ISSN No. 0976-5697

Volume 2, No. 2, Mar-Apr 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An O(n^2) Algorithm to Compute a Square that contains at least k points

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 two dimensional plane. In this paper we study the minimum enclosing square problem. An O(n^2) time and space algorithm is proposed to locate a minimum enclosing axis-parallel square (S_k) that encloses at least k ($1 \le k \le n$) points of P.

Keywords: Enclosing problem, Computational Geometry, Facility Location, Covering Location Problem, Algorithm, Optimization Technique.

I. INTRODUCTION

The logistics for distribution of product or services has been a subject of increasing importance over the years, as part of the strategic planning of both public and private enterprise. Decisions concerning the best configuration for the installation of facilities within convex objects in order to enclose demand requests are the subject of a wide class of problems, known as enclosure problem. In facility location [14, 18, 19], enclosure problem computes a subset P' of a finite set of demand points $P = \{p_1, p_2, \ldots, p_m\}$ (i.e. $P' \subseteq P$) to be enclosed by convex objects $C = \{c_1, c_2, \ldots, c_m\}$ such that facility f_i is placed within convex object c_i , $1 \le i \le m$ and any demand p_i , $1 \le i \le n$ may get service from at least one facility f_j , $1 \le j \le m$ maintaining some optimality criteria.

Enclosure problem is an important area of algorithmic research in computer science community which arise in facility location, material cutting, geometrical shapes inspection, material salvage, and antenna coverage and are well studied in the operation research literature [3] and computational geometry [2, 17]. In some cases, the enclosing object is orientation-invariant, that is, the region bounded by the object remains same under rotation (for example circle). However, if the enclosing object is em orientation-dependent, then it sometimes becomes more difficult to compute the optimal orientation over all possible orientations. In facility location, a classical problem in the domain of enclosure problem is location set covering problem (LSCP). The LSCP [10] determines the minimal number of facilities that are necessary to attend all clients, for a given covering distance. Owing to formulation restrictions, maximum covering location problem (MCLP) model [14] does not consider the individual demand of each client. In addition, the number of needed facilities can be large, incurring high installation and recurring costs. An alternative formulation considers the installation of a limited number of facilities, even if this amount is unable to address the total demand. In this formulation, the condition that all clients must be served is relaxed and the objective is changed to locate p facilities such that all the maximum part of the existing demand can be addressed, for a given covering distance. In *constrained* MCLPs, the location of each facility is restricted to some feasible region. In the *unconstrained* model, there is no restriction on the location of facilities. The real world applications always imposes some spatial constraints on the corresponding problem, whereas their unconstrained counterparts are rather of theoretical interest. It is noteworthy that sometimes the solutions of both constrained and unconstrained versions may coincide, i.e, solution of unconstrained version falls in one of the feasible regions of the constrained problem. As solution of a unconstrained problem P may give some crucial insight to the constrained version of P, the study of former is of great importance. Maximum covering location models can differ in their objective function, the distance metric applied, the number and size of the facilities to locate and several other indices. Depending upon the specific application, inclusion and consideration of these indices in the problem formulation will lead to very different maximum covering location models.

In this paper we study minimum enclosure problem that computes a minimum area axis-parallel square enclosing at least k points of P. A square (rectangle) is said to be a k *- square* (k -rectangle) if it encloses exactly k points of P and there does not exist another square (rectangle) S'_k having area less than that of S_k. The problem of computing k *- square* and k *- rectangle* are studied by different computer scientists during last decades [4, 1, 6, 5, 9, 7]. Interested readers can have a look on different existing algorithmic complexity results in the work of Das et al. [4] for details.

Another motivation of studying different types of enclosure problems comes from pattern recognition [12, 13], where essential features are represented by a point set, and the objective is to identify clusters [15] containing at least k number of features. In this paper cluster region is the region enclosed by an axis-parallel square or rectangle.

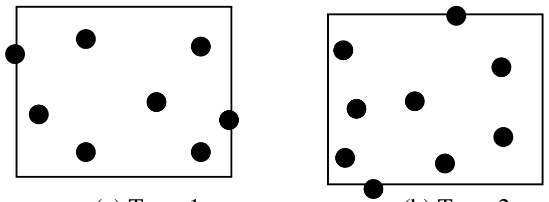
Deterministic (also known as polynomial-time) as well as probabilistic algorithms can be developed using differents optimization techniques [20, 21, 22]. Deterministic lgorithms are most often used if a clear relation between the characteristics of the candidate solutions and their utility for a

given problem exists. Then, the search space can efficiently be explored using different techniques such as divide and conquer scheme. If the relation between a potential solution and its Fitness [23] are not so simple or too complicated, or the dimensionality of the search space is very high, it becomes difficult to design a deterministic algorithm to solve a problem. For this type of problem, it not possible to frame an exhaustive enumeration of the search space even for relatively small problems. Then, probabilistic algorithms come into action. The initial work in this domain which now has become one of most important research area in optimization was started long time ago. An especially relevant family of probabilistic algorithms are the Monte Carlo6-based approaches, Evolutionary Computation (EC) [24], Memetic Algorithms, Random Optimization [25], Tabu Search (TS), Genetic Algorithms (GA) [26] etc.
We use sophisticated tools from computer science, computational geometry and combinatorial optimization. The approach to solve this problem does not require more than the standard geometric insight that has been extensively studied in computational geometry literature. In Section 2, first some basic results are developed to propose a deterministic algorithm to find S_k. Then an $O(n^{2})$ time and space algorithm is presented to find S_k. We make concluding remark in section 3.

II. BASICS

Let $P = \{p_1, p_2, \dots p_n\}$ be the set of given points in the plane. Our objective is to locate k-square S_k. Without loss of generality, we describe our method under general position assumptions for the points in P, i.e., (i) no two points lie on a vertical line, and (ii) no three points are collinear. We denote by $x(p)$ and $y(p)$ the x-coordinate and the y-coordinate of any point p respectively. The size of a square is denoted by the length of it's side. Let L_1 and L_2 be the pairs of opposites sides of S_k. We have the following observation.

Observation 1 *Each side in L_1 or each side in L_2 of the k-square S_k must contain at least one point of P.*


(a) Type-1 (b) Type-2

Consider a set of axis-parallel squares whose each vertical side contains at least one point of P. We can generate at most such squares by considering all pair of points p_i and $p_j \in P$ lying on the vertical sides of the square. Similarly a set of axis-parallel squares can be generated whose each horizontal sides contain a point of P. From observation 1, S_k must belong to these two sets and can be obtained by locating a minimum sized square in this set containing at least k points. The problem of locating S_k is equivalent to computing an axis-parallel square of minimum area among the set of squares whose vertical sides or horizontal sides pass through p_i and $p_j \in P$, $i \le n$ and $j \le n$ and it contains at least k points from P.

The number of such squares will be $\frac{n(n-1)}{2} = O(n^2)$. Using brute force algorithm we can determine the squares from the set that contain at least k points from P; among them the square with the minimum area is the desired solution. This naive algorithm requires $O(n^3)$ time and $O(n)$ space. To expedite our search process we can use divide and conquer method to solve our problem. At each branching point we solve a decision problem. Depending upon the solution to that decision problem we choose our search path. For each optimization problem, there is a corresponding decision problem that asks whether there is a feasible solution for some particular measure. For example, consider graph coloring problem for a graph G that computes the minimum number of colors required to make the graph G colorable. The corresponding decision problem of graph coloring problem would be “Is the graph G is k colorable?”. This problem can be answered with a simple “yes” or “no”. For computing S_k the decision problem can be described as follows: “*given a length l, does a square of size l enclose at least k points of P?*” This decision problem motivates us to use the following result.

Result 1 *[8, 16] Given a set P of n points in the plane we can compute an axis-parallel square of fixed size containing maximum number of points from P in $O(n \log n)$ time using $O(n)$ space.*

Let us define $Max\text{-}Sq(\alpha)$ as an axis-parallel square of size α that covers maximum number of points from P. Assume that the sequence $< p_1, p_2, \dots, p_n >$ is in non decreasing order of the x-coordinate values. In first pass, we compute the horizontal distances $(x(p_j) - x(p_i)), j > i$ for each pair of points p_i and $p_j \in P$ and store these $\frac{n(n-1)}{2}$ distances in an array Φ. Thus the array Φ contains the following sequences of vertical distances and the total number of horizontal distances is $\frac{n(n-1)}{2}$.

$$\Phi_1 = x(p_2) - x(p_1), x(p_3) - x(p_1), \dots, x(p_n) - x(p_1)$$
$$\Phi_2 = x(p_3) - x(p_2), x(p_4) - x(p_2), \dots, x(p_n) - x(p_2)$$
$$\vdots$$
$$\Phi_i = x(p_{i+1}) - x(p_i), x(p_{i+2}) - x(p_i) \dots, x(p_n) - x(p_i)$$
$$\vdots$$
$$\Phi_{n-2} = x(p_{n-1}) - x(p_{n-2}), x(p_n) - x(p_{n-2})$$
$$\Phi_{n-1} = x(p_n) - x(p_{n-1})$$

We use binary search in the array Φ to locate minimum length square enclosing k points. Observe that the vertical distances array Φ is in non decreasing order. This is very required as binary search is used as a prune and search tecnnique to locate S_k. Compute the median λ of the elements in and locate the $Max\text{-}Sq(\lambda)$ using Result [8]. Distance λ partitions the array in two halves. Right half contains the elements greater than or equal to λ. In case the number of points enclosed by $Max\text{-}Sq(\lambda)$ is less than k, the elements in the left half of Φ lesser than λ are not the candidates for generating S_k. So for each step of the binary search, half of the elements of the array Φ are discarded. The running time T of this algorithm is governed by the time required to compute the median λ and the time required to locate the $Max\text{-}Sq(\lambda)$. Observe that $O(n^2)$ time is required to compute the median from a set of n^2 elements and locating $Max\text{-}Sq(\lambda)$ needs $O(n \log n)$ time (See Result 2). Theforefore, the time complexity of the proposed algorithm is governed by the following recurrence relation.

$$T(n^2) = T(n^2/2) + O(n^2) + O(n \log n)$$
$$= O(n^2)$$

Therefore computation of S_k with at least one point on each of its left and right side requires $O(n^2)$ time and space.

In second pass, similar technique (i.e. binary search) is used to compute S_k with at least one point on each of its top and bottom side. For this, let $< p'_1, p'_2, \ldots, p'_n >$ be the elements of P in non decreasing order of their y-coordinate values. Here binary search algorithm is also used on the vertical distances $(y(p'_j) - y(p'_i)),\ j > i$ for each pair of points p'_i and $p'_j \in P$. In this case the vertical distances are expressed by the following sequences.

$$\Psi_1 = y(p'_2) - y(p'_1), y(p'_3) - y(p'_1), \ldots, y(p'_n) - y(p'_1)$$
$$\Psi_2 = y(p'_3) - y(p'_2), y(p'_4) - y(p'_2), \ldots, y(p'_n) - y(p'_2)$$
$$\vdots$$
$$\Psi_i = y(p'_{i+1}) - y(p'_i), y(p'_{i+2}) - y(p'_i) \ldots, y(p'_n) - y(p'_i)$$
$$\vdots$$
$$\Psi_{n-2} = y(p'_{n-1}) - y(p'_{n-2}), y(p'_n) - y(p'_{n-2})$$
$$\Psi_{n-1} = y(p'_n) - y(p'_{n-1})$$

Like first pass, observe that an array that contains all these vertical distances is also in non decreasing order.

In each pass, we keep the smallest area square enclosing at least k points of P found so far. Finally, the minimum area square containing at least k points of P is reported. We thus have the following result.

Theorem 1 Given a set P of n points in the plane and an integer $k \le n$, the smallest area axis-parallel square containing at least k points of P can be located in $O(n^2)$ time and space.

III. CONCLUSIONS

Given a set P of n points in two dimensional plane and an integer $k\ (\le n)$, we have considered the enclosure problem of finding a minimum area axis-parallel square that contains at least k points of P. A k point enclosing square (rectangle) S_k is said to be a k-*square* (k-*rectangle*) if there does not exist another square (rectangle) having area less than that of S_k and containing k points from P. In Section 3 an $O(n^2)$ time and space algorithm is presented to find S_k.

IV. REFERENCES

[1] A. Aggarwal, H. Imai, N. Katoh and S. Suri, Finding k points with minimum diameter and related problems. Journal of Algorithms, Vol. 12, pp. 38--56,1991.

[2] M. de Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf, Computational Geometry: Algorithms and Applications. Springer, Berlin, 1997.

[3] S. Martello and P. Toth, Knapsack Problems: lgorithms and Computer Implementations. John Wiley & Sons Ltd. Chichester, Baffins lane, West Sussex, England, 1990.

[4] S. Das, P. P. Goswami and S.C. Nandy, Smallest k-point enclosing rectangle and square of arbitrary orientation. Information Processing Letters, Vol. 94, pp. 259--266, 2005.

[5] A. Datta, H.E. Lenhof, C. Schwarz and M. Smid, Static and dynamic algorithms for k-point clustering problems. Lecture Notes in Computer Science, Vol. 709, Springer, Berlin, pp. 265--276, 1993.

[6] D. Eppstein and J. Erickson, Iterated nearest neighbors and finding minimal polytopes. Discrete Computational Geometry, Vol. 11, pp. 321--350, 1994.

[7] J. Matoušek, On geometric optimization with few violated constraints. Discrete Computational Geometry, Vol. 14, 365--384, 1995.

[8] Priya Ranjan Sinha Mahapatra, Partha P. Goswami and Sandip Das, Covering Points by Isothetic Unit Squares. Proc. 19th Canadian Conference on Computational Geometry, 169--172, 2007.

[9] M. Segal and K. Kedem, Enclosing k points in the smallest axis parallel rectangle. Information Processing Letters, Vol. 65, pp. 95--99, 1998.

[10] Toregas C, Swain R, Revelle C, Bergman L, The location of emergency service facilities. Operation Research, Vol. 19,1971.

[11] Greg N. Frederickson and Donald B. Johnson, Generalized Selection and Ranking: Sorted Matrices. SIAM Journal on Computing, Vol. 13, pp. 14--30, 1984.

[12] H.C. Andrews, Indroduction to mathematical techniques in pattern recognition, Wiley-Intersciences, New York, 1972.

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

[14] Zvi Drezner and Horst W. Hamacher, Facility location: applications and theory, Springer, Berlin, Heidelberg, New York, 2001.

[15] H. Spath, Cluster Analysis Algorithms, Ellis Horwood, Chichester, UK, 1989.

[16] M. J. Katz, K. Kedem and M. Segal, Improved algorithm for placing undesirable facilities. Computer & Operation Research, Vol. 29, 1859--1872, 2002.

[17] Herbert Edelsbrunner, Algorithms in Computation Geometry, Springer-Verlog, Berlin, Heidelberg, New york, London, Paris, Tokyo, 2001.

[18] R. L. Francis and L. Mirchandani, Discrete Location Theory, Wiley, New York, 1990.

[19] R. L. Francis and F. Leon and J. McGinnis and J. A.White, Facility Layout and Location: An Analytical Approach, Prentice-Hall, New York, 1992.

[21] Kenneth A. De Jong, Evolutionary Computation: A Unified Approach, MIT Press, 2006.

[22] Jon Kleinberg and Eva Tardos, Algorithm Design, First, Pearson Education, 2006

[23] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.

[24] Kenneth A. De Jong, Evolutionary Computation: A Unified Approach, MIT Press, 2000.

[25] Ketan Mulmuley, Computational Geometry: An Introduction Through Randomized Algorithms, Prentice-Hall, 1994.

[26] L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991