Disc Covering Problem with Application to Digital Halftoning

Tetsuo Asano[(1)], Peter Brass[(2)], and Shinji Sasahara[(3)]

(1) School of Information Science, JAIST, 1-1 Asahidai, Tatsunokuchi, Ishikawa, 923-1292 Japan. t-asano@jaist.ac.jp

(2) Department of Computer Science, City College, CUNY, Convent Avenue at 138th Street, New York, NY-10031,USA. peter@cs.ccny.cuny.edu

(3) Fuji Xerox Co., Ltd., 430 Sakai, Nakai, Ashigarakami, Kanagawa 259-0157, Japan.

Abstract

This paper considers the following geometric optimization problem: Input is a matrix $R = (r_{ij})$. Each entry r_{ij} represents a radius of a disc with its center at (i, j) in the plane. We want to choose discs in such a way that the total area covered by exactly one disc is maximized. This problem is closely related to digital halftoning, a technique to convert a continuous-tone image into a binary image for printing. An exact algorithm is given for the one-dimensional version of the problem while approximation algorithms are given for the two-dimensional one. The approximation algorithms are verified to be satisfactory in practice through experiments in applications to digital halftoning.

1 Introduction

One of the popular geometric optimization problems is that of finding the maximum radius r_n of n equal and non-overlapping discs to be packed in a unit square. It has been widely explored with a number of surprising results (see e.g. [2]). It is equivalent to that of scattering n points into the unit square such that the minimum pairwise distance becomes as large as possible.

The problem to be considered in this paper is similar to the above-stated problems but different in many ways. In our case possible locations of disc centers are restricted to predefined lattice points. Furthermore, the radii of discs are given as a matrix. The problem is to choose discs so that the total area covered by exactly one disc is maximized.

This problem seems to be computationally hard although no proof of its NP-hardness is known. In this paper we first consider the one-dimensional version of the problem and give a polynomial-time algorithm. Then, we propose some approximation algorithms with theoretical guarantee on their performance ratios.

This problem originates from an application to digital halftoning, a technique to convert continuous-tone images into binary images for printers. Theoretically speaking, the approximation ratio of our proposed algorithm is not so attractive, but it works quite satisfactorily in practice, which is verified through experiments on its applications to digital halftoning of various images. Our goal for the application to digital halftoning is to distribute points so that the Voronoi diagram associated with them contains no skinny Voronoi region. In this sense the problem is similar to mesh generation, in which the interior of a given polygonal region is partitioned into simplicies to avoid skinny simplicies. For that purpose some part may be partitioned into small regions. In our case no polygon is given and the sizes or areas of simplices (convex polygons) are determined by spatial frequency of an image. The idea of using Voronoi diagram for designing screen elements is not new. In fact, it is seen in the literatures [3] and [4], in which Voronoi diagram is used to generate non-regular arrangement of screen elements. The first and third authors [5] formulated the problem as a disc covering problem based on spatial frequency information of an image and

presented a heuristic algorithm based on bubble-packing algorithm. It is an iterative improvement algorithm and took much time before convergence. This paper achieves efficient implementation while keeping the quality of the resulting Voronoi diagram and output images.

This paper is organized as follows: Section 2 describes a motivation of this study and applications to digital halftoning. Section 3 gives a formal definition of the geometric optimization problem. Section 4 presents an effective algorithm for the same problem in the one dimension. Sections 5 and 6 include some approximation algorithms with bounded performance ratios and practical heuristics with some experimental results. Finally, concluding remarks and some future plans are given in Section 7.

2 Motivation and Application

Digital halftoning is a technique to convert a continuous-tone image into a bi-level image for output on bi-level printing devices. Conventional halftoning algorithms are classified into two categories depending on resolution of printing devices. In a low-resolution printer such as an ink-jet printer individual dots are rather clearly separated. On the other hand dots are too small in a high-resolution printer such as off-set printer to make fine control over their positions. Therefore, dots should form clusters whose sizes are determined by their corresponding intensity levels. Such a halftoning algorithm is called a cluster-dot halftoning.

This algorithm consists in partitioning the output image plane into repetitive polygons called screen elements, usually of the same shape such as rectangles or parallelograms. Each screen element is then filled in by dots according to the corresponding intensity levels. Dots in a screen elements are clustered around some center point to form a rounded figure. Denoting by k the area or the number of pixels of a screen element, only $k+1$ different intensity levels instead of 2^k levels are reproduced since the gray level in a screen element is determined only by the number of dots in the region. So, large screen element is required to have effective tone scale. On the contrary the size of a screen element should be small for effective resolution. This suggests a serious tradeoff between effective resolution and effective tone scale. The algorithm to be proposed in this paper resolves it by introducing adaptive mechanism to determine cluster sizes.

In most of the conventional cluster-dot halftoning algorithms the output image plane is partitioned into screen elements in a fixed manner independent of given input images. A key idea of our algorithm is to partition the output plane into screen elements of various sizes to reflect spatial frequency distribution of an input image. This adaptive method is a solution to balance effective resolution and effective tone scale in the following sense. The two indices are both important, but one is more important than the other depending on spatial frequency distribution of an input image. That is, resolution is more important in a high-frequency part to have a sharp contour, so that the sizes of screen elements should be kept small. On the other hand, tone scale is more meaningful in a low-frequency part with intensity levels changing smoothly, and so larger sizes of screen elements are preferred. All these requirements suggest the following geometric optimization problem. Given a continuous-tone image A and a scaling factor to define the size of an output image, we first compute spatial frequency distribution by applying Laplacian or Sobel [6] differential operator. Then, each pixel in the output image plane is associated with a disc of radius reflecting the Laplacian value at the corresponding point. Now, we have a number of discs of various radii. Then, the problem is to choose a set of discs to cover the output plane in an optimal way. The optimality criterion should reflect how large area is covered by exactly one disc from the set, which implies minimization of the area of unoccupied region and intersection among chosen discs to make the resulting screen elements rounded figures.

3 Problem Formulation

This section gives a formal definition of the problem. Input is an $M \times N$ matrix $R = (r_{ij})$, $0 \leq i < M, 0 \leq j < N$ of positive real numbers. Each matrix entry r_{ij} specifies a radius of a disc to be placed at the corresponding position (i, j) in the plane. A matrix B of the same size is a binary matrix to be calculated. Each binary value b_{ij} is 1 when a disc of radius r_{ij} is placed at position (i, j) in the plane and 0 otherwise. Given two matrices R and B, the area covered by exactly one accepted disc, which is denoted by $g(R, B)$, is our objective function to be maximized. In other words, given a matrix R, we want to find a binary matrix B that maximizes the objective function $g(R, B)$.

We could also consider the problem by replacing discs by squares. Then, each r_{ij} represents a side length of a square to be placed at (i, j). This problem is similar to that of packing n equal discs of largest possible radius in a unit square and to that of covering a unit square by n equal discs of smallest possible radius. In fact, if all the input values r_{ij} are equal, it becomes a discrete decision problem of the above problems in such a sense that disc center locations are restricted to grid points. The computational hardness of the problem is well recognized in the literature, but it is still open whether this problem is NP-hard or not.

4 An Efficient Algorithm for 1-D Version of the Problem

4.1 Graph-based approach

Let us first consider the one-dimensional version of the problem. We will show that an optimal solution is found in polynomial time in this case. We give an algorithm by reducing the problem to a longest path finding problem on a directed acyclic graph, which runs in $O(n^2)$ time and space. Later on, the space complexity will be improved. The idea of the algorithm is utilized in an approximation algorithm for the two-dimensional problem.

Let $R = \{r_1, r_2, \ldots, r_n\}$ be the set of n positive real numbers. For each r_i, we define an interval $I_i = [s_i, t_i]$ where $s_i = i - r_i$ and $t_i = i + r_i$, referred to as the left and right endpoints of the interval I_i. Let $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ be a set of all such intervals. For a subset $\mathcal{I}'$ of $\mathcal{I}$ we define its gain $g(\mathcal{I}')$ by

$$g(\mathcal{I}') = \text{the total length of intervals covered exactly once by intervals in } \mathcal{I}'. \tag{1}$$

Given a set $\mathcal{I}$ of n intervals, our objective is to find a subset $\mathcal{I}^*$ of $\mathcal{I}$ maximizing the gain, the total length covered exactly once.

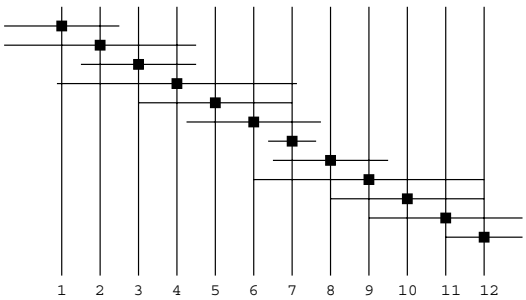

Figure 1: An example of the problem: $r_1 = 1.5, r_2 = 2.5, r_3 = 1.5, r_4 = 3.1, r_5 = 2.0, r_6 = 1.8, r_7 = 0.7, r_8 = 1.6, r_9 = 3.0, r_{10} = 2.0, r_{11} = 2.0, r_{12} = 1.0$.

An example is shown in Fig. 1 with 12 intervals. Of course, there are 2^{12} different choices of intervals. When we choose three intervals $I_2 = [s_2, t_2], I_6 = [s_6, t_6]$, and $I_{10} = [s_{10}, t_{10}]$, shown in

bold lines in Fig. 2, the interval $[s_6 = 4.2, t_2 = 4.5]$ is doubly covered , and $[t_6 = 7.8, s_{10} = 8.0]$ and $[t_{10} = 12, t_{12} = 13]$ are empty. The remaining part is covered exactly once. Thus, the gain of this subset is given by $2r_2 + 2r_6 + 2r_{10} - 0.3 * 2 = 5 + 3.6 + 4 - 0.6 = 12$.

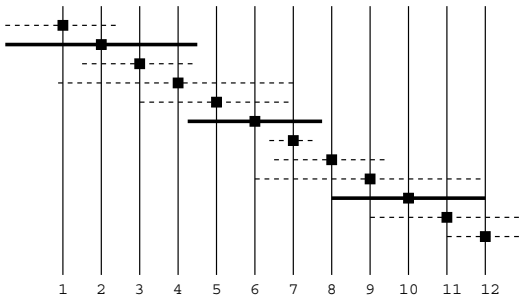

Figure 2: An example of a subset $\mathcal{I}'$ of intervals where bold lines indicate those intervals in the subset.

The following is a key observation to an efficient algorithm for the problem.

Lemma 1 *For any set $\mathcal{I}$ of intervals there is an optimal subset $\mathcal{I}^*$ of $\mathcal{I}$ such that no point is covered by three intervals from $\mathcal{I}^*$.*

Proof Suppose a point x is contained in three intervals, I_u, I_v and $I_w \in \mathcal{I}^*$. If one of them is properly included in another, then we can safely remove the shorter interval without decreasing the total gain. So, assume no inclusion relationship among the three intervals. Then, all the interval endpoints are distinct, and hence one of the intervals must be contained in the union of the other two intervals since at most two intervals can contribute to the left and right endpoints of the union. This implies that we can remove the interval contained in the union from $\mathcal{I}^*$ without decreasing the total gain since the union of those intervals is preserved. □

A set $\mathcal{I}$ of intervals is called an *at most doubly overlapping interval set* if no three of them have non-empty intersection, or any point is covered by at most two intervals in $\mathcal{I}$. The lemma 1 guarantees that restriction to at most doubly overlapping interval sets does not lose all of optimal solutions. An at most doubly overlapping interval set can be expressed as a sequence of intervals $(I_{\sigma(1)}, I_{\sigma(2)}, \ldots, I_{\sigma(k)})$ such that

$$s_{\sigma(1)} < s_{\sigma(2)} < \cdots < s_{\sigma(k)}$$
$$I_{\sigma(i)} \cap I_{\sigma(j)} = \emptyset \text{ if } |i - j| \geq 2,$$

that is, one interval $I_{\sigma(i)}$ possibly overlaps the next interval $I_{\sigma(i+1)}$ and the previous $I_{\sigma(i-1)}$, but no others.

Lemma 2 *When an at most doubly overlapping interval set $\mathcal{I}$ is given as a sequence of intervals $(I_{\sigma(1)}, I_{\sigma(2)}, \ldots, I_{\sigma(k)})$, then the gain of $\mathcal{I}$ is given by*

$$g(\mathcal{I}) = \sum_{i=1}^{k} |I_{\sigma(i)}| - 2 \sum_{i=1}^{k-1} |I_{\sigma(i)} \cap I_{\sigma(i+1)}|, \tag{2}$$

where $|I_{\sigma(i)}|$ is the length of interval $I_{\sigma(i)}$, i.e., $|I_{\sigma(i)}| = t_{\sigma(i)} - s_{\sigma(i)}$, and $|I_{\sigma(i)} \cap I_{\sigma(i+1)}|$ is that of the intersection of the two consecutive intervals.

Proof The total length of the union of all intervals is given by $\sum_{i=1}^{k} |I_{\sigma(i)}| - \sum_{i=1}^{k-1} |I_{\sigma(i)} \cap I_{\sigma(i+1)}|$. Since the intersection of consecutive intervals should be excluded from the singly-covered region, we have to reduce its length by $\sum_{i=1}^{k-1} |I_{\sigma(i)} \cap I_{\sigma(i+1)}|$. □

Now we can reduce our problem to that of finding a maximum-weight path in a directed acyclic graph defined as follows: Given a set $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ of intervals, an interval traversing graph $G = (V, E, W)$ has vertices corresponding to those intervals and two special vertices s and t. Edge set is defined as follows.
(1) $(I_i, I_j) \in E,\ I_i, I_j \in \mathcal{I}$ if and only if (i) $i < j$, (ii) $j > t_i$, (iii) $i < s_j$, and (iv) there is no I_k such that $i < k < j$ and $I_i \cap I_k = I_k \cap I_j = \emptyset$.
(2) $(s, I_i) \in E, I_i \in \mathcal{I}$ if and only if there is no I_j such that $j < i$ and $I_i \cap I_j = \emptyset$, and
(3) $(I_i, t) \in E, I_i \in \mathcal{I}$ if and only if there is no I_j such that $j > i$ and $I_i \cap I_j = \emptyset$.

Edge weights are defined as follows.
(4) $w(I_i, I_j) = |I_i| - 2|I_i \cap I_j|$ for each $(I_i, I_j) \in E$,
(5) $w(s, I_i) = 0$, and
(6) $w(I_i, t) = |I_i|$.

An example of an interval traversing graph is given in Fig. 3. A longest $s - t$ path $s \to I_1 \to I_2 \to I_4 \to t$, which corresponds to a set of $\{I_1 = [0.5, 1.5], I_2 = [1.3, 2.7], I_4 = [2.8, 5.2]\}$. The singly covered intervals are $[0.5, 1.3], [1.5, 2.7], [2.8, 5.2]$. The total length of the intervals is $0.8 + 1.2 + 2.4 = 4.4$, which equals the length of the path $0 + 0.6 + 1.4 + 2.4 = 4.4$.

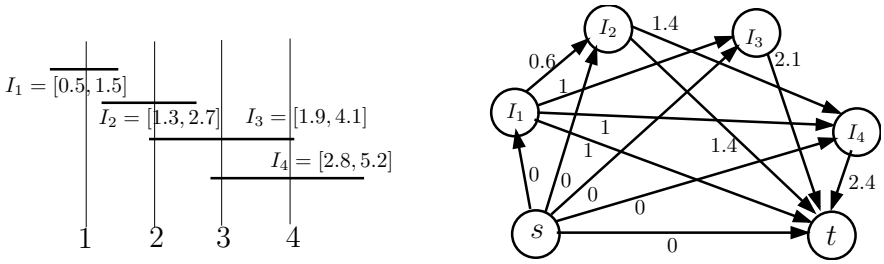

Figure 3: An example of an interval traversing graph.

Lemma 3 *Let $r_{\max}$ be the maximum among $\{r_1, r_2, \ldots, r_n\}$. Then, outgoing degree of a vertex in an interval traversing graph G associated with a set of intervals defined by $\{r_1, r_2, \ldots, r_n\}$ is at most $3\lceil r_{\max} \rceil + 1$.*

Proof Let I_i be any vertex. We know that $(I_i, I_j) \notin E$ if (1)$j \le t_i$ or (2) $j > t_i + 3\lceil r_{\max} \rceil$. In case (1), $j \le i$ or $i < j \le t_i$. So, if $t_j \ge t_i$ then the right half of I_i (the part in the interval $[i, t_i]$) is included in $I_i \cap I_j$. If $t_j < t_i$ then the left half of I_j is contained in $I_i \cap I_j$. Therefore, in either case we have $|I_i \cap I_j| \ge 1/2 \min\{|I_i|, |I_j|\}$. In case (2), even if the interval $I_j, j = \lceil t_i \rceil + 3\lceil r_{\max} \rceil$ is longest, i.e., $r_j = r_{\max}$, there is a large gap between the right endpoint t_i of I_i and the left endpoint s_j is to include an interval since their gap is longer than $2r_{\max}$. There are at most $3\lceil r_{\max} \rceil$ integers between t_i and the center point $j = \lceil t_i \rceil + 3\lceil r_{\max} \rceil$. Only those intervals defined for such integers plus the special vertex t can be the destinations of edges outgoing from the vertex I_i. Thus, the lemma follows. □

The largest value $r_{\max}$ can be assumed to be at most n since otherwise the problem becomes trivial. If $r_{\max} = O(n)$ then the number of edges is $O(n^2)$. In many practical cases $r_{\max}$ is a constant independent of n and then we have only linear number of edges.

Let $\mathcal{I}'$ be a subset of the whole interval set $\mathcal{I}$. $\mathcal{I}'$ is *redundant* if there is an interval $I_i \in \mathcal{I} \backslash \mathcal{I}'$ such that I_i does not intersect any interval in $\mathcal{I}'$. Obviously, if $\mathcal{I}'$ is redundant then $\mathcal{I}'$ is not optimal since we can increase its gain by inserting an interval intersecting no interval in $\mathcal{I}'$. $\mathcal{I}'$ is also redundant if it contains two intervals I_i and I_j such that $|I_i \cap I_j| \ge 1/2 \min\{|I_i|, |I_j|\}$. The condition (1) in the definition of the interval traversing graph guarantees that an edge (I_i, I_j) is defined only if it is not redundant. When $i < j$, $|I_i \cap I_j| \ge 1/2 \min\{|I_i|, |I_j|\}$ if and only if $j > t_i$ and $i < s_j$, that is, $j > i + r_i$ and $i < j - r_j$.

Lemma 4 *Let $\mathcal{I}$ be a given set of intervals and G be its associated interval traversing graph. Then, there is one-to-one correspondence between directed paths from s to t in G and at most doubly overlapping non-redundant interval sets. Furthermore, the sum of edge weights of such a path coincides with the gain of the corresponding set of intervals.*

Proof Let $P = (s, I_{u_1}, I_{u_2}, \ldots, I_{u_k}, t)$ be any directed path from s to t in G. Then, $u_1 < u_2 < \cdots < u_k$ since $u_i > t_{u_{i-1}} > u_{i-1}$ for $i = 2, 3, \ldots, k$. The set of intervals $\{I_{u_1}, I_{u_2}, \ldots, I_{u_k}\}$ is an at most doubly overlapping interval set. If three intervals have common intersection, one of their center points must be in the union of the other two intervals and thus it must be in one of the other intervals, which cannot happen by the definition of the graph. The interval set is not redundant again by the definition.

The proof for the other direction is similar. Lastly we can observe that the sum of edge weights of P and the gain of the corresponding interval set are both given by

$$\sum_{i=1}^{k} |I_{u_i}| \quad - \quad 2\sum_{i=1}^{k-1} |I_{u_i} \cap I_{u_{i-1}}|.$$

□

Theorem 5 *Given a set of n intervals associated with n real numbers $r_1, r_2, \ldots, r_n$, an optimal subset can be found in time $O(nr_{\max})$ as a maximum-weight path in the corresponding interval traversing graph, where $r_{\max}$ is the largest among $r_1, r_2, \ldots, r_n$.*

Proof The lemma 4 guarantees that a maximum-weight path in the graph gives an optimal subset. Since the graph is a directed acyclic graph, such a path can be found in time linear in the number of edges, that is, in time $O(nr_{\max})$ by Lemma 3. □

The graph is built by checking at most $3\lceil r_{\max} \rceil + 1$ vertices at each vertex to define its outgoing edges, which is done in $O(nr_{\max})$ time and space.

One disadvantage of the above-described approach is high space complexity. The number of edges is $O(n^2)$. Fortunately, we can reduce the space complexity while keeping the running time. An idea for the efficiency is dynamic programming combined with plane sweep paradigm.

4.2 Plane sweep approach

Plane sweep is a powerful technique in computational geometry to convert two-dimensional problem to a sequence of one-dimensional problems. In our case we regard intervals as horizontal line segments and sweep the plane from left to right while keeping a set of horizontal line segments (intervals) intersecting a vertical sweep line and an optimal set of intervals to the left of the sweep line that maximizes the total length of singly-covered intervals. To simplify the notations, we assume that all the endpoints are different and rename the intervals so that $T = (t_1, t_2, \ldots, t_n)$ is the increasing order of right endpoints of n given intervals, that is,

$t_1 \leq t_2 \leq \cdots \leq t_n$.

At each right endpoint t_i we want to maintain an optimal set S_i of intervals that maximizes the total length of singly-covered intervals using the interval I_i. Let g_i be the gain (the total length of singly-covered intervals) of S_i, and let $\hat{g}_i$ be the maximum among $g_1, g_2, \ldots, g_i$. Then, our goal here is to compute the value $\hat{g}_n$.

Initially, we have $g_1 = \hat{g}_1 = t_1 - s_1$. Then, the sweep line proceeds to $t_2, t_3, \ldots, t_n$ in order. At the i-th event point t_i, there are two cases to consider depending on which set S_j should be combined with the current interval I_i to form an optimal set S_i. The set S_j is associated with the interval $I_j, j < i$, and t_j is the rightmost endpoint in S_j. The two cases are (1) $t_j < s_i$ and (2) $s_i < t_j < t_i$.

In the first case, the whole interval I_i contributes to the gain. So, the gain achieved is given by $g_j + t_i - s_i$.

In the second case, the contribution of I_i to the gain depends on the location of the second rightmost right endpoint t_k in S_j. If $t_k > s_i$ then the three intervals I_i, I_j, and I_k share the point s_i. Therefore, by Lemma 1 we can exclude the possibility to include these three intervals simultaneously. So, it suffices to consider the case where $t_k < t_i$. The gain achieved by incorporating I_i to S_j is calculated as follows. The interval $[t_j, t_i]$ is singly-covered. The one $[\max(s_i, s_j, t_k), t_j]$ used to be singly-covered, but it is now doubly covered due to the new interval I_i. Here, if $s_j > s_i$ then I_j is a subinterval of I_i, that is, $I_j \subseteq I_i$. Therefore, any set including both of them cannot be optimal. Also, if $t_k > s_i$ then either the three intervals have non-empty intersection or I_j is included in I_i. Accordingly, the gain in this case is given by

$g_j + t_i - t_j - (t_j - s_i) = g_j - 2t_j + t_i + s_i$.

Summarizing the above argument, we can compute the gain g_i by

$$g_i = \max \quad \{ \quad \max_{t_j < s_i}(g_j + t_i - s_i), \quad \max_{s_i < t_j < t_i}(g_j - 2t_j + t_i + s_i)\}.$$

The terms $t_i - s_i$ and $t_i + s_i$ above are independent of j. So, what we need is to find
(1) $j < i$ such that $t_j < s_i$ and g_j is largest, and
(2) $j < i$ such that $s_i < t_j < t_i$ and $g_j - 2t_j$ is largest.

(1) is easy because such g_j is given as $\hat{g}_k$ where t_k is the right endpoint just to the left of s_i, that is, $t_k < s_i < t_{k+1}$. To find an index such that $g_j - 2t_j$ is largest under the condition that $s_i < t_j < t_i$, we maintain a set of points defined by $(t_j, g_j - 2t_j), j = 1, 2, \ldots, i-1$ in such a data structure that we can find a highest point in the vertical region between s_i and t_i. One of such a data structure is a variation of a segment tree. The root node is defined by $[1, n]$ and its left and right subtrees are recursively defined for $[1, \lceil n/2 \rceil]$ and $[\lceil n/2 \rceil + 1, n]$. Leaves are $[1, 1], [2, 2], \ldots, [n, n]$. Initially, all the nodes have keys $-\infty$. Whenever we compute a gain g_j, we follow the path from the leaf $[j, j]$ to the root $[1, n]$ while updating their key values if $g_j - 2t_j$ is larger than their keys.

At the i-th endpoint t_i we want to find an index j such that $g_j - 2t_j$ is largest among those j such that $s_i < t_j < t_i$. Such a largest value is given as the largest key in the nodes associated with an interval $[k, i-1]$, where t_k is the endpoint to the right of s_i, that is, $t_{k-1} < s_i < t_k$. Obviously, we can update the date structure and answer the query in $O(\log n)$ time. Thus, the total running time is $O(n \log n)$. The space required by the data structure is $O(n)$.

Now the algorithm is obvious.

[Plane Sweep Algorithm]
(1) Sort the right endpoint in the increasing order:
Rename intervals so that the resulting sorted order is given as $t_1 < t_2 < \ldots < t_n$.
(2) $g_1 = \hat{g}_1 = t_1 - s_1$, and insert $(t_1, g_1 - 2t_1)$ into the data structure T.
(3) for $i = 2$ to n do
(3.1) Find an index j in T such that $g_j - 2t_j$ is largest among those satisfying $s_i < t_j$.
(3.2) $g_i = \max\{\hat{g}_{i-1} + t_i - s_i, g_j - 2t_j + t_i + s_i\}$.
(3.3) $\hat{g}_i = \max\{\hat{g}_{i-1}, g_i\}$, and insert $(t_i, g_i - 2t_i)$ into T.

Theorem 6 *The plane sweep algorithm finds an optimal solution in $O(n \log n)$ time and $O(n)$ space.*

5 2-Dimensional Problem: Problem Formulation and Approximation Algorithm

In this section we consider the original problem in the two dimensions. Let $R = (r_{ij})$ be a matrix of positive real numbers. For each (i, j) we define a disc C_{ij} of radius r_{ij} with its center at a lattice point (i, j). For a subset $\mathcal{S}'$ of the set $\mathcal{S}$ of all such discs we define its gain $g(\mathcal{S}')$ by

$$g(\mathcal{S}') = \text{the total area of regions covered exactly once by discs in } \mathcal{S}'. \tag{3}$$

Given a set $\mathcal{S}$ of discs, the problem is to find a subset $\mathcal{S}'$ of $\mathcal{S}$ maximizing the gain. This problem seems to be NP-hard although no proof is obtained.

Unfortunately, there is no such nice property stated in Lemma 1, that is, restriction to at most doubly overlapping disc sets may lose all of optimal solutions in this case. A simple example is shown in Fig. 4, in which three discs have intersection but an optimal solution contains all of them if there is no other disc. Our approximation algorithm has the performance ratio 5.83 and runs in $O(n \log n)$ time where $n = MN$ is the total number of discs.

To describe the algorithm we prepare some terminology. C_u denotes a disc with its center at point u. $r(C_u)$ denotes the radius of a disc C_u. By $R_\rho(C_u)$ we denote a disc obtained by contracting C_u by a factor of $\rho, 0 < \rho < 1$, and therefore $r(R_\rho(C_{ij})) = \rho r(C_u)$. The contracted disc $R_\rho(C_u)$ is called the *core* of the disc. We say that a disc C_v *violates* another disc C_u if C_v intersects the core of C_u and C_v *safely intersects* C_u otherwise. Fig. 5 shows two safely intersecting discs and their cores are painted dark.

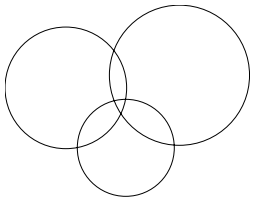

Figure 4: Three discs with non-empty intersection

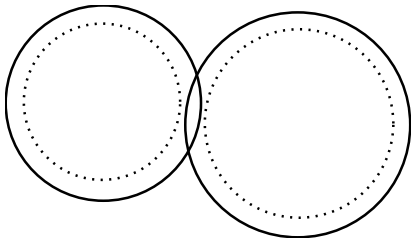

Figure 5: Safely intersecting discs with cores painted.

We start with the following simple approximation algorithm.

[Algorithm 1]

· Sort all the discs in the decreasing order of their radii.
· for each disc C_u in the order do
· if C_u does not intersect any previously accepted disc
· then accept C_u else reject C_u.
· Output all the accepted discs.

Lemma 7 *Algorithm 1 finds a 9-approximate solution.*

Proof Let $\mathcal{S}_{approx}$ be a solution (set of discs) obtained by Algorithm 1. We want to compare the singly-covered area by S_{approx} with that by an optimal solution S_{opt}. When we denote them by $|SCA(S_{approx})|$ and $|SCA(S_{opt})|$, respectively, what we want to prove is the following:

$$\frac{|SCA(S_{approx})|}{|SCA(S_{opt})|} \geq \frac{1}{9}. \tag{4}$$

Let C_u be any given disc which has not been accepted by Algorithm1. Then, there must be some disc which has been accepted before examining C_u and intersects C_u. Since discs are examined in the decreasing order of their radii, we have $r(C_v) \geq r(C_u)$. Thus, if we enlarge the accepted disc C_v by a factor of 3 (we denote the resulting disc by $E_3(C_v)$), then the rejected disc C_u is completely contained in the enlarged disc $E_3(C_v)$. It implies that if we blow up every accepted disc then the union of all those enlarged discs covers the union of all given discs. More precisely, we have

$$\begin{aligned}
\frac{|SCA(S_{approx})|}{|SCA(S_{opt})|} &\geq \frac{|SCA(S_{approx})|}{|\bigcup_{C_u \in S_{opt}} C_u|} \geq \frac{|SCA(S_{approx})|}{|\bigcup_{C_u \in S} C_u|} \\
&\geq \frac{|SCA(S_{approx})|}{|\bigcup_{C_u \in S_{approx}} E_3(C_u)|} = \frac{\sum_{C_u \in S_{approx}} |C_u|}{|\bigcup_{C_u \in S_{approx}} E_3(C_u)|} \geq \frac{\sum_{C_u \in S_{approx}} |C_u|}{|\sum_{C_u \in S_{approx}} |E_3(C_u)|} \\
&\geq \frac{1}{9}.
\end{aligned}$$

□

The performance ratio can be improved to 5.83. The idea is to allow some overlap among accepted discs.

[Algorithm 2]
· Sort all the discs in the decreasing order of their radii.
· for each disc C_u in the order do{
· if C_u is not violated by any previously accepted disc (that is, if the core of C_u does not intersect any such disc)
· then accept C_u else reject C_u.
· Output all the accepted discs.

Lemma 8 *Algorithm 2 finds a* 5.83*-approximate solution.*

Proof Let C_v be any disc rejected in the algorithm. Then, there must be a disc C_u such that
(1) C_u has been accepted before examining C_v, which implies $r(C_u) \geq r(C_v)$, and
(2) C_v violates C_u, that is, C_v intersects the core of C_u.

We enlarge each accepted disc C_u by a factor of $2+\rho$ while keeping its center. The resulting disc coincides with the region swept around the core boundary by a copy of the disc C_u. The enlarged disc has radius $(2+\rho)r(C_u)$. It implies that if we enlarge every accepted disc by a factor of $(2+\rho)$ then each rejected disc is contained in some such enlarged disc.

The proof proceeds in a similar but slightly different way. A difference is that accepted discs may not be disjoint, but it is important to note that their cores are disjoint. We have

$$\begin{aligned}
\frac{|SCA(S_{approx})|}{|SCA(S_{opt})|} &\geq \frac{\sum_{C_u \in S_{approx}} |C_u \cap SCA(S_{approx})|}{|\bigcup_{C_u \in S_{approx}} E_{2+\rho}(C_u)|} \\
&\geq \frac{\sum_{C_u \in S_{approx}} |Vor(C_u) \cap SCA(S_{approx})|}{\sum_{C_u \in S_{approx}} |Vor(C_u) \cap E_{2+\rho}(C_u)|}.
\end{aligned}$$

A key idea for our analysis is to partition the union $\bigcup_{C_u \in S_{approx}} E_{2+\rho}(C_u)$ into disjoint regions each associated with a disc accepted by Algorithm 1 to evaluate how much area is covered exactly

once by those discs. For this purpose we take an additively weighted Voronoi diagram of the accepted disc, weighting each accepted disc C_u by its radius. Then we truncate each Voronoi cell $Vor(C_u)$ associated with C_u by its blown-up copy $E_{2+\rho}(C_u)$. Notice that each Voronoi cell $Vor(C_u)$ is star-shaped. We partition $Vor(C_u)$ by extending rays from the center of C_u through Voronoi vertices in the cells and through intersections between the disc boundary and Voronoi edges. See Fig. 6 for illustration. The cell $Vor(C_u)$ is partitioned into angular sectors. We want to evaluate how much area is covered by the disc C_u itself or its core in each such angular sector.

There are two cases to consider. Look from the center of C_u in some direction. The boundary of $Vor(C_u)$ in that direction is either outside or inside the disc C_u. In fact, if two discs are disjoint then the Voronoi edge between them if any lies between those discs. If two discs intersect each other, then the Voronoi edge between them must lie completely inside their intersection and thus it does not intersect the cores of the two discs since they safely intersect each other.

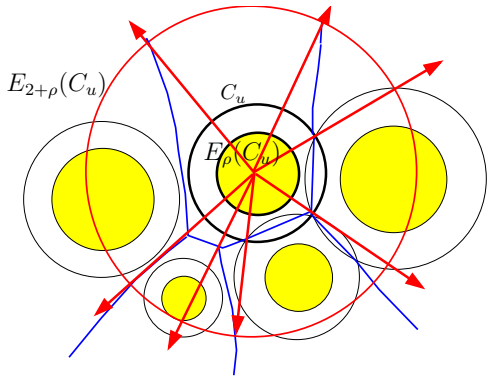

Figure 6: Partition of an additively weighted Voronoi cell associated with a disc C_u by rays from the center of the disc.

If it is outside C_u, then we know that it is not too far outside (at most a factor $2+\rho$ of the disc radius), and the disc cannot be overlapped by any other disc in this direction. So we have the ratio $1/(2+\rho)^2$ in an angular sector in that direction. On the other hand, if it is inside C_u, the part of the disc C_u is overlapped by another disc, but at least we know that the core of C_u is not overlapped by another disc. So in that angular sector between two intersection between the Voronoi edge and the disc boundary we have a ratio at least ρ^2.

We choose the ρ value so that $\min\{\frac{1}{(2+\rho)^2}, \rho^2\}$ is maximized, that is, we choose $\rho = \sqrt{2}-1 = 0.41421356\ldots$ so that $\rho^2 = \frac{1}{(2+\rho)^2}$, that is, $(\rho-(\sqrt{2}-1))(\rho+(1+\sqrt{2}))(\rho+1)^2 = 0, 0 < \rho < 1$ is satisfied. Then, the ratio of the singly-covered area to the total area of the union of those blown-up discs is at least $\frac{1}{(1+\sqrt{2})^2} = \frac{1}{5.83\ldots}$. More precisely, we have

$$\begin{aligned}
\frac{|SCA(S_{approx})|}{|SCA(S_{opt})|} &\geq \frac{\sum_{C_u \in S_{approx}} |C_u \cap SCA(S_{approx})|}{|\bigcup_{C_u \in S_{approx}} E_{2+\rho}(C_u)|} \\
&\geq \frac{\sum_{C_u \in S_{approx}} |Vor(C_u) \cap SCA(S_{approx})|}{\sum_{C_u \in S_{approx}} |Vor(C_u) \cap E_{2+\rho}(C_u)|} \\
&\geq \min\{\rho^2, \frac{1}{(2+\rho)^2}\} \geq \frac{1}{5.83..}
\end{aligned}$$

□

6 Heuristic Algorithms and Experimental Results

Now we propose heuristic algorithms. Although the algorithms have no theoretically guaranteed performance ratios, experimental results verify their effectiveness by their reasonable outputs. We have two heuristic algorithms. One is quite simple but very fast. The other takes some time but it produces better outputs because it is a natural extension of a polynomial-time exact algorithm for the one-dimensional version of the problem.

6.1 Heuristic Algorithm 1

Consider an $M \times N$ rectangular regions associated with an $M \times N$ matrix $R = (r_{ij})$. The first heuristic algorithm is a simple greedy algorithm. We first fix a contraction factor ρ appropriately. Then, we scan the matrix elements in a raster order. At each lattice point we accept the disc at the point if its core does not intersect the core of any previously accepted disc and reject it otherwise. Note that we use contracted discs by the factor ρ for the intersection test. If we can assume that the number of accepted disc centers in an influence region is bounded by some constant, then the algorithm runs in linear time. An influence region for a point (i, j) is a rectangular region with two corners at $(i - \rho(r_{ij} + r_{\max}), j - \rho(r_{ij} + r_{\max}))$ and $(i, j + \rho(r_{ij} + r_{\max}))$, where $r_{\max}$ is the maximum value of $r_{up} \in R$. We can maintain a set of center points of accepted discs using some appropriate data structure such as a range tree. Because of the nature of the algorithm the first disc at the lower left corner is always accepted, which may be a bad choice for future selection.

We have implemented the Heuristic Algorithm 1 using LEDA 4.1. We applied it to sample images of sizes 106×85 and 256×320 each to be enlarged into those of 424×340 and 1024×1280, respectively. Fig. 7 is the one of the smaller size. The running time is 0.06 seconds for the smaller image and 0.718 seconds for the larger one on a personal computer, DELL Precision 350 with Pentium 4. Fig. 8 illustrates an arrangement of contracted discs and its associated Voronoi diagram for the set of those center points, which are outputs for the smaller image. At each output pixel we computed a differential value using so-called Sobel operator [6] and normalized the value roughly between 3 and 8. So, r_{ij} ranges from 3 to 8, which implies that the largest screen element contains 16×16 pixels while the smallest one does 6×6.

Figure 7: An input image for experiments.

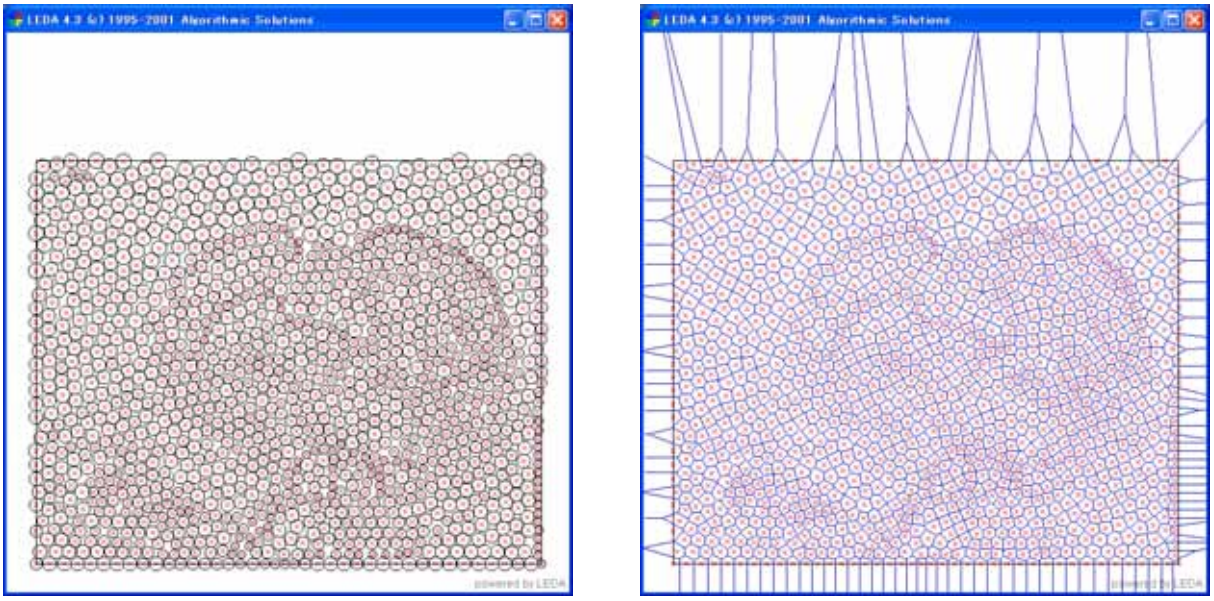

Figure 8: An arrangement of discs accepted by Heuristic Algorithm 1 (above) and Voronoi diagram associated with a set of center points of the discs.

6.2 Heuristic Algorithm 2

The second heuristic is based on Algorithm 2 in the previous section, which examines discs in the decreasing order of their sizes and accepts discs if their core parts do not intersect the core part of any previously accepted disc. It took more time than Heuristic Algorithm 1 for all of our input images and the output quality was also worse. For example, it ran in 0.109 and 1.031 seconds for small and large input images, respectively and the gains achieved were 151474 and 1358940 for small and large images. The gains achieved by Heuristic 1 were 174022 and 1582940, respectively. Heuristic 1 achieved better results for all the test images than Heuristic 2.

Fig. 9 shows an arrangement of contracted discs that are accepted by Heuristic Algorithm 2 and its associated Voronoi diagram.

6.3 Heuristic Algorithm 3

The third heuristic is based on the idea of disc traversing graph. In this heuristic we applied the idea for the first few rows. That is, after computing the smallest radius $d_{\min}$ of discs we examine all the discs with their centers in the first $\lfloor d_{\min}/2 \rfloor$ rows. With this restriction at most doubly overlapping disc sets lead to an optimal set of discs in this restricted region. Finding such an optimal solution for the first several rows is followed by the same raster scan used in Heuristic 1. Unfortunately, this heuristic could not give better solutions than Heuristic 1 in most of the cases.

Fig. 10 shows an arrangement of contracted discs that are accepted by Heuristic Algorithm 3 and its associated Voronoi diagram.

6.4 Iterative Improvement

Set of discs obtained by heuristic algorithms can be improved by applying a standard iterative improvement strategy based on flipping of discs. That is, for each accepted disc we check whether more area can be covered by replacing it with neighboring discs. The running time of this iterative improvement depends on the number of flipping operations. Typical time for the improvement was

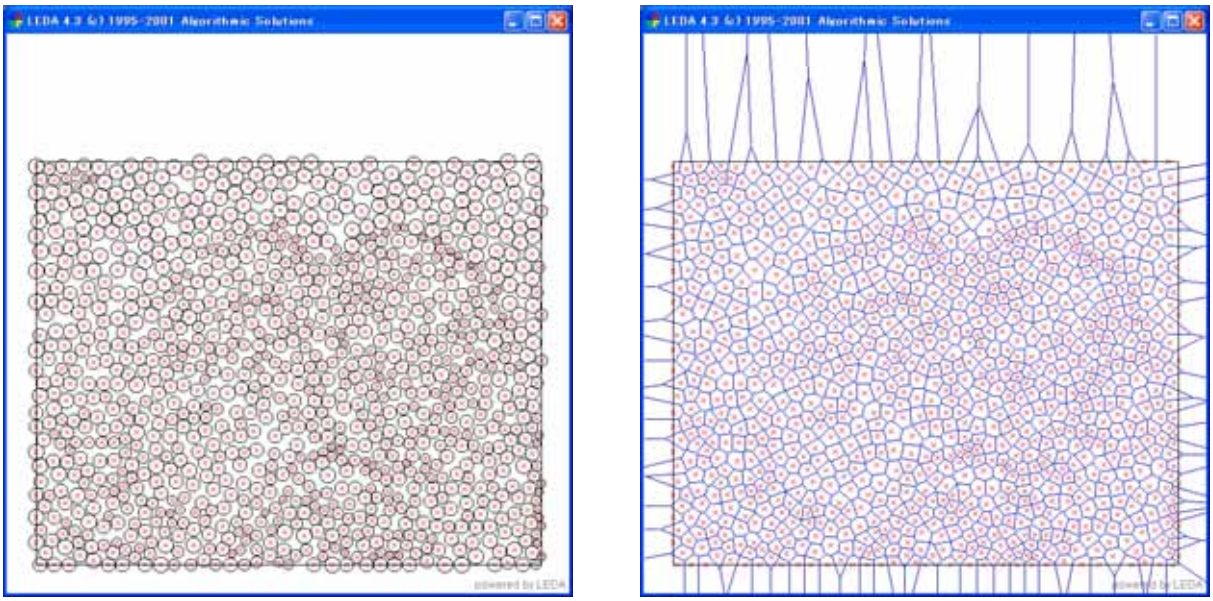

Figure 9: An arrangement of discs accepted by Heuristic Algorithm 2 (above) and Voronoi diagram associated with a set of center points of the discs.

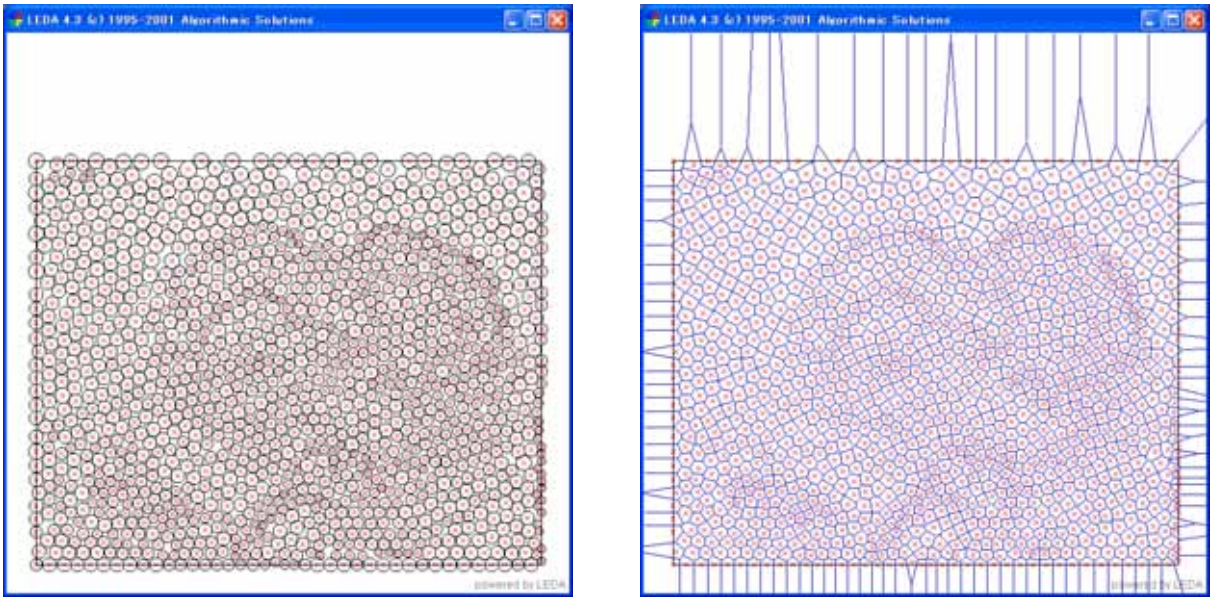

Figure 10: An arrangement of discs accepted by Heuristic Algorithm 3 (above) and Voronoi diagram associated with a set of center points of the discs.

0.469 and 4.39 seconds for small and large images, respectively. Improvement over the area was roughly 1% in most cases.

6.5 Halftoning based on resulting Voronoi diagram

Center points of accepted discs specify center points of screen elements for digital halftoning. Each Voronoi region in an associated Voronoi diagram is filled in according to the gray level at corresponding place.

A part of an output image is shown in Fig. 11. The whole output image is given in Fig. 12 although it is largely contracted.

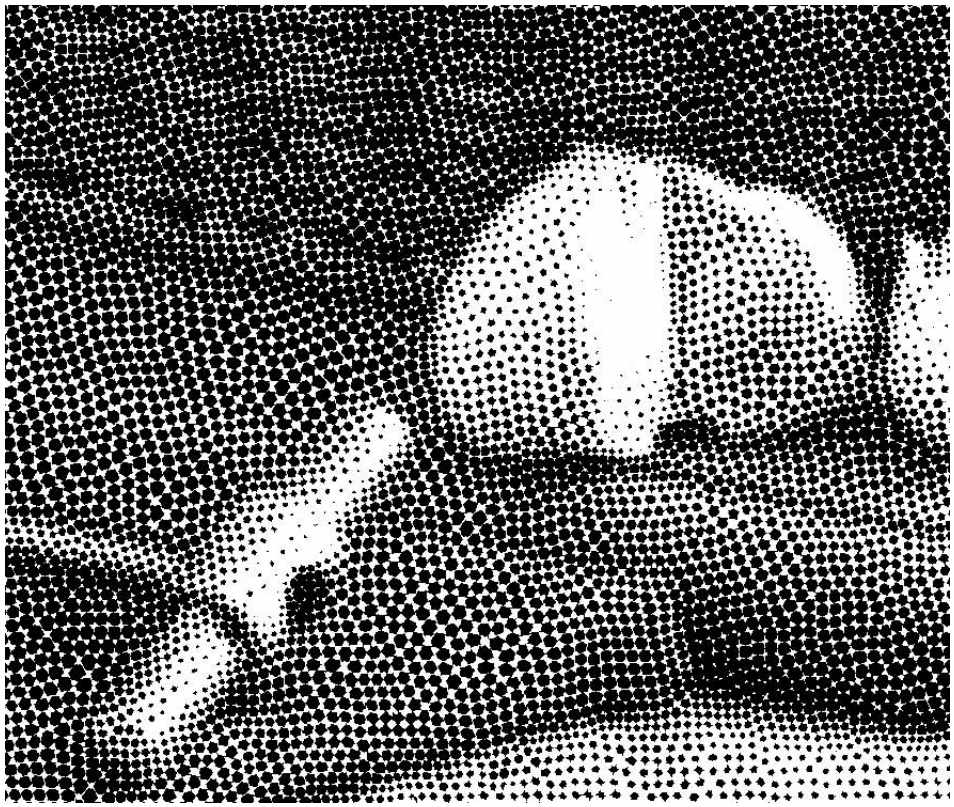

Figure 11: A part of an output image by Heuristic Algorithm 1 followed by iterative improvement.

7 Concluding Remarks

We have considered a geometric optimization problem to choose discs among n given discs to maximize the singly-covered area in conjunction with applications to adaptive digital halftoning. It finds a hard problem from a theoretical point of view, but a heuristic algorithm proposed in this paper works well for our practical applications.

8 Acknowledgment

The first author would like to thank Chee K. Yap of New York University for his valuable suggestion to the plane sweep algorithm. The authors thank to all the participants to the Korean workshop in Seoul, 2003 for their stimulus discussions. This research for the first author was partially supported

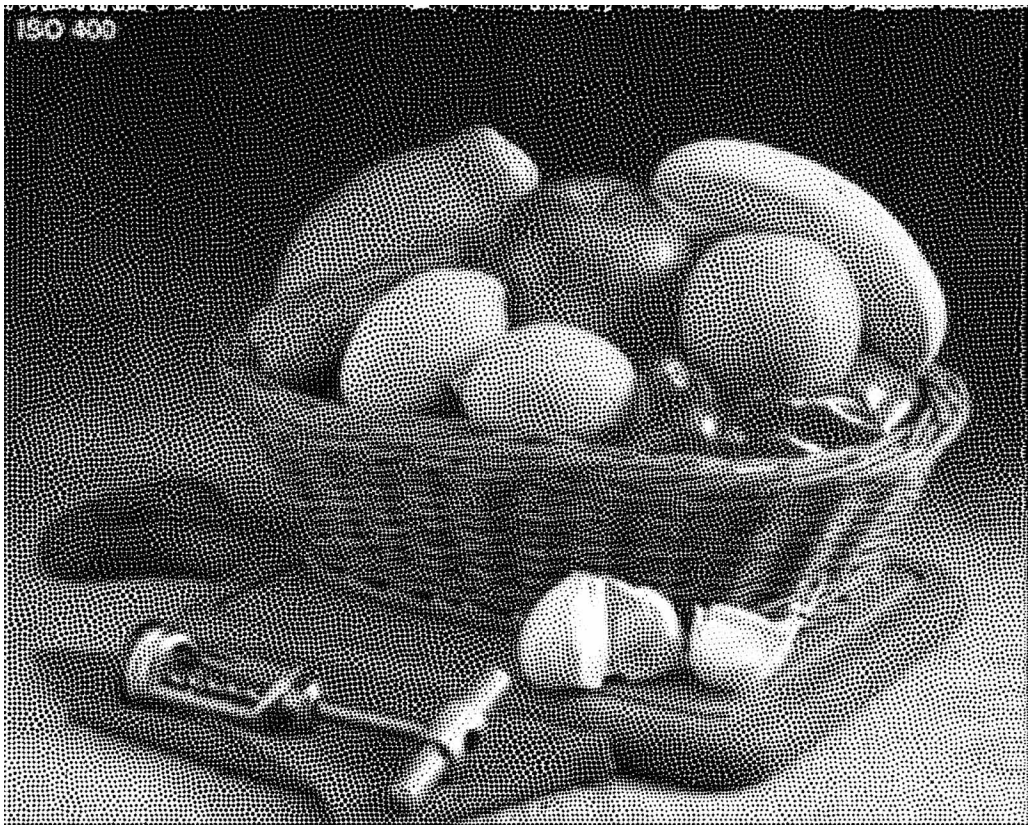

Figure 12: The whole output image.

by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B) and Exploratory Research.

References

[1] T. Asano, T. Matsui, and T. Tokuyama: "Optimal Rounding of Sequences and Matrices," *Nordic Journal of Computing*, Vol.7, No.3, pp.241-256, Fall 2000.

[2] K.J. Nurmela and P.R.J. Oestergard, "Packing up to 50 Equal Circles in a Square," Discrete Comput. Geom., 18:111-120, 1997.

[3] V. Ostromoukhov: "Pseudo-Random Halftone Screening for Color and Black&White Printing," *Proceedings of the 9th Congress on Advances in Non-Impact Printing Technologies*, Yokohama, pp.579-581, 1993.

[4] V. Ostromoukhov, R. D. Hersch: "Stochastic Clustered-Dot Dithering," *Journal of Electronic Imaging*, Vol.8, No.4, pp.439-445, 1999.

[5] S. Sasahara T. Asano: "Adaptive Cluster Arrangement for Cluster-dot Halftoning Using Bubble Packing Method," *Proceeding of 7th Japan Korea Joint Workshop on Algorithms and Computation*, pp.87-93, Sendai, July 2003.

[6] I. Sobel: "Camera Models and Machine Perception," AIM-21, Stanford Artificial Intelligence Lab., Palo Alto, 1970.