Local Properties of Geometric Graphs

Jean Cardinal[1] Sébastien Collette[2] Stefan Langerman[3]

Computer Science Department, Université Libre de Bruxelles, CP212, Boulevard du Triomphe, 1050 Bruxelles, Belgium

Abstract

We propose a definition of locality for properties of geometric graphs. We measure the local density of graphs using *region-counting distances* [8] 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 Θ-graphs, Ordered Θ-graphs and Skip List Spanners. 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.

1 Introduction

We consider here *geometric graphs*, where the vertices are points in the plane. In particular we consider graphs generated by a set of points. Some of these graphs belong to the the class of *proximity graphs* [11], where given a set of points S, two points in S are adjacent if they are close in some sense. Many such graphs have been studied in computational geometry, such as Delaunay triangulations, Nearest Neighbor graphs, Relative Neighborhood graphs, Gabriel graphs and Θ-graphs [13, 11, 9, 12]. The properties defined on such graphs are most of the time expressed as a function of n, the number of vertices in the graph. For instance, the Nearest Neighbor graphs have n edges, the degree of the vertices in a Delaunay triangulation is $O(n)$, etc. In this context, this is what we call *global properties*.

[1] jcardin@ulb.ac.be
[2] Aspirant du F.N.R.S., sebastien.collette@ulb.ac.be
[3] Chercheur qualifié du F.N.R.S., stefan.langerman@ulb.ac.be

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 *region-counting distances* [8, 10]: 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 an upper bound on the size (number of edges) of the shortest path between any pair of vertices. We define the corresponding *local diameter* as the same upper bound, but as a function of the region-counting distance between the pair of points.

Geometric graphs are used in numerous fields: motion planning, VLSI design, Geographic Information Systems [7, 16]; in these contexts they are often used as data structures for a set of points and they have to support different kinds of queries: check if a point in $\mathbb{R}^2$ belongs to the set S defining the graph (point query), find the closest vertex in the graph to a given point in $\mathbb{R}^2$ (nearest neighbor query), or find a path in a graph from one vertex to another (path query).

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. The complexity of this operation is usually expressed as a function of n, the number of keys in the set. For that problem, interesting properties were studied. The *dynamic finger property* certifies that a query can be answered in a time logarithmic in the rank difference between the current and the previous query. The *rank difference* between two items in $\mathbb{R}$ is the number of items contained in the interval between them. For instance, Level-Linked Trees of Brown and Tarjan [3] and Splay Trees of Tarjan and Sleator [17] have the dynamic finger property. These properties have been extended to the 2-dimensional case. Demaine, Iacono and Langerman generalized the rank difference and the dynamic finger property to the plane [8]. The proposed generalizations of the rank difference are region-counting distances, and the dynamic finger property in $\mathbb{R}^2$ is the same as in $\mathbb{R}$ but uses these distances. In this work, 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 has not, the dynamic finger property generalized to the plane.

In section 2, we describe the region-counting distances and give examples of neighborhood regions used in what follows. In section 3, we introduce some properties of geometric graphs and we describe t-Spanner graphs. Section 4 is dedicated to the study of locality: we give bounds on the local diameter of Ordered Θ-graphs [2], Skip List Spanners [1] and Proximate Point Searching structures [8]. We study the complexity of point and path queries and see how they are related to the local diameter. A table summarizing the results is presented and commented in section 5.

2 Region-counting Distances

The region-counting distances are a natural class of 2-dimensional distance functions introduced in [8] by Demaine, Iacono and Langerman. They were used in different contexts: point searching and location [8, 10] and definition of proximity graphs [5]; their properties were studied in [4].

The point set S is an implicit parameter of these distance functions. An influence region defines the neighborhood around any pair of points which must be considered to compute the distance:

Definition 1 *An* influence region R *is a function mapping a pair* (p, q) *of points in* $\mathbb{R}^2$ *to a subset* $R(p, q)$ *of* $\mathbb{R}^2$*.*

Definition 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*.*

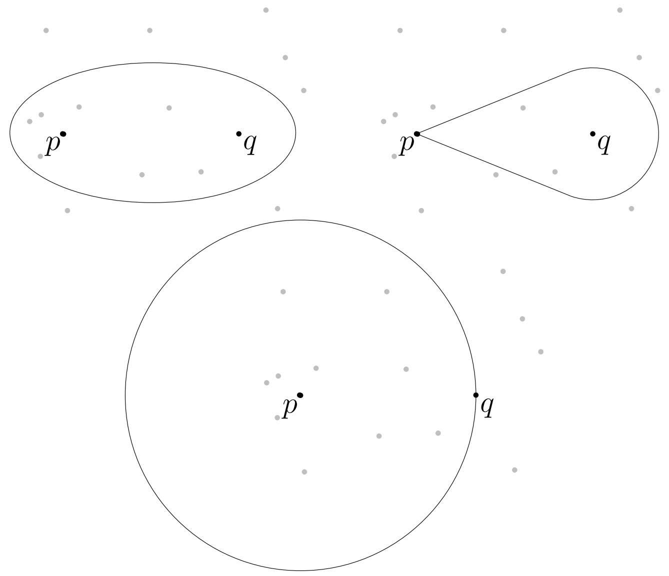


Fig. 1. Ellipse region-counting distance $\mathrm{d}_{E_t}(p, q) = 9$, ice-cream cone region-counting distance $\mathrm{d}_{I_t}(p, q) = 4$ and circle region-counting distance $\mathrm{d}_C(p, q) = 12$.

The distance is the cardinality of the intersection of the dataset and the neighborhood:

Definition 3 *A* region-counting distance $\mathrm{d}_R = \mathrm{d}_R^S(p,q)$ *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 1 shows three different sample regions which can be used to compute the region-counting distance between two vertices p and q. We denote by d_{R_t} a region-counting distance whose shape depends on a parameter t. A given region-counting distance is defined by an unique template region, with fixed parameter.

The *ice-cream cone* [8] $I_t(p,q)$ is the convex hull of p and a disk of radius $t \cdot \mathrm{d}_2(p,q)$ centered at q, where d_2 is the Euclidean distance. The ellipse $E_t(p,q)$ is the locus of the points r such that $\mathrm{d}_2(p,r) + \mathrm{d}_2(r,q) = t \cdot \mathrm{d}_2(p,q)$.

3 t-Spanner Graphs

t-Spanner graphs are a subclass of geometric graphs in which the length of the shortest path joining two vertices is within a factor of the Euclidean distance between them. We will concentrate on this class of graphs as we will see that they are good candidates for the study of local properties.

Definition 4 *The* length of a path $P = \{u_1, u_2, \ldots, u_k\}$ *in the graph* G *is* $\mathrm{d}_G(P) = \sum_{i=1}^{k-1} \mathrm{d}_2(u_i, u_{i+1})$ *where* d_2 *is the Euclidean distance.*

Definition 5 *A graph in the plane is a* t-Spanner *if and only if*

$$\forall u, v \in V \ : \ \frac{\mathrm{d}_G(u,v)}{\mathrm{d}_2(u,v)} \leq t \in [1, \infty),$$

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

In what follows, we present several classes of t-Spanner graphs and give bounds on their spanning ratio. Local properties of these graphs are studied in the next sections.

3.1 Θ-graph

Chew [6] defined the t-Spanners with the aim of approximating the complete Euclidean graph. Keil [12] introduced the Θ-graph to provide a way to construct t-Spanners efficiently. This structure is similar to the Yao graph [19], which was introduced ten years before.

A Θ-graph G is a directed graph defined by a point set V. In a Θ-graph, each vertex has up to k outgoing edges connected to the closest vertex in $k = 2\pi/\Theta$ different cones.

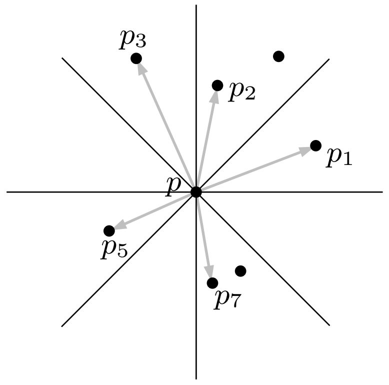


Fig. 2. Plane subdivision using cones, as used in Θ-graphs and variants.

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. There are $k = 2\pi/\Theta$ cones for each vertex. Figure 2 shows the cones associated with one vertex. In each cone i of the vertex p, the outgoing edge is connected to the closest vertex, denoted by p_i. Various definitions of the closest vertex can be found in the literature; while the Euclidean distance (Figure 3a) seems to be the most natural, the closest vertex in a cone is defined in [12, 19] as the one having the closest orthogonal projection onto the border of the cone (Figure 3b) in e.g. [2]. Other authors define it as the vertex with the closest orthogonal projection onto the bisector of the cone (Figure 3c) e.g. [15].

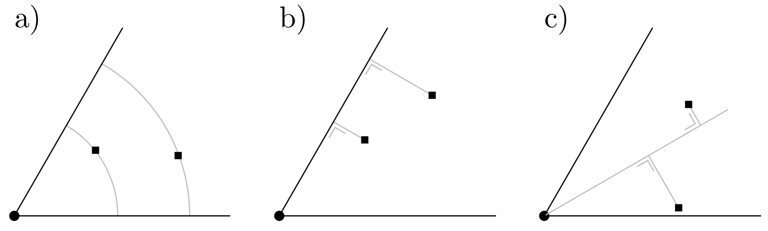


Fig. 3. Different ways to see which is the closest vertex in a cone.

Θ-graphs have at most $k(n-1)$ edges, at most k outgoing edges per vertex and at most $n-1$ in-going edges. Ruppert and Seidel [15] 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. The variant in which the points are projected onto the closest border of the cone leads to t-Spanner graphs with the same spanning ratio. To find a t-Spanner path, which is a path in the graph achieving the spanning ratio, the algorithm always follows the edge in the cone containing the other endpoint of the path, until it is reached.

3.2 Ordered Θ-graph

The Ordered Θ-graph $G = (V, \pi)$ [2] 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, as a solution to some weaknesses of the original Θ-graph. The difference is that in the Ordered Θ-graph the order of insertion of the vertices π matters: each vertex is connected to up to k edges, which are the closest previously inserted vertices in the k cones. If the order π is a random permutation of the set of vertices, the corresponding graph has a small diameter. 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 a constant out-degree. Its spanning ratio is $t \leq \frac{1}{\cos(\Theta)-\sin(\Theta)}$.

3.3 Skip List Spanner

The Skip List is a data structure introduced by Pugh [14] to create lists with low access cost. Using this structure, any item in the list can be reached in $O(\log n)$ time. The Skip List Spanner $G = (V, \omega)$ [1] is a Skip List extension of the Θ-graph introduced by Arya, Mount and Smid, where every vertex is assigned to a level. 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 regular 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 logarithmic with high probability. The spanning ratio is the same as the Θ-graph, as the Skip List Spanner contains it.

The Proximate Point Searching graph, as defined in [8], 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 [1]: a vertex is surrounded by k non-overlapping cones; the edges in the cone k handle travel in a direction with absolute angle between $2\pi i/k$ and $2\pi(i+1)/k$.

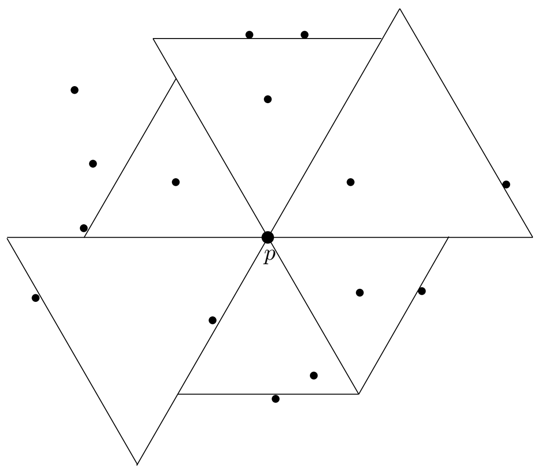
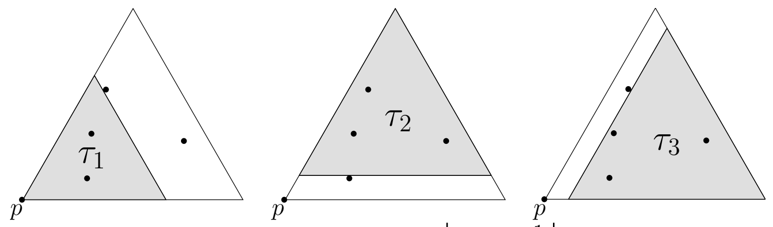


Fig. 4. Triangles in each cone at level 2.

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 triangle τ, containing the $(3/2)^r$ closest points in that cone, if possible. Figure 4 shows the triangles present at level two. We denote by $\tau_{(p,r,i)}$ the triangle τ associated with vertex p, in cone i and at level r.



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

Fig. 5. Subdivision of τ in three overlapping sub-triangles.

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 5 shows three views of the triangle τ at level four, and thus containing at most five points. For each cone 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 [8] that $\tau_1 \cup \tau_2 \cup \tau_3 \supseteq \tau$.

Lemma 6 *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 such a vertex is present, and that 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 3c). $\square$

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 [8].

4 Local Properties of Geometric Graphs

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

Definition 7 *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 8 *A geometric graph* G *is said to* have the property $\mathcal{P}$ *when this relation holds for all pair* p, q *in* V.

Intuitively, a property is local if it depends only on vertices close to the ones considered. The next subsection shows why t-Spanner graphs are good candidates to study locality. We then analyze three local properties: 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.

4.1 A Property of t-Spanner Paths

Definition 9 *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 path between two vertices is not always the path with minimal length or with minimal size. We know however that in any t-Spanner graph, there are t-Spanner paths between every pair of vertices.

Lemma 10 *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.*

PROOF. An ellipse with foci p and q and parameter t is the locus of the points whose sum of the distance to the two foci is equal to t times the Euclidean distance between the foci. 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. □

4.2 Local Diameter

The *local diameter* of a graph is the upper bound on the size of the shortest path between any pair of vertices, expressed as a function of the region-counting distance between these vertices.

Theorem 11 *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. □

For the Ordered Θ-graph and the Skip List Spanner, we can however improve this result, as stated in the following theorem:

Theorem 12 *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. We consider a path from p to q in G and G'. We will simulate the use of the path algorithm in Ordered Θ-graphs as described in [2].

Beginning from both ends p and q, the path algorithm adds an edge and a vertex to the path at each step. By Lemma 10 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. 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 [2] 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 an expected 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. □

4.3 Queries and Computational Model

Graphs are often used to represent data structures. For instance in the *pointer model* [18], the algorithms must use nothing else than a graph as data structure, where every vertex has a fixed number of out-going labeled edges. One of the edges is the *center* or *root*, by which we begin to visit the graph. The allowed operations in that model consist in following the edges, compare two vertices to determine if they are the same, create or modify existing edges, create or modify 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, delete, modify and compare an edge or a vertex in $O(1)$ time, 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, 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.

4.4 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 10, 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 latest vertex added), following the edge in the cone containing the other endpoint of the path, until it is reached. 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 13 *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 12, 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 algorithm can be achieved in constant time, resulting in a $O(\log \mathrm{d}_{E_t}(p, q))$ bound for path queries in Skip List Spanners and Ordered Θ-graphs. □

4.5 Point Query

The *point query* consists in, given a data set S and a query q, finding the item q in S if such an item exists. In order to exploit locality, the query algorithm should use the position of the previous query point. For this, the algorithm will follow edges in the graph representing the search structure, along a path from the previous query point p to the present query point q. The number of edges traversed can then be expressed as a local property, i.e. as a function of $\mathrm{d}_R(p, q)$.

Note that the point query has not necessarily the same complexity as the path query. Path queries return t-spanner paths, while for point queries we just want to reach the destination as quickly as possible.

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 [8], whose point query time is $O(\log \mathrm{d}_{I_t}(p, q))$ where d_{I_t} is the ice-cream cone region-counting distance [8]. The ice-cream cone distance is in some sense better than the ellipse distance, because $\forall\, t_0, \exists\, t_1 : I_{t_1}(p, q) \subseteq E_{t_0}(p, q)$ while the converse is not true.

For the other graphs, we know that the number of nodes visited to reach the query is at most the length of the 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 14 *The time complexity of point queries is bounded by the time to perform a path query between the 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 to answer point queries, because the algorithm relies on a method where we construct the path from both ends. This does not make any sense for the point query, as we do not have any information concerning the target, and thus we cannot construct the path by this end. That is why the point query complexity is the same as the one for the Θ-graph.

5 Results and Discussion

The table given by Fig. 6 shows properties of various graphs where n is the number of vertices and d is the region-counting distance between the two considered items and using the corresponding region: the ellipse or the ice-cream cone [8]. Other results summarized in that table come from the original papers describing these graphs [8, 12, 1, 2], and are detailed in section 3. 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 the path query algorithm to perform a point query, as some of them construct the path by 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 the number of edges. The path query is in $O(\log \mathrm{d})$ if we use the point query algorithm, but this is not necessarily a t-Spanner path.

	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.
Minimal Region	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.	$\mathbf{O(\log d)}$
Minimal Region	Ellipse E_t	Ice-Cream Cone I_t

Fig. 6. 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.).

5.1 Future Work

An extension of this work is the study of locality for other classes of graphs. In particular, is there a graph or a class of graph combining all the "good" aspects: a small local diameter, dynamic finger, logarithmic path query complexity, linear number of edges?

We studied extensively the properties of proximity graphs defined by different region counting distances in [5]. We could determine the extremal regions ensuring various properties. We could also try to minimize the region used to characterize locality here: what is the minimal region needed, what are the properties common to all the regions which can be used in that context?

We introduced locality with the local diameter and the complexities of point and path queries. What are the other graph properties which are local, or where a local definition can be found?

Acknowledgment

We would like to thank Pat Morin and the anonymous referees for their helpful comments.

References

[1] S. Arya, D. Mount, and M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 703–712, 1994.

[2] P. Bose, J. Gudmundsson, and P. Morin. Ordered theta graphs. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, pages 17–21, 2002.

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

[4] J. Cardinal, S. Collette, and S. Langerman. Region counting circles. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG05)*, pages 278–281, August 10–12 2005.

[5] J. Cardinal, S. Collette, and S. Langerman. Region counting graphs. In *Proceedings of the 21st European Workshop on Computational Geometry (EWCG05)*, 2005.

[6] L. P. Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the 2nd Annual ACM Symposium on Computational Geometry (SoCG'86)*, pages 169–177, 1986.

[7] K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the 19th ACM Symposium on Theory of Computing (STOC'87)*, pages 56–65, 1987.

[8] E. D. Demaine, J. Iacono, and S. Langerman. Proximate point searching. In *Proceedings of the 14th Canadian Conference on Computational Geometry (CCCG)*, 2002.

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

[10] J. Iacono and S. Langerman. Proximate point location. In *Proceedings*

of the 2003 ACM Symposium on Computational Geometry (SoCG 2003), pages 220–226, 2003.

[11] J. Jaromczyk and G. Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1571, 1992.

[12] J. Keil and C. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete and Computational Geometry*, 7(1):13–28, 1992.

[13] D. Lee and A. Lin. Generalized delaunay triangulations for planar graphs. *Discrete and Computational Geometry*, 1:201–217, 1986.

[14] W. Pugh. Skip lists: A probabilistic alternative to balanced trees. *Commun. ACM*, 6:668–676, 1990.

[15] J. Ruppert and R. Seidel. Approximating the d-dimensional complete euclidean graph. In *Proceedings of the 3rd Canadian Conference on Computational Geometry (CCCG'91)*, pages 207–210, 1991.

[16] N. Sherwani. *Algorithms for VLSI Physical Design Automation*. Kluwer, 1993.

[17] D. Sleator and R. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985.

[18] P. van Emde Boas. *Handbook of Theoretical Computer Science: Volume A: Algorithms and Complexity*. Elsevier, 1990.

[19] A. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.