Information Processing Letters 25 (1987) 77–86
North-Holland

6 May 1987

A LINEAR EXPECTED-TIME ALGORITHM FOR COMPUTING PLANAR RELATIVE NEIGHBOURHOOD GRAPHS

Jyrki KATAJAINEN, Olli NEVALAINEN and Jukka TEUHOLA

Department of Computer Science, University of Turku, 20500 Turku, Finland

Communicated by R. Wilhelm
Received 26 June 1986
Revised 13 November 1986

A new algorithm for computing the relative neighbourhood graph (RNG) of a planar point set is given. The expected running time of the algorithm is linear for a point set in a unit square when the points have been generated by a homogeneous planar Poisson point process. The worst-case running time is quadratic on the number of the points. The algorithm proceeds in two steps. First, a supergraph of the RNG is constructed with the aid of a cell organization of the points. Here, a point is connected by an edge to some of its nearest neighbours in eight regions around the point. The nearest region neighbours are chosen in a special way to minimize the costs. Second, extra edges are pruned from the graph by a simple scan.

Keywords: Relative neighbourhood graph, cell technique, region approach, computational geometry

1. Introduction

The *relative neighbourhood graph* (RNG) of a set $V = \{p_1, p_2, \ldots, p_n\}$ of n points in the Euclidean plane is obtained by connecting two points p_i and p_j by an edge if they are *relative neighbours*, i.e.,

$$d(p_i, p_j) \leqslant \max\{d(p_i, p_k), d(p_j, p_k)\}$$
$$\text{for all } k = 1, 2, \ldots, n, \quad k \neq i, j.$$

Here, $d(p_i, p_j)$ denotes the Euclidean distance between the two points p_i and p_j. A geometric interpretation for the definition is that points p_i and p_j are connected with an edge if no other point lies inside lune(p_i, p_j), which is defined as the intersection of two circles with radii equal to $d(p_i, p_j)$ centered at p_i and p_j (see Fig. 1).

The relative neighbourhood graph problem, in which the RNG must be computed for a given point set V, was originally introduced by Toussaint [10]. Since then, several papers have been published on RNG-algorithms. For recent developments, the reader is referred to [5], for example.

In this article we consider the case of Euclidean point sets and give a *simple* RNG-algorithm which runs in linear expected-time for a special distribution of points. Katajainen and Nevalainen [5] proved that there exists a linear expected-time RNG-algorithm on the assumption that the point set, for which the RNG is determined, has been

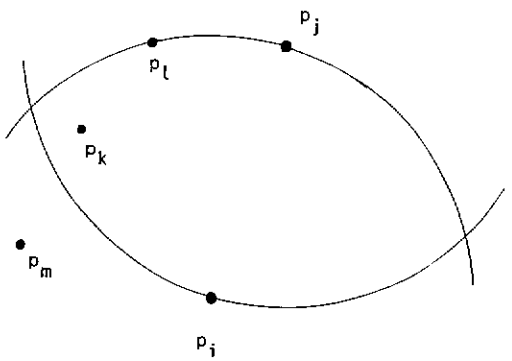

Fig. 1. The lune of p_i and p_j. Point p_k is inside and points p_l and p_m are outside of lune(p_i, p_j).

generated by a homogeneous planar Poisson point process. First, a supergraph of the RNG is determined. Here, each point is connected with an edge to its nearest neighbours in a number of sectors as seen from that point. The wanted RNG is found by deleting from the supergraph the edges the lune of which is not empty. However, the algorithm of Katajainen and Nevalainen is quite involved and hard to implement. This is why they used a simple variant of the optimal algorithm in their experiments. The variant worked very efficiently in practice and its observed running time turned out to be almost linear. Here, we shall give a modification of the practical algorithm of Katajainen and Nevalainen and prove its linearity on the homogeneous Poisson process assumption.

The main idea is that we can restrict in a new way the region of the search for the nearest neighbours of a given point. This is done by collecting information of the geometric shape of the region around the point where possible candidates for the nearest neighbours might be located. This results in a local search for the edges and when this is done by a suitable data structure, i.e., a cell organization of the points, the linearity can be proved for the Poisson distribution.

We start in Section 2 by recalling the algorithm of Katajainen and Nevalainen and by stating the new RNG-algorithm. In Section 3 we show that the algorithm runs in $\Theta(n)$ average time but needs $\Theta(n^2)$ worst-case time. A number of open questions are mentioned in Section 4.

2. A linear expected-time algorithm

The algorithm of Katajainen and Nevalainen [5] combines two widely used techniques for solving geometric problems: the region approach [2] and the cell technique. Let us briefly recall the algorithm. The points are located in a unit square. The space around each point p is divided into eight 'narrow' regions $R_j(p)$, $j = 1, 2, \ldots, 8$ (see Fig. 2), and the points closest to p, called the *region neighbours*, are searched in each region. If we connect each point with edges to the region neighbours, we get the so-called *all region neighbours* (ARN) graph, which is a supergraph of RNG [5]. To get the wanted RNG, the ARN graph is reduced from extra edges. Because the cardinality of the edges in the ARN graph is O(n), a straightforward implementation of the above method will give an $O(n^2)$ algorithm.

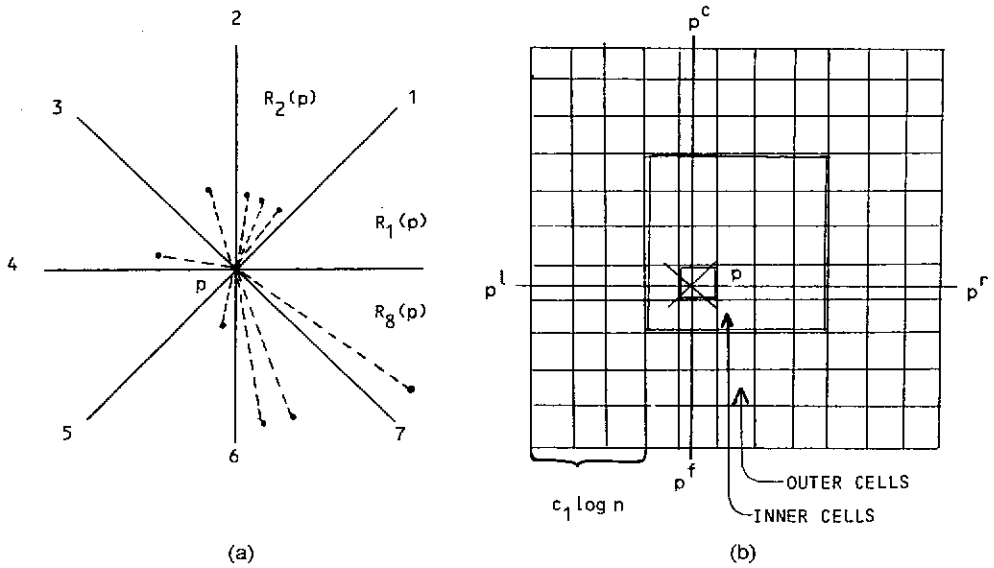

Fig. 2. The search for the region neighbours. Note that ray i belongs to region $R_i(p)$. (a) The regions and the region neighbours to p. (b) The spiral search.

To 'localize' the search for region neighbours and to delete the extra edges we can use the cell technique. A square grid of size $\sqrt{n} \times \sqrt{n}$ is placed on the set of points. The grid forms n cells, each of which is implemented as a linear list of the points which fall into the corresponding square of the grid. Now, the region neighbours of a point p_i are searched for by the *spiral search*. All cells in the neighbourhood of p_i are visited in a spiral manner until the region neighbours are found in each eight regions. During the search, all points, which *may hit* a lune determined by p_i and one of its region neighbours, are collected into a list and the lune test is performed only with the points in the list. However, in order to prove the O(n) average case running time, points in the border areas of the square grid had to be handled differently in [5]. Therefore, the cells are divided into *inner* and *outer cells* according to their distance from the boundaries of the unit square. The outer cells are the cells among the $c_1 \log n$ (c_1 is a constant) layers from the boundary (see Fig. 2(b)). The points of the outer cells are called the *outer points* and their relative neighbours are computed by an optimal worst-case RNG-algorithm, for example by the O(n log n) algorithm of [9].

In what follows we shall give a modification of the spiral search such that it will work efficiently also for the outer points. This will simplify the algorithm considerably. The intuitive idea behind the modification is the observation that, instead of finding all region neighbours for a point p, *it is necessary to find only some of its region neighbours, and by using these we can exclude those region neighbours which are far from* p. This is because the nearest neighbour points which are found to be close to p lie inside the lune of p and its far region neighbours. Thus, we have to do the searching for the region neighbours and the lune tests simultaneously.

During the search for the region neighbours, point p is said to be *closed* with respect to a region $R_j(p)$ if we have either found one or more points in $R_j(p)$ or hit the boundary of the square grid in such a way that we can be sure that all the following layers in the spiral search will be outside the square grid in $R_j(p)$. Otherwise, p is *open* with respect to $R_j(p)$.

When searching for the relative neighbours of a point $p = (x_p, y_p)$ we visit all the cells in the neighbourhood of p, layer by layer until *either no two neighbouring regions are open for it, or all the cells have been searched through*. Then, p is said to be closed (globally). To facilitate the description of the RNG-algorithm we temporarily include into V four artificial points

$$p^f = (x_p, 0), \quad p^e = (x_p, 1), \quad p^{\ell} = (0, y_p),$$

and $$p^r = (1, y_p).$$

These are the intersections of the sides of the unit square with the two coordinate axes when p is taken as the origin. The four points are the 'representatives' of the sides and, in case the side is met in a particular region, the representative is taken as the region neighbour.

Let us suppose that point p is closed and the region neighbours which made it closed are $q_1, q_2, \ldots, q_k$ ($k \geq 4$). Further, suppose that, of these points, q_j is farthest from p when the distances are measured in the L_∞-metric, i.e.,

$$d_\infty((x, y), (u, v)) = \max\{|x - u|, |y - v|\}.$$

Let

$$c_{\max,p} = \max_{j=1,2,\ldots,k} \{d_\infty(p, q_j)\}.$$

The parameter $c_{\max,p}$ defines a square with side length $2c_{\max,p}$, centre at p, and the sides of which are parallel to the coordinate axes. The region neighbours of p are in the closed regions inside this square. In the open regions there may exist some points with an L_∞-distance greater than $c_{\max,p}$ from p and still determining an edge of the RNG. However, we shall show in Lemma 2.2 that it is sufficient to visit cells only to the limit distance $4c_{\max,p}$ from p in order to find all candidates for the nearest neighbours. (Here, the distance is given by the L_2-metric.)

The *square of influence* SI_p of a point p denotes a square with side length $8c_{\max,p}$, p at the centre, and the sides of which are parallel to the coordinate axes. In the first step of the algorithm we search points in spiral order from the eight regions around p until there exists no pair of open adjacent regions. After this we continue the search for

the remaining cells of SI_p. In the process we collect all the points of SI_p in list L and the region neighbours in list G. The edges from p to the points of G belong to the All Region Neighbour (ARN) graph. In the second step of the algorithm, (the subgraph of) the ARN graph is reduced from extra edges to form the wanted RNG. This is done by checking for each edge whether there is at least one point in L which lies inside the lune corresponding to the edge. The method is described below in Procedure RNG_A. It is observed that RNG_A will not necessarily search for all region neighbours of a particular point p.

Procedure RNG_A (points V) returns (edges E)
(* This procedure determines the set E of edges of the RNG for a point set V *)
$E := \emptyset$
Create the cell structure
for each $p_i \in V$ **do**
Visit all the cells in the neighbourhood of p_i in spiral manner, collect the points and maintain SI_{p_i}. Stop searching when there does not exist a pair of two adjacent open regions, or all the regions in SI_{p_i} have been searched through.
Collect the region neighbours of point p_i inside SI_{p_i} into list G.
Collect all the points in SI_{p_i} into list L.
for each $p_j \in G$ such that $i < j$ **do**
lune_empty := true
for each $q \in L - \{p_i, p_j\}$ **do**
if $q \in$ lune(p_i, p_j) **then** lune_empty := false **break**
if lune_empty **then** $E := E \cup \{\langle p_i, p_j \rangle\}$
return(E)
end RNG_A

The validity of the algorithm follows from the next two lemmas. The first of these is a direct consequence of the cosine law for triangles.

2.1. Lemma ([9, p. 431]). *Let* p, q, *and* v *be points in the plane. Define* ***degree(angle*** pvq) *as* 180° *if* v *lies on the line segment* pq, *and as the degree of the smaller of the two angles formed by* pv *and* vq, *otherwise. Assume that* p, q, *and* v *are distinct, and assume that degree(angle* pvq) $\geq 90°$. *Then,* $v \in$ lune(p, q).

2.2. Lemma. *Let* p *and* q *be two different points of* V *and suppose that* $q \notin SI_p$, *where* SI_p *is the square of influence of point* p. *Then,* $\langle p, q \rangle \notin$ RNG.

Proof. Without loss of generality we can suppose that $q \in R_1(p)$.

(1) Suppose that $R_1(p)$ is closed and q_1 is the (proper) region neighbour which closes it. Because in $R_1(p)$ degree(angle pq_1q) $\geq 90°$, by Lemma 2.1, edge $\langle p, q \rangle$ is excluded from RNG.

(2) Suppose that $R_1(p)$ is open. At the termination of the spiral search we know that the regions $R_8(p)$ and $R_2(p)$ are closed. Now we have to consider three different cases by which the regions $R_2(p)$ and $R_8(p)$ have been closed.

Case (a). The regions $R_8(p)$ and $R_2(p)$ are both closed by proper points q_8 and q_2 of V (see Fig. 3). This means that

$$d(p, q_2) \leq \sqrt{2}\, c_{\max,p}$$

and

$$d(p, q_8) \leq \sqrt{2}\, c_{\max,p}.$$

By Lemma 2.1, no point q which is in the region $R_1(p)$ and outside the shaded area of the figure, can be connected with p in RNG because either q_2 or q_8 is in lune(p, q). Also, it is easily seen that

$a \leqslant 4c_{max,p}$ and thus the shaded area is totally inside the square of influence for p. Because q is outside SI_p, it is surely outside the shaded area of Fig. 3 and thus either q_2 or q_8 excludes the edge $\langle p, q\rangle$ from the RNG.

Case (b). A side of the unit square is met in one of the two regions $R_8(p)$ and $R_2(p)$. Then, region $R_1(p)$ is restricted in the direction of the empty region by a side of the square. An argument similar to that of Case (a) applies here, too.

Case (c). Both of the regions $R_8(p)$ and $R_2(p)$ are closed by the sides of the unit square. Then, the q-point does not exist. □

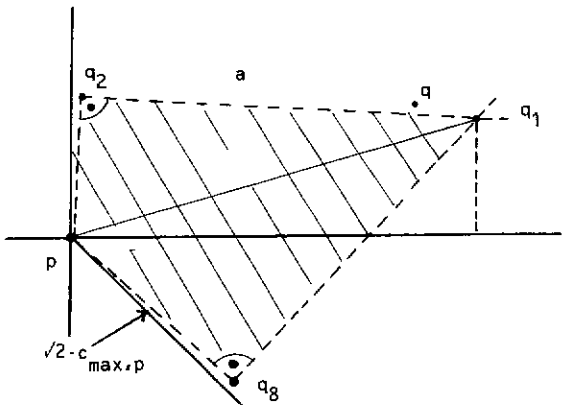

Fig. 3. Regions $R_8(p)$ and $R_2(p)$ are closed.

3. Analysis

The running time of Algorithm RNG_A is determined by the following parameters:

n_p: the number of region neighbours for a point p,

C_p: the number of cells which are contained in or intersected by the square of influence SI_p,

V_p: the total number of points in those C_p cells.

For any point p, the cost to find the region neighbours is proportional to $C_p + V_p$, and the cost to perform the lune tests is proportional to n_pV_p. The actual running time per point (or per cell of SI_p) is constant. Hence, we have a formula for W, the total work done by the algorithm:

$$W = k_0 n + \sum_{p \in V} (k_1 C_p + k_2 n_p V_p + k_3), \tag{1}$$

where the k_i, $i = 0, 1, 2, 3$, are constants. The time $k_0 n$ is needed for the construction of the cell structure during the initialization phase of the algorithm. A constant time floor function is presupposed here.

The All Region Neighbour (ARN) graph can [5] be expressed as a union of eight graphs called the *Geographic Neighbourhood Graphs* (GNG_i) of V in the *ith* region ($i = 1, 2, \ldots, 8$). Here, each edge of GNG_i connects a point with its nearest neighbour in the *ith* region. Because each GNG_i is a planar graph [5], and thus contains O(n) edges, the ARN contains $\Sigma_{p \in V} n_p = O(n)$ edges, too. In the *worst case*, the finding of each of the region neighbours and a lune test demands O(n) work, i.e., $C_p \leqslant n$ and $V_p \leqslant n$. Hence, the worst-case running time of RNG_A is $O(n^2)$. This upper bound is also reached because it may happen that almost all the points hit a single cell, which means that the square of influence always contains $\Omega(n)$ points.

To analyse the *average running time* of the algorithm we assume that the n points in the unit square have been generated by a homogeneous planar Poisson point process of intensity n (for details, see [5]). This means that the number of points N(S) in a bounded region S is distributed according to a Poisson distribution with expectation $n|S|$, where $|S|$ denotes the area of S. Additionally, for disjoint regions $S_1, S_2, \ldots, S_m$, the number of points $N(S_1), N(S_2), \ldots, N(S_m)$ are independent.

From (1) we get the expected running time

$$E(W) = k_0 n + \sum_{p \in V} (k_1 E(C_p) + k_2 n_p E(V_p) + k_3) = O(n) + O(1) \sum_{p \in V} (E(C_p) + n_p E(V_p)). \tag{2}$$

The random variables $C_{p_1}, C_{p_2}, \ldots, C_{p_n}$ ($V_{p_1}, V_{p_2}, \ldots, V_{p_n}$) are identically distributed and, as Lemma 3.2 below shows, $E(C_p) = E(V_p) = O(1)$. Hence, we have the following theorem.

3.1. Theorem. *Let* V *be a set of* n *points. Algorithm* RNG_A *finds the Euclidean relative neighbourhood graph of* V, *and the running time of* RNG_A *is*

(a) $\theta(n^2)$ *in the worst case, and*

(b) $\theta(n)$ *in the average case assuming that set* V *is located in a unit square and has been generated by a homogeneous Poisson process with an intensity of* n.

In [5] it has been shown that, for an inner point of the unit square, the square of influence will intersect O(1) cells and these cells will contain only O(1) points, on the average. The next lemma will generalize this result to any point in the unit square.

3.2. Lemma. *Let set* V *contain* n *points in a unit square generated by a homogeneous planar Poisson process. Let* p *be an arbitrary point of* V, *and let* W_p *denote the number of cells which are contained in or intersected by* SI_p ***plus*** *the total number of points in those cells, i.e.,* $W_p = C_p + V_p$. *Then,* $E(W_p) = O(1)$.

Proof. We first divide the unit square containing the points of V into eight disjoint regions $A_1, A_2, \ldots, A_8$ as shown in Fig. 4. We shall consider the points in region A_1 only. This is because the situation is symmetric in the other regions and their analysis can thus be omitted.

Let p be an arbitrary point in $V \cap A_1$. Then, the narrow regions around p cut the unit square into four triangles and four parallelograms (see Fig. 5). Of these regions, $R_2(p)$, $R_4(p)$, and $R_7(p)$ are ill-formed. That is, when point p is close to the right side or the upper corner of triangle A_1, the regions become thin slabs containing only a small number of points on the average. To simplify the analysis one can think that two changes are made to the spiral search. These actions slightly deteriorate the power of the search but the overall performance of the algorithm remains the same. In a practical application of RNG_A, the spiral search can be implemented in a natural way. The assumptions are as follows.

(1) Only the points in regions $R_1(p)$, $R_3(p)$, $R_5(p)$, $R_6(p)$, and $R_8(p)$ are considered during the search, because these regions are triangle-like or contain several complete layers of cells as counted outwards from point p. This assumption allows us to consider well-formed layers of cells only. (In practice, the inclusion of regions $R_2(p)$, $R_4(p)$, and $R_7(p)$ increases the speed of finding all necessary closest points to close the regions around p.)

(2) The spiral search is performed in regions $R_1(p)$, $R_3(p)$, $R_5(p)$, $R_6(p)$, and $R_8(p)$ only for the $\log_e n$ innermost layers. The choice will become clear later on. If the search does not stop before this limit, i.e., there is at least one pair of adjacent regions which are still open, then we give up the spiral search and simply scan all the cells in order to find the relative neighbours to point p.

Now let $E(W_p \mid c_{max,p})$ denote the conditional expectation of W_p, given $c_{max,p}$. By using the equation

$$E(W_p) = E\big(E(W_p \mid c_{max,p})\big),$$

we can give an upper bound for $E(W_p)$, the expected number of cells and points to be searched per a point p. Because the area of SI_p is $(8c_{max,p})^2$, it contains or intersects in total at most $1 + 4(8c_{max,p})^2/(1/n)$ cells. (At least the home cell of p intersects SI_p, and a square smaller than a cell intersects at most four cells.) Hence, we have

$$E(C_p \mid c_{max,p}) \leqslant 1 + 236nc_{max,p}^2.$$

The Poisson process of intensity n gives an average of An points in an area of size A. In case of the spiral search we know that some of the regions $R_1(p)$, $R_2(p)$, ..., $R_8(p)$ are empty in the vicinity of the base

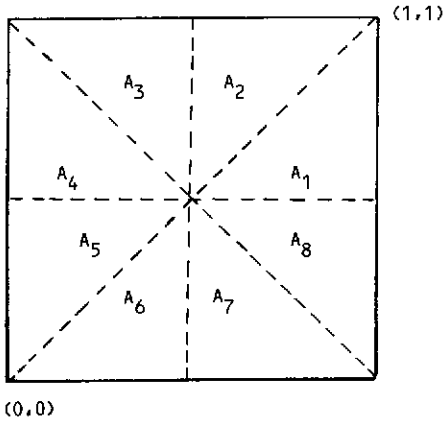

Fig. 4. Partition of the unit square into eight regions.

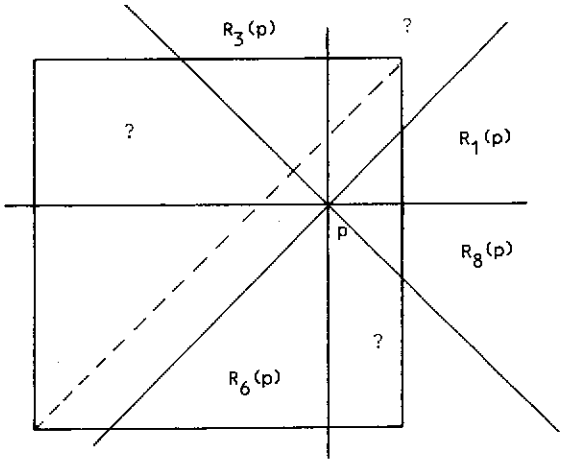

Fig. 5. The narrow regions around point p.

point p (see Fig. 6). More precisely, the cells in the innermost layers of the regions are known to contain no points. However, the number of points which are in the cells visited after the one containing the first point found is independent of the empty cells. Thus, given $c_{\max,p}$, we have an upper bound

$$E(V_p \mid c_{\max,p}) \leqslant n\left(1/n + 4\left(8c_{\max,p}\right)^2\right) = 1 + 236nc_{\max,p}^2$$

for the expected number of points in the cells contained in or intersected by SI_p.

From the above we get an upper bound $2(1 + 236nc_{\max,p}^2)$ for $E(W_p \mid c_{\max,p})$. However, we shall use this bound only for the $\log_e n$ innermost layers or, more precisely, when $c_{\max,p} < \log_e n/\sqrt{n}$. If $c_{\max,p} \geqslant \log_e n/\sqrt{n}$, we use the trivial upper bound 2n. Hence, we have

$$E(W_p \mid c_{\max,p}) \leqslant \begin{cases} 2\left(1 + 236nc_{\max,p}^2\right) & \text{for } 0 \leqslant c_{\max,p} < \log_e n/\sqrt{n}, \\ 2n & \text{for } c_{\max,p} \geqslant \log_e n/\sqrt{n}. \end{cases}$$

From this it follows that

$$E(W_p) = E\left(E(W_p \mid c_{\max,p})\right) \leqslant 2 + 472nE\left(c_{\max,p}^2 \mid [0, \log_e n/\sqrt{n})\right) + P\left(c_{\max,p} \geqslant \log_e n/\sqrt{n}\right)2n.$$

To complete the proof we shall first show that $P(c_{\max,p} \geqslant \log_e n/\sqrt{n})$, the probability that any two adjacent regions are open after visiting $\log_e n$ layers, is exponentially decreasing. Then we shall show that

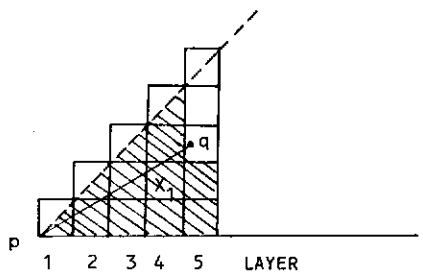

Fig. 6. A point and its region neighbour $q \in R_1(p)$.

$E(c^2_{max,p} | [0, \log_e n/\sqrt{n}))$, the expectation of $c^2_{max,p}$ restricted to the interval $[0, \log_e n/\sqrt{n})$, is proportional to $1/n$, and thus $E(W_p) = O(1)$.

The probability that there is at least one pair of two adjacent open regions after the scan of $\log_e n$ layers is certainly smaller than the probability that at least one open region exists after $\log_e n$ layers. The latter probability, on the other hand, is bounded by [5]

$$\sum_{i=1}^{8} e^{-nF_i(p)} = 8e^{-(1/2)\log_e^2 n} = 8n^{-\log_e \sqrt{n}},$$

where $F_i(p)$ stands for the area of the intersection of region $R_i(p)$ and the $\log_e n$ first layers for p. For a point which is open after $\log_e n$ layers, the work is at most 2n and thus the contribution of an open point to the expectation of W_p is

$$(2n)8n^{-\log_e \sqrt{n}} = 16n^{1-\log_e \sqrt{n}},$$

which approaches zero when n increases.

Next, we shall give an upper bound for $E\left(c^2_{max,p} | \left[0, \log_e n/\sqrt{n}\right)\right)$. For this purpose, let $X_{max,p}$ denote the distance which is to be moved from point p until the search detects at least one point (not necessarily a region neighbour), hits the side of the unit square, or $\log_e n$ layers have been searched through unsuccessfully. That is, if the distance between p and the first point met in $R_i(p)$ (or the boundary) is denoted by X_i, $i = 1, 3, 5, 6, 8$, then

$$X_{max,p} = \max\{X_1, X_3, X_5, X_6, X_8\}.$$

In RNG_A, *all* narrow regions $R_1(p), R_2(p), \ldots, R_8(p)$ are considered in the spiral search while determining $c_{max,p}$ and therefore

$$c_{max,p} \leqslant X_{max,p} \quad \text{and} \quad E\left(c^2_{max,p} | \left[0, \log_e n/\sqrt{n}\right)\right) \leqslant E\left(X^2_{max,p}\right).$$

Now we have (see, e.g., [7, Section 3.2])

$$E\left(X^2_{max,p}\right) = 2\int_0^{\log_e n/\sqrt{n}} xP\left(X_{max,p} > x\right) = 2\int_0^{\log_e n/\sqrt{n}} x\left(1 - P\left(X_{max,p} \leqslant x\right)\right) dx.$$

Because the random variables X_i are independent of each other, it holds that

$$P\left(X_{max,p} \leqslant x\right) = \prod_{i=1,3,5,6,8} P(X_i \leqslant x),$$

where $P(X_i \leqslant x)$ is the probability of the case that the first point found (or the boundary) in region $R_i(p)$ is closer than distance X to point p. Let $B_{i,p}$ denote the distance of point p from the boundary in region $R_i(p)$. Then we have

$$P(X_i \leqslant x) = \begin{cases} 1 - P(X_i > x) & \text{for } 0 \leqslant x < \min\left\{B_{i,p}, \log_e n/\sqrt{n}\right\}, \\ 1 & \text{for } \min\left\{B_{i,p}, \log_e n/\sqrt{n}\right\} \leqslant x < \log_e n/\sqrt{n}. \end{cases}$$

Here, $P(X_i > x)$ can also be regarded as the probability that a right-angled triangle with a side equal to x contains no points of V. For a Poisson distribution of the points, this equals $e^{-nx^2/2}$ and we have

$$P(X_i \leqslant x) = \begin{cases} 1 - e^{-nx^2/2} & \text{for } 0 \leqslant x < \min\left\{B_{i,p}, \log_e n/\sqrt{n}\right\}, \\ 1 & \text{for } \min\left\{B_{i,p}, \log_e n/\sqrt{n}\right\} \leqslant x < \log_e n/\sqrt{n}. \end{cases}$$

Because point p was selected from region A_1 of the unit square we have

$$\text{dist}(p, \text{right\_boundary}) \leqslant \text{dist}(p, \text{upper\_boundary}).$$

Depending on the location of p, the analysis can be divided into three different cases.

Case 1:

$$d(p, \text{upper\_boundary}) \geqslant d(p, \text{right\_boundary}) \geqslant \log_e n/\sqrt{n}\,.$$

Then we have

$$E\left(X^2_{\max,p}\right) = 2\int_0^{\log_e n/\sqrt{n}} x\left(1-\left(1-e^{-nx^2/2}\right)^5\right) dx$$
$$\leqslant 2\int_0^{\infty} x\left(1-\left(1-e^{-nx^2/2}\right)^5\right) dx = O(n^{-1}).$$

Case 2:

$$d(p, \text{right\_boundary}) < \log_e n/\sqrt{n}\,, \qquad d(p, \text{upper\_boundary}) \geqslant \log_e n/\sqrt{n}\,.$$

Denote $b = d(p, \text{right\_boundary})$. Then

$$E\left(X^2_{\max,p}\right) = 2\int_0^{b} x\left(1-\left(1-e^{-nx^2/2}\right)^5\right) dx + 2\int_b^{c\log n/\sqrt{n}} x\left(1-\left(1-e^{-nx^2/2}\right)^3\right) dx$$
$$= O(n^{-1}) + O(n^{-1}) = O(n^{-1}).$$

Case 3:

$$d(p, \text{right\_boundary}) \leqslant d(p, \text{upper\_boundary}) < \log_e n/\sqrt{n}\,.$$

Denote $b_0 = d(p, \text{right\_boundary})$ and $b_1 = d(p, \text{upper\_boundary})$. Then we have

$$E\left(X^2_{\max,p}\right) = 2\int_0^{b_0} x\left(1-\left(1-e^{-nx^2/2}\right)^5\right) dx + 2\int_{b_0}^{b_1} x\left(1-\left(1-e^{-nx^2/2}\right)^3\right) dx$$
$$+ 2\int_{b_1}^{c\log n/\sqrt{n}} x\left(1-\left(1-e^{-nx^2/2}\right)^2\right) dx$$
$$= O(n^{-1}) + O(n^{-1}) + O(n^{-1}).$$

This concludes the proof of the lemma: the average work per point is constant. □

4. Concluding remarks and open questions

The new algorithm RNG_A is simpler than the previous algorithms but it still gives the linear average running time. First, *only some* of the region neighbours are searched for and by means of these usually only a local neighbourhood, the square of influence for a point, is determined such that it contains all relative neighbours to that point. It is interesting to note that the same idea can be used to solve other geometric problems, too. For example, one can easily show that the square of influence contains all *Voronoi neighbours* of a point (for a definition, see, e.g., [1,8]). This observation gives us a way to simplify the Voronoi diagram algorithm of Bentley, Weide, and Yao [1].

A drawback of the algorithms based on the cell technique is the weak worst-case performance.

The worst-case running time of RNG_A is $O(n^2)$. It were therefore interesting to try to decrease these worst-case running times to O(n log n), say.

In this article, we gave only the theoretical analysis of the algorithm. Practical tests where RNG_A is compared with other efficient RNG-algorithms are still to be done. The actual performance of RNG_A heavily depends on the implementation details. One problem is how to determine the smallest possible square of influence. For instance, for an inner point of the unit square it is very probable that each of the eight regions contains a point close to the inner point in question. Then, the square of influence could be much smaller than that given by Lemma 2.2. Thus, it would be advisable to introduce a dynamic scheme capable of adapting itself to the actual distribution of the closest points. Another open problem is whether it is practical to collect the points in the square of influence into a separate list: The points can be directly recovered from the cell structure. According to Lemma 3.2, this will not deteriorate the average case performance. In case a separate list is created, it could be economical to have eight separate lists. In a particular lune test, at most five of the eight point lists are needed, namely the lists of the regions intersecting the lune.

The *storage requirement* of RNG_A also depends on the details of the implementation. A number of 2n storage locations (words) are needed for the input, and 6n for the output (RNG is a planar graph). The cell structure can be implemented in $n + O(1)$ extra space (see, e.g., [6]) if the order of input points may be changed. However, if the order must be preserved, $2n + O(1)$ storage locations are needed for the cell structure [4]. For the region neighbours, $n - 1$ storage locations must be reserved. This is because a point can have $n - 1$ region neighbours in the worst case. Thus, the working storage of RNG_A is $2n + O(1)$ locations if it is kept minimal, and $4n + O(1)$ locations if an improvement to the running time is required.

In this paper we have considered the relative neighbourhood graphs for the Euclidean metric only. However, the basic algorithm will work for *any* L_p*-metric* [3]. Moreover, the square of influence always includes an L_p-lune determined by an arbitrary point and one of its region neighbours.

A theoretically interesting open problem is to analyse the average running time of RNG_A for a *more general point process*. It seems that it is essential that the distribution does not possess very strong peaks. Further, it is evident that the algorithm would work efficiently for point sets in rectangles which are not 'too' thin.

References

[1] J.L. Bentley, B.W. Weide and A.C. Yao, Optimal expected-time algorithms for closest point problems, ACM Trans. Math. Software 6 (1980) 563–580.

[2] H.N. Gabow, J.L. Bentley and R.E. Tarjan, Scaling and related techniques for geometric problems, Proc. 16th Ann. ACM Symp. on Theory of Computing, Washington, D.C. (1984) 135–143.

[3] J. Katajainen, The region approach for computing relative neighbourhood graphs in the L_p metric, Rept. B38, Dept. of Computer Science, Univ. of Turku, Finland, 1986.

[4] J. Katajainen and O. Nevalainen, Three programs for computing the relative neighbourhood graphs in the plane, Rept. D29, Dept. of Computer Science, Univ. of Turku, Finland, 1985.

[5] J. Katajainen and O. Nevalainen, Computing relative neighbourhood graphs in the plane, Pattern Recognition 19 (1986) 221–228.

[6] A. Maus, Delanay triangulation and the convex hull of n points in expected linear time, BIT 24 (1984) 151–163.

[7] V.K. Rohatgi, An Introduction to Probability Theory and Mathematical Statistics (Wiley, New York, 1976).

[8] M.I. Shamos, Computational geometry, Ph.D. Thesis, Dept. of Computer Science, Yale Univ., New Haven, CT, 1978.

[9] K.J. Supowit, The relative neighbourhood graph with an application to minimum spanning trees, J. ACM 30 (3) (1983) 428–448.

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