ACSIJ Advances in Computer Science: an International Journal, Vol. 3, Issue 4, No.10 , July 2014
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

Finding Two Maximum Separating Circles

Afsaneh Hellatabadi Farahani [1] and Alireza Bagheri [2,1*]

[1] Faculty of Engineering, Tehran North Branch, Islamic Azad University,Tehran, Iran
afsan.farahani@gmail.com

[2] Department of Computer Engineering and Information Technology, Amirkabir University, Tehran, Iran
ar_bagheri@aut.ac.ir

Abstract

In this paper, for the first time an algorithm is presented for separating the points by two circles. Separating the points by circles is one of the problems in computational geometry being applied in different fields of science, some of which are: facility location, image processing and clustering. Our algorithm finds the locations of two smallest circles in the plane such that these two circles cover as many as points of *Q* while avoid the points of *P*. The running time of the algorithm is $O(n^2)$, where *n* is total number of input points. The experimental results show that our algorithm succeeds to find near-optimal solutions, with an approximation factor of 1.32 in average. The proposed algorithm is not optimal, but in some cases it yields optimum solutions.

Keywords: *computational geometry, point-set covering, point-set separation.*

1. Introduction

Computational geometry is one of the important branches of computer science and it has applications in robotic knowledge, geographical information systems, integrated circuit design and discussions related to clustering [1]; this field is also applied in different branches of computer science such as statistics, machine vision and graphics.

The issue of coverage is one of the important issues in computational geometry. This issue has application in the field of facility location. In this problem the aim is covering a set of geometrical objects such as point by one or more certain geometrical objects such as circle, rectangle, square, triangle, etc.

In one version of coverage problem it is assumed that points are not of the same color but of two colors. In this case, we want to separate points by a certain geometrical shape or just cover the point of one color. This version of coverage problem is known as separation problem and the separating geometrical shape is called separator.

Separation problem has also applications in the fields of image processing, pattern recognition, statistical computations and clustering. Sometimes, perfect separation is not possible and the aim is separating maximum points called maximal separating or optimal separating.

One of the important shapes used for separating a set of points is circle and it has many applications in solving the problems relating to clustering, location of facilities and image processing. Up to now, some methods have been presented for separating with one circle, but separation by two circles has not been considered yet.

The problem of finding smallest circle that covers points was firstly presented in 1875 by Sylvester at time $O(n^4)$ by testing all cases. Later some algorithms were developed for solving this problem, all of which aimed to reduce time order of the algorithm [2,3,4]. In 1975 an algorithm was presented of time order $O(nlogn)$ for solving this problem [5]. Finally, in 1983, this algorithm was solved using linear programing by Megiddo in the time $O(n)$[6]. Also, in 1994, an algorithm of time order $O(n^2log^3n)$ was presented for covering the points using two circles [7].

Another problem relating to covering with circle is covering with disjoint unit circles. In this problem a set called P including *n* points in the plane and a set called D including unit circles are given with fixed positions. The aim of the problem is finding $D' \subseteq D$ of the minimum cardinality such that covers all points of P. This problem is geometrical version of set cover problem, and it is a NP-hard problem and an approximation algorithm with constant approximation factor is presented for it. Because of extensive applications of this problem in wireless networks, attempts have been done to reduce this factor. According to last attempts this factor was reduced from 72 to 38 [8].

* Corresponding Author

The problem of finding the smallest covering circle that covers points has many applications in real world, one of which is facility location [9]. Although coverage problems have been presented as full covering, sometimes this is not possible due to resource limitation.

The problem of maximal covering by one circle with fixed size was posed in 1981 by Drezner [10]. He introduced an algorithm of time order $O(n^2 logn)$ for solving this problem. Later in 1986, Chazelle & Lee could develop an algorithm of time order $O(n^2)$ for solving this problem [11].

In the problem of covering weighted points by a unit circle, a set including *n* points with positive real weights in two-dimensional space is given. The aim is putting a circle with unit radius in the plane such that total weight of points inside the circle is maximized. An approximation algorithm has been presented [12] for solving this problem.

Deberg et al. in 2006 [13] indicated that the problem of putting *m* circles (for constant *m* and *m* > 1) such that maximum points are covered is solvable in time $O(n^{2m-1} logn)$.

In 2008, Cabello et al. [14] investigated the problem of maximum coverage of points with two separate circles with constant size, and developed an algorithm of time order $O\left(n^{8/3} log^2 n\right)$ for solving it.

The point sets R and B are called circular separable, if there is a circle that covers the points of B; while the points of R located outside this circle.

In 1984, Kim and Anderson [15] developed an algorithm, of time order $O(n^2)$ for distinguishing circular separability of two sets of points of total size n.

In 1986, Orouke et al [16] indicated that the problem of recognizing circular separability of two sets of points P and Q with total size of *n* is solvable in optimum time $O(n)$. They also indicated that if the set of points is able to be separated circularly then finding smallest separating circle in time $O(n)$ and finding the largest separating circle in time $O(nlogn)$ is possible which indicates reduction of circular separability problem to linear separability problem in three- dimensional space.

In 2010, Bitner et al. [17] investigated the problem of finding smallest covering circle for blue points aiming to minimize the number of red points in this circle. This circle is computable in linear time.

In the problem of covering with outlier points using one circle, *n* points and *k* outlier points are given. The aim is covering *n-k* points using one circle such that the area of circle is minimized. An algorithm was developed in 1995 by Matousek with time $O(nlogk + k^3 n^{\epsilon})$ [18].

Later in 2008 an algorithm, was presented by Agarwal et al. for the problem of covering with outlier points using two circles such that the area of bigger circle is minimized at the time of $O(nk^7 log^3 k)$ [19].

In this paper, it is assumed that there are a number of points with two colors (blue and red) in R^2 space. The algorithm presented in this paper finds a solution for determining the positions of two circles among blue points such that maximum number of blue points are covered while red ones are avoided. Among circles that cover the same number of blue points, the aim is to finding circles that have smallest radiuses.

2. The characteristics of smallest separating circle

Observation: the smallest separating circle is a circle that besides having characteristics of separating circle, it has two blue points on its boundary located on a diameter or surrounded by three blue points or two blue points and one red point.

For investigating this issue, we follow the process of converting a separating circle into the smallest separating circle. In figure 1, a separating circle is indicated, its radius is reduced while its center is fixed. In figures red point are shown by stars, and blue points by bullets. Reduction of circle radius continues until the circle become tangent to one of the blue point located inside it.

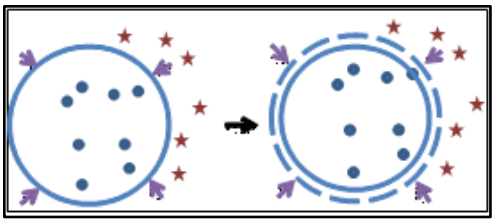

Fig. 1 Reduction of circle radius until becomes tangents on a blue point.

In this step, we observe that the circle still could become smaller. By moving the center of the circle toward the blue point that had become tangent to circle at previous step. This is continued until the circle becomes tangent to other point as well. see figure 2(a). For investigating whether the separating circle can become smaller, we move the center of the circle toward. The middle of the line-segment connecting the two points tangent to the circle. see figure 2(b).

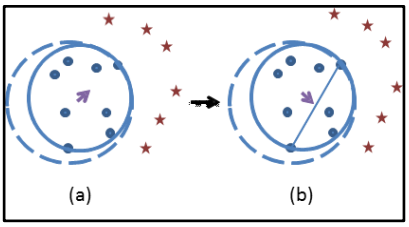

Fig. 2 closing center of separating circle to the point being tangent to the circle and then the middle of two tangent points.

In this step, three cases may be encountered. If two primary points located on the boundary of circle are two end points of circle diameter, the center is not able to become closer to the line passing these two points, because it is already located on it; this is the case indicated in figure 3(a). Otherwise, we move the center towards the middle of line segment connecting two tangent points until it becomes tangent to third point. This could be seen in figure 3(b). Other case happens when the separating circle before becomes tangent to the third point, it reaches to a red point outside the circle, the case indicated in figure 3(c), in which the circle becomes tangent to two blue points and one red point.

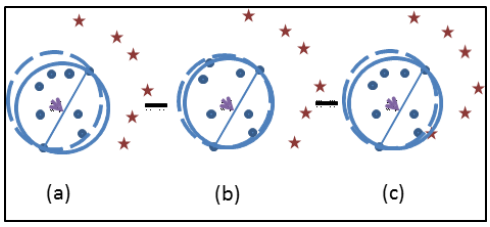

Fig. 3 The cases may occur during of closing center of separating circle to the middle of the line segment connecting two tangent points.

The idea described here has been taken from the smallest enclosing circle technique presented in [1,20].

3. The optimal algorithm for solving the problem

As said before, the smallest separating circle has two blue points, or three blue points, or two blue and one red points on its boundary. So, for obtaining a certain solution for two maximum separating circles, all possible cases that respectively require times $O(n^2)$, $O(n^3)$ and $O(n^3)$ should be investigated.

In next step, we explore the circles in the candidate list and two circles that totally cover maximum number of blue points and among circles covering similar number of blue points having the smallest radius are in the final solution. Since this part of algorithm investigates two circles among all candidate circles added to the list in the previous step, it needs time $O(n^6)$ that is not practical.

4. The approximation algorithm for solving the problem

Optimal algorithm has time complexity $O(n^6)$ that is not practical. In this section, an approximation algorithm is presented for solving this problem of time complexity $O(n^2)$.

In fact, this algorithm, intends to find two smallest circles with maximum number of blue points in them where there is no red point. In this method, briefly, the given points are ordered based on a technique relying on point distance. Then, the centers of two circles are found based on a criterion which is having maximum distance to red points. Then it begins to find enclosing circle from that point using incremental coverage idea in order to cover blue points, until it reaches to red points and no further progress is possible. This algorithm has time complexity $O(n^2)$ and it gives a near-optimal solution. In the following the details of the algorithm is presented by an example.

4.1 the details of the proposed algorithm

This algorithm runs on an array of points represented by their (x,y) coordinates and colors. Blue colors in the array are showed by B and red colors by R.

We firstly order the points based on their distances. For this purpose, we need a starting point for ordering. Starting point for ordering must be a point located at the extreme of the plane, i.e. there is no point before it. For this purpose, finding two farthest points helps us in this regard. Therefore, the distance between each two points i and j of the given blue and red points are obtained and we find two points that have the farthest distance to each other. For example, consider the array filled by random data in table1.

Based on the data, the two points having the farthest distance to each other are the seventh point with (2,28) coordinates colored red and the last point with (7,29) coordinates colored blue. Running this part of algorithm is possible in time $O(n^2)$. Then we start to one of these points for example point (2,28) here as starting point, find the closest point and put it on next position in the order. Next we select the point for putting in the next position of the order that have the least total distance from the previous points in the order, and so on. We would obtain ordered array of table 2. Running time of this part of algorithm is $O(n^2)$.

Table 1: The array filled by random data

X	6	26	4	8	7	19	2	12	11	17	16	18	14	23	25	20	27	5	10	29
Y	2	4	18	30	20	12	28	25	3	14	6	26	5	24	29	15	8	23	11	7
color	R	B	B	B	B	B	R	B	B	B	B	B	B	R	B	B	R	B	B	B

Table 2: The array ordered based on the mentioned ordering method

X	2	5	8	7	4	12	18	10	17	23	20	25	19	14	16	6	11	27	26	29
Y	28	23	30	20	18	25	26	11	14	24	15	29	12	5	6	2	3	8	4	7
color	R	B	B	B	B	B	B	B	B	R	B	B	B	B	B	R	B	R	B	B
rank	1	2	3	4	5	5	4	3	2	1	2	3	4	3	2	1	2	1	2	3

Then we are going to find two candidate blue points to pass our two separating circles through them. The blue points that have many blue points around and are far from red points seems good candidates. To find these good candidates, we rank points as follows. If this rank is high it indicates that this point has more distance from red points and there are more blue points in its neighborhood.

For computing these ranks we begin from red points and assign to each one rank = 1. Then we assign to each of array cells located in the left or right side of red points a rank =2. Then rank = 3 is assigned to each of surrounding cells with rank = 2, and so on each of cells are assigned a rank based on their distance from red points. This rank is indicated in fourth row of the array in table 2.

The first candidate which is selected is a point that has the highest rank and the second one is the point that has the second highest rank and its distance from the first selected point is at least the same as the rank of the first candidate point. This limitation is due to the fact that since separating circles are surrounded by red points, the second starting point shall be selected in a position that there is at least one red point in the distance between these two starting points and the second point is selected in a different position. This is done for separating as more as blue points. In this example, two points (18,4) and (12,19) are two points that meet this condition. This part of algorithm, needs the time $O(n)$.

After finding the two candidate blue points, we try to construct the two separating circles. To do this, first we sort the given array of points two times. In each time, it is sorted according to ascending distance of point from one of the two candidate points. Then we consider the three first points of array in each time. The following cases may occur:

- If two of them are red points, the initial point is considered as initial circle.
- If one of points is red, the circle is considered against the diameter passing through it as starting point and other blue point.
    - If a red point is included circle, starting point is considered as the initial circle, otherwise the circle is considered as initial circle.
- If all three points are blue, at first two farthest points are considered and we pass a circle with its diameter passing through these two points.
    - If there is no blue point in this circle, the circle passing through these three points is considered as the initial circle.
        - If a red point is located in the circle, initial starting point itself is considered as the initial circle. Otherwise, the circle passing through these three points is considered as initial circle.
    - Otherwise, the circle passing through two points is considered as initial circle.

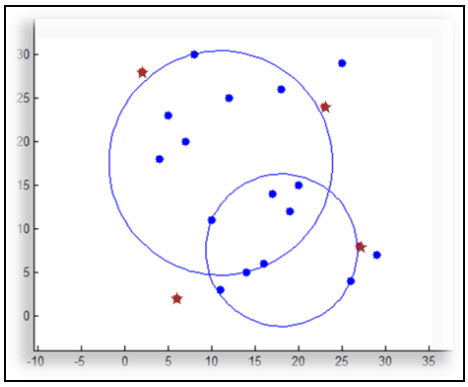

Fig. 4 the result of algorithm for separation with two circles.

In the next phase, the initial circles are extended. In this phase, we start from fourth cell of the array and progress to the end of the array. If selected point is blue located outside the circle, following operations are performed on each of initial circles:

- We consider the selected point with farthest and the second farthest points in the circle. We consider the circle with its diameter passing through selected point and the farthest point.
    - If a red point is in the circle or the second farthest point is not located inside the circle, we consider the circle passing through the three points.
        - If a red point is inside the circle, we withdraw the selected point and consider the next point in the array. Otherwise, the circle is considered as the extended circle.
    - Otherwise, the circle with two points is considered as the extended circle.
- We continue this approach of extending the circles up to the end of the array. When we reach to the end of the array, we have two final circles that are not extendable and they have separated blue points. The result of this example is indicated in figure 4.

For computing the two separating circles we need the time $O(n^2)$. By reviewing all the phase of algorithm, we see that the algorithm can be done in time $O(n^2)$. Pseudo code of the algorithm is given in figures 5- 9.

The algorithm for separation by two circles
Input: an array of blue and red points with x & y coordinates
Output: two circles in the plane

1- Finding points d1 and d2 with the maximum distance from each other among the points and choosing one of them as the basic point F.
2- Arranging all points by putting F in position 1 and putting a point in position i, where total distance from that point to i-1 previous points is less than the following points.
3- Assigning rank = 1 to each of red points and rank +1 to each of surrounding cells with highest rank value until the whole array takes the value of rank.
4- $a) f = maxrank$ $b)\ k = maxrank\ after\ f\ and\ dist(k_i, f_i) \geq f$
5- Arranging the array based on each of f and k points and performing steps 6 – 8 for each of them (s = f or k).
6- If two points of s +1 and s +2 are red, then initial circle = s.
7- If one of s +1 and s+2 are red, then run algorithm 2.
8- If none of s+1 and s+2 points are red, then run algorithm 3.
9- Until reaching to the end of array and obtaining two extended circles, if selected point is blue and out of circle, consider it on the boundary of circle and run stages 10- 12.
10- We consider point s with two points, namely mds1 and mds2 inside extended circle or initial circle which respectively have maximum distance from point s.
11- We plot circle c with s diameter to mds1.
12- If there is red point in circle c or mds2 is not in circles, run algorithm 5, otherwise extended circle = c.

Fig. 5 Algorithm 1 The separating algorithm, with two circles.

The algorithm for drawing initial circle with presence of one of red initial points

1- Plotting circle c with a diameter passing through point s until one blue point s+1 or s+2 is reached.
2- If there is no red point in the circle, initial circle = c, otherwise initial circle = s.

Fig. 6 Algorithm 2 drawing initial circle with presence of one of red initial points.

The algorithm for drawing initial circle with presence of no initial red point

1- Consider two farthest points between s, s+1 and s+2 as sd1 and sd2 and the other point as sd3.
2- Draw circle c with diameter of sd1 to sd2.
3- If sd3 point is not inside circle c and or there is a red point inside circle c, run algorithm 4, otherwise let initial circle = c.

Fig.7 Algorithm 3 drawing initial circle without any red initial point.

The algorithm for drawing initial circle using three points
1- Draw circle c passing through three points: s, s+1 and s +2.
2- If no red point is inside circle c, then initial circle = c, otherwise initial circle = s.

Fig.8 Algorithm 4 drawing initial circle using three points.

The algorithm for drawing extended circle using three points
1- Draw a circle passing through s, mds1 and mds2.
2- If there is a red point in c, then we left selected point, otherwise let extended circle = c.

Fig.9 Algorithm 5 drawing extended circle using three points.

4.2 Evaluation of the proposed algorithm

The proposed approximation algorithm and the optimal algorithm are implemented in MATLAB environment. In order to evaluate efficiency of the proposed algorithm, and also finding approximation factor of the algorithm, we use table 3 in which the results of 35 tests are presented on approximation algorithm and optimal algorithm are indicated.

Mean of approximation factor = $\frac{\sum_1^{35} af}{35} = 1.32$ (1)

In order to evaluate the performance of the proposed algorithm compared to optimal algorithm, an approximation factor was calculated for it. The result obtained from 35 iterations, yielded 1.32 approximation factor for the proposed algorithm. This number indicates the result of the proposed algorithm is not worse than 1/1.32 of the result of the optimal algorithm. Figure 10 indicates the result of one of those tests that gave similar result for both algorithms. Of course it must be noted that the proposed algorithm has much less running time $O(n^2)$, against running time $O(n^6)$ of the optimal algorithm.

Table 3: Computing approximation factor of the proposed algorithm

Test No.	Number of blue points	Number of separated blue points using the proposed algorithm	Number of separated blue points using the optimal algorithm	approximation factor
1	15	11	15	$^{15}/_{11}$
2	16	16	16	$^{16}/_{16}$
3	14	13	13	$^{13}/_{13}$
4	16	15	16	$^{16}/_{15}$
5	17	14	17	$^{17}/_{14}$
6	18	17	18	$^{18}/_{17}$
7	13	9	11	$^{11}/_{9}$
8	14	9	10	$^{10}/_{9}$
9	15	8	11	$^{11}/_{8}$
10	15	8	12	$^{12}/_{8}$
11	17	11	15	$^{15}/_{11}$
12	15	10	15	$^{15}/_{10}$
13	13	9	11	$^{11}/_{9}$
14	16	11	15	$^{15}/_{11}$
15	14	9	12	$^{12}/_{9}$
16	13	8	10	$^{10}/_{8}$
17	13	8	10	$^{10}/_{8}$
18	17	12	17	$^{17}/_{12}$
19	16	8	14	$^{14}/_{8}$
20	13	6	10	$^{10}/_{6}$
21	17	9	17	$^{17}/_{9}$
22	13	9	11	$^{11}/_{9}$
23	16	10	12	$^{12}/_{10}$
24	13	10	10	$^{10}/_{10}$
25	13	7	10	$^{10}/_{7}$
26	13	7	9	$^{9}/_{7}$
27	15	9	12	$^{12}/_{9}$
28	16	12	16	$^{16}/_{12}$
29	11	7	8	$^{8}/_{7}$
30	14	9	11	$^{11}/_{9}$
31	17	14	17	$^{17}/_{14}$
32	16	7	12	$^{12}/_{7}$
33	13	4	7	$^{7}/_{4}$
34	15	12	13	$^{13}/_{12}$
35	16	10	15	$^{15}/_{10}$

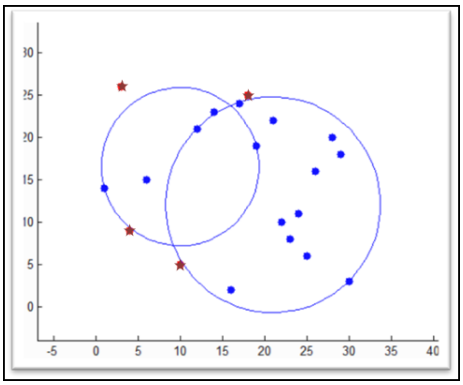

Fig.10 the similar result of two algorithms for separating by two circles.

5. Conclusions and future work

In present study, we proposed an approximation algorithm for separating the points by two circles. Since investigating all cases of two separating circles which is done by optimal algorithm is not suitable, since it is time consuming, an approximation algorithm was proposed for solving this problem which has time complexity of $O(n^2)$ and it gives near- optimal solution.

The proposed algorithm could be generalized for separating blue points with more than two circles. Also with some changes, we may find two biggest separating circles by this method. If we pose limitations such that the circles must be disjoint from each other or certain points must be covered and or separating circle's radius is a given fixed number, a new problem would emerge. Moreover, similar methods may be used when separation by two circles is required while each circle must cover a certain color. Also this method may be used for separating by two circles when we have more than two colors.

References

[1] M. de Berg, M. van Kreveld, M. Overmars and O. Cheong Schwarzkopf, "Computational geometry: Algorithms and applications," pp. 86-89, Springer 2000.
[2] J. J. Sylvester, "On Poncelet's approximate linear Valuation of surd forms," Philosophical Magazine Series, vol. 20, no. 132, pp. 203-222, 2009.
[3] L. J. Bass and S. R. Schubert, "On Finding the Disc of Minimum Radius Containing a Given Set of Points," Mathematics of Computation, vol.21 , pp. 712-714, 1967.
[4] D. J. Elzinga and D. W. Hearn, "Geometrical solutions for some minimax location problems," Transportation Science, pp. 379-394, 1972.
[5] M. I. Shamos and D. Hoey, "Closest-point problems," 16th Annual IEEE Symposium on Foundations of Computer Science, pp. 151-162, 1975.
[6] N. Megiddo, "Linear-time algorithms for linear programming in R3 and related problems," SIAM J. COMPUT, vol. 12, no. 4, pp. 759-776, November 1983.
[7] P. K. Agarwal and M. Sharir, "Planar Geometric Location Problems,",Algorithmica 11, pp.185-195, 1994.
[8] P. Carmi, M.J. Katz and N. Lev-Tov, "Covering points by unit disks of fixed location", in: 18th Internat. Symp. on Algorithms and Comput., pp.644-655, 2007.
[9] J. M. Robert and G. T. Toussaint, "Computational Geometry and Facility Location," in Proc. International Conference on Operations Research and Management Science, pages 1-19, 2007.
[10] Z. Drezner, "On a modified one-center model," Management Science, vol. 27, no. 7, pp. 848-851, June 1991.
[11] B. Chazelle and D. T. Lee, "On a circle placement problem," Computing, vol. 36, no. 1-2, pp. 1-16, 1986.
[12] C. Figueiredo and G. Fonseca, "Enclosing weighted points with an almost-unit ball," Information Processing Letters, vol. 109, no. 21-22, pp. 1216-1221, 31 October 2009.
[13] M. de Berg, S. Cabello and S. Har-Peled, "Covering many or few points with unit disks," Theory Comput. Syst., 45(3):446-469, 2009.
[14] S. Cabello, J. M. Díaz-Báñezc, C. Seara, J. A. Sellares, J. Urrutia and I. Ventura, "Covering point sets with two disjoint disks or squares," Computational Geometry, vol. 40, no. 3, pp. 195-206, August 2008.
[15] T. A. Kim and A. Anderson, "Digital disks and a digital compactness measure," in Annual ACM symposium on Theory of computing, New York, pp.117-124, 1984.
[16] J. O'Rouke, S. R. Kosaraju and N. Megiddo, "Computing circular separability," Discrete & Computational Geometry, vol. 1, no. 1, pp. 105-113, 1986.
[17] S. Bitner, Y. Cheunge and O. Daescu, "Minimum Separating Circle for Bichromatic Points in the Plane," In ISVD, pp.50-55, 2010.
[18] J.Matousek. "On geometric optimization with few violated constraints," proc.10th Annual ACM Symposium on Computational Geometry, pp.312-321, 1994.
[19] P.K. Agarwal, J.M. Phillips. "An Efficient algorithm for Euclidean 2-Center With Outlier," proc. Of 16th ESA, pp. 64-75,2008.
[20] J N. Megiddo, "Linear-time algorithms for linear programming in R 3 and related problems," SIAM J. Comput., vol. 12, pp. 759-776, 1983.

Afsaneh Hellatabadi Farahani received the B.S. degrees in computer engineering from Shariaty University and M.S. degrees in computer engineering from Islamic Azad University at Tehran. Currently she is a Ph.D. candidate in the computer engineering. Her research interests include computational geometry, graph drawing and graph algorithms.

Alireza Bagheri received his B.S. and M.S. degrees in computer engineering from Sharif University of Technology (SUT) at Tehran. He received his Ph.D. degree in computer science from Amirkabir University of Technology (AUT) at Tehran. Currently he is an assistant professor in the computer engineering and IT department at Amirkabir University of Technology (AUT) at Tehran. His research interests include computational geometry, graph drawing and graph algorithms.