. We sort these edges by increasing weight in $O(n \log^{O(1)} n)$ time and keep the $3n - 6$ first ones. This gives us an upper bound on the value of σ, as a planar graph has $3n - 6$ edges or less.

Given a set of n line segments in $\mathbb{R}^2$, the algorithm proposed by Bentley and Ottmann [BO79] returns all pairs of segments intersecting each other in $O((n+k) \log n)$ time, where k is the number of intersections in the set. The same algorithm can be used to check if at least one segment intersects with any other, by stopping as soon as it finds one intersection; this can be achieved in $O(n \log n)$ time.

We do a binary search on the size s of the set of edges (initialized to $3n - 6$), using Bentley-Ottmann to check if the first s edges in sorted order intersect or not. This gives a time complexity of $O(n \log^2 n)$ to determine the maximum value of σ corresponding to a planar σ-local graph. □

Chapter 7

Conclusion

The anchored regions and the region-counting distances have applications going far beyond their original purpose of rank difference generalization. We proposed approaches for the two objectives described in the introduction, namely the analysis and the definition of geometric graphs.

While there is definitively space for further improvements, as we made several hypotheses and restrictions on the regions and graphs we analyzed, we hope that this work will be the starting point for user-oriented geometric graphs, where the constraints are given as input, and not as a by-product of the study of some graph.

The next section regroups original contributions of this work. We conclude with the description of perspectives and future work.

Summary of Results

Our results mainly concern three topics: the analysis of region-counting distances and neighborhood regions, their use for the analysis of geometric graphs, and the definition of new graphs where the required properties are part of the input.

Region-counting Distances

The region-counting distances were introduced to efficiently solve point searching and point location problems [DIL04, IL03]. Although they have been successfully used in these contexts, their study was incomplete. In Chapter 3, we went further in their analysis.

First, we presented their properties (Section 3.2), and we gave a model for defining anchored regions (Subsection 3.3.1). We formalized the concepts of region-counting disks and circles (Section 3.3). Interestingly, it turned out that for a wide class of regions, the complexity of a region-counting circle of radius k in a set of n points is either at least as large as the complexity of the k^{th} level in an arrangement of n lines, or is linear in n. We gave a complete characterization of regions falling into one of these two cases.

As region-counting disks and circles can be complex objects, we defined ϵ-approximations of region-counting distances, leading to approximated disks and circles (Subsection 3.3.3).

We also compared region-counting distances with another family of parameterized distances: the speed distances (Section 3.4). We were able to show that those families are related and that in some sense the region-counting distances are discrete speed distances.

Analysis of Geometric Graphs

Our results on that topic were presented in Chapter 4. We wanted to give improved bounds for close successive queries in geometric graphs. Therefore we introduced the local properties (Section 4.2), and illustrated the concept with an analysis of the local diameter (Subsection 4.3.2), and local point and path query complexities.

We considered local properties of various t-spanners such as Θ-graphs and their variants, and proximate point searching graphs (Section 4.4). In particular, we showed that path queries can be answered in time linear in the size of the neighborhood of the considered vertices for Θ-graphs and proximate point searching graphs, while path queries can be answered in logarithmic time for ordered Θ-graphs and skip list spanners, using the ellipse region-counting distance. For point queries, we obtained a time linear in the size of the neighborhood for Θ-graphs and ordered Θ-graphs.

These results achieve our primary goal: given any pair of vertices we now have a way to upper bound the time needed to perform queries which only depends on the local density of the considered graph, and not on the size of the entire graph.

Empty Region Graphs

A new family of proximity graphs, the empty region graphs, was introduced in Chapter 5. We defined them using anchored template regions (Section 5.3). Two vertices form an edge if and only if the anchored region around them is empty.

The main result is a new way of defining geometric graphs, in which one can choose (a combination of) some monotone properties that the graphs are required to satisfy (Subsection 5.8).

We first analyzed how properties were related to the choice of template region. The foundation of our approach is the fact that for every monotone property, there exists a corresponding set (possibly infinite) of extremal anchored regions, that we call tight regions (Subsection 5.4). From there, any region satisfies the property if it contains, or is contained in, one of these tight regions.

We considered geometric properties (to have a planar embedding, to have a bounded spanning ratio) which can be ensured by the proper choice of template regions. We studied how to ensure that the resulting graph does not contain k-stars or k-cycles; we also determined the template regions corresponding to cycle freeness, planarity, connectivity and bipartiteness. (Section 5.4).

This way of defining proximity graphs is appealing, but it is useless if we cannot construct the corresponding graphs efficiently. At the end of Chapter 5, we gave a generic framework to construct ERGs.

In Chapter 6, we chose the example of σ-local graphs to illustrate how we can efficiently construct a subclass of the empty region graphs. We showed how to construct them in $\mathbb{R}^d$ in time $O(\sigma^d n + n \log n)$ and, in the plane, in time $O(n \log^3 n \log\log n + k)$, where k is the size of the output. We also gave algorithms to find the optimal value of the parameter σ such that the corresponding σ-local graphs have various properties, such as planarity and connectivity (Section 6.3).

Erickson [Eri03] used this definition of σ-locality as a realistic input model for nonconvex polyhedra, and showed that Boolean combinations of two σ-local polyhedra defined by n vertices in $\mathbb{R}^d$ can be performed in $O(n \log n)$ time. Further to Erickson's work, the study of σ-local graphs might prove useful for the analysis of all kinds of objects defined in that realistic input model.

Perspectives and Future Work

Neighborhoods

In Chapter 4 we studied local properties so that we could get better bounds on the complexities of queries for geometrically close query points. Our approach consisted in analyzing some properties which depended only on the vertices in the neighborhood of the considered points.

We considered the ellipse neighborhood to analyze locality, because of the nice properties of t-spanner graphs (see Subsection 4.3.1). Of course, if we had considered other neighborhood regions, our results would have been different.

Recall that the ***local diameter*** of a graph is an upper bound on the size of the shortest path between any pair of vertices, expressed as a function of the region-counting distance $\mathrm{d}_R(p, q)$ between these vertices. If the neighborhood R is large enough, the local diameter of any graph is the same as its "global" diameter. On the other hand, if the neighborhood is the line segment, the local diameter is unbounded. This is why we should try to determine the class of template regions, or at least the smallest region, such that some local property holds.

For instance, in the case of Θ-graphs we proved that the local diameter is $O(\mathrm{d}_{E_t}(p, q))$. Then, we know that for the Θ-graphs, the local diameter is also $O(\mathrm{d}_R(p, q))$ for any region $R(p, q) \supseteq E_t(p, q)$. Thus, for a given class of graphs, "to have a local diameter in $O(f(\mathrm{d}_R))$" using the region-counting distance d_R is a monotone property with respect to the partial order of inclusion of template regions R, raising similar issues as the ones analyzed in Chapter 5: are there tight regions? For a given local property, is that region unique?

Tight Regions and Monotone Properties

In Chapter 5, we showed that some properties of empty region graphs are related to what we called ***tight regions***. For some properties, a unique tight region could be found, others had a finite number of tight regions, and some properties corresponded to an infinite set of tight regions.

In some sense, tight regions are similar to threshold values for monotone graph properties studied in random graph theory [FK96]. A random graph on a set V of vertices is a graph $G(V, E)$ where every edge appears in E according to some probability dis-

tribution. In that context, it is known that every monotone property corresponds to a unique threshold p; every random graph on n vertices with more than pn edges satisfies the property with high probability.

Another result which is similar concerns forbidden minors. A proof of more than 500 pages, due to Robertson and Seymour [RS83, RS04], shows that every monotone graph property can be expressed as a finite set F of forbidden minors, such that any graph G satisfies the property if and only if it has no element of F as minor.

This leads us to wonder why, in our analysis, we could prove that some properties admit an infinite number of tight regions. Or in other words: can we characterize the family of properties for which the tight region is unique, or for which there is a finite set of tight regions? Of course as we proved the existence of tight regions, we know that all monotone properties can be ensured with an infinity of tight regions.

Note that even if any monotone property can be ensured by a finite set of forbidden minors, each of these minors represent an infinity of forbidden subgraphs. In the case of empty region graphs, can we find a simple finite representation of the infinite set of tight regions corresponding to one property?

A first step to the understanding of why multiple tight regions exist, is perhaps to analyze the constraint added by the class of anchored regions we consider. We analyzed the class of convex and symmetric regions. Imagine that we restrict the set of anchored regions to the ellipses $E_t(p, q)$ for all t. As $E_{t_1}(p, q) \subseteq E_{t_2}(p, q)$ for all $t_1 \leq t_2$, there is a total order of inclusion and every monotone property has a unique tight region among the ellipses. What is the least restricted class of anchored regions which would still ensure a unique tight region (or at least a finite number of tight regions) for every monotone property?

Even for the class of regions we considered (convex and symmetric regions), in which we know that some properties require an infinite number of tight regions, there are still properties which cannot be satisfied together: cycle-freeness and connectedness is a typical example of combination of properties that empty region graphs cannot ensure, while the properties do not seem to be incompatible. Minimum spanning trees, for instance, combine these properties. This suggests to try a model in which we would not be restricted to anchored regions, as these limitations seems to be bound to the approach that we used.

Bibliography

[ADD+93] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, *On sparse spanners of weighted graphs*, Discrete & Computational Geometry **9** (1993), no. 1, 81–100.

[ADM+95] S. Arya, G. Das, D.M. Mount, J.S. Salowe, and M. Smid, *Euclidean spanners: short, thin, and lanky*, Proceedings of the 27th ACM Symposium on Theory of Computing (STOC), 1995, pp. 489–498.

[AE99] P.K. Agarwal and J. Erickson, *Geometric range searching and its relatives*, Advances in Discrete and Computational Geometry (B. Chazelle, J. E. Goodman, and R. Pollack, eds.), vol. 223, American Mathematical Society, 1999, pp. 1–56.

[AG86] N. Alon and E. Györi, *The number of small semispaces of a finite set of points in the plane*, Journal of Combinatorial Theory Series A **41** (1986), no. 1, 154–157.

[Aga04] P.K. Agarwal, *Range searching*, Handbook of Discrete and Computational Geometry (Jacob E. Goodman and Joseph O'Rourke, eds.), CRC Press LLC, 2004, pp. 809–837.

[AHL90] A. Aggarwal, M. Hansen, and T. Leighton, *Solving query-retrieval problems by compacting Voronoi diagrams*, Proceedings of the 22nd ACM Symposium on Theory of Computing (STOC), 1990, pp. 331–340.

[AM92] P.K. Agarwal and J. Matoušek, *Relative neighborhood graphs in three dimensions*, Computational Geometry: Theory and Applications **2** (1992), 1–14.

[AMS94] S. Arya, D.M. Mount, and M. Smid, *Randomized and deterministic algorithms for geometric spanners of small diameter*, Proceedings of the 35th

IEEE Symposium on Foundations of Computer Science (FOCS), 1994, pp. 703–712.

[AP95] N. M. Amato and F. P. Preparata, *A time-optimal parallel algorithm for three-dimensional convex hulls*, Algorithmica **14** (1995), no. 2, 169–182.

[BDEK06] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick, *On the spanning ratio of Gabriel graphs and β-skeleton*, SIAM Journal on Discrete Mathematics **20** (2006), no. 2, 412–427.

[BDLL95] P. Bose, G. Di Battista, W. Lenhart, and G. Liotta, *Proximity constraints and representable trees*, Proceedings of the DIMACS International Workshop on Graph Drawing (GD '94), 1995, pp. 340–351.

[BGM04] P. Bose, J. Gudmundsson, and P. Morin, *Ordered theta graphs*, Computational Geometry: Theory and Applications **28** (2004), no. 1, 11–18.

[BLL96] P. Bose, W. Lenhart, and G. Liotta, *Characterizing proximity trees*, Algorithmica **16** (1996), no. 1, 83–110.

[BO79] J.L. Bentley and T.A. Ottmann, *Algorithms for reporting and counting geometric intersections*, IEEE Transactions on Computers **C-28** (1979), no. 9, 643–647.

[Bor26] O. Borůvka, *O jistém problému minimálním*, Práce Moravske Přírodovědecke Spolecnosti 3 (1926), 37–58.

[BT80] M. Brown and R. Tarjan, *Design and analysis of a data structure for representing sorted lists*, SIAM Journal on Computing **9** (1980), 594–614.

[CCJ91] B. Clark, C. Colbourn, and D. Johnson, *Unit disk graphs*, Discrete Mathematics **86** (1991), no. 1-3, 165–177.

[CCL04] J. Cardinal, S. Collette, and S. Langerman, *Local properties of geometric graphs*, Proceedings of the 16$^{\text{th}}$ Canadian Conference on Computational Geometry (CCCG), 2004, pp. 145–148.

[CCL05a] J. Cardinal, S. Collette, and S. Langerman, *Region counting circles*, Proceedings of the 17$^{\text{th}}$ Canadian Conference on Computational Geometry (CCCG), 2005, pp. 278–281.

[CCL05b] J. Cardinal, S. Collette, and S. Langerman, *Region counting graphs*, Proceedings of the 21st European Workshop on Computational Geometry (EuroCG), 2005, pp. 1–4.

[CCLar] J. Cardinal, S. Collette, and S. Langerman, *Local properties of geometric graphs*, Computational Geometry: Theory and Applications (to appear), Special issue of selected papers from the 16th Canadian Conference on Computational Geometry (CCCG'04).

[Cha05] T.M. Chan, *On levels in arrangements of curves, II: A simple inequality and its consequences.*, Discrete & Computational Geometry **34** (2005), no. 1, 11–24.

[Che86] L.P. Chew, *There is a planar graph almost as good as the complete graph*, Proceedings of the 2nd Annual ACM Symposium on Computational Geometry (SoCG), 1986, pp. 169–177.

[Che89] L.P. Chew, *There are planar graphs almost as good as the complete graph*, Journal of Computer and System Sciences **39** (1989), no. 2, 205–219.

[Cie97] K. Ciesielski, *Set theory for the working mathematician*, Cambridge University Press, 1997.

[Cim92] R.J. Cimikowski, *Properties of some Euclidean proximity graphs*, Pattern Recognition Letters **13** (1992), no. 6, 417–423.

[CK95] P.B. Callahan and S.R. Kosaraju, *A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields*, Journal of the ACM **42** (1995), no. 1, 67–90.

[Cla87] K. Clarkson, *Approximation algorithms for shortest path motion planning*, Proceedings of the 19th ACM Symposium on Theory of Computing (STOC), 1987, pp. 56–65.

[CLR90] T. Corman, C. Leiserson, and R. Rivest, *Introduction to algorithms*, MIT Press, 1990.

[CMSS00] R. Cole, B. Mishra, J.P. Schmidt, and A. Siegel, *On the dynamic finger conjecture for splay trees. Part I: Splay sorting log n-block sequences*, SIAM Journal on Computing **30** (2000), no. 1, 1–43.

[Col00] R. Cole, *On the dynamic finger conjecture for splay trees. Part II: The proof*, SIAM Journal on Computing **30** (2000), no. 1, 44–85.

[Dey98] T.K. Dey, *Improved bounds on planar k-sets and k-levels*, Discrete & Computational Geometry **19** (1998), 373–382.

[DFS90] D. Dobkin, S. Friedman, and K. Supowit, *Delaunay graphs are almost as good as complete graphs*, Discrete & Computational Geometry **5** (1990), 399–407.

[DIL04] E.D. Demaine, J. Iacono, and S. Langerman, *Proximate point searching*, Computational Geometry: Theory and Applications **28** (2004), no. 1, 29–40, Special issue of selected papers from the 14$^{\text{th}}$ Canadian Conference on Computational Geometry (CCCG'02).

[DJ89] G. Das and D. Joseph, *Which triangulations approximate the complete graph?*, Proceedings of the International Symposium on Optimal algorithms, 1989, pp. 168–192.

[DK83] D.P. Dobkin and D.G. Kirkpatrick, *Fast detection of poyhedral intersection*, Theoretical Computer Science **27** (1983), 241–253.

[DLL95] G. Di Battista, W. Lenhart, and G. Liotta, *Proximity drawability: a survey*, Proceedings of the DIMACS International Workshop on Graph Drawing (GD '94), 1995, pp. 328–339.

[Ede87] H. Edelsbrunner, *Algorithms in combinatorial geometry*, Springer-Verlag, 1987.

[ELSS73] P. Erdös, L. Lovász, A. Simmons, and E.G. Straus, *Dissection graphs of planar point sets*, A Survey of Combinatorial Theory, North-Holland, 1973, pp. 139–149.

[EPY97] D. Eppstein, M. Paterson, and F. Yao, *On nearest-neighbor graphs*, Discrete & Computational Geometry **17** (1997), 263–282.

[Eri03] J. Erickson, *Local polyhedra and geometric graphs*, Proceedings of the 19$^{\text{th}}$ Annual ACM Symposium on Computational Geometry (SoCG), 2003, pp. 171–180.

[FK96] E. Friedgut and G. Kalai, *Every monotone graph property has a sharp threshold*, Proceedings of the AMS, 1996, pp. 2993–3002.

[GS69] K. Gabriel and R. Sokal, *A new statistical approach to geographic variation analysis*, Systematic Zoology **18** (1969), 259–278.

[HP00] S. Har-Peled, *Taking a walk in a planar arrangement*, SIAM Journal on Computing **30** (2000), no. 4, 1341–1367.

[HW87] D. Haussler and E. Welzl, *Epsilon-nets and simplex range queries*, Discrete & Computational Geometry (1987), no. 2, 127–151.

[IL03] J. Iacono and S. Langerman, *Proximate point location*, Proceedings of the 19$^{\text{th}}$ Annual ACM Symposium on Computational Geometry (SoCG), 2003, pp. 220–226.

[IS85] M. Ichino and J. Sklansky, *The relative neighborhood graph for mixed feature variables*, Pattern Recognition **18** (1985), 161–167.

[JT92] J. Jaromczyk and G. Toussaint, *Relative neighborhood graphs and their relatives*, Proceedings of the IEEE **80** (1992), no. 9, 1502–1571.

[KG89] J.M. Keil and C.A. Gutwin, *The Delauney triangulation closely approximates the complete Euclidean graph*, Proceedings of the Workshop on Algorithms and Data Structures (WADS '89), 1989, pp. 47–56.

[KG92] J.M. Keil and C.A. Gutwin, *Classes of graphs which approximate the complete Euclidean graph*, Discrete & Computational Geometry **7** (1992), no. 1, 13–28.

[KR85] D. Kirkpatrick and J. Radke, *A framework for computational morphology*, Computational Geometry (1985), 217–248.

[Lov71] L. Lovász, *On the number of halving lines*, Ann. Univ. Sci. Budapest, Eötvös, Sect. Math. **14** (1971), 107–108.

[LS93] A. Lubiw and N. Sleumer, *Maximal outerplanar graphs are relative neighbourhood graphs*, Proceedings of the 5$^{\text{th}}$ Canadian Conference on Computational Geometry (CCCG), 1993, pp. 198–203.

[Mat99] J. Matoušek, *Geometric discrepancy (an illustrated guide)*, Springer-Verlag, 1999.

[Meg83] N. Megiddo, *Applying parallel computation algorithms in the design of serial algorithms*, Journal of the ACM **30** (1983), no. 4, 852–865.

[Mor01] P. Morin, *Online routing in geometric graphs*, Ph.D. thesis, Carleton University, School of Computer Science, 2001.

[MR95] R. Motwani and P. Raghavan, *Randomized algorithms*, Cambridge University Press, 1995.

[MS80] D. Matula and R. Sokal, *Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane*, Geographical Analysis **12** (1980), 205–222.

[OBSC00] A. Okabe, B. Boots, K. Sugihara, and S.N. Chiu, *Spatial tessellations: Concepts and applications of voronoi diagrams*, 2nd ed., Wiley Series in Probability and Statistics, John Wiley & Sons Ltd., 2000.

[Pug90] W. Pugh, *Skip lists: A probabilistic alternative to balanced trees*, Communications of the ACM **6** (1990), 668–676.

[RL73] B. Rosenberg and D.J. Langridge, *A computational view of perception*, Perception **2** (1973), 415–424.

[RS83] N. Robertson and P.D. Seymour, *Graph minors. I. Excluding a forest*, Journal of Combinatorial Theory Series B **35** (1983), no. 1, 39–61.

[RS91] J. Ruppert and R. Seidel, *Approximating the d-dimensional complete Euclidean graph*, Proceedings of the 3rd Canadian Conference on Computational Geometry (CCCG), 1991, pp. 207–210.

[RS92] J. Reif and S. Sen, *Optimal parallel randomized algorithms for three-dimensional convex hulls and related problems*, SIAM Journal on Computing **21** (1992), no. 3, 466–485.

[RS04] N. Robertson and P. D. Seymour, *Graph minors. XX. Wagner's conjecture*, Journal of Combinatorial Theory Series B **92** (2004), no. 2, 325–357.

[SA96] M. Sharir and P.K. Agarwal, *Davenport-schinzel sequences and their geometric applications*, Cambridge University Press, 1996.

[Sha91] M. Sharir, *On k-sets in arrangements of curves and surfaces*, Discrete & Computational Geometry **6** (1991), 593–613.

[She93] N. Sherwani, *Algorithms for VLSI physical design automation*, Kluwer, 1993.

[ST85] D. Sleator and R. Tarjan, *Self-adjusting binary search trees*, Journal of the ACM **32** (1985), no. 3, 652–686.

[TÓ0] G. Tóth, *Point sets with many k-sets*, Proceedings of the 16$^{\text{th}}$ Annual ACM Symposium on Computational Geometry (SoCG), 2000, pp. 37–42.

[Tou80] G.T. Toussaint, *The relative neighbourhood graph of a finite planar set*, Pattern Recognition **12** (1980), 261–268.

[Vai89] P. Vaidya, *An $O(n \log n)$ algorithm for the all-nearest neighbors problem*, Discrete & Computational Geometry **4** (1989), 101–115.

[VC71] G. Vapnik and A. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of probability and its applications **16** (1971), no. 2, 264–280.

[vEB90] P. van Emde Boas, *Handbook of theoretical computer science: Volume a: Algorithms and complexity*, Elsevier, 1990.

[Vel92] R. Veltkamp, *The γ-neighbourhood graph*, Computational Geometry: Theory and Applications **1** (1992), 227–246.

[WLM+03] W. Wang, X. Li, K. Moaveninejad, Y. Wang, and W. Song, *The spanning ratio of beta-skeletons*, Proceedings of the 15$^{\text{th}}$ Canadian Conference on Computational Geometry (CCCG), 2003.

[Yao82] A. Yao, *On constructing minimum spanning trees in k-dimensional spaces and related problems*, SIAM Journal on Computing **11** (1982), no. 4, 721–736.

[Zah71] C.T. Zahn, *Graph-theoretical methods for detecting gestalt clusters*, IEEE Transactions on Computers **C-20** (1971), 68–86.

Index

G

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X