Local Properties of Geometric Graphs

Jean Cardinal* Sébastien Collette† Stefan Langerman‡ §

Abstract

We introduce a new property for geometric graphs: the *local diameter*. This property is based on the use of the *region counting distances* introduced by Demaine, Iacono and Langerman as a generalization of the rank difference. We use it as a characterization of the local density of the vertices. We study the properties of those distances, and show that there is a strong relation between region counting distances and speed distances. We define the local diameter as the upper bound on the length of the shortest path between any pair of vertices, expressed as a function of the number of vertices in their neighborhood. We determine the local diameter of several well-studied graphs such as the Ordered Θ-graph and the Skip List Spanner. We also analyze the impact of the local diameter for different applications, such as path and point queries on geometric graphs.

1 Introduction

We consider here *geometric graphs*, which are graphs in the plane with vertices in general position, and in particular we consider graphs generated by a set of points. Many such graphs have been studied in computational geometry, such as the Delaunay triangulation, Nearest Neighbor graphs, Relative Neighborhood graphs, Gabriel graphs or Θ-graphs [8, 6, 5, 7].

The properties of such graphs are most of the time expressed as a function of n, the number of vertices in the graph. This is what we call a *global property*. For instance, the *diameter* of a graph is an upper bound on the length (number of edges) of the shortest path between any pair of points, and it is expressed as a function of n. We define the corresponding *local diameter* as the same upper bound, but as a function of the number of vertices in the neighborhood of the pair of points. More generally, a *local property* is any property expressed as a function of the cardinality of a local subset of vertices.

To formalize the local properties, we need to find a distance function which could characterize the local composition of the graph. This function should give the relation between any pair of sites and the number of items in its neighborhood.

Therefore we consider distance functions not only taking two sites as parameters, but also depending on a set of points or on a function. The *speed distances* are a family of distance functions where the distance between two points p and q depends on the function f associating the inverse of the instant speed with each point in the plane, and is defined as the time needed to go from p to q in straight line. The *region counting distances* [4] are distance functions parameterized by a finite point set in which the distance between two points is the count of items inside a shape surrounding those points.

Geometric graphs are used in numerous fields: motion planning, VLSI design, Geographic Information Systems. They can also be considered as data structures e.g. for a set of points. For example, 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 the other (path query).

The point query problem is a bidimensional extension of the dictionary problem, which consists in retrieving the information associated with a totally ordered key in a dataset. The complexity of this operation is most of the time a function of n, the number of keys in the set.

The *dynamic finger property* certifies that a query can be answered in a time logarithmic to the rank difference between the current and the previous query. The *difference in rank* between two items in $\mathbb{R}$ is the number of items contained in the interval between them and does not depend on the whole data set. Thus the dynamic finger is a local property. For instance, Level-Linked Trees of Brown and Tarjan [3] and Splay Trees of Tarjan and Sleator [9] have the dynamic finger property.

Those properties have been extended to the bidimensional case. Demaine, Iacono and Langerman generalized the rank difference and the dynamic finger to the plane [4]. The proposed generalization of the rank difference is the region counting distance, and the dynamic finger in $\mathbb{R}^2$ is the same as in $\mathbb{R}$ but uses the rank difference generalization.

In section 2, we study the region counting distances and the speed distances, and we compare them. We prove a relation: if points are sampled independently from some distribution, then as the number of points

*jcardin@ulb.ac.be
†Aspirant du F.N.R.S., sebastien.collette@ulb.ac.be
‡Chargé de recherches du F.N.R.S., stefan.langerman@ulb.ac.be
§Département d'Informatique, Université Libre de Bruxelles, CP212, Boulevard du Triomphe, 1050 Bruxelles BELGIUM

grows, region counting distances behave like speed distances.

In section 3, we introduce some properties of geometric graphs and describe t-spanner graphs. Then, we compute the local and global diameter of the Ordered Θ-graph [2], the Skip List Spanner [1], the Proximate Point Searching structure [4] and the Multiple Delaunay triangulation. We study implications on the other properties of those graphs, and in particular the impact of the local diameter on the point query and the path query.

A table displaying the bounds and grouping the properties of various graphs is presented and commented in section 4.

2 Region Counting Distances vs. Speed Distances

The region counting distances are a natural class of 2-dimensional distance functions generalizing the one-dimensional rank-difference. The point set S is an implicit parameter of those distance functions.

A **region counting distance** $\mathrm{d}_R = \mathrm{d}_R^S(p,q)$ parameterized by a finite point set $S \subseteq \mathbb{R}^2$ is defined by a triple $R = (a,b,D)$, where a and b are points and D is a region of the plane such that inclusion in D can be computed in $O(1)$ time. $\mathrm{d}_R(p,q) = |S \cap R(p,q)|$ where $R(p,q)$ is obtained by translating, rotating and uniformly scaling D so that a maps to p and b maps to q.

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

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

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

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

Speed distances constitute another class of parameterized distance functions, depending on a function $f(p)$ representing the inverse of the instantaneous speed at point p.

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 $\sigma(t) = p + t\frac{p-q}{\mathrm{d}_2(p,q)}$.

Theorem 1 *Let d_f be a speed distance with $k = \int_P f(x,y)dxdy < \infty$; let $\mathrm{d}_{R_t}^{S_n}$ be a region counting distance, defined by $R_t = (a,b,D_t)$ with D_t strictly containing the segment from a to b and depending on parameter t. If the set S_n 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*

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

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

To create the density function g, we consider the plane P as bounded. If $k = \int_P f(x,y)dxdy$ is defined and is not infinite, we can define $g(x,y) = f(x,y)/k$.

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

3 Local Properties

3.1 Property of t-spanner paths

A geometric graph G is a **t-spanner** if between any two vertices p and q there exists a path whose length is at most t times the Euclidean distance between p and q.

We refer to such a path as a **t-spanner path**, and t is called the spanning ratio of G.

Lemma 2 *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.*

The Θ-graph, introduced by Keil [7], is an example of t-spanner graph. In the Θ-graph, each vertex p has up to k outgoing edges connected to the closest vertex in $k = 2\pi/\Theta$ different cones, each with apex p and absolute angle between $i\Theta$ and $(i+1)\Theta$.

To find a t-spanner path, the algorithm always follows the edge in the cone containing the other extremity of the path, until it is reached.

The Ordered Θ-graph [2] and the Skip List Spanner [1] are Θ-graph variants, and they use the same cone subdivision of the plane. They both are t-spanners.

In the Ordered Θ-graph, we consider the order of insertion of the vertices in the graph: each outgoing edge is connected to the closest previously inserted vertex in a given cone.

In the Skip List Spanner, each vertex is assigned to a level by throwing a fair coin. Half of the vertices are assigned to the first level, one quarter to the second level, one eighth to the third one, etc. For each level, a regular Θ-graph is constructed containing the vertices present in the current and the lower levels.

Other graphs use Skip List-like hierarchical structures. For example the Multiple Delaunay triangulation is built like the Skip List Spanner, but constructing a Delaunay triangulation at each level.

3.2 Path Query

The *path query* consists in, given a graph G and a pair of query vertices (p, q), finding a path between those vertices, if such a path exists. The algorithm must use nothing else than the graph itself as a data structure.

We show that for some graphs, the path query complexity is a function of the region counting distance between the two ends of the path.

We consider the t-spanner path queries. The path query algorithm consists, for both graph, 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 extremity 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 extremity of the path, until it is reached.

Theorem 3 *Path queries are answered in a time $O(\log \mathrm{d}_{E_t}(p, q))$ with high probability in an Ordered Θ-graph or a Skip List Spanner.*

Proof. (Sketch) It is shown in [1, 2] that the path query between p and q is in $O(\log n)$ with high probability for those Θ-graph variants, and we know that it depends only on the items contained in the ellipse $E_t(p, q)$. We prove that the path query is in $O(\log \mathrm{d})$, with d the ellipse region counting distance : let G be an Ordered Θ-graph or a Skip List Spanner, let G' be the Ordered Θ-graph or Skip List Spanner generated with the points contained in $E_t(p, q)$. We show that the path in G is the same as in G' because the algorithm used is the same and that the edges considered by the path algorithm are present in both graphs. The cardinality of G' is d, and thus its global diameter is $O(\log \mathrm{d})$, which is an upper bound on the length of the path.

As the path query algorithm considers $O(1)$ edges for each vertex, its complexity is linear in the number of vertices in the path. □

3.3 Local Diameter

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

We show that the local diameter of the Ordered Θ-graph is logarithmic, because the diameter of the subgraph contained in the ellipse defined by any pair of sites is logarithmic in the size of this subgraph, which is the rank difference between those sites.

Theorem 4 *For every t-spanner, the local diameter for a given pair (p, q) of sites is at most the diameter of the subgraph induced by the vertices in $E_t(p, q)$.*

This comes from the fact that there exists a t-spanner path in this ellipse. This implies a local diameter in $O(\log \mathrm{d})$ for the Ordered Θ-graph and the Skip List Spanner.

3.4 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. The algorithm must use nothing else than the graph itself as a data structure.

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 $\mathrm{d}_R(p, q)$.

Typically, this method can be used when we know that close queries occur frequently. Close vertices influence the query time, while far vertices are not taken into account to answer the query. This is the approach used in the Proximate Point Searching data structure [4], 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 [4].

The ice-cream cone distance is 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.

4 Discussion

Table 1 shows properties of various graphs. n is the number of vertices and $\mathbf{d}$ is the region counting distance between the two considered items and using the mentioned shape: the ellipse or the ice-cream cone [4]. 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.

In the Multiple Delaunay triangulation, the path algorithm considers both ends of the path. As in the Skip List Spanner, the end in the lowest level is used to extend the path with the edge whose angle with the

	Θ-graph	Ordered Θ-g. & Skip List S.	Proximate Point Search.	Multiple Delaunay
Edges	$\frac{2\pi}{\theta}n$	$\frac{2\pi}{\theta}n$	$O(n \log n)$	$O(n)$
Indegree	$O(n)$	$O(n)$	$O(n \log n)$	$O(n)$
Outdegree	$\frac{2\pi}{\theta}$	$\frac{2\pi}{\theta}$	$O(\log n)$	$O(n)$
Spanning Ratio	$t \leq \frac{1}{1-2\sin(\frac{\Theta}{2})}$	$t \leq \frac{1}{\cos(\Theta)-\sin(\Theta)}$	Contains Θ-graph	$t \leq 2.42$
Path query	$\mathbf{O(d)}$	$\mathbf{O(\log d)}$ w.h.p.	$\mathbf{O(d)}$	$O(n)$
Point query	$\mathbf{O(d)}$	$\mathbf{O(d)}$ (Skip List S. only)	$\mathbf{O(\log d)}$	$O(n)$
Diameter	$O(n)$	$O(\log n)$ w.h.p.	$O(\log n)$	$O(\log n)$ w.h.p.
Local diameter	$\mathbf{O(d)}$	$\mathbf{O(\log d)}$ w.h.p.	$\mathbf{O(\log d)}$	$\mathbf{O(\log d)}$ w.h.p.
Minimal Shape	Ellipse	Ellipse	Ice-Cream Cone	Ellipse

Table 1: Comparison of properties of geometric graphs. d is the region counting distance using the given shape. Bounds are worst cases, or hold with high probability when mentioned (w.h.p.).

segment joining both ends is the lowest, until the other end is reached.

This structure has good properties: a small global and local diameter. However, it has bad query performance: at each vertex and for each level, the upper bound on the number of outgoing edges is $O(n)$, resulting in a higher complexity to find the path.

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 $O(\log d)$ if we use the point query algorithm, but this is not a t-spanner path.

4.1 Future Work

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

The shape used to characterize the local properties is also a point we can analyze: what is the minimal shape needed, what are the properties common to all the shapes which can be used in that context.

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

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

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

[7] J. Keil and C. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete Computational Geometry*, (7):13–28, 1997.

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

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