ISSN No. 0976-5697

Volume 2, No. 3, May-June 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Locate a Pair Squares for Maximum Points Containment

Priya Ranjan Sinha Mahapatra
Department of Computer Science and Engineering
University of Kalyani
Kalyani, India
Priya_cskly@yahoo.co.in

Abstract: Given a set P of n points in R^2, locate two disjoint or overlapping axis-parallel squares, say S_1 and S_2, covering maximum number of points from P. However, in case of overlapping, their overlapped zone is empty. This work proposes $O(n^2 \log n)$ time and $O(n^2)$ space algorithm to locate S_1 and S_2.

Keywords: Facility Location, Axis-parallel square, Range tree, Staircase.

I. INTRODUCTION

The problem for finding two disjoint axis-parallel unit squares containing maximum number of points from $P = \{p_1, p_2, \ldots, p_n\}$, was first studied in [5] along with an $O(n^2)$ time algorithm. They later improved the complexity to $O(n \log n)$ [4].

It was mentioned in the same paper that the problem find applications in facility location, Pattern Recognition and Classification, etc. This work considers an extension of this problem [5] to compute two axis-parallel unit squares S_1 and S_2 such that the number of points covered by S_1 and S_2 is maximum; S_1 and S_2 may disjoint or intersecting. In case of intersecting, their common area is empty. Here a pair of square having empty overlapping zone may be called as disjoint as they donot contains any point of P. So a pair of intersecting axis parallel unit squares with common empty area may be taken as a candidate for finding better solution of the problem. In other words, the search space for finding such pair of disjoint or overlapping squares containing essential features is larger than that of two disjoint squares only. In this work, an $O(n^2 \log n)$ and $O(n^2)$ space algorithm is proposed to locate S_1 and S_2. The motivation of studying this problem comes from facility location [9, 1, 6], where essential features are represented as a point set, and the objective is to identify an optimum pair of disjoint precise convex regions containing maximum number of points.

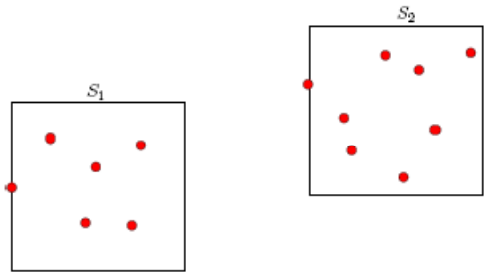

Figure.1 Optimum disjoint pair

Observation 1 Here the two squares are of same size. Moreover each square is a unit square. This condition can be lifted easily, leaving the results unchanged. In the rest of the paper, a square means an axis-parallel unit square. Let S_1 and S_2 be two intersecting axis-parallel unit squares with common empty area such that the number of points covered by them is maximum and S_g the global optimum axis-parallel unit square containing maximum number of points from P. Then S_1 and S_2 will both intersect S_g.

Proof: We will prove this observation by contradiction. Let us suppose that S_1 and S_2 will not intersect S_g. Then it is very clear that either the pair of squares S_g and S_1 or the pair of squares S_g and S_2 constitutes the desired solution of our problem. Hence a contraction arrives. W

II. SOME IMPORTANT OBSERVATIONS

This section describes some important observations. Let $P = \{p_1, p_2, \ldots, p_n\}$ be the n points in R^2. Without loss of generality assume that the coordinates of all points are distinct, since a small perturbation [7] can enforce

it. Pre-sort the points (in non-decreasing order) from P on their x- and y-coordinates and store them in lists L_x and L_y respectively. The coordinates of a generic point p are denoted by (p_x, p_y) and those for a specific point p_i are denoted by (p_{x_i}, p_{y_i}). Let R be the bounding rectangle containing the given n points from P. Clearly R is defined by the boundaries $p_{x_{min}}, p_{y_{max}}, p_{x_{max}}, p_{y_{min}}$, where $p_{x_{min}}, p_{x_{max}}$ are minimum and maximum x-coordinate and $p_{y_{min}}, p_{y_{max}}$, minimum and maximum y-coordinate respectively.

Suppose S_i is an axis-parallel unit square with bottom boundary at the level p_{y_i} and enclose maximum number of points from set P. Similarly, other three types of unit squares can be defined. Diaz Banez et. al. in [5] computes all such axis-parallel unit squares(i.e at most $4n$) and the number of points covered by each such squares in $O(n^2)$ time. Observe that given this set of axis-parallel unit squares, the problem of finding two disjoint axis-parallel unit squares jointly covering maximum number of points can be solved in linear time by sweeping horizontal and vertical lines over this set. The details can be found in [5]. Other important observations come during solving our problem are described below.

(i) From Figures 1 and 2, it is very clear that only these computed axis-parallel unit squares are not sufficient to find the solution of problem.
(ii) Unfortunately, the result in observation 1 is not used to find the solution of the problem.
(iii) If the desired pair of axis-parallel unit squares are intersecting having common empty area then the boundary of the common empty area may not aligned with point from P.
(iv) When the adjacent boundaries of the common empty area are aligned with point from P then the problem can be tackled in simple way.
(v) If anyone try to find the solution of the problem in brute force manner then they have to search S_1 and S_2 from $O(n^4)$ axis-parallel unit squares.

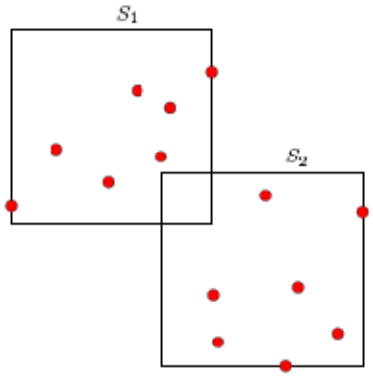

Figure.2 Optimum overlapping pair

Observation 2 Let $P_x = \{x_1, x_2, \ldots, x_n\}$ be sorted x-coordinates in non-decreasing order of n points in R^2. Define $l(x_i) = x_j$ where j is minimum index of x such that $(x_i - x_j)$ is less than or equal to one. Then we can compute $l(x_i), 1 \le i \le n$ in $O(n)$ time. Obviously, $l(x_1) = x_1$.

Algorithm for proving Observation 2:

Initialization: Initialize two indices i and j of x with 1.

(1) Repeat following steps until i becomes n.
(2) Let $d = x_i - x_j$.
(2.1) If the d is less than or equal to one then (i) $l(x_i) \leftarrow x_j$ (ii) increment i by one.
(2.2) Else increment j by one.

Clearly the runtime of this algorithm to compute $l(x_i), 1 \le i \le n$ in $O(n)$. Hence the Observation 2 is proved.

III. SOLUTION OF THE PROBLEM

From the Observations 1 and 2 in the section 2, the idea of grid is used to find the pair of axis-parallel unit squares S_1 and S_2 such that the number of points covered by them is maximum using $O(n^2 \log n)$ time and $O(n^2)$ space. We allow S_1 and S_2 to be disjoint or intersecting. However, in case they are intersecting, their common area is empty. Construct the grid for n points from P with the help of horizontal grid lines and vertical grid lines through these points. Clearly, the largest rectangle generated by horizontal grid line and vertical grid line is R. Observe that co-ordinates of the grid point, generated i-th horizontal grid line and j-th vertical grid line is (p_{x_j}, p_{y_i}) and these resulting n^2 grid points can be traversed by previously defined sorted lists L_x and L_y in appropriate manner. If, for some grid point (i, j), the corresponding x- and y-coordinates are of the same point, then the grid point is occupied by a point from P. For brevity, we sometimes specify a grid point by its coordinates also. Define a function l over sorted x-coordinates in non-decreasing order for n^2 grid points in the same way as in Observation 2. Similarly for each grid point, define $r(p_{x_j}) = p_{x_i}$ over sorted x- coordinates where i is maximum index of x such that $(p_{x_i} - p_{x_j})$ is less than or equal to one. Also define functions u and b over sorted y-coordinates in non-decreasing order for n^2 grid points. Therefore by Observation 2, each function l, r, u, r can be computed in $O(n^2)$ time . Our

algorithm works in the following steps.

Theorem 1 Let P be a set of n points in R^2. A range tree T [3] for P uses $O(n\log n)$ storage and can be constructed in $O(n\log n)$ time. By querying this range tree T one can report the points in P that lie in a rectangular query range [3] in $O(\log^2 n+k)$ time, where k is the number of reported points.

Theorem 2 The query time in the Theorem 1 can be improved to $O(\log n+k)$ by fractional cascading[3].

A. Step I to compute optimum axis-parallel unit squares within grid

Define $S_{i,j}$ as the square with the grid point (p_{x_j}, p_{y_i}) as top-left corner and $Count_{i,j}$ is the number of points within $S_{i,j}$. Let $S_{i,j}^{opt}$ be the optimum axis-parallel unit square containing maximum number of points within the sub-rectangle defined by the grid points (p_{x_j}, p_{y_i}) and $(p_{x_{max}}, p_{y_{min}})$, say $R_{i,j}^{sub}$, embedded in the grid. Let $COUNT_{i,j}^{opt}$ be the number of points within $S_{i,j}^{opt}$. Clearly $R_{i,j}^{sub}$ is generated by the grid point (p_{x_j}, p_{y_i}) as the top-left corner. Observe that, other types of $S_{i,j}^{opt}$ and $COUNT_{i,j}^{opt}$ can be defined and computed for the sub-rectangles within grid generated by other corner positions of $S_{i,j}$. However, since they are similar, we describe only the top-left corner case.

First of all, we compute $Count_{i,j}$ and $COUNT_{i,j}^{opt}$ within bottom-most unit strip having boundaries $p_{x_{min}}$, $p_{x_{max}}$, $p_{y_{min}}$ and $u(p_{y_{min}})$. To do so, construct a range tree R_T [3] from the points in P in the following way.

The main tree is T_1 is a AVL tree [3] built on x -coordinate of the points in L_x.

For any intermediate or leaf node v in T_1, the canonical subset $P(v)$ is stored in a AVL tree $T_{assoc}(v)$ on the y-coordinate of the points in L_y. The node v stores a pointer to the root of $T_{assoc}(v)$, which is called the associated structure of v.

We compute $Count_{i,j}$ by querying the range tree R_T that lie in the range $[p_{x_j}, p_{x_{max}}]X[p_{y_{min}}, p_{y_i}]$ using Theorem 1 where $l(p_{x_{max}}) = p_{x_j}$ and $u(p_{y_{min}}) = p_{y_i}$. Obviously $COUNT_{i,j}^{opt} = Count_{i,j}$. Then by another range query, we can compute $Count_{i,j-1}$. Clearly $COUNT_{i,j-1}^{opt} = max\{COUNT_{i,j}^{opt}, Count_{i,j-1}\}$ and the corresponding optimum square is $S_{i,j}^{opt}$. Hence by Theorem 1, we can compute at most n -1 $COUNT_{i,j}^{opt}$, $S_{i,j}^{opt}$ generated by the bottom-most unit strip in $O(n\log n)$ time. Similarly one can also compute at most n -1 $COUNT_{i,j}^{opt}$, $S_{i,j}^{opt}$ generated by the right-most unit vertical strip having boundaries $p_{y_{min}}$, $p_{y_{max}}$, $p_{x_{max}}$ and $l(p_{x_{max}})$ in $O(n\log n)$ time. In the next step, we compute $O(n)$ $Count_{i-1,j}$ corresponding to $S_{i-1,j}$, generated by the unit horizontal strip having boundaries $p_{x_{min}}, p_{x_{max}}, p_{y_{i-1}}$ and p_{y_k} where $b(p_{y_{i-1}}) = p_{y_k}$ in the following fashion. It is very clear that we have already computed $S_{i-1,j}$, $S_{i-1,j}^{opt}$ at the time of processing all $S_{i,j}$ in the unit vertical strip. So we first process $S_{i-1,j-1}$ to compute $count_{i-1,j-1}$.

(i) If the point on the $(j-1)$-th vertical grid line is within $S_{i-1,j-1}$ then increase the value of $Count_{i-1,j-1}$ by one.

(ii) Traverse all vertical grid lines from the right side of the right boundary of $S_{i-1,j-1}$ to the right boundary of $S_{i,j}$ and find the number of points ,say m, on these vertical grid lines. Clearly these m points was within $S_{i,j}$ but not within $S_{i-1,j-1}$. So decrease the value of $S_{i-1,j-1}$ by m.

(iii) Increase $Count_{i-1,j-1}$ by $Count_{i,j}$.

Clearly we can compute $O(n)$ $Count_{i-1,j}$ corresponding to $S_{i-1,j}$ in this horizontal unit strip in $O(n)$ time. Since there are at most $n-2$ unit horizontal strip excluding the bottom-most horizontal strip, so we can compute $O(n^2)$ $Count_{i,j}$ corresponding to $S_{i,j}$ generated by $O(n)$ unit horizontal strips in $O(n^2)$ time.

Now we are in a position to compute $COUNT_{i,j}^{opt}$ corresponding to $S_{i,j}^{opt}$ with the help of (i) already computed $COUNT_{i,j}^{opt}$ corresponding to $S_{i,j}^{opt}$ within the bottom-most unit horizontal strip and right-most unit vertical strip. (ii) all already computed $Count_{i,j}$ corresponding to $S_{i,j}$. Let $l(p_{x_{max}}) = p_{x_i}$ and $u(p_{y_{min}}) = p_{y_j}$. Then $COUNT_{i-1,j-1}^{opt}$ = max $\{COUNT_{i,j-1}^{opt}, COUNT_{i-1,j}^{opt}, Count_{i-1,j-1}\}$. It is very

clear that we can compute $COUNT^{opt}_{i-1,j-1}$ and $S^{opt}_{i,j}$ in constant time. Hence the time required to compute $S^{opt}_{i,j}$ and $COUNT^{opt}_{i,j}, 1 \le i, j \le n$ is $O(n^2)$.

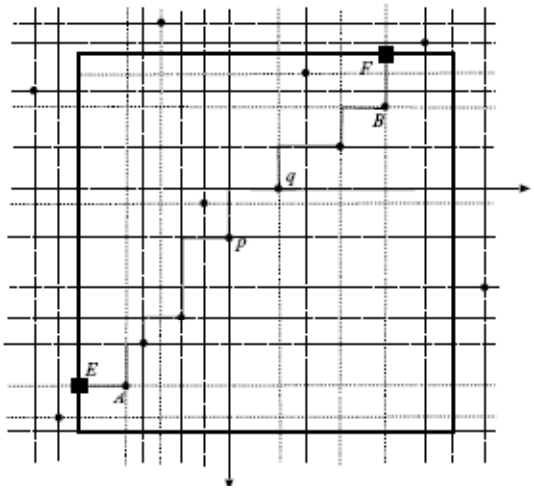

Figure.3 Stair case within unit square $S_{i,j}$

B. Step II to compute staircase AB and additional grid points E, F within $S_{i,j}$

Define $S_{i,j}$ as an axis-parallel unit square with bottom-right corner at the grid point (p_{x_j}, p_{y_i}) on the i -th horizontal grid line and $r(p_{x_{min}}) = p_{x_j}$. In this subsection, we first construct staircase AB between lowest point A and top-most point B within $S_{i,j}$ and then compute two additional grid points E, F with respect to bottom-right corner position of the square $S_{i,j}$. Observe that other types of staircase can be constructed with respect to other corner positions of the square $S_{i,j}$. However, since they are similar, we describe the bottom-right corner case.

a)Algorithm for constructing AB

Preface: By Theorem 2, all points within unit square $S_{i,j}$ can be found and these are in non-deceasing order of y -coordinate. We process all points in $S_{i,j}$ according to nondecreasing order of y -coordinate using two sweep lines named horizontal sweep line HSL lines and vertical sweep line VSL.

Denote two consecutive points of the staircase by p and q. Let $R^{sub}_{p_x,q_y}$ be sub-rectangle embedded in the grid, defined by the grid points (p_x, q_y) and $(p_{x_{max}}, p_{y_{min}})$. Observe that we have already computed $COUNT^{opt}_{p_x,q_y}$ and the corresponding square $S^{opt}_{p_x,q_y}$. Create a binary search tree (BST) T with q_y as key to represent the staircase AB between lowest point A and top-most point B within $S_{i,j}$. Attach the variables p, q and $COUNT^{opt}_{p_x,q_y}$, $S^{opt}_{p_x,q_y}$ with each node of T. Here p and q generate the grid point (p_x, q_y) as a member of the staircase AB. Create a max heap H with $COUNT^{opt}_{p_x,q_y}$ as key. Attach the variable $S^{opt}_{p_x,q_y}$ with each node of H.

Initialization: Initialize p with lowest point within $S_{i,j}$ and VSL at p.

(1)For each point q encountered by the HSL

(2) If the point q is on the right side of the current position of the VSL

(2.1) These HSL and VSL generate the grid point (p_x, q_y) as a member of the staircase AB. Find $COUNT^{opt}_{p_x,q_y}$ corresponding to $S^{opt}_{p_x,q_y}$. Insert p, q and $COUNT^{opt}_{p_x,q_y}, S^{opt}_{p_x,q_y}$ into T with q_y as key. Also insert $COUNT^{opt}_{p_x,q_y}$ as key along with $S^{opt}_{p_x,q_y}$ into H.

(2.2) $p \leftarrow q$

(2.3) Move the VSL to the point p.

(3) Else discard the point q.

b) Compute additional grid points E and F

If the lowest point within $S_{i,j}$ is not on the left-boundary of $S_{i,j}$ then the horizontal line through the lowest point within $S_{i,j}$ and the vertical line along the left-boundary of the square $S_{i,j}$ will generate the additional grid point E. Find the optimum square, say S^{opt}_E, from the sub-rectangle defined by the grid points E and (p_{max}, p_{min}). Let C^{opt}_E be the count of S^{opt}_E.

If top-most point within $S_{i,j}$ is not on the right boundary of $S_{i,j}$ then the vertical line through the top-most point and the horizontal line along the top-boundary of $S_{i,j}$ will generate the grid point F. Find S^{opt}_F and C^{opt}_F in the similar way. Observe that E and F may not exist.

Clearly the optimal square, say S_{AB}, at the root of heap H contains maximum number of points ,say C_{AB}, among all optimal squares in the heap H. Compute $C_{EF} = \max(C_{AB}, C^{opt}_E C^{opt}_F)$ and the corresponding

square be S_{EF}. Observe that the pair S_{EF} and $S_{i,j}$ may be disjoint or intersecting with empty common zone. So the optimal square S_{EF} together with the square $S_{i,j}$ is a candidate pair jointly covering number of point, say SUM, from P.

C. Step II to compute staircase $A'B'$ and additional grid point E' within $S(i, j+1)$

Let $A'B'$ be the staircase between lowest point A' and top-most point B' within $S_{i,j+1}$ and E' be the additional grid point. Observe that the additional grid point of type F within square $S_{i,j}$ does not exist and the additional grid point E' may not exist.

If the point on the $(j+1)-$th vertical grid line is within $S_{i,j+1}$ then compute staircase $A'B'$ within square $S_{i,j+1}$ from the staircase AB and additional grid point E' in the following way.

(1) If the left boundary of $S_{i,j+1}$ coincides with the left boundary of $S_{i,j}$

(1.1) Don't update the the lower end of the staircase AB and E becomes E'.

(1.2) Update the upper end of the staircase AB by the algorithm 3.3.1

(2) Else if left boundary of $S_{i,j+1}$ is on the left of the lowest point within $S_{i,j}$

(2.1) Don't update the the lower end of the staircase AB and compute the additional grid point E' in the same way as described in the subsection 3.2.2.

(2.2) Update the upper end of the staircase AB by the algorithm 3.3.1

(3) Else compute the staircase $A'B'$ from AB by the algorithms 3.3.1 and 3.3.2 and the additional grid point E' in the similar way.

Algorithm to update the upper end of the staircase AB to compute the staircase $A'B'$

(1) Find the largest key from T and denote the corresponding node by α.

(2) If y-coordinate of the point under processing on the right boundary of $S_{i,j+1}$ is greater than q_y associated with α.

(2.1) The point p becomes the point q associated with α and q becomes the point under processing. The horizontal line through the point q and the vertical line through the point p generate the grid point (p_x, q_y) as a member of $A'B'$.

(2.2) Find $COUNT^{opt}_{p_x,q_y}$ and corresponding square $S^{opt}_{p_x,q_y}$. Insert q_y as key along with p, q and $COUNT^{opt}_{p_x,q_y}, S^{opt}_{p_x,q_y}$ into T. Also insert $COUNT^{opt}_{p_x,q_y}$ as key along with $S^{opt}_{p_x,q_y}$ into H.

(3) Else-if the point under processing is above the point p associated with α

(3.1) Find the node, say β, from heap H containing $COUNT^{opt}_{p_x,q_y}$ associated with α.

(3.2) Clearly, the point q associated with α will be replaced by the point under processing. The vertical line through the point p associated with α and the horizontal line through the point under processing generate the grid point (p_x, q_y) as a member of $A'B'$. Find $COUNT^{opt}_{p_x,q_y}$ and corresponding $S^{opt}_{p_x,q_y}$ Save $COUNT^{opt}_{p_x,q_y}$ and corresponding $S^{opt}_{p_x,q_y}$ at the node α.

(3.3) Replace the values associated with node β in H with the modified values $COUNT^{opt}_{p_x,q_y}$ and corresponding $S^{opt}_{p_x,q_y}$ in α.

(4) Else delete the node α from T and β from H.

Repeat **Step 1** until the **Step 2** or **Step 3** is executed.

Algorithm to update the lower end of the staircase AB to compute the staircase $A'B'$

(1) Find the lowest key from the BST T and denote the corresponding node by γ.

(2) If the left-boundary of $S(i, j+1)$ is between points p and q associated with γ

(2.1) Find the node, say δ, from H containing $COUNT^{opt}_{p_x,p_y}$ associated with γ.

(2.2) Delete the node γ from BST T and δ from heap H.

Repeat **Step 1** until the **Step 2** is executed.

Clearly the square, say $S_{A'B'}$, at the root of heap H contains maximum number of points, say $C_{A'B'}$, among all squares in the heap H. Compute $C_{E'B'} = \max(C_{A'B'}, C^{opt}_{E'})$ and the corresponding square be $S_{E'B'}$. Observe that the pair $S_{E'B'}$ and $S_{i,j+1}$ may be disjoint or intersecting with empty common zone. So the

square $S_{E'B'}$ together with the square $S_{i,j+1}$ is a candidate pair jointly covering number of points, say SUM', from P.

If SUM is greater than SUM' then squares S_{EF} and $S_{i,j}$ is current optimal candidate pair. Otherwise squares $S_{E'B'}$ $S_{i,j+1}$ is current optimal candidate pair.

This is done for all unit squares $S_{i,j}$ with right-bottom corner at the grid point on i-th horizontal grid line. So after processing all such unit squares on i-th horizontal grid line, we get an optimal candidate pair.

Repeat Step II to find $O(n)$ optimal candidate pairs generated in similar way.

At the end of processing $O(n^2)$ axis-parallel unit squares generated from $O(n)$ horizontal grid lines, S_1 and S_2 can be found.

D. Complexity analysis of Phase II

It should be observed that points constructing the staircase AB within $S_{i,j}$ may not be present in the staircase $A'B'$ within $S_{i,j+1}$. So a point within $S_{i,j}$ may be deleted from the BST T or inserted into BST T for atmost one time to process $S_{i,j+1}$. The number nodes in heap H is $O(n)$ to process all $S_{i,j}$ on the $i-$ th horizontal grid line and insertion and deletion operation takes of $O(\log n)$ time. Hence the time required to process all axis-parallel unit squares on the i-th horizontal grid line is $O(n \log n)$. Consequently, the time required to process $O(n^2)$ axis-parallel unit squares on $O(n)$ horizontal grid lines is $O(n^2 \log n)$. We thus have the result.

Theorem 3 Given a set P of n points in R^2. Two disjoint or intersecting (with empty overlapping zone) axis-parallel unit squares covering maximum number of points can be found using $O(n^2 \log n)$ time and $O(n^2)$ space.

Observation 3 This algorithm can be easily extended to find and report the points, covered by a pair of rectangles say R_1 and R_2 with pairwise parallel side with given directions and with disjoint interiors or intersecting (with empty overlapping zone) that maximizes the sum of the points covered by R_1 plus the number of points covered by R_2 in $O(n^2 \log n)$ time and $O(n^2)$ space.

IV.CONCLUSION

We have taken the help of range tree to compute all axis-parallel unit squares within the bottom-most horizontal and right-most vertical strips as our proposed complexity is $O(n^2 \log n)$. It can be observed that the way of computing all axis-parallel unit squares can be avoided by traversing the grid points in L_x or L_y. The proposed time complexity can be reduced to $O(n^2)$ if we can manage the updating of the staircase in better way.

V. REFERENCES

[1] T. Asano, B. Bhattacharya, M. Keil, F. Yao *Clustering algorithms based on maximum and minimum spanning trees.* Proc. 4th Annual Symposium on Computational Geometry, 252--257, 1988.

[2] A. Aggarwal and S. Suri, *Fast Algorithm for Computing the Largest Empty Rectangle*, Proceedings of the third annual symposium on Computational Geometry, 278--290, 1987.

[3] M. de Berg, M. Van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry, Algorithms and Applications*, Springer, 1997.

[4] Sergio Cabello, J. Miguel Diaz-Banez, Carlos Seara, J. Antoni Sellares, Jorge Urrutia, Inmaculada Ventura, *Covering point sets with two disjoint disks or squares.* Computational Geometry Theory and Applicaions, 40, 195--206, 2008.

[5] J. Miguel Diaz-Banez, Carlos Seara, J. Antoni Sellares, Jorge Urrutia, Imma Ventura, *Covering Points Sets with Two Convex Objects*, EWCG, 2005.

[6] Z. Drezner and H. Hamacher, *Facility Location: Applications and Theory.* Springer Verlag, Berlin, 2002.

[7] Edelsbrunner, H. and Mucke, E.P. *Simulation and simplicity: A technique to cope with degenerate cases in geometric algorithms*, ACM Trans. Graphics, vol. 9, 66--104, 1999.

[8] B. Chazelle, R.L. Drysdale, D.T. Lees *Computing The Largest Empty Rectangle*, SIAM J. COMPUTING, vol. 15, 1986.

[9] J. A. Hartigan, *Clustering Algorithms*, Wiley, New York, 1975.

[10] D. T. Lee, Y. F. Wu, *geometric complexity of some location problems*, Journal of Algorithmica, vol. 1, 193--211, 1986.