DISCRETE DISC COVER PROBLEM

A Project Report Submitted
in Partial Fulfilment of the Requirements
for the Degree of

BACHELOR OF TECHNOLOGY

in

Mathematics and Computing

by

Raj Kamal

(Roll No. 09012321)

to the

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, INDIA

April 2013

CERTIFICATE

This is to certify that the work contained in this project report entitled "**Discrete Disc Cover Problem**" submitted by **Raj Kamal** (**Roll No.: 09012321**) to Indian Institute of Technology Guwahati towards partial requirement of **Bachelor of Technology** in Mathematics and Computing has been carried out by him under my supervision and that it has not been submitted elsewhere for the award of any degree.

Guwahati - 781 039 (Dr. Gautam K. Das)

April 2013 Project Supervisor

ABSTRACT

Given a set R of n red points and a set B of m blue points on a 2-dimensional plane, Discrete Disc Cover problem is about finding a subset C of B of minimal cardinality such that if we draw circles with radius r with centers around points in C all the red points from R will lie inside the unions of circles in C. The purpose of this project is to come up with an efficient algorithm to cover points with circle of given radius. This is a variant of set cover problem. The decision version of set covering is NP-complete. Lund and Yannakakis [1] showed that set covering cannot be approximated in polynomial time within a factor of $0.72 \ln n$. Discrete Unit Disc Cover [2] is an variant of this problem. We adopted an efficient algorithm for the problem using K-Mean Clustering algorithm. We apply both continuous and discrete version of K-Mean Clustering algorithm and perform experiments and our approach converged in every feasible case. Experiments are done with varying number of blue, red points and in all feasible case, our algorithm is able to find cover.

Contents

Chapter 1

Introduction

1.1 Introduction

Discrete Disc Cover problem is an instance of covering problem. In combinatorics and Computer Science, covering problems are those problems that ask whether a certain combinatorial structure covers another, or how large the structure has to be do that. Covering problem are minimization problem, packing problems are dual of the covering problems.

Recent interest in specific geometric set cover problems is partly motivated by applications in wireless networking. In particular, when wireless clients and servers are modeled as points in the plane and the range of wireless transmission is assumed to be constant (say one unit), the resulting region of wireless communication is a disk of certain radius centered on the point representing the corresponding wireless transmitting device. Under this model, sender a successfully transmits a wireless message to receiver b if and only if point b is covered by the unit disk centered at point a. This model applies more generally to a variety of facility location problems for which the Euclidean distance between clients and facilities cannot exceed a given radius, and clients and candidate facility locations are represented by discrete sets of points. Other instances are selecting locations for wireless servers (gateways) from a set of candidate locations to cover a set of wireless clients, selecting a set of weather radar antennae to cover a set of cities. These problems can be modeled by Discrete Disc Cover Problem.

Definition: Given a set R of n red points and a set B of m blue points on a 2-dimensional plane, Discrete Disc Cover problem is about finding a subset C of B of minimal cardinality such that if we draw circles with radius r with centers around points in C all the red points from R will lie inside the union of circles in C.

1.2 Related Works

- **Minimum geometric disk cover**: Here we are given set of points and we have to find unit disk of minimum cardinality whose union covers the points. Here disk centers are not restricted to the set but can be arbitrary points from the plane. Again this problem is NP-hard [3] and has a PTAS solution [[4], [5]].

- **Discrete K centers** : Given two sets of point in the plane P and Q and an integer K and we have to find a set of K disks centered on points in P whose union covers Q such that the radius of the largest disk is minimized. This is constraint optimization problem. We have to get a set Q that consists of K disks whose centers are from points in P and constraint is that disks can have atmost radius one. This problem is NP-Hard.

- **Discrete unit disk cover Problem** Given a set P of n points and a set D of m unit disks and DUDC problem is finding minimum cardinality subset D^* covering all the points in P[2]. The research was done by Das et al., where they proposed constant factor approximation algorithm to the problem in O(nlog n + m log m + mn). They brought down the approximation approximation factor to 18 which was previously 22 by Claude et al., [6].

- Efficient Algorithm for placing a given number of base stations to cover a convex region. Here Voronoi diagram is used for covering [7].

Chapter 2

Algorithm

2.1 Testing Feasibility

Checking feasibility is an important step while solving an optimization problem. In discrete disc cover problem we have to find minimum subset set of blue points from a given set of blue points in 2 dimensional plane such that circles with radius *rad* is drawn around points in the subset to cover all the red points which are also given in 2 dimensional plane. In order to proceed for the solution first we have to be sure whether the solution exists or not i.e, checking the feasibility of problem. For feasibility we fix a radius *rad*. We select a red point, we find out distance with every blue point, if any of the distance with respect to that red point is found to be less than *rad*, then we can say that red point can be covered, otherwise it can't be covered. We repeat the same with every red point. If any of the red point is found to be uncovered, then we say that problem is not feasible, otherwise feasible.

2.2 Algorithm

We have been given a set R of n red points and a set B of m blue points. We define a red point is covered if there exist a blue point such if circle of given radius rad is drawn with center around that blue point then that red point lies inside the circle. In order to find the cover of all red points, we have adopted two approaches. One is based on continuous version of K Mean Algorithm and other is based on discrete version of K Mean Algorithm.

2.2.1 K Mean Continuos Version

This approach is based on continuous version of K Mean Algorithm. First we define some basic structure that we will be using in describing algorithm.

Cluster: Cluster is an entity containing cluster head and a set of red points. Cluster head is the point according to which red points are selected and we create cluster around the head. All those red point we pick in the set we call that set as set of red points. We denote ith cluster by $\mathrm{c1}_i$ and corresponding set of red points as $s1$.

Cluster Set: It is a set containing Clusters. We denote the Cluster set by $C1$

Blue Cluster: Blue Cluster is an entity containing cluster head and a set of red points. Cluster head is the point according to which red points are selected and we create cluster around it. All those red point we pick in the set we call that set as set of red points. We denote ith cluster by c_i and corresponding set of red points as s.

Blue Cluster Set: It is a set containing Blue Clusters. We denote the blue cluster set as C.

We assume that feasibility criteria is satisfied and we are using circles of radius *rad* for covering.

Algorithm

1. Fix value K.
2. Randomly select K blue point from set B of blue points.
3. Create K clusters with cluster head selected from K randomly selected blue points. Each of K randomly selected blue point in step 2 can be cluster head of one and only one cluster. We denote Cluster by c1$_i$, i=1, 2, 3, ..., K as defined. Now for each cluster we create set of red points $s1$ as defined. For each r$_i$ $\in R$, i=1, 2, 3, ..., n, we calculate the euclidean distance between r$_i$ and cluster head c1$_j$, j=1, 2, 3, ..., K, denote it by d1$_{ij}$, i=1, 2, 3, ..., n j=1, 2, 3, ..., K and we put r$_i$ to the set of red points $s1$ of that c1$_j$, j=1, 2, 3, ..., K whose distance d1$_{ij}$, is minimum. Thus Cluster set $C1$ containing K clusters is created.
4. Change Cluster head. For each cluster c1$_i$, i=1, 2, ..., K, we find out the mean of X-coordinates and Y-coordinates of all red points in the set of red points $s1$ of that cluster and assign the mean of X and Y coordinates as the new coordinates of cluster head. The Euclidean distance between old and new cluster head is calculated for each cluster c$_i$, i=1, 2, 3, ..., K in cluster set $C1$ and denote by diff1$_i$, i=1, 2, 3, ..., K.
5. We find the minimum of diff1$_i$, i=1, 2, 3, ..., K and denote it by *min*.

6. If *min* is not less than a *tolerance*, we repeat step 3 to 5.

7. Calculate the radius r_i of each cluster $\mathrm{c1}_i$ where i=1, 2, 3, ..., K. Radius is defined as maximum distance between cluster head of $\mathrm{c1}_i$ and red points from corresponding set of red points $s1$.

8. Calculate the maximum of the radius r_i i=1, 2, 3, ..., K and denote it by *max*.

9. If *max* is greater than *rad* than we increment the K by some factor and repeat step 2 to 8.

10. Create Blue Cluster c_i, i=1, 2, 3, ..., K and Blue Cluster set C similar to cluster and cluster set. For each cluster head of $\mathrm{c1}_i$, i=1, 2, 3, ..., K we find a blue point $b_j \in B$, j=1, 2, 3, ..., m such that Euclidean distance between head of cluster $c1_i$ and b_j is minimum for all blue points in B. Thus K corresponding Blue Points are selected.

11. Create K Blue Clusters with cluster head selected from K randomly selected blue points from step 10. Each of K randomly selected blue point in step 10 can be cluster head of one and only one Blue Cluster. We denote Blue Cluster by c_i, i=1, 2, 3, ..., K as defined. Now for each Blue Cluster we create set of red points s as defined. For each $\mathrm{r}_i \in R$, i=1, 2, 3, ..., n, we calculate the Euclidean distance between r_i and Blue Cluster head c_j, j=1, 2, 3, ..., K, denote it by d_{ij}, i=1, 2, 3, ..., n j=1, 2, 3, ..., K and we put r_i to the set of red points s of c_j, j=1, 2, 3, ..., K whose distance d_{ij}, j=1, 2, 3, ..., K is minimum and mark that red point as covered. Thus Blue Cluster set C containing K Blue clusters is created.

12. Eliminate unfavorable Blue Clusters from Blue Cluster set and refine each

Blue Cluster. Unfavorable Blue Cluster means the Blue Cluster in which the Euclidean distance between the cluster head and the each of red point from set s is greater than rad. Refining the Blue Cluster means we remove all those red point from the set s of the Blue Cluster whose distance from cluster head is greater is greater than rad and mark that red point as uncovered. For each Blue Cluster $c_i \in C$, i=1, 2, 3, ..., K we do eliminate unfavorable and refine the set of that Blue Cluster.

13. All those red point which are not yet covered, we search for the blue point $b_j \in B$, j=1, 2, 3, ..., K which can cover it, that is whose distance from the uncovered red point is less than rad. We create Blue Cluster with the cluster head as that blue point b_j. For that cluster head we create set of red points s. The set is created by taking all those red point which are uncovered and whose distance from cluster head is less than rad and then marking those red point as covered. We then insert the newly formed Blue Cluster in the Blue Cluster set.

14. Hence we obtain cover of red point. The size of Blue Cluster set C determine the number of blue points $b_j \in B$, j=1, 2, 3, ..., m that are required to cover all the red points in R.

Lemmas and Proofs

Lemma: Algorithm Converges in every feasible condition.

Proof: In order to prove this we have to prove that maximum of the radii of cluster will decrease or remain same, leading to decrease in min diff calculated in step 5. Suppose radius of cluster $c1_j$ is maximum and we call it as maximum radius max. After this we find the new cluster head by taking mean of X and Y coordinates in set of red points $s1$ in c1$_i$, i=1, 2, 3, ..., K. Now for the cluster with maximum radius *max* i.e, c1$_j$ if we calculate the Euclidean distance between new cluster head and all red points in set of red points, we find that distance is less than the radius *max* as previously calculated as the center moves closer towards farthest red point. It might happen that while we create a new cluster than some new point may come and some point may go out of cluster. In case new point come inside that means the distance between cluster head and that new point is minimum for all cluster head but previously calculated statistics says it is closer to some another cluster and that distance is less than radius *max* so newly distance must be less than radius *max* as it prepare set by taking nearest red point. Similarly if a point goes out of the set to another cluster than too the distance decrease by same argument. Hence there can be only two cases possible either radius will decrease or remain same leading to min diff less than tolerance after some iteration. Also there are fixed number of red points so it will stop.

Lemma:Algorithm finds the cover.

Proof:From previous lemma we can be assure of that algorithm will converge. Now it might happen that radius it converges is more than *rad* so in that case we set new K and repeat the step 2 to 8 till we find our algorithm converge with the radius less than *rad*. Now here we are assure that most of the red points are cover by the cluster and all lies with in radius *rad*. Now a new set Blue Cluster is created according to step 10 to 12. Here we find a blue point corresponding to cluster head in Cluster set because it might happen the cluster head in Cluster may not coincide with blue point. Now there it happen that some of the red point are still are not covered and because of the fact that we have a feasible case we can definitely find out cover for all those uncovered red points. Thus our algorithm finds out cover.

2.2.2 K Mean Discrete Version

This approach is based on discrete version of K Mean Algorithm. First we define some basic structure that we will be using in giving algorithm.

Red Point: It is an entity containing coordinates of red point and a boolean variable cover. The variable cover is true if it can be cover by at least one blue point. It is denoted by r$_i$.

Red Point Set: It is a set containing Red Point. It is denoted by R.

Blue Point: It is an entity containing coordinates of blue point and a boolean variable cover. The variable cover is true if it is a cluster head. It is denoted by b$_i$.

Red Point Set: It is a set containing Blue Point. It is denoted by B.

Cluster: It is an entity containing coordinates of cluster head, radius of cluster, size of cluster and a set of red point denoted by h, r, $size$ and s respectively. We denote cluster by c$_i$. Set of red points is a set containing all those red points which forms cluster around cluster head based on some condition which we will state further.

Cluster Set: It is a set containing cluster. It is denoted by C.

Here are some assumptions and definitions of some terms that we will using again and again in algorithm.

rad: We assume that we will be working with circle with radius rad. We can change its value according to our requirement.

cover: We say that a red point is cover if there exist at least one blue point such that if a circle is drawn with radius *rad* and center around that blue point then that red point lie inside the circle.

Radius of cluster: It is defined as the maximum distance between cluster head and the points inside the set s in the cluster.

Algorithm

INPUT: A set of n red points and a set of m blue points in 2 Dimensional Plane.

1. For each n red points we create a corresponding entity Red Point r_i, i=1, 2, 3, ..., n as defined. This contains coordinates of red point and a boolean variable *cover*. *cover* is true if it can be cover by at least a blue point. Initially it is false for all Red Point. We then create a Red Point set containing Red Point entity. The cardinality of Red Point set is n.

2. For each m blue points we create a corresponding entity Blue Point b_i, i=1, 2, 3, ..., m as defined. This contains coordinates of blue point and a boolean variable *cover*. Initially it is false for all Blue Point. We then create a Blue Point set containing Blue Point entity. The cardinality of Blue Point set is m.

3. Fix K.

4. Select K randomly Blue Point b_i out of Blue Point Set and make the cover of that Blue Point entity true.

5. Create K Cluster with cluster head selected from K randomly selected chosen Blue Point. Each of K randomly selected Blue Point can be cluster head of

one and only one cluster. We denote the cluster by c_i, i=1, 2, 3, ..., K. Now for each cluster c_i we create a set of red points, s. For each $r_i \in R$, i=1, 2, 3, ... n, we calculate the Euclidean distance between r_i and cluster head c_j, j=1, 2, 3, ..., K, denote it by d_{ij}, i=1, 2, 3, ..., n j=1, 2, 3, ..., K and we put r_i to the set of red points s in c_j, j=1, 2, 3, ..., K whose distance d_{ij}, j=1, 2, 3, ..., K is minimum and update the size parameter of that cluster. If the minimum distance calculated comes out to be more than *rad*, we mark Red Point cover as false otherwise true. Thus Cluster set C containing K clusters is created.

6. We find the radius of each cluster and update the radius parameter of the cluster.

7. Change Cluster Head. For each cluster c_i, i=1, 2, 3, ..., K, we take the set s and find out the radius of the set considering each Blue Point as cluster head and out of the them the one which has minimum radius we take that Blue Point and assign it as the new cluster head and mark cover of that Blue Point true. The old Blue Point is marked false. We calculate the distance between old cluster head and new cluster head, denoted as diff_i for i=1, 2, 3, ..., K.

8. Calculate *cur_rad* of the Cluster Set. It is the maximum of all cluster radius.

9. Calculate the maximum of diff_i, i=1, 2, 3, ..., K.

10. If the maximum is more than a given tolerance, we repeat step 4 to 9.

11. If the *cur_rad* is more than *rad* we increment K by increment factor and repeat step 3 to 10.

12. Eliminate unfavorable cluster. There may be cases such that the set of red point s is empty, i.e size is 0, those are unfavorable cluster and should be

removed. For each cluster c_i, i=1, 2, 3, ..., K we check the size, if it is 0 then we remove the c_i from the C.

13. All those Red Point which are not yet covered, we search for the Blue Point which can cover it. We create Cluster with the cluster head as the Blue Point. For that cluster head we create set of red points s. The set is created by taking all those Red Point which are uncovered and whose distance from cluster head is less than rad and mark that Red Point as covered. We then insert the newly formed Cluster in the Cluster set.

14. Hence we obtain cover of Red Point. The size of Cluster set determine the number of Blue Points that are required to cover all the Red Point.

Lemma and Proofs

Lemma:The Algorithm II converges.

Proof:In order to prove the convergence we have to show that the maximum radii of the cluster decreases or remains same, which leads to decrease in max diff calculated in step 9. Suppose radius of cluster c_j is maximum and we call it as cur_rad calculated in step 8. We have to prove that this cur_rad will decrease or remain constant. In order to change the cluster head, we search for a blue point $b_j \in B$, j=1, 2, 3, ..., K such that considering it as cluster head if we find the radius corresponding to the set s in cluster, then that radius is minimum of all. We then change the cluster head to that Blue Point. Now for all points in set the maximum distance between points in set and cluster head is decreased, it might happen some distance may increase. Now we create cluster according to the new cluster head, it might happen some points may come in and some may go out of the set. Those points which come in have distance less than *cur_rad* as its distance with its previous cluster head is less than *cur_rad* and in that decreased radius after the change in cluster head, that distance is also less than *cur_rad* so it mush have moved to some cluster whose distance is further lesser. Similarly the case with Red Points that go out of the set. Thus in every step the radius is decreasing or remaining same and we know that there are fixed number of blue points and red points so it must converge after some time. Hence the max diff calculated in step 8 will decrease, for faster convergence we put tolerance. Hence Algorithm II converges

Lemma:Algorithm II finds the cover.

Proof:From above lemma we know that the algorithm will converge. Now it might happen it has converged but than *cur_rad* is greater than *rad*. In that case we increment the K and repeat the steps and since we know the algorithm will converge and feasibility condition, we will be able to find a K such that algorithm will converge such that *cur_rad* is less than *rad*. Now it might happen some cluster don't contain any red point so we avoid that in step 12. Further there can be some Red Points which are not yet covered so we found out cover for them in step 13 and we are able to do since we are given feasible condition. Hence algorithm will be able to find cover.

Chapter 3

Experimental Results

3.1 Results

We have tested the Algorithm I, covering based on continuos K Mean Algorithm on various domains ie range of x and y coordinates of blue or red points and with varrying number of red and blue points and finding the mean and tabulating the results.

Table 3.1: Results obtained in interval setting domain 1 to 100 for x and y both

Red Points	Blue Points	K
0100	145	59
0200	150	52
0500	190	65
1000	210	71
2000	225	75
3000	245	78

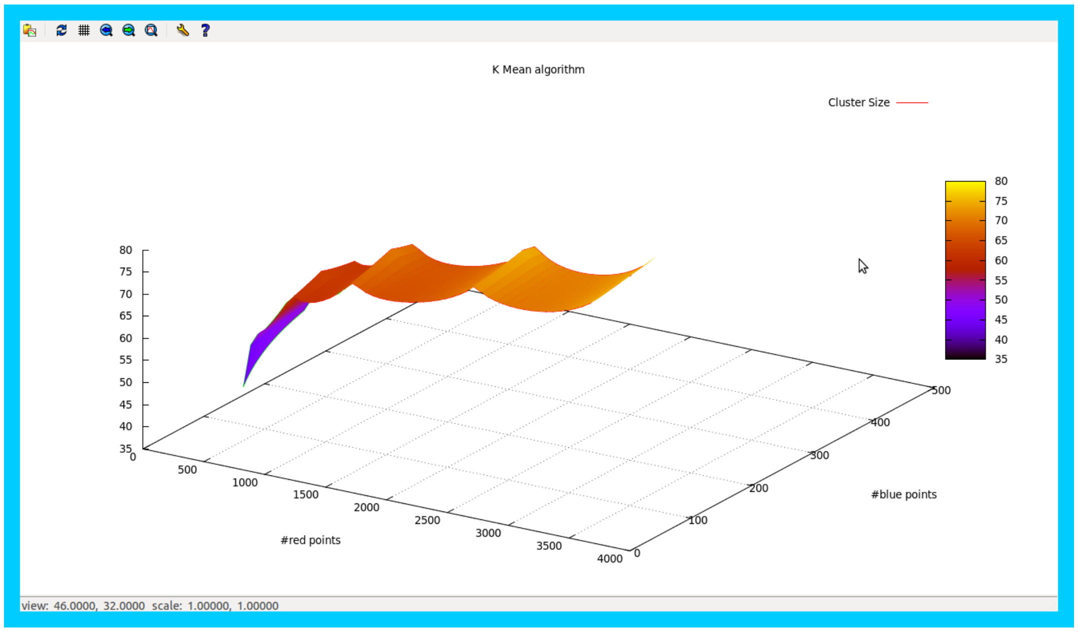

Figure 3.1: Surface plot for the cluster for interval 1 to 100

Table 3.2: Results obtained in interval setting domain 1 to 1000 for x and y both

Red Points	Blue Points	K
0100	1485	098
0200	1680	194
0500	2745	465
0700	0071	630
1000	1250	867

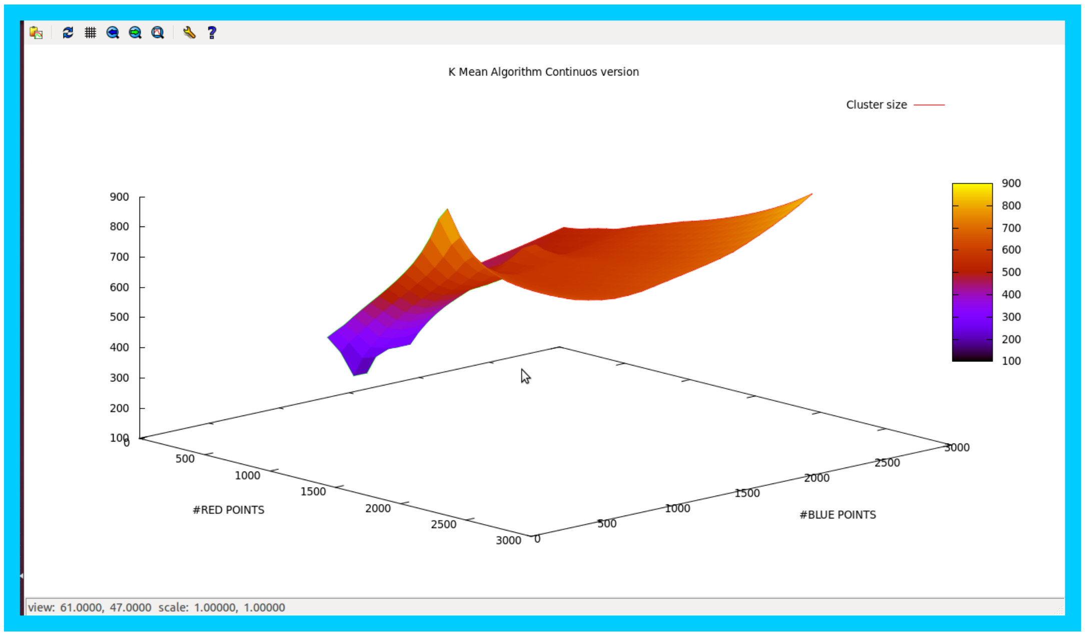

Figure 3.2: Surface plot for the cluster for interval 1 to 1000

Table 3.3: Results obtained in interval setting domain 1 to 100 for x and y both for descrete version of K Mean algorithm

Red Points	Blue Points	K
100	140	056
200	168	082
500	185	121
700	208	137
1000	210	144

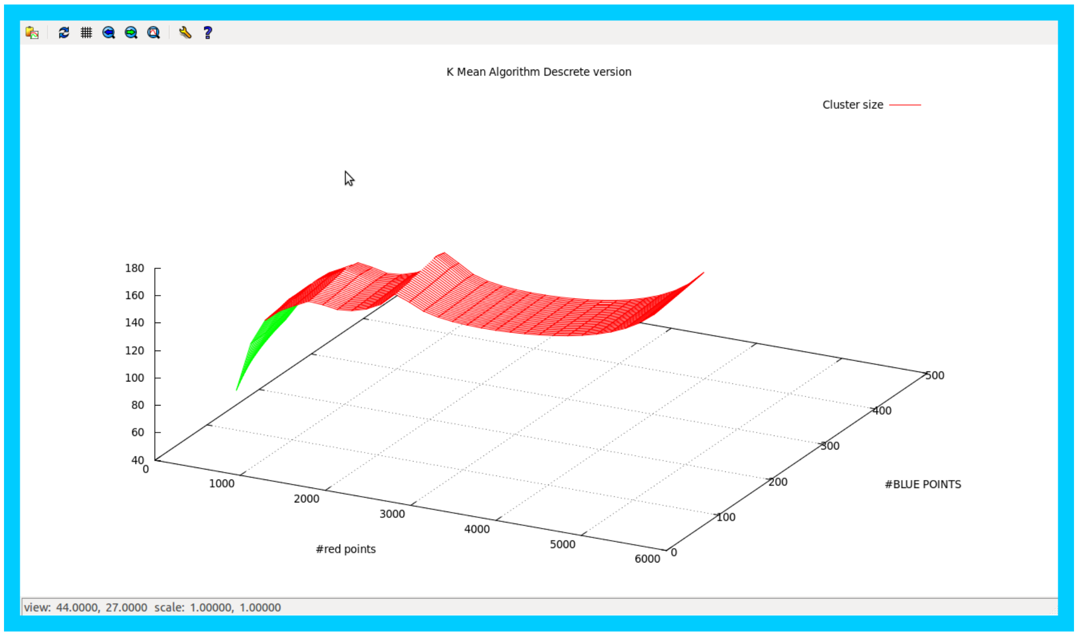

Figure 3.3: Surface plot for the cluster for interval 1 to 100

Table 3.4: Variation of radius

Radius	K	actual K	red points covered
30.46310	010	10	011
23.76970	010	10	013
23.76970	010	10	019
24.35160	025	22	050
20.02500	025	22	060
17.00000	025	22	067
17.00000	025	22	071
16.27880	040	31	057
16.27880	040	31	072
16.27880	040	31	075
17.20470	055	37	064
15.23150	055	37	087
18.97310	070	40	086
12.64910	070	40	093
12.64910	070	40	096
15.13270	085	44	080
11.31370	085	44	093
11.31370	085	44	095
15.26430	100	46	083
12.08300	100	46	098
12.64910	130	52	095
10.04990	130	52	096
16.27880	145	53	095
15.81140	145	53	096
11.40180	160	55	093
09.48683	160	55	100

3.2 Conclusion

Both K Mean Continuos Version and K Mean Discrete Version algorithm are able to find the cover. In cases where we have tremenduosly huge data the convergence rate is slow. Also when the domain is very large then discrete version works better than continuos version but still both have slow convergence rate. Discrete Version converges faster than continuos version.

Bibliography

[1] Lund, Carsten, Yannakakis, Mihalis *On the hardness of approximating minimization problems* J. ACM, 41, (1994) (960-981)

[2] Gautam K. Das, Robert Fraser, Alejandro Lopez-Ortiz, and Bradford G. Nickerson *On the descrete unit disk cover problem.* 5th International Workshop, WALCOM 2011, New Delhi, India, February 18-20, 2011. LNCS-6552 (2011) (146-157)

[3] Fowler, M. Paterson and S. Tanimoto, *Optimal packing and covering in the plane are np-complete*, Inf. Proc. Let., 12, (1981) (133-137).

[4] T. Gonzalez, *Covering a set of points in multidimensional space*, Inf. Proc. Let., 40, (1991) (181-188).

[5] D. Hochbaum and W. Maass, *Approximation schemes for covering and packing problems in image processing and VLSI*, J. ACM, 32, (1985) (130-136).

[6] F. Claude, G. K. Das, R. Dorrigiv, S. Durocher, R. Fraser, A. Lopez-Ortiz, B. G. Nickerson and A. Salinger, *An improved line-separable algorithm or discrete unit disk cover*, Disc. Math, Alg. and Appl., 2, (2010) (77-87).

[7] Gautam K. Das, Sandip Das, Subhas C. Nandy, Bhabani P. Sinha, *Efficient Algorithm for placing a given number of base stations to cover a covex region*J. Parallel Distrib. Comput. 66 (2006) (1353-1358).