Optimal Placement of Base Stations in a Cellular Mobile Network

A dissertation submitted in partial fulfilment of the requirements of M.Tech (Computer Science) degree of Indian Statistical Institute , Kolkata

By

Subhojyoti Ganguly

Under the supervision of

Dr. B.P. Sinha

Advanced Computing and Microelectronics Unit

Indian Statistical Institute
203 , Barrackpore Trunk Road
Kolkata – 700 108 .

June 2^{nd} , 2002

Indian Statistical Institute

203, Barrackpore Trunk Road ,

Kolkata - 700 108

Certificate of Approval

This is to certify that the thesis titled " Optimal Placement of Base Stations in a Cellular Mobile Network " submitted by **Subhojyoti Ganguly** towards partial fulfilment of requirements for the degree of M.Tech in Computer Science at Indian Statistical Institute , Kolkata embodies the work done under my supervision .

Dr. B. P. Sinha .
Advanced Computing and Microelectronics Unit ,
Indian Statistical Institute ,
Kolkata - 700 108 .

Acknowledgements

I take the pleasure in thanking Dr. B. P. Sinha for his friendly guidance throughout the dissertation period. His pleasant and encouraging words have always kept my spirits up.

I take the opportunity to thank my classmates, friends and my family for the their encouragement to finish this work.

Subhojyoti Ganguly

Contents :

Abstract : Minimising the transmitting power has gained importance in the state – of – the – art research activities in the field of mobile computing . We know that surrounding a base station there exists a region within which any mobile node can receive signals emitted by the base station. This region , called cell , has a span depending on the transmitting range and hence on the transmitting power of the concerned base station . Thus diminishing the transmitting power decreases the coverage of the cell . This work attempts to position the given number of base stations with equal transmitting capacity (i.e. equal size of the surrounding cells) so that the base stations can completely cover a given square region with minimum overlapping area and hence with minimum transmission capacity . We have assumed each cell to be circular instead of hexagonal . Thus we attempt to minimise the radius of a fixed number of equal circles covering a given square region . We have proposed a heuristic in this work for tackling the above problem and compared our results with the best possible results so far . Although we could not minimise the radius upto that extent but we could drastically reduce the time complexity as compared with them sacrificing the accuracy of the result to a small extent .

Introduction

The rapid development of wireless digital communication technology can be attributed to the extensive research works being carried out in the field of mobile computing . The falling cost of both communication and mobile computing devices is making wireless computing affordable not only to business users but also to consumers. Efficient energy management plays a vital role in decreasing the mobile communication cost . Energy management has already been addressed in the hardware level like switching off backlighting , providing the CPU with a power efficient doze mode , replacing the disk with more energy efficient semiconductor memories etc . But there is a growing pressure to improve the energy management at the transmission level .

In mobile **cellular network** a given area is covered by a number of base stations each of which can transmit signals omnidirectionally over a particular region surrounding it . This region is called cell . The main challenge lies in decreasing the size of the cell and hence the transmitting power , but still covering the entire area with the given number of base stations . An **ad hoc network** , however , is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralised administration . In this environment a mobile node transmits information to some other nodes via its neighbours i.e. those nodes who are presently under its transmitting range .

We have selected the case of mobile cellular network and tried to decrease the size of the cell , keeping the number of base stations fixed , by placing the base stations optimally so as to cover the given area entirely .

Previous works in this topic were unsuccessful in providing a formal Solution to this problem . Some attempts were made to minimise the radius through some Heuristic approaches .

We have selected the work of Kari J. Nurmela and Patric R. J. Ostergard Of Helsinki University of Technology , titled "Covering a square with upto thiry equal circles " [1] . This work which adopted the concepts of mechanics in guiding the movement of the circles has claimed to achieve the minimum result so far . We have selected their results in comparing the results of our work on covering a unit square with up to thirty equal circles .

We have developed a heuristic based on ' vector ' approach for tackling the above problem . Although we could not minimise the radius upto their extent (results varying mostly between 0.48 % to 10.7 % , excepting a few shoot ups on stray occassions) we have reduced the time complexity to a large extent . The results obtained in our case is almost instantaneous as compared with a few hours of CPU time in their case - worst timing for them being two weeks of CPU time on a 233 MHz Pentium PC for 27 circles .

Our Approach :

Since our aim is to optimise the radis of the circles , by Rearranging their positions so as to cover entirely the given square field , we first try to move a circle in the most optimum direction without changing its radius . By most optimum direction we mean that direction where the excessive overlaps of the circle should tend to minimise as well as the gaps across its circumference should also be diminished . The optimum directon which is different for different circles is evaluated by using a 'vector' approach as explained below :

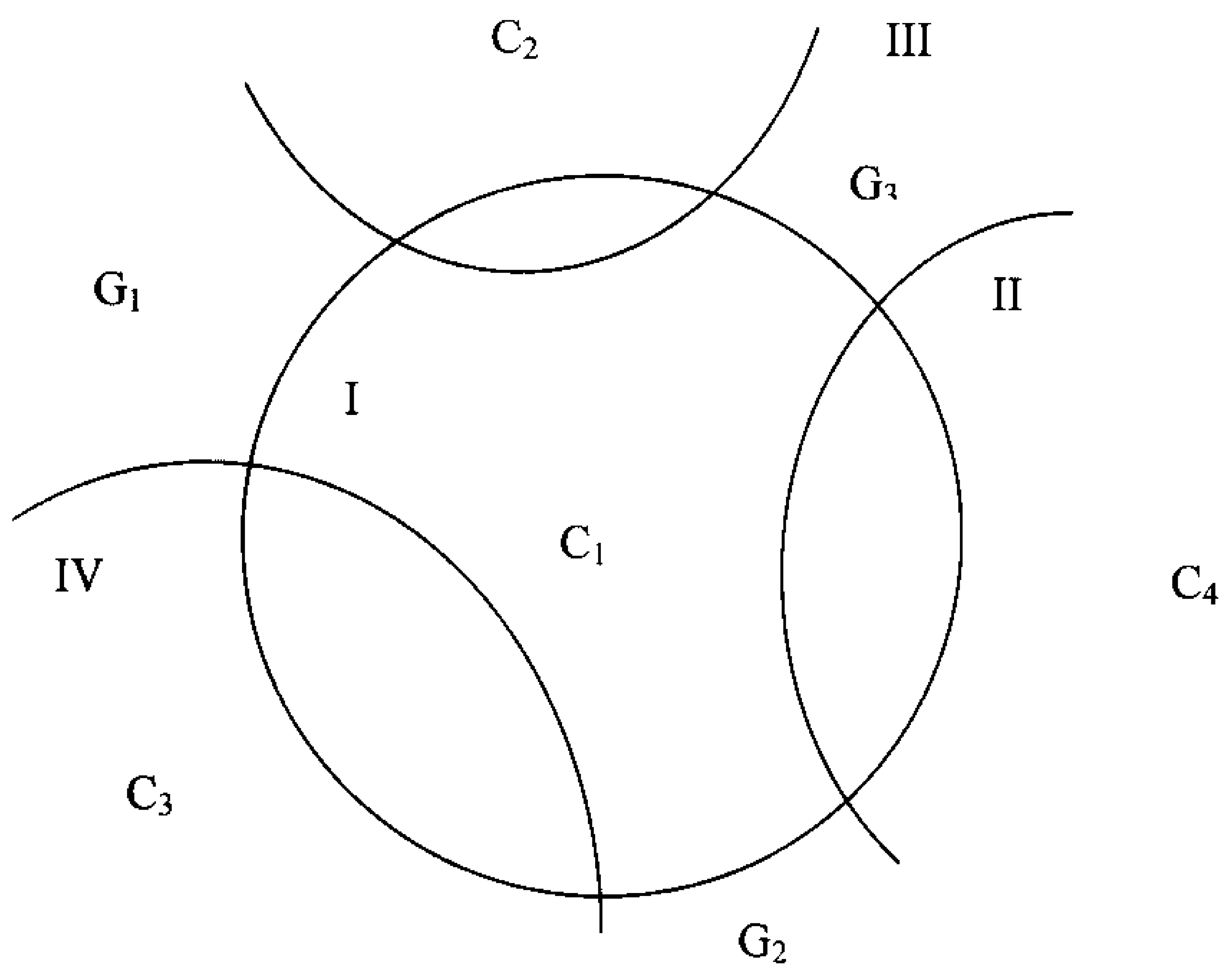


Fig . 1

Let the circle 1 with centre C1 has overlaps with its Neighbouring circles 2 , 3 and 4 with centres C2 , C3 and C4 respectively . We try to Find the most optimum direction for circle 1 .

With the help of basic geometry we calculate the overlapping areas of circle 1 with each of its neighbours 2 , 3 and 4 .Let these areas be A_{12} , A_{13} and A_{14} . We represent each of these overlaps as a vector with magnitude equal to the overlapping area and direction given by the line joining the centres of the respective circles starting from the centre of the circle under consideration to the centre of the neighbouring circle . All angles are measured relative to the positive direction of x axis. Thus overlapping vectors $\mathbf{A_{12}}$, $\mathbf{A_{13}}$ and $\mathbf{A_{14}}$ are represented as shown below :

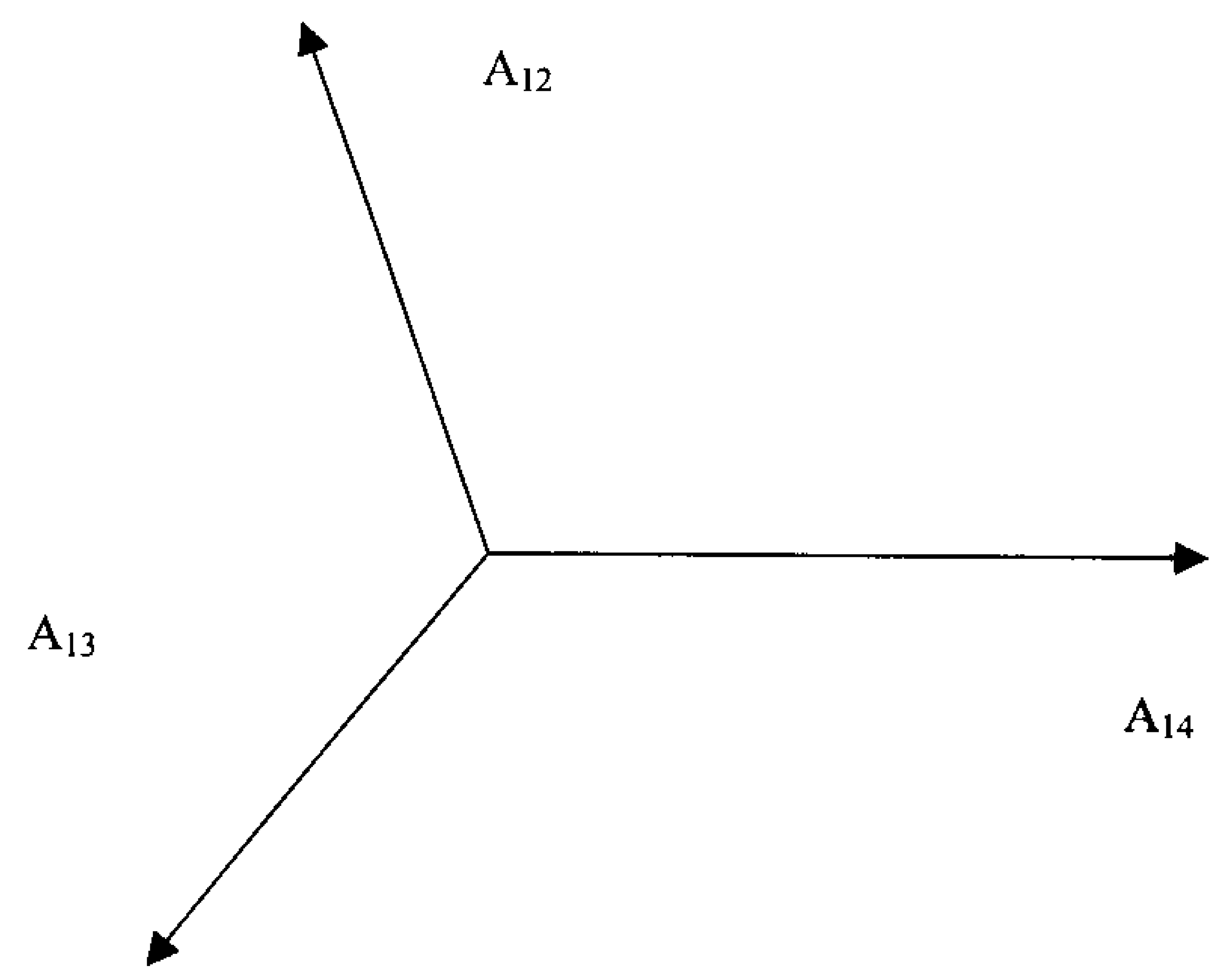


Fig . 2

In case of overlap with any side of the given field , we similarly evaluate the overlapping area which is the magnitude of the overlap and its direction is given by the bisector of the overlapping angle , pointing from the center of the circle to the side .

Since our aim is to minimize the overlap by repositioning the circle , keeping its radius fixed , we first find out the resultant vector **O** of the system of overlapping vectors and hence evaluate the negative of the resultant vectors **-O .** In order to minimise the overlap of the circle under consideration , we move the circle in the direction of **-O .**

Similarly , we find out a system of ' gap ' vectors for a particular circle . The term ' gap ' denotes that portion of the circumference which is bare i.e. without any overlap . In Fig. 1 the circle 1 has three gaps along its circumference g1 , g2 and g3 . Magnitude of the gap vector is given by the magnitude of the angular span of that gap . Its direction is given by the bisector of the gap angle , pointing outwards from the centre of the circle . Thus the three gap vectors **g1 , g2** and **g3** of Fig. 1 can be represented as a system of vectors similar to Fig. 2 . Now we find the resultant of the gap vectors **G .**

Since we want to move the circle in the most optimum direction , we should allow the circle to move more in the direction of the gap minimising its overlap and hence the optimum diredtion of the circle is given by the resultant of the two vectors **G** + (**-O**) .

This optimum direction is evaluated for each of the individual circles and the circles are given a minute shift along their respective optimum directions . Thus in this process of covering the gap and minimizing the overlap , a situation may arise when all the circles are isolated from each other with the gap angle of individual circles being 360^0 . This corresponds to a situation when all the circles are existing inside the given field , without overlapping each other and also without overlapping any of the sides of the given field . This situation naturally suggests that the given field is big enough to accommodate all the circles without any overlap and hence all the circles placed in a non-overlapping manner are not adequate to cover the given field entirely . At this point we increase the radius of all the circles by the same amount keeping their respective positions fixed .

Some situations may arise when the gap across each of the circles is less than 360^0 i.e. all the circles are not completely isolated whereas the circles tend to oscillate about their respective positions without any resultant move in any of the directions and so is the resultant gap across the circles . This type of situation prevents the termination of the algorithm but compels to run it in an infinite loop . Such types of deadlocks are handled by keeping track of the radius for a fixed number of successive iterations .If the radius does not change over that span for iterations , the radius is deliberately increased by a small percentage so as to restore the normal flow of the algorithm .

The above procedure is carried on iteratively until the centers of all the circles lie inside the given field and the sum of the gaps across the circles tend to zero. The radius thus obtained gives the optimum radius for that many number of circles covering the given field .

The initial distribution as well as the starting radius is not taken arbitrarily . If A be the given area of the square field and N be the number of circles , then we try find the area of N number of equal squares those can be fitted inside the given square . If N be a perfect square then we get a square array of small squares where each of the small square has an area given by A/N and side is equal to $\sqrt{A/N}$. Thus we get an array of size $\sqrt{N}$ X $\sqrt{N}$ consisting of small subsquares . In case when N is not a perfect square we cannot place the small squares in a square shaped array . We start with the initial value of a side equal to $\sqrt{A/N}$ and continue fitting the squares row wise . If all the squares get fitted into the given square then we get an array where the number of squares in the last row is different from the previous rows . If the squares do not get fitted initially then we try with a lesser value of the side shrinking the area of the square by a small percentage . We continue this process iteratively till all the squares get fitted inside the given squares . The centres of these N squares are taken as initial positions of the circles and the radius of the incircle of a small square is taken as the starting radius of the algorithm . This radius gives a reasonable lower bound to start with.

Data Structure and Algorithm :

Data Structure :

1> To store the details of the circles we have used the following linked list :

```
struct circ{
  int sln;                /* serial number of the circle */
  float cx;               /* abscissa of the center */
  float cy;               /* ordinate of the center */
  struct circ *next;
};
```

2> The information regarding the neighbours are stored using two linked lists :

```
a> struct nebr_det{
      int sln ;         /* serial number of the neighbouring circle */
      float x1;         /* coordiantes
      float y1 ;           of the
      float x2;            points of
      float y2 ;           intersection */
      float area ;      /* magnitude of the overlapping area */
      float angle ;     /* direction of overlapping vector */
      struct nebr_det *next;
};
```

```
b> struct nebr{                   /* This structure stores the serial number of the
      int sln ;                      circle and details of its neighbours in another
      struct nebr_det *next;         list pointed by the link */
      struct nebr*next;
};
```

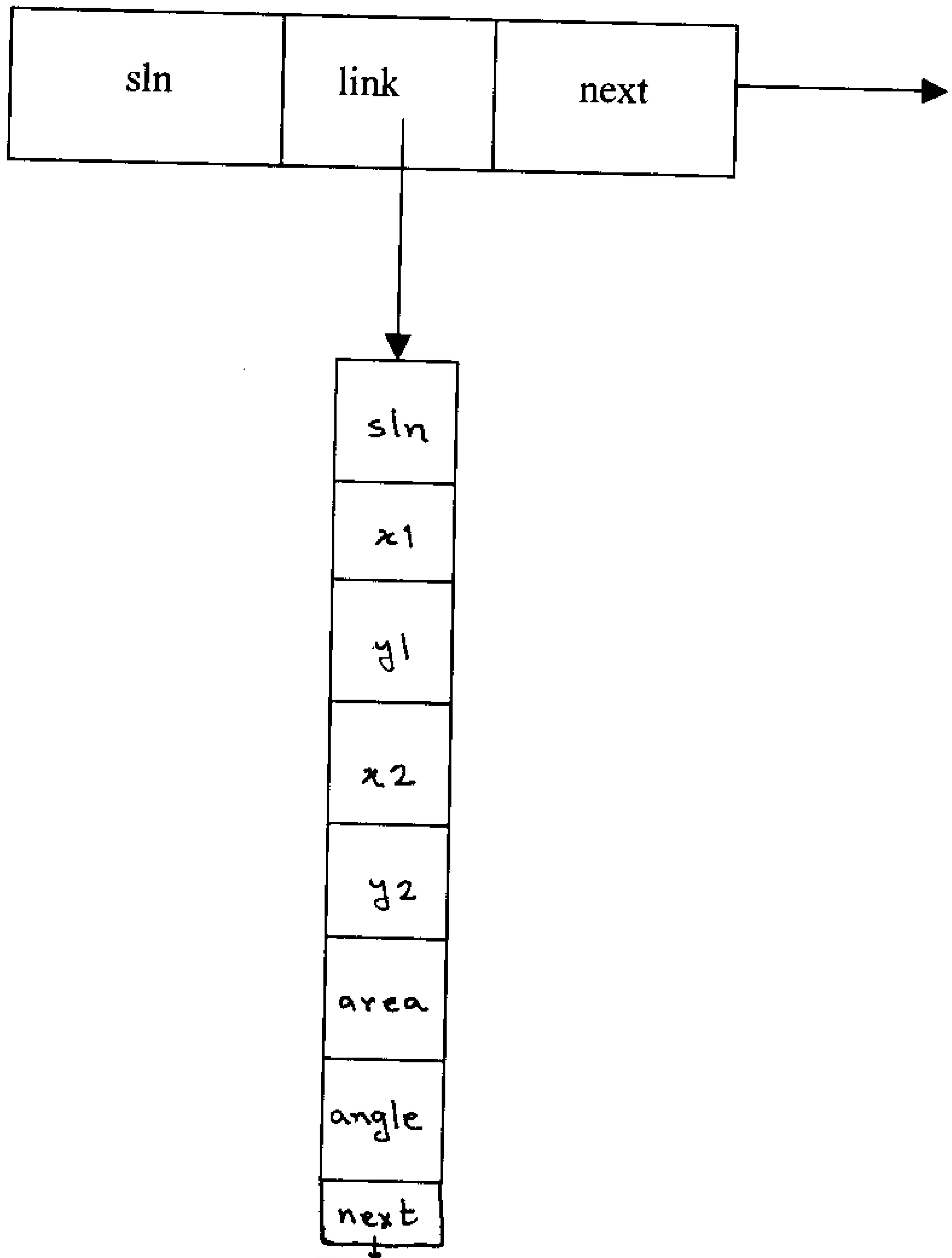


Fig. 3

3> The information regarding overlap of the circles are stored using two linked lists :

a>
```
struct angle{
    float st_angle ;        /* starting angle of the overlap */
    float end_angle ;       /* end angle of the overlap */
    struct angle *next;
};
```

b>
```
struct overlap{             /* This structure stores the list of overlap
    int sln ;                  along its circumference pointed by link and the
    struct angle *link ;       sum of the overlap in the field sum */
    float sum;
```

```
        struct overlap*next;
};
```

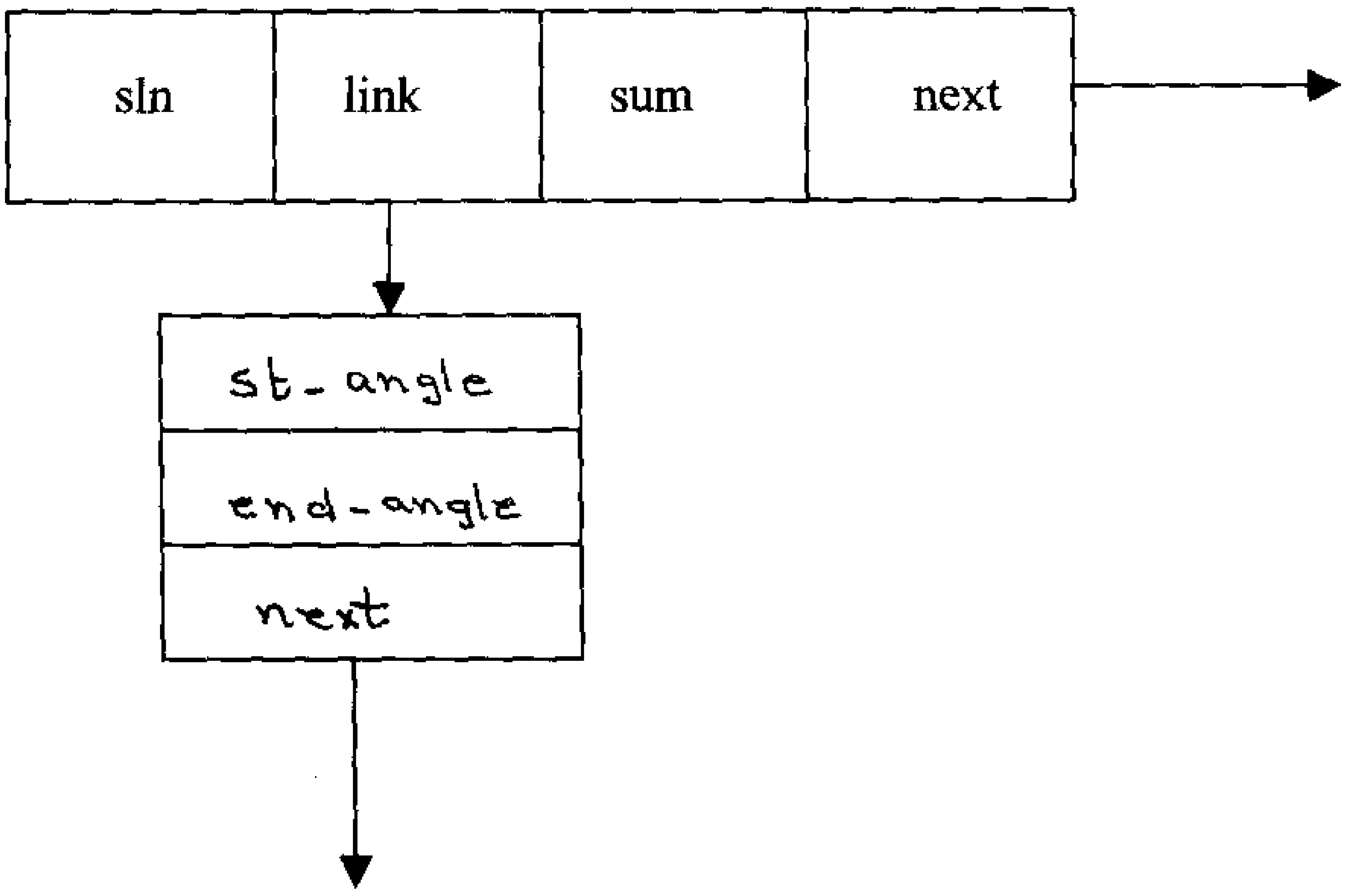


Fig. 4

4> The information regarding the gaps along the circumference of the circle is stored in two linked lists similar to overlap :

```
a> struct g_ ang{
       float s_ang;          /* starting angle of the gap  */
       float e_ang;          /* end angle of the gap  */
       float value;          /* magnitide fo the gap */
       float angle ;         /* direction of the gap vector */
       struct g_angle *next;
};
```

```
c> struct gap{                        /* This structure stores the list of gap
        int sln;                      along its circumference pointed by link and the
        struct g_ang *link;             sum of the gap in the field sum */
        float sum;
        struct glap*next;
};
```

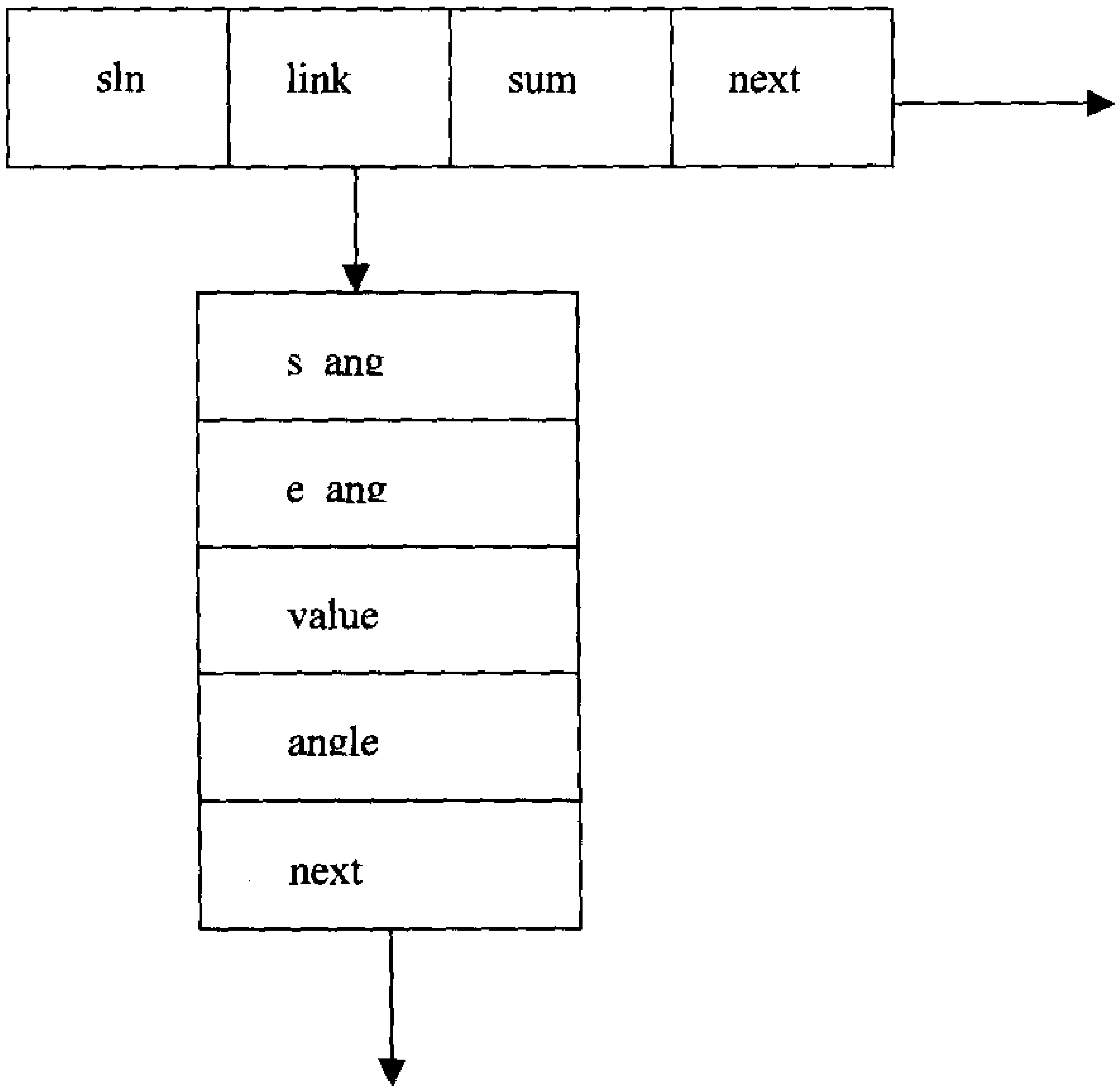


Fig . 5

Algorithm :

Step 1 : Get the number of circles ' n ' and the boundaries of the field .

Step 2 : Evaluate the area of n squares iteratively which can be fitted inside the the given field in the formm of an array with the last row probably having a fewer number of small squares as compared with the previous rows .

Step 3 : List the centers of these squares as the centers of the circles and radius of incircle of the square as the radius of the circle to start with .

Step 4 : Create a list of neighbours of each circle containing the coordinates of intersection .

Step 5 : Create a list of overlap of each circle containing the area and direction of overlap with respective circle serial number .

Step 6 : Evaluate , for each circle , whether it intersects any of the sides of the given square field . If it intersects , then append the overlap list created in step 5 with this overlap magnitude and direction .

Step 7 : Create a list of gap of each circle containing gap magnitude and direction .

Step 8 : For each circle find the sum of the gap across it and hence evaluate the total gap sum .

Step 9 : If the total gap sum tends to zero and any of the center goes outside the Boundary then go to step 12 . Otherwise , go to step 10.

Step10 : For each circle find the negative resultant overlap vector as well as the Resultant gap vectors . Add these two vectors to find the resultant direction And shift the circle in this direction of the resultant .

Step11: Go to step 4 .

Step12 : Stop .

Experimental Results :

Number of Circles	Observed Radius	Standard Results	Error (%)
1	0.708301	0.707106	0.168
2	0.586293	0.559016	4.87
3	0.539043	0.503891	6.9
4	0.354151	0.353553	0.169
5	0.340942	0.326160	4.8
6	0.344859	0.298729	15.4
7	0.344812	0.274291	25.7
8	0.266021	0.260300	2.19
9	0.291069	0.230636	26.2
10	0.241686	0.218233	10.7
11	0.242567	0.212516	14.1
12	0.211176	0.202575	4.2
13	0.229499	0.194312	18.1
14	0.201093	0.185510	8.4
15	0.199322	0.179661	10.1
16	0.177075	0.169427	4.5
17	0.176980	0.165680	6.8
18	0.176462	0.160639	9.8
19	0.159834	0.157841	1.26
20	0.159834	0.152246	4.9
21	0.160034	0.148953	7.4
22	0.160417	0.143693	11.6
23	0.160967	0.141244	13.9
24	0.178242	0.138302	28.8
25	0.141660	0.133548	6.07
26	0.133174	0.131764	1.07
27	0.127695	0.128633	0.7
28	0.128652	0.127317	1.04
29	0.126414	0.125553	0.68
30	0.121446	0.120865	0.48

Conclusion :

In this report we identify a new method of repositioning equal circles within a given square so as to achieve minimum radius , but still covering the entire square . Although the results cannot claim to have achieved the minimum radius but the results are encouraging with respect to time complexity . The major drawback as identified by us is that our algorithm moves the circle based on local optimum direction and not on global consideration . The results might have improved had we considered movements of few circles within elementary strips (both horizontally and vertically) of the given square region , instead of moving all the circles in one run . The problem could have been tried with Voronoi diagram approach considering a random initial distribution of circles and then trying to partition the given field into Voronoi Cells iteratively till the centers of all the cells converge in successive iterations . But this approach also suffers from the same drawback of local optimization . However , our work leaves a scope of further improvement of the results using the vector approach as suggested by us .

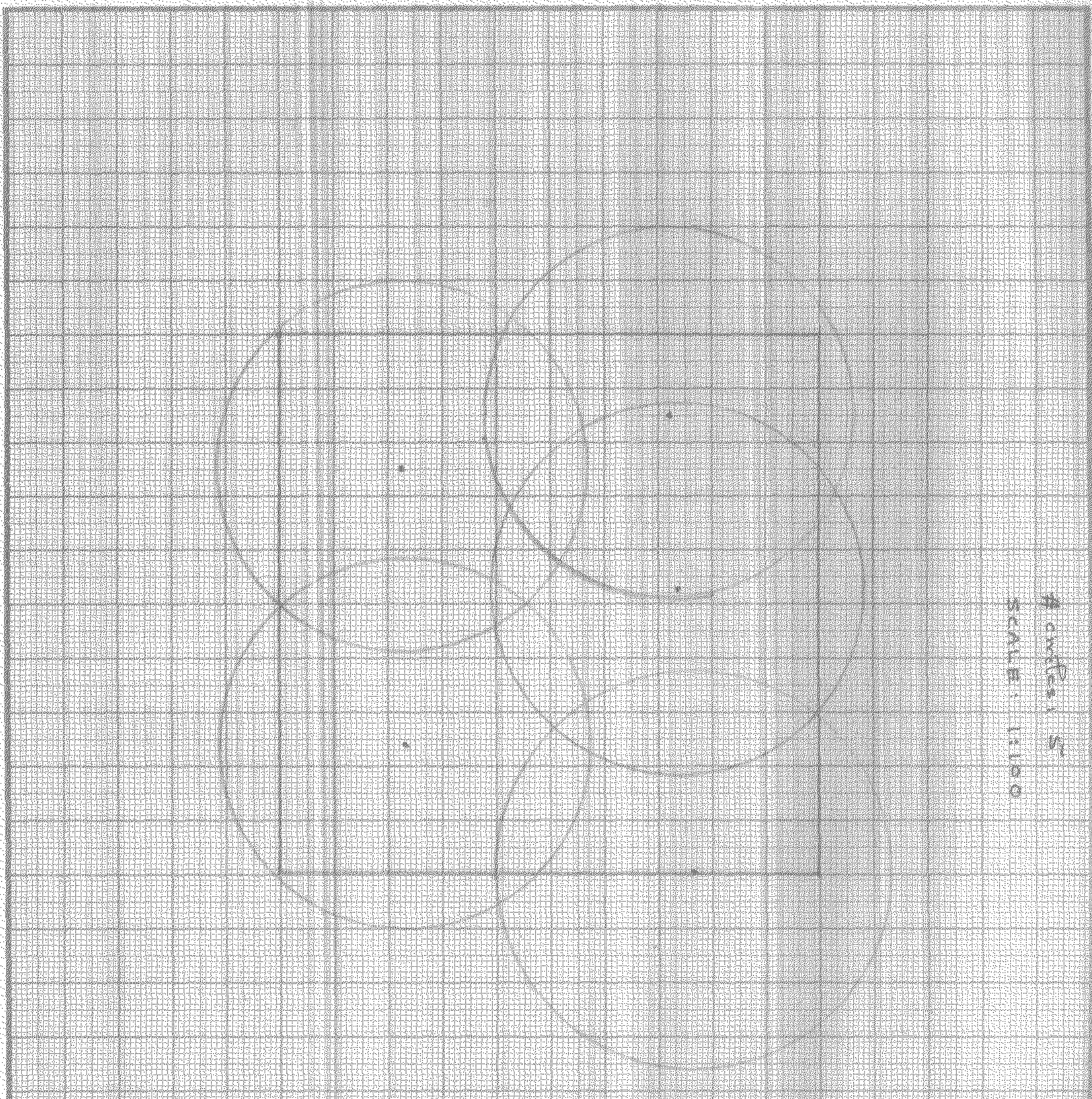
References :

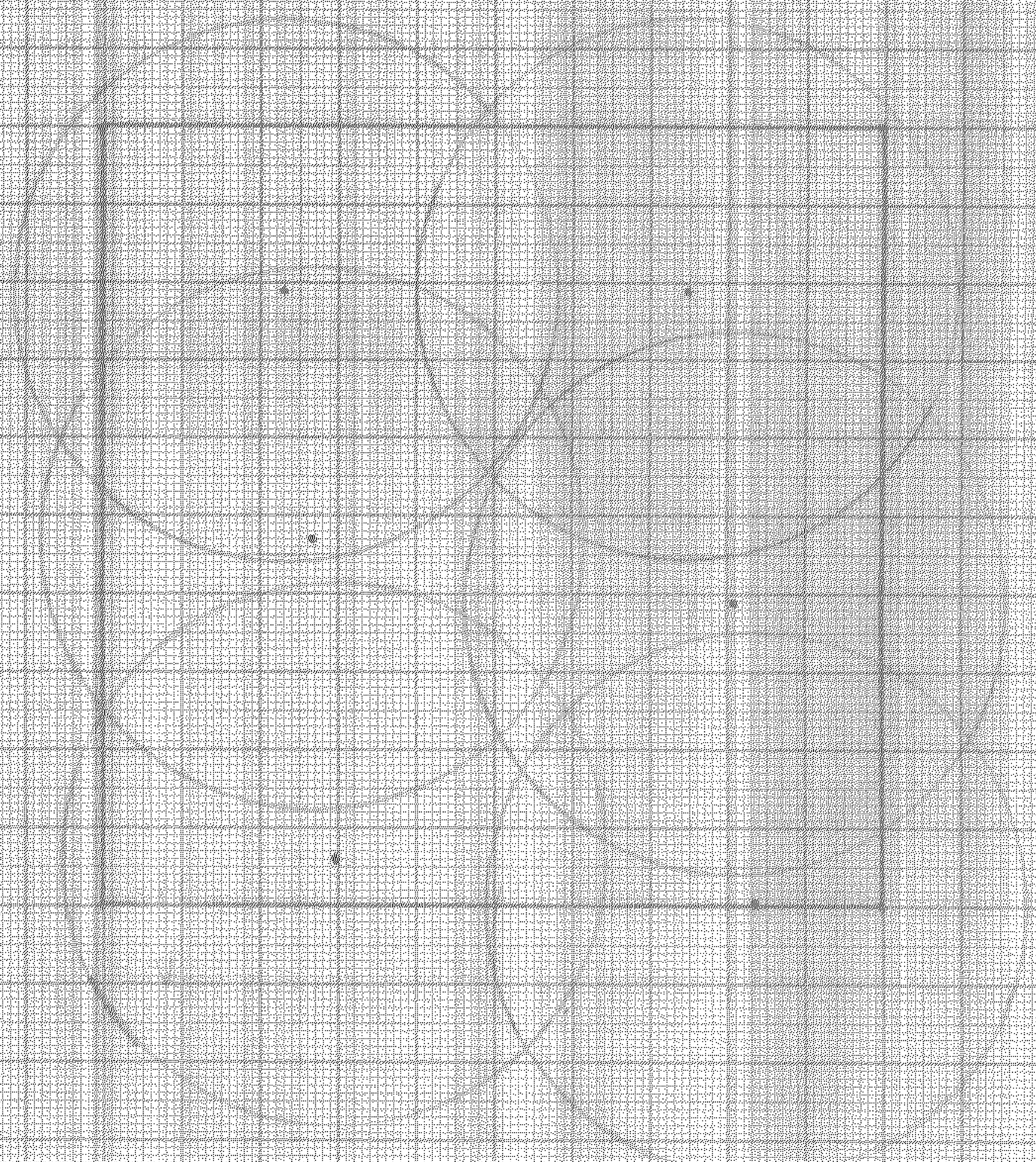
[1] Kari J. Nurmela and Patric R.J. Ostergard , Helsinki University of Technology , Covering a square with up to thirty equal circles .

[2] A . Heppes and H. Melissen , Covering a rectangle with equal circles , Period Math. Hungar 34 (1997) , 65 – 68 .

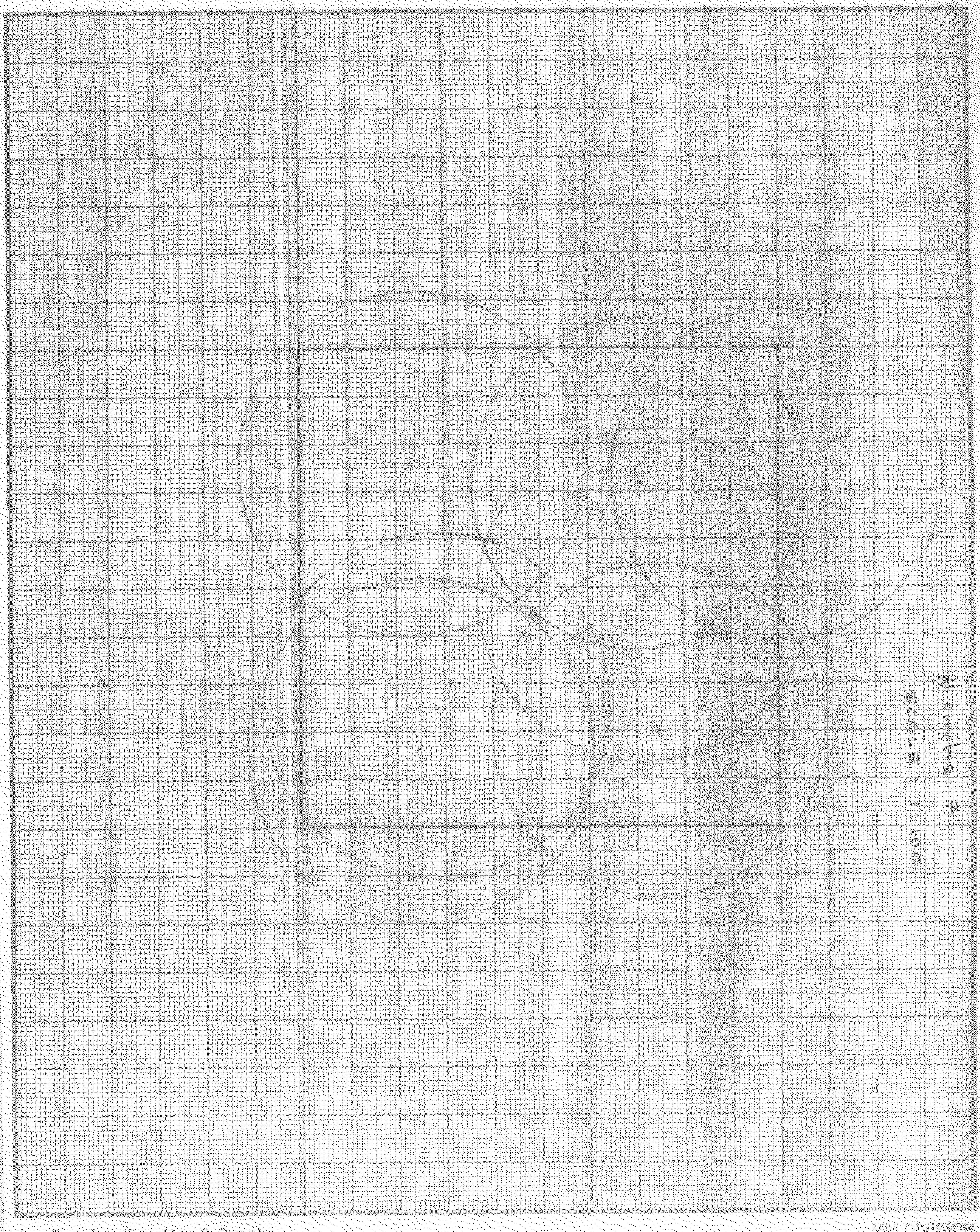
[3] J.B.M. Melissen and P.C. Schur , Improved coverings of a square with six and eight Equal circles , Electron J. Combin 3 (1996) R32 , 10 pp (electronic).

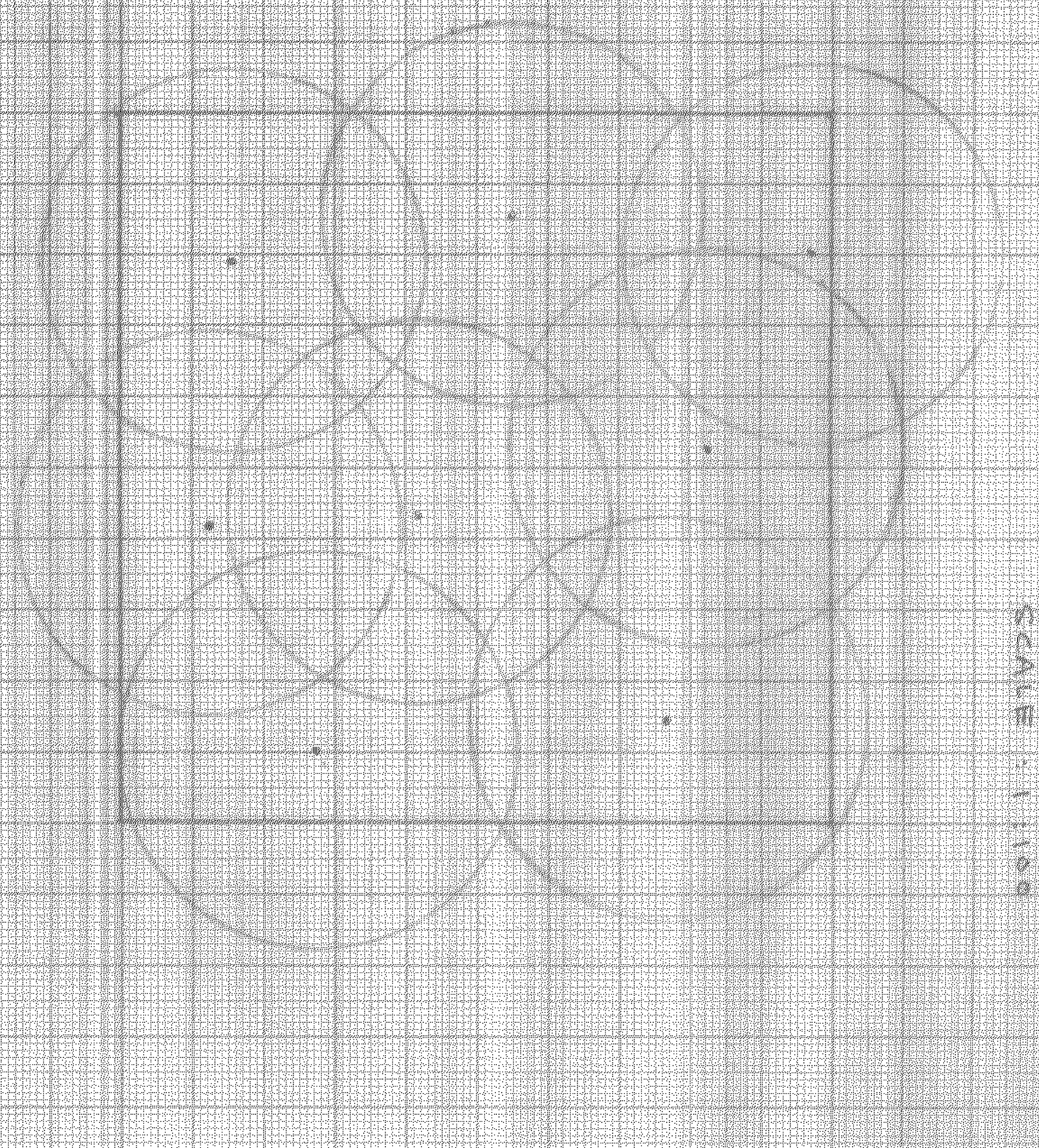
[4] J.B.M. Melissen and P.C. Schur , Covering a rectangle with six and seven circles , Discrete Appl. Math 99 (2000) , 149 – 156.

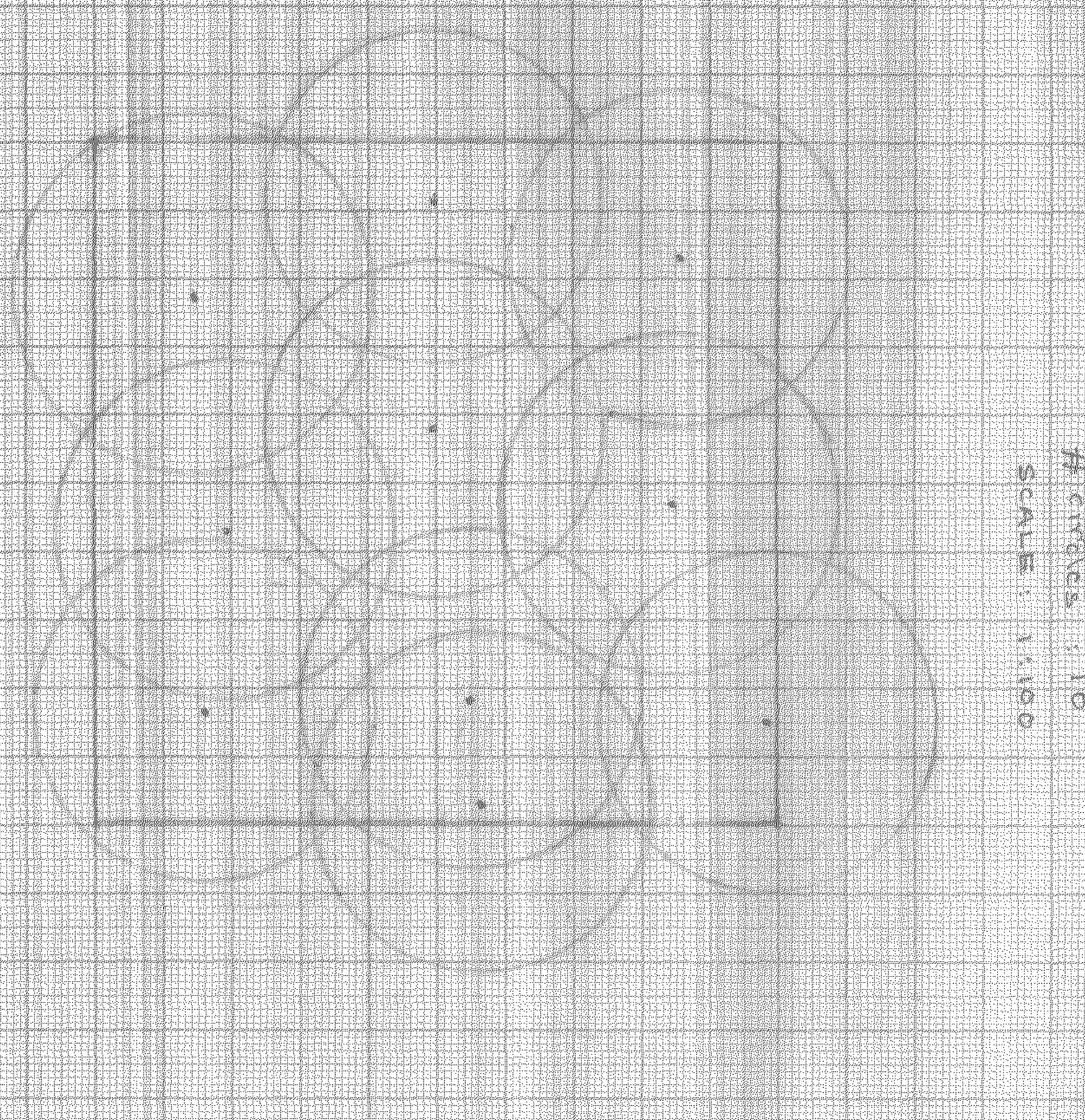

circles: 5
SCALE: 1:100



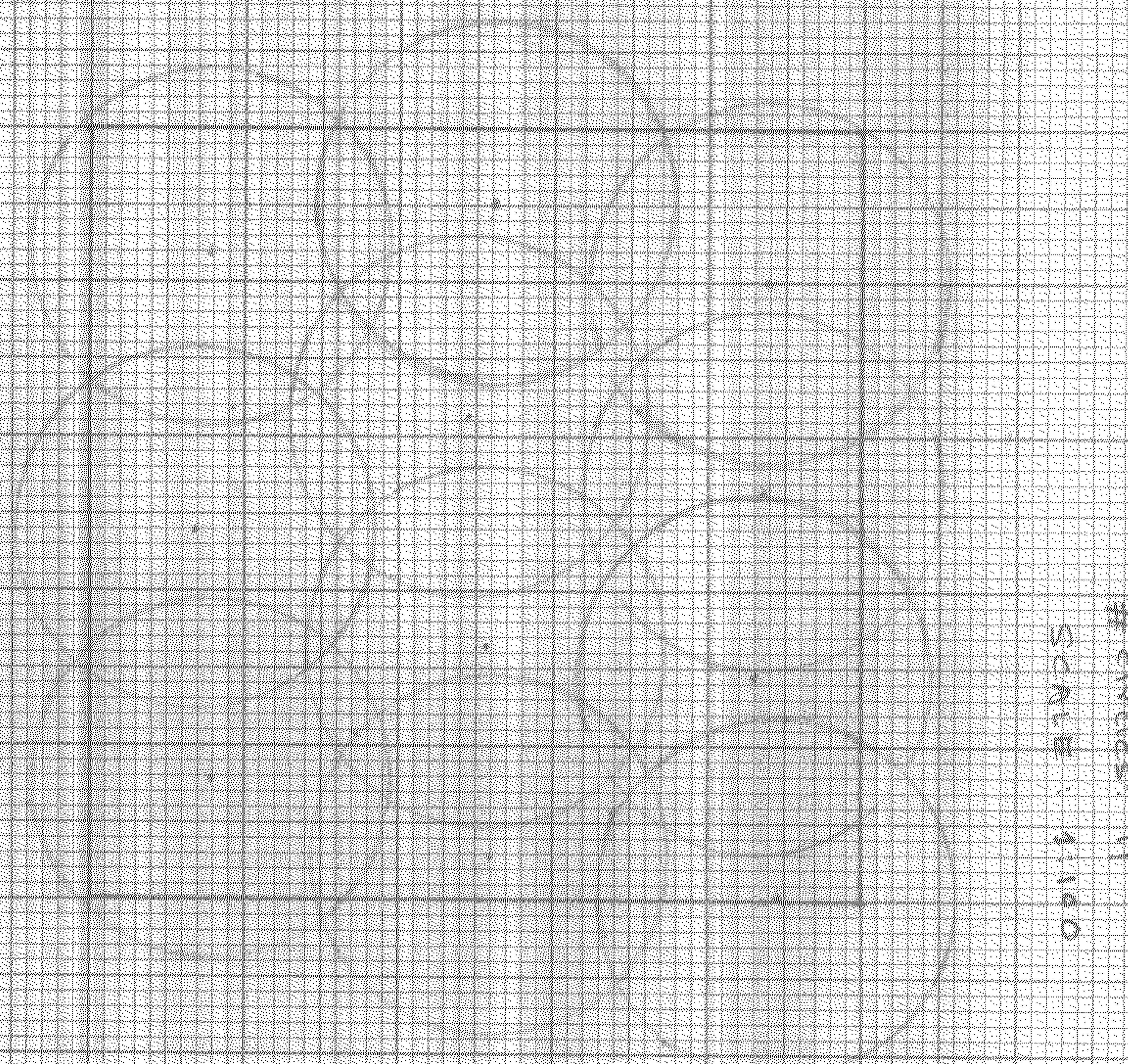
circles: 6
SCALE 1:100


circles 7
SCALE : 1:100


#circles: 8
SCALE: 1:100



circles : 10
SCALE : 1:100


circles: 11
SCALE : 1:100