INVITED SURVEY PAPER *Special Issue on Selected Papers from LA Symposium*

Digital Halftoning: Algorithm Engineering Challenges

Tetsuo ASANO[†], ***Regular Member***

SUMMARY Digital halftoning is a technique to convert a continuous-tone image into a binary image consisting of black and white dots. It is an important technique for printing machines and printers to output an image with few intensity levels or colors which looks similar to an input image. This paper surveys how algorithm engineering can contribute to digital halftoning or what combinatorial problems are related to digital halftoning. A common criterion on optimal digital halftoning leads to a negative result that obtaining an optimal halftoned image is NP-complete. So, there are two choices: approximation algorithm and polynomial-time algorithm with relaxed condition. Main algorithmic notions related are geometric discrepancy, matrix (or array) rounding problems, and network-flow algorithms.

key words: *approximation algorithm, combinatorial optimization, matrix rounding, network flow*

1. Introduction

The quality of color printers has been drastically improved in recent years, mainly based on the development of fine control mechanism. On the other hand, there is no great invention on the software side of the printing technology. Required is a technique to convert a continuous-tone image into a binary image consisting of black and white dots that looks similar to the input image. Theoretically speaking, the problem is how to approximate an input continuous-tone image by a binary-tone image. Since this is one of the central techniques in computer vision and computer graphics, a great number of algorithms have been proposed to the date (see, e.g., [8], [18], [24], [26], [28], [30], [49]). However, there have been few studies discussing reasonable criteria for evaluating the quality of output binary images; maybe because the problem itself is very practically oriented. Actually, the most popular criterion is to judge the quality by human eyes. It is desirable to establish a good evaluation system for halftoning methods (instead of the "human eye's judgment"), and to handle the digital halftoning problem fully mathematically or algorimically. Unfortunately, to the author's knowledge there has been no attempt to investigate computational complexities of the halftoning problem. This paper surveys recent challenges of algorithm engineers and describe some related problems.

Manuscript received February 8, 2002.

[†]The author is with the School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923–1292 Japan.

2. Known Basic Algorithms

Throughout the paper we put the following assumptions to simplify the discussion to investigate inherent computational complexity of the halftoning problem. We take as an input image an $N \times N$ real-valued matrix $A = (a_{ij})$, $0 \leqq a_{ij} \leqq 1$ for each (i, j) and output a binary matrix $B = (b_{ij})$ of the same size. Usually, the intensity level of "black" is 0 while that of "white" is 1. For color images, we repeat the same halftoning process three times for each of R (Red), G (Green), and B (Blue) components.

We briefly describe several popular halftoning algorithms with their variations.

2.1 Simple Thresholding

Given an $N \times N$ array A of real numbers between 0 and 1, we want to obtain a binary array B of the same size which looks similar to A, where entry values represent light intensities at corresponding locations. The most naive method for obtaining B is simply to binarize each input entry by a fixed threshold, say 0.5. It is simplest, but the quality of the output image is worst since any uniform gray region could become totally white or totally black depending on the intensity levels.

2.2 Ordered Dither

Instead of using a fixed threshold over an entire image, we could use different thresholds. A simple way of implementing this idea is as follows: We prepare an $M \times M$ matrix of integers from 1 to M^2. This matrix *(dither array)* is tiled periodically to cover the image. Each pixel in the image is compared with the corresponding threshold from the dither array to decide which color should be put at that location. Figure 1 shows the dither matrix given by Bayer [8].

2.3 Dither Algorithm

Ordered dither algorithm tries to keep the average intensity level between input and output images by using many different thresholds. However, it is also very easy to define an input image for which the algorithm produces an output image which looks totally different

from the input. For example, if intensity level at each pixel is only slightly smaller than the corresponding entry in the dither matrix, the output image becomes totally black. Dither algorithm to be described preserves the average intensity level. It is achieved if the number of white dots in an output image is about the same as the sum of intensity levels in the input image.

The algorithm is as follows. We first compute the total sum of intensity levels of all pixels (matrix elements) and round it to its nearest integer k. Then, we partition the image plane into two or four disjoint regions and distribute k white dots according to the sums of intensity levels in the corresponding regions. Ties are resolved arbitrarily. We iterate this process recursively until a region is partitioned into pixels.

2.4 Error Diffusion

The dither algorithm is designed to preserve the average intensity level between input and output images. There is another standard algorithm called *"error diffusion"* that also possesses the same property by propagating the quantization errors to unprocessed neighboring pixels according to some fixed ratios. More precisely, pixels are processed in a raster order, from left to right and top to bottom. Each pixel level is compared with a fixed threshold, 0.5 and round it up if it is greater than or equal to the threshold and round it down otherwise. The quantization error caused by the rounding is diffused over the pixels around it with fixed ratios. For example, if a pixel level is 0.7, it is rounded up to 1 and the error -0.3 is diffused to the unprocessed pixels nearby. The ratios suggested by Floyd and Steinberg in their paper [18] are shown in Fig. 2:

We can also distribute error over wider region. One such pattern is given in Fig. 3.

This method certainly preserves the average intensity level because the rounding error is distributed to neighboring pixels. When the process terminates, the difference between the sums of intensity levels in the input and output images is at most 0.5.

This method not only preserves the average intensity level but also gives excellent image quality in many

1	33	9	41	3	35	11	43
49	17	57	25	51	19	59	27
13	45	5	37	15	47	7	39
61	29	53	21	63	31	55	23
4	36	12	44	2	34	10	42
52	20	60	28	50	18	58	26
16	48	8	40	14	46	6	38
64	32	56	24	62	30	54	22

Fig. 1 8×8 dither matrix by Bayer.

	•	7/16
3/16	5/16	1/16

Fig. 2 Diffusion ratios in error diffusion by floyd and steinberg.

		•	7/48	5/48
3/48	5/48	7/48	5/48	3/48
1/48	3/48	5/48	3/48	1/48

Fig. 3 Diffusion ratios by Jarvis-Judice-Ninke [23].

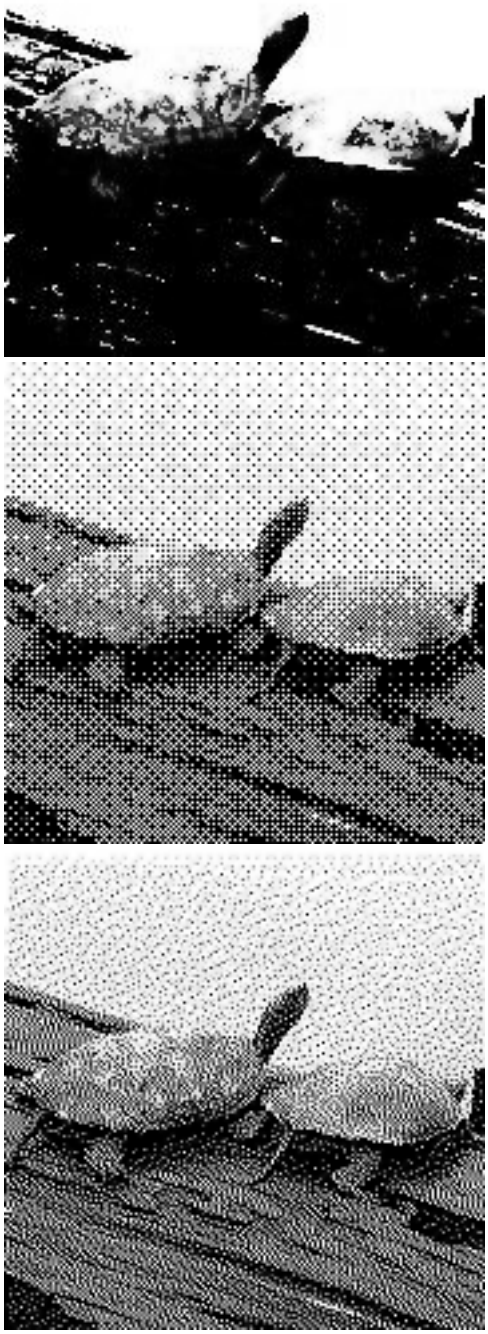


Fig. 4 Output images: Simple thresholding, ordered dither, and error diffusion (color images).

34	48	40	32	29	15	23	31
42	58	56	53	21	5	7	10
50	62	61	45	13	1	2	18
38	46	54	37	25	17	9	26
28	14	22	30	35	49	41	33
20	4	6	11	43	59	57	52
12	0	3	19	51	63	60	44
24	16	8	27	39	47	55	36

Fig. 5 Matrix used for dot diffusion.

cases, but it tends to produce visible artifacts in an area of uniform intensity, which are caused by the fixed error-diffusing coefficients.

Figure 4 shows output images of simple thresholding, ordered dither and error diffusion. Theoretically the simple thresholding is too bad, but it can produce meaningful pictures in many cases. Its serious drawback is poor reproduction ability of details. Ordered dither is as fast as the simple thresholding and produces reasonable pictures, and thus it is popularly used in cheap printers. Its drawback is visual artifacts due to regular patterns. Error diffusion is much slower than those algorithms, but it is much superior in the quality of output images. It is unfortunate to produce visible artifacts in a region with uniform intensity.

2.5 Dot Diffusion

Yet another drawback of error diffusion is its serial nature. Since the last pixel (at the lower right corner of an image) is affected by the first pixel (at the upper left corner), it is an inherently serial process, hard to be parallelized. Dot diffusion algorithm proposed by Knuth [24] tries to parallelize error diffusion. This algorithm uses a matrix just like a dither matrix. An example of such a matrix is shown in Fig. 5. The matrix entries represent the order of processing. The matrix is tiled periodically to cover the image. First, we binarize all the pixels numbered 1 and propagate rounding errors to their neighboring unprocessed pixels. This process can be done in parallel. Then, we round all the pixels numbered 2, and so on. It is easy to choose unprocessed pixels. To round pixels numbered i, unprocessed pixels are those neighboring pixels of numbers greater than i. The pixel whose number is locally highest cannot distribute its rounding error over neighboring pixels. Such a pixel is called a "baron." The matrix of Fig. 5 has two barons. We could design a matrix which has only one baron, but it is known that it would not lead to good results in experiments [24].

3. Variation of Known Algorithms with Related Problems

3.1 Variation of Simple Thresholding

The serious drawback of the simple thresholding is poor expression of intermediate intensity due to its independent process at each pixel and its use of a fixed threshold. One of the method to improve the expression is to use random thresholds. Precisely, we generate white Gaussian noise over an input image and use the noise as threshold. This method is considered as variation of ordered dither with a dither matrix with random numbers. Thus, theoretically speaking, the expected average intensity level of the output image is expected to be equal to that of input image.

The same idea is popular in randomized algorithms under a different name, i.e., *randomized rounding* [36], [43], [44], in which a real number x, $0 \leqq x \leqq 1$, is rounded up with probability x. It is one of the standard techniques in randomized algorithms. Unfortunately, experimental results due to our implementation of the algorithm are not so satisfactory.

3.2 Variation of Ordered Dither Algorithm

The previous subsection has described a rounding algorithm using variable random numbers as thresholds to generalize the simple thresholding using one fixed threshold. We can use a table of random numbers instead of generating a random number for each pixel. Dither matrix corresponds to this table of random numbers. Small table size tends to generate visible artifacts. So, the largest possible table size would be better. In fact, there is an algorithm along this idea, which is known as a *blue-noise mask* algorithm [33], [50], [51], [56] in general. This algorithm uses a large dither matrix (blue-noise mask) of size, say 256×256.

Properties of Dither Matrix

The performance of the ordered dither algorithm heavily depends on a dither matrix used. Then, how can we define an optimal dither matrix? The dither matrix depicted in the previous subsection is defined as follows [25].

First, starting from a 1×1 matrix $D_0 = [1]$, we recursively define $D_k (k = 1, 2, \ldots)$ as follows:

$$D_k = \begin{bmatrix} 4D_{k-1} - 3U_{k-1} & 4D_{k-1} - U_{k-1} \\ 4D_{k-1} & 4D_{k-1} - 2U_{k-1} \end{bmatrix}.$$

Here, U_k is a $2^k \times 2^k$ square matrix consisting of all 1s.

We have known how to construct the dither matrix. Then what is a merit to use this dither matrix? In other words, does it optimize anything? If the purpose were only to distribute numbers 1 through 2^{2k} over the $2^k \times 2^k$ matrix, there would be a number of different ways. Imagine an artificial image of gradually increasing intensity from left to right. During the transition from dark to bright, the number of white dots should gradually increase. This means that for any number i between 1 and 2^{2k} those entries having numbers greater than i must be as uniformly distributed as possible in the dither matrix. The uniformity can be measured in

several different ways. One measure is based on the ratio of the minimum pairwise distance against the diameter of the maximum empty circle. Another possible measure is based on the notion of *discrepancy* which is related to the difference between the area and the relative number of white dots.

The above regular grid-like construction of the dither matrix is optimal in the former measure since it is constructed under the notion of *incremental Voronoi insertion*. An optimal dither matrix under the former measure is designed as follows. Before construction we have to note that dither matrix is used to cover an entire image by repeatedly arranging the matrix. First we choose an arbitrary entry, say, the upper left corner of the matrix, to assign number 1. Because of the periodicity, it means that we have placed points numbered 1 on regular grids $(8i, 8j), i, j = 0, 1, \ldots$. The entry 2 must be placed at a grid point farthest from the points numbered 1. Such a place coincides with a Voronoi vertex of the Voronoi diagram (see Fig. 6) for the set of points numbered 1. Similarly, the location of the entry 3 should be chosen among the Voronoi vertices for the Voronoi diagram of the set of points numbered 1 or 2. This strategy is called "incremental Voronoi insertion" which is rather easy to be implemented. If we resolve ties appropriately we obtain the dither matrix.

Unfortunately, this dither matrix is not good enough in practice. What is wrong? The measure may be wrong. That is, the measure based on the ratio between the minimum pairwise distance and the radius of the maximum empty circle may not be good enough. The latter measure based on discrepancy suggested above seems to be more promising. In the measure we take a number of regions of the same area. If points are uniformly distributed, every such region contains roughly the same number of points in it. In the discrepancy measure we can take regions of arbitrary shapes. The former measure based on the minimum pairwise distance is obtained if we take a family of circular regions. In this sense the discrepancy measure is a generalization of the former measure.

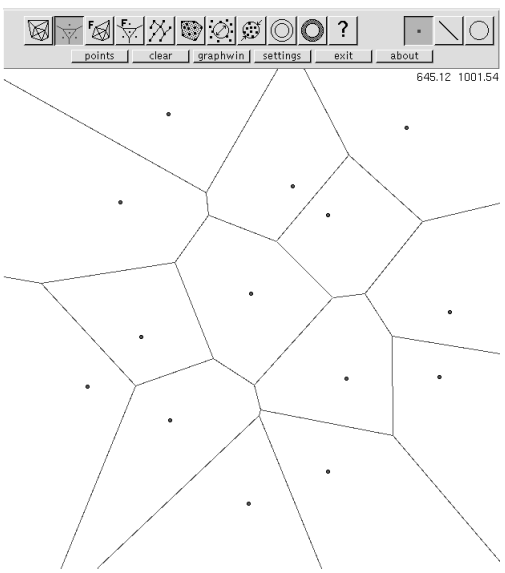


Fig. 6 Voronoi diagram.

To define the *discrepancy measure*, we introduce a family $\mathcal{F}$ of regions over an image. For each region R in $\mathcal{F}$, let $A(R)$ denote the area of R and $card(R)$ denote the number of points in R. Then, we take the difference

$$D(R) = |n \cdot A(R) - card(R)|,$$

as the discrepancy for the region R, assuming that the area of the whole image is 1.

Consider a regular pattern in which n points are placed in a $\sqrt{n} \times \sqrt{n}$ grid. Take a rectangular region R defined by two rows of points. Then, the area of the rectangle is $(1/\sqrt{n}) \times 1 = 1/\sqrt{n}$. If we locate the rectangle so that the two sides exactly coincide with two rows of points, we have $card(R) = 2\sqrt{n}$. Otherwise, it contains only one of rows of points, and so $card(R) = \sqrt{n}$. Thus, we have $D(R) = |n/\sqrt{n} - 2\sqrt{n}| = \sqrt{n}$ in the former case and $D(R) = |n/\sqrt{n} - \sqrt{n}| = 0$ in the latter case. In fact, we can prove that the maximum value of $D(R)$ is $O(\sqrt{n})$. Furthermore, it is known that it remains $O(\sqrt{n})$ when n points are randomly distributed. However, there are deterministic algorithms which achieves the discrepancy $O(\log n)$. Refer to the textbooks by Chazelle [15] and Matoušek [31].

Packing Problem

The problem of distributing a specified number of points as uniformly as possible is closely related to the so-called *packing problem* in which we are required to place a given number of congruent disks within a unit square so that the radius of those disks is largest possible. It is known that this problem is equivalent to that of placing a given number of points within a unit square so that their pairwise minimum distance is maximized. In fact, denoting by r_k the largest radius of k congruent disks within a unit square and by d_k the largest possible minimum pairwise distance between k points within a unit square, we have

$$r_k = \frac{d_k}{2d_k + 2}.$$

The packing problem is very interesting in itself and in fact it has a long history (see for a survey [41]). The first non-trivial result was reported in 1960s. During the last three decades the development in this field has been steady but there was no break-through. Very recently Nurmera and Östergård presented surprising results [39]. Using a computer-aided proof technique, they proved the optimality of the circle packing patterns up to 27 circles. In addition, they provided best packing patterns up to 50 circles [38]. More recently, Casado-Garcia-Szabó-Csendes find more optimal solutions and better packing results up to 100 circles [13],

[14].

Rotation of Dither Matrix

Another related problem comes from the human perception. An interesting feature of human perception is that horizontal and vertical patterns are more sensitive to human eyes than skewed patterns [40]. This fact suggests us of rotating a dither matrix to have better results. Then, the problem is how to design such a rotated pattern consisting of M^2 elements. This is not so easy since a rotated pattern must be tiled to cover the entire image and the area (number of entries) is fixed. Figure 7 illustrates how a rotated dither matrix covers the entire plane.

We shall explain how to design a pattern which satisfies the following conditions:

(1) area condition: The rotated matrix must have the same number of entries (or grid points) as that of the original matrix, and those grid points form a connected cluster without any hole.

(2) tiling condition: The rotated matrix must be tiled to cover the entire grid plane, that is, the entire plane must be tiled without any gap by repeating the corresponding pattern.

(3) angle condition: The rotated matrix must be bounded by four digital line segments. The angles of those segments from the axes should be close enough to given angles. Furthermore, the angle between two such segments should be almost perpendicular.

The most important observation behind the scheme for rotation is the following Pick's theorem [35].

[Pick's Theorem] The area of any simple polygon P in a grid (not necessarily convex) whose vertices are lattice points is given by

$$\text{area}(P) = L_{in}(P) + L_{bd}(P)/2 - 1,$$

where $L_{in}(P)$ denotes the number of grid points in the interior of the polygon P and $L_{bd}(R)$ that of grid points on the boundary.

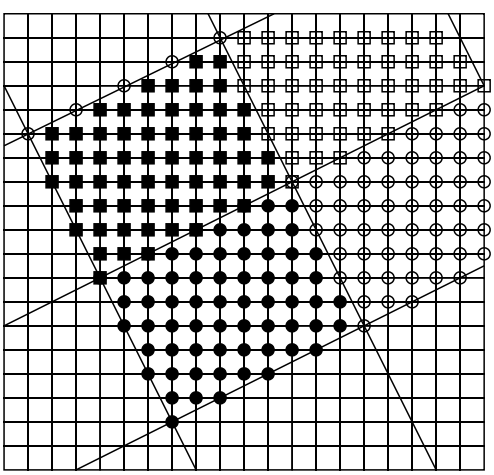


Fig. 7 Tiling the entire grid by a pattern.

Our objective is to design a rotated square region R consisting of M^2 grid points rotated approximately by an angle θ. We have four vertices denoted by A, B, C and D, as shown in Fig. 8. Among the four vertices only the vertex A is included in the rotated pattern. Since this is a tiling pattern, the other three vertices become the positions at which the A corner of the pattern R is located. The rotated square R has four sides, AB, AC, BD and CD. The grid points on the lower sides AB and AC are included into the rotated pattern R while those grid points on the upper sides BC and CD are not. Here note that by symmetry the number of grid points on the lower sides is equal to that of grid points on the upper sides. See Fig. 8 for illustration.

Then, the number of grid points included in the rotated pattern R is given by the sum of the number of grid points lying in the interior of the rotated square $R = ABCD$, half the number of grid points on the four sides excluding the vertices, and 1 for the vertex A. Thus, by the Pick's theorem, the number of grid points in the rotated square pattern R is

$$\begin{aligned} &L_{in}(R) + \frac{1}{2}(L_{bd}(R) - 4) + 1 \\ &\quad = L_{in}(R) + \frac{1}{2}L_{bd}(R) - 1 \\ &\quad = area(R). \end{aligned}$$

Now, given a size of a rotated pattern (the number of grid points) and an angle θ, we can construct such a rotated pattern as follows:

Designing a Rotated Pattern

(1) Find four integers a, b, c, and d such that

$$\begin{aligned} &ad + bc = M^2, \\ &\frac{b}{a} \cong \frac{c}{d} \cong \tan\theta. \end{aligned}$$

(2) Determine a quadrangle $R = ABCD$ such that

1. The bottom, right, left, and top vertices of R are A, B, C, and D.
2. The coordinates of the vertices B, C, D are determined by $(x_A + a, y_A + b)$, $(x_A - c, y_A + d)$, $(x_A - c + a, y_A + d + b)$, respectively.

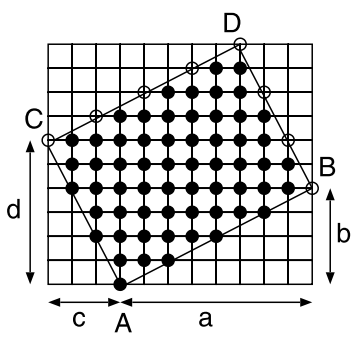


Fig. 8 A tiling pattern $R = ABCD$ and four parameters a, b, c, and d defining it.

3. The grid points on the lower side are included in R.
4. The grid points on the upper side are not included in R.
5. Among the four vertices, only the bottom vertex A is included in R.

Lemma 3.1: The rotated pattern R designed above satisfies the three conditions mentioned above.

Proof
Area Condition is satisfied since the area of the rotated pattern R is

$$(a+c)(b+d) - ab - cd = ad + bc.$$

Tiling Condition: The pattern can tile the entire grid. When we translate R so that the A corner coincides with the location of the vertex B, the side AC of the translated pattern coincides with the side BD of the pattern in the original location. By the definition of the rotated pattern, they coincide with each other and those grid points on the side are included only in the translated pattern. It is just the same for the other sides. Finally, there is no collision of vertices since we have chosen only one vertex among the four vertices.
Angle condition: We can choose the four parameters a, b, c, d so that the slopes of the sides AB and CD are roughly equal to $\tan\theta$ and those of the sides AC and BD are roughly equal to $\tan(\theta - \pi/2)$. Thus, we can choose the best possible parameter values among those values satisfying the area condition. □

Blue-noise Mask: a Huge Dither Matrix

One way to remove the artifact texture pattern of the Ordered dither algorithm is to rotate the dither matrix. There is another way. Just use a huge dither matrix of size, say, 256×256. If we carefully design such a huge dither matrix, artifact textures are not visible anymore. The problems are how to design such a huge dither matrix and the large storage requirement.

Such a huge dither matrix is generally referred to as a blue-noise mask. Important is to remove periodicity. Consider a dither matrix of size 256×256. When we have 256 intensity levels, each number between 1 and 256 appears 256 times in the matrix. For each number p between 1 and 256, those entries numbered 1 through p should be distributed as uniformly as possible. A desired pattern is not a regular one but somewhat random-looking pattern as is explained using the notion of discrepancy. There are several ways to incorporate randomness. One such method is the one called "void-and-cluster" algorithm [51].

The algorithm starts with a random distribution of points and gradually tries to reform the pattern for uniform distribution. There are two factors to break uniformity: cluster parts in which many points are located closely to each other and void parts in which points are sparsely distributed. An idea to achieve uniform distribution is to remove a point in a cluster to put it at the center of a void.

Such an operation is well supported in computational geometry. Given n points, the Voronoi diagram is constructed in $O(n \log n)$ time [42]. When a point is surrounded by many points, its associated Voronoi region tends to be small. Thus, cluster parts are found by checking areas of Voronoi regions. On the other hand, void parts correspond to sparse regions. Such locations are found as centers of large empty circles with no point contained. A largest empty circle can be found in linear time in two ways, one based on linear programming [10] and the other on randomization [42].

3.3 Variation of Dither Algorithm

A standard dither algorithm iteratively partitions an image into four parts by orthogonal lines. Partitions by orthogonal lines may produce visible artifacts because of the sensitivity of the human visual system. One way to resolve the problem is to use non-orthogonal line for partition, or to use a rotated tiling pattern.

3.4 Variation of Error Diffusion

Error diffusion is one of the most successful algorithms for digital halftoning and thus a number of variations of the algorithm have been proposed so far.

One drawback of the error diffusion algorithm is to generate artifact textures in a region of uniform intensity. We could remove such textures by introducing randomness in the diffusion ratios, but then it would cause blurring effect. One way to resolve this problem is based on region segmentation. That is, we distinguish those pixels in uniform intensity areas from those near edges or parts characterized by drastic change of intensity levels. In uniform areas we diffuse rounding errors in a random manner and in edge parts we use the conventional ratios or diffuse error only along directions of edges [52].

Another idea to remove textures is to replace the raster scan by a different order of scanning pixels. In other words, we scan pixels in a different order. Such orders are characterized by *"space-filling curves"* which visit every pixel exactly once.

The idea of using space-filling curves for digital halftoning is not new. Velho and Gomes [53] use space-filling curves for cluster-dot dithering. Zhang and Webber [57] give a parallel halftoning algorithm based on space-filling curves. Peano curves are also used [55]. Asano, Ranjan and Roos [7] formulate digital halftoning as a mathematical optimization problem and obtain an approximation algorithm based on space-filling curves. Asano [2] diffuses rounding errors along a random space-filling curve. So, the digital halftoning techniques based on space-filling curves seem to be promising.

Figures 9 show several representative space-filling

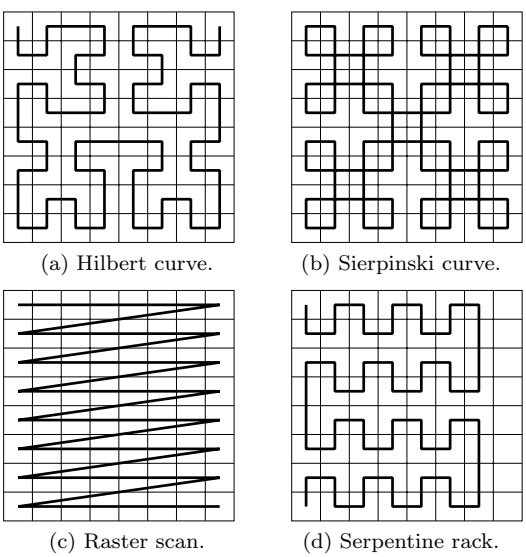
curves which are recursively defined.

A serious drawback of the traditional space-filling curves is a severe constraint on image sizes, that is, most of them are usually applied only to squares and also feasible image sizes are quite discrete. Another drawback is that their regularity causes some inherent visible artifact textures. It seems to be hard to resolve these problems as far as we rely on recursively defined space-filling curves.

There are two algorithms for generating random space-filling curves. One of them is based on random spanning tree of a lattice graph [2] (see Fig. 10) and the other is based on successive local changes of a curve [3]. Especially, Asano-Katoh-Tamaki-Tokuyama [3] defines a local transformation of a space-filling curve into another and discusses whether any two space-filling curves can be mutually transformed when their entrance and exit are fixed. More precisely, the following theorem is established.

Theorem 3.2: Given two space-filling curves with the same entrance and exit on a rectangular grid, we can transform one from the other by applying local transformations $O(n^3)$ times, where n is the number of grid points, i.e., the length of the curve.

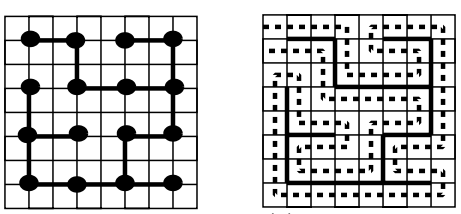
Another theoretical results on grammatical definitions of space-filling curves and some other applications are seen in the literature [45].



(a) Hilbert curve. (b) Sierpinski curve.

(c) Raster scan. (d) Serpentine rack.

Fig. 9 Representative space-filling curves.



(a) Spanning tree. (b) Traverse along the tree.

Fig. 10 Maze which defines a random space-filling curve.

4. Digital Halftoning as a Combinatorial Problem

4.1 Human Visual System

In the conventional halftoning algorithms their output images have been evaluated by human eyes in many cases and there are very few studies for quantitative evaluation of the quality of output images. There are many studies on image quality in computer vision (see e.g., [34]) while most of them are for continuous-tone images and not for binary images. In the Direct Binary Search [1] an operation to flip pixel values between 0 and 1 so that the difference between average intensity levels in small regions is minimized between input and output images. Geist et al. [19], who introduced a mathematical framework into digital halftoning, uses simulated annealing technique for optimization. Another algorithm considering human visual system is also proposed [29].

A criterion common to those studies is the following. Let $A = (a_{ij})_{i,j=1,\ldots,N}$ be an input continuous-tone image. Each matrix element a_{ij} represents an intensity between 0 and 1 at a pixel (i, j). The index i corresponds to the row index and j does to the column index, as usual. For simplicity we assume an $N \times N$ square matrix and denote the number of matrix elements (pixels) by n, i.e., $n = N^2$.

An output image is a binary matrix $B = (b_{ij})$ where each $b_{ij} = 0, 1$. An output image should look like an input image. When we look at a pixel, what we perceive is not the intensity level of the pixel but average intensity level around the pixel. Thus, if we computed the sum or squared sum of average intensity levels at all pixels and took their difference between input and output image, it would be a good criterion to determine the similarity between the two images [29], [34], [54]. It is known as Frequency Weighted Mean Square Error (FWMSE). Formally, with a weight matrix $V = (v_{|k||l|}), k, l = -M, \ldots, 0, \ldots, M$, we compute the following sum:

$$W(A,B) = \sum_{(i,j)\in[1,N]^2} \left[\sum_{k=-M}^{M} \sum_{l=-M}^{M} v_{|k||l|} a_{i+k,j+l} - \sum_{k=-M}^{M} \sum_{l=-M}^{M} v_{|k||l|} b_{i+k,j+l} \right]^2 .$$

Here, $V = (v_{|k||l|}), k, l = -M, \ldots, 0, \ldots, M$ is an impulse response that approximates the frequency characteristics of the human visual system and M is some small constant, say 3. An example of the matrix V is the following:

$$V = \begin{pmatrix} 1 & 0.8 & 0.5 & 0.2 \\ 0.8 & 0.8 & 0.5 & 0.2 \\ 0.5 & 0.5 & 0.5 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 \end{pmatrix}.$$

With the distance measure between input and output images, our goal is to find a binary image B that minimizes the distance $W(A, B)$ to a given input image A.

4.2 Optimization Criterion

We can rewrite the above optimization criterion by introducing a family of regions $\mathcal{F} = \{R_{ij}\}$, where R_{ij} is a square region centered at (i, j), i.e.,

$$R_{ij} = \{(i + k, j + l) | k, l = -M, \ldots, 0, \ldots, M\}.$$

Defining $A(R_{ij})$ and $B(R_{ij})$ for a region R_{ij} by

$$A(R_{ij}) = \sum_{k=-M}^{M} \sum_{l=-M}^{M} v_{|k||l|} a_{i+k,j+l}, \text{ and}$$

$$B(R_{ij}) = \sum_{k=-M}^{M} \sum_{l=-M}^{M} v_{|k||l|} b_{i+k,j+l},$$

our objective is to

$$\text{minimize} \quad W(A, B) = \sum_{R_{ij} \in \mathcal{F}} [A(R_{ij}) - B(R_{ij})]^2.$$

To analyze the computational complexity of digital halftoning as a combinatorial problem, we further simplify the criterion with the following assumptions.
(1) First we neglect frequency characteristics of the human visual system, in other words, we assume that every weight v_{kl} is 1. Then, we can replace the weighted average in each region just by the sum in the region.
(2) For the time being we replace the sum of squared difference by the sum of difference, or by the maximum difference, that is, L_2-metric by L_1-metric or L_∞-metric. Later, we generalize it to that of L_p-metric.
(3) We generalize regions to compute the difference. We allow not only square regions but rectangular regions and even sets of pixels which do not form connected regions. In other words, we treat regions just as sets of pixels. Therefore, a family of regions is considered as a part of input.

To have a general description of the problem, we denote the set of all pixels by G_n:

$$\begin{aligned} G_n &= \{(1, 1), (1, 2), \ldots, (N, N)\} \\ &= \{p_1, p_2, \ldots, p_n, \} \end{aligned}$$

where $n = N^2$ is the total number of pixels. Each region in a given family is a subset of G_n. Hereafter a family of regions is denoted by a symbol $\mathcal{F}$.

Let $\mathcal{A} = \mathcal{A}(G_n)$ be a set of all $[0, 1]$ matrices with the index set G_n and $\mathcal{B} = \mathcal{B}(G_n)$ be its subset consisting of all $\{0, 1\}$ matrices. For a region R in a family $\mathcal{F}$, the sum of elements of A and B in R are denoted by $A(R)$ and $B(R)$, respectively:

$$A(R) = \sum_{p_i \in R} a_{p_i}, \text{ and } B(R) = \sum_{p_i \in R} b_{p_i}.$$

Now, given a family $\mathcal{F}$, the distances $Dist_1^{\mathcal{F}}(A, A')$ and $Dist_\infty^{\mathcal{F}}(A, A')$ between two images A and A' in $\mathcal{A}$ is defined by

$$Dist_1^{\mathcal{F}}(A, A') = \sum_{R \in \mathcal{F}} |A(R) - A'(R)|, \text{ and}$$

$$Dist_\infty^{\mathcal{F}}(A, A') = \max_{R \in \mathcal{F}} |A(R) - A'(R)|.$$

Now, given a $[0, 1]$-matrix A as an input image, the digital halftoning problem is defined as that of finding a $\{0, 1\}$-matrix B that minimizes the distance $Dist_1^{\mathcal{F}}(A, B)$ or $Dist_\infty^{\mathcal{F}}(A, B)$ to A.

4.3 Variation of Problems

The most interesting problem is to find an optimal binary image, but we are also interested in finding a reasonable solution. More precisely, given an input image A, a family $\mathcal{F}$ of regions, and some number d, we are interested in finding a binary image B that satisfies

$$|A(R) - B(R)| < d$$

for every region R in $\mathcal{F}$. If we have a polynomial-time algorithm for determining the existence of such a binary image, we can obtain an optimal solution in the $Dist_\infty^{\mathcal{F}}$ distance in polynomial time in n, the number of pixels, and $m = |\mathcal{F}|$. We are also interested in finding an approximate solution with some performance guaranteed.

In summary, problems we consider in this paper are
(1) Optimal Digitization to find an optimal binary matrix for a given real-valued matrix and a family of regions,
(2) Feasible Digitization to determine whether there is any binary matrix satisfying all the constraints associated with regions in a given family $\mathcal{F}$, and
(3) Approximate Digitization to give an approximate binary matrix with some performance guaranteed.

5. One-Dimensional Problem

5.1 Problem Definition

We shall begin with a basic case for the rounding problem. We take a $[0, 1]$-valued one-dimensional array $A = (a_i)_{i=1,\ldots,n}$ instead of a two-dimensional array as an input, and a binary array $B = (b_j)_{j=1,\ldots,n}$ of the same size as an output. Accordingly, a family $\mathcal{F}$ of regions is a family of subsets of the index set

$G_n = \{1, \ldots, n\}$. A basic family is the one of intervals in the entire interval $[1, n]$, which are regarded as subsets consisting of consecutive indices. Among them, the most basic family of intervals is that of all intervals of length 2.

5.2 Feasibility Problem

We first consider the feasibility problem, which is formally described as follows:

[Feasibility Problem] Given a $[0, 1]$-valued array $A = (a_i)_{i=1,\ldots,n}$ of length n and a family $\mathcal{F}$ of all the subintervals of length 2 of $[1, n]$, determine whether there is any binary array B of the same length such that

$$|A(I) - B(I)| < 1$$

holds for any interval I in $\mathcal{F}$.

Fortunately, the answer to the above question is always affirmative and in fact it is easy to find such a binary array in linear time. An algorithm is as follows:

[1D-Error Diffusion]

```
e = 0;
for i = 1 to n do{
    if a_i + e < 0.5 then b_i = 0;  else b_i = 1;
    e = a_i + e − b_i;
}
```

Let e_i be the quantization error generated after rounding a_i in the above algorithm. Then, we can prove by induction that each error e is between -0.5 and 0.5. Thus, for any interval I of length 2 the difference between $A(I)$ and $B(I)$ is bounded by 1, that is, $|A(I) - B(I)| < 1$. Furthermore, the same inequality holds even for a family of all intervals of length k, $1 \leqq k < n$.

The bound 1 is almost tight because we also have the following interesting observation [4].

Lemma 5.1: For a family $\mathcal{F}$ of all intervals of length $k \geqq 2$ there exists an input sequence $A = (a_i)$ for which there is no binary sequence B attaining

$$\max_{I \in \mathcal{F}} |A(I) - B(I)| < 1 - \frac{1}{n-1}. \tag{1}$$

For $k = 2$, given a sequence $(0, 1/(4p-1), (4p-3)/(4p-1), (4p-5)/(4p-1), \ldots, (2p-2)/(4p-1), 2p/(4p-1))$, the best bound is $(4p-2)/(4p-1) = 1 - 1/(4p-1)$.

Another interesting result is known about the number of feasible solutions. Recall that our constraint is that the rounding error $|A(I) - B(I)|$ is bounded by 1 for any interval I in a given family $\mathcal{F}$. How many different roundings satisfying the constraint exist? Sadakane, Takki-Chebihi, and Tokuyama [46] proved the following theorem that says there are at most $n + 1$ different roundings independently on k.

Theorem 5.2: Given a $[0, 1]$-valued array $A = (a_j)$ of size n and an integer $k < n$, there are at most $n + 1$ integer-valued array $B = (b_i)$ of the same size such that

$$\left| \sum_{j \in R} a_j - \sum_{j \in R} b_j \right| < 1$$

for all intervals of length k.

5.3 Optimization Problem

We have seen that the feasibility problem is solved in linear time. Then, what about the optimization problem?

[L_∞-Optimization Problem] Given a $[0, 1]$-valued array $A = (a_i)_{i=1,\ldots,n}$ of length n and a family $\mathcal{F}$ of all subintervals of length 2 of $[1, n]$, find a binary array B of the same length that minimizes

$$\max_{I \in \mathcal{F}} |A(I) - B(I)|.$$

This problem is also easily solved based on dynamic programming. In fact, we can solve a generalized problem for a family of all intervals of length k. An algorithm known as the Viterbi's algorithm is one such algorithm which runs in $O(2^k n)$ time and $O(2^k n)$ space [7]. It is also possible to reduce the space complexity from $O(2^k n)$ to $O(n + 2^k \sqrt{n})$ without increasing the running time [7]. Anyway, the running time is polynomial in n while exponential in k, the length of intervals.

A natural question is whether there is an algorithm of polynomial complexity both in n and k. Such polynomial algorithms are presented by the authors [4]. Precisely, two algorithms are proposed. Algorithm 1 runs in $O(n^{1.5} \log^2 n)$ time and $O(n)$ space, while Algorithm 2 does in $O(k^2 n \log n)$ time and $O(nk)$ space. Algorithm 1 is advantageous for a large k value and Algorithm 2 is better if $k < n^{1/4}$. The basic idea of the algorithms is a procedure to detect a negative cycle in a network.

5.4 Generalization on Families

We could generalize the optimal quantization problems 1-2 in two ways. First we consider generalization of families of intervals. That is, we allow intervals of different lengths. One basic family is that of intervals of the form $[1, i]$, that is, $\mathcal{F}_{ed} = \{[1, 1], [1, 2], \ldots, [1, n]\}$. Another one is associated with a tree structure. That is, the whole interval $[1, n]$ is partitioned into two (non-intersecting) subintervals, and each of the subintervals is also partitioned into two smaller intervals in a recursive manner. This operation naturally corresponds to a tree. For simplicity, this tree has all entries 1 through n in its leaves. A family of intervals defined in this manner is denoted by $\mathcal{F}_{tree}$. See Fig. 11 for an example.

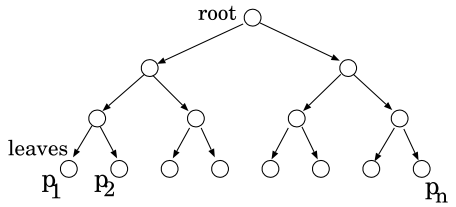


Fig. 11 A family of intervals of tree structure.

For the family $\mathcal{F}_{ed}$, the 1D-error diffusion gives us a feasible solution, which is also optimal in many cases.

5.5 More General Framework

We have seen that the L_∞-optimization problem in the one-dimensional case is solved in polynomial time in the array size if a family $\mathcal{F}$ of subsets of G_n is a set of intervals in $[1, n]$. What is the reason for it? To explain the reason we reformulate the problem in the form of linear programming.

We are interested in the L_1-optimization problem instead of the L_∞-optimization, which is described as follows:

minimize $\sum_{I_i \in \mathcal{F}} |\sum_{j \in I_i} a_j - \sum_{j \in I_i} b_j|$
subject to $0 \leqq a_j \leqq 1,\ j = 1, \ldots, n$, and
$b_j = 0, 1,\ j = 1, \ldots, n$.

If we introduce new variables $y_i = B(I_i)$ and constants $c_i = A(I_i)$ which depend only on input values, the problem can be restated by

minimize $\sum_{I_i \in \mathcal{F}} |y_i - c_i|$
subject to $y_i = \sum_{j \in I_i} b_j,\ i = 1, \ldots, m = |\mathcal{F}|$, and
$b_j = 0, 1,\ j = 1, \ldots, n$.

The constraints concerning the variables y_i are represented in a matrix form:

$$(-I, \mathcal{C}(G_n, \mathcal{F}))Y = 0,$$

where I is an identity matrix and $Y = (y_1, \ldots, y_m, b_{p_1}, \ldots, b_{p_n})^T$, and $\mathcal{C}(G_n, \mathcal{F}) = (c_{ij})$ is the incidence matrix defined by

$$c_{ij} = \begin{cases} 1 & \text{if } b_{p_j} \in I_i \\ 0 & \text{otherwise.} \end{cases}$$

The objective function is decomposed into m piecewise-linear functions by a standard conversion technique depicted in Fig. 12. That is, we break the function $|y_i - c_i|$ at integral points, $\lceil c_i \rceil$ and $\lfloor c_i \rfloor$ so that the new function passes through the points $(\lceil c_i \rceil, |\lceil c_i \rceil - c_i|)$ and $(\lfloor c_i \rfloor, |\lfloor c_i \rfloor - c_i|)$. Note that this conversion does not change the value of the objective function of an optimal solution since we are interested in its values at integers. Let $f_i(y_i)$ be the resulting function. Then, the LP relaxation of the problem is of the form:

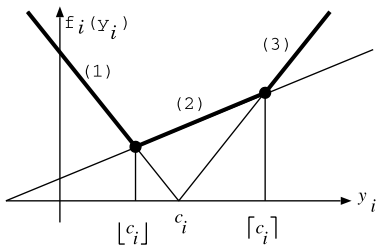


Fig. 12 Conversion of the convex objective function into a piecewise linear convex function.

minimize $\sum_{I_i \in \mathcal{F}} f_i(y_i)$
subject to $(-I, \mathcal{C}(G_n, \mathcal{F}))Y = 0$.

The most fundamental observation behind the arguments hereafter is the following theorem [22], [47]

Theorem 5.3 (Hoffman and Kruskal 1956): Let C be a totally unimodular matrix and let B be an integral vector. Then the polyhedron $P := \{x | Ax \leqq B\}$ is integral.

Based on the theorem, our linear program has an integer solution if the incidence matrix $\mathcal{C}(G_n, \mathcal{F})$ is *totally unimodular*, where a matrix C is totally unimodular if the determinant of each square submatrix of C is equal to $0, 1$, or -1.

Totally unimodular matrices have several interesting properties characterized in the following theorem [37]:

Theorem 5.4: The following statements are equivalent:

1. C is totally unimodular.
2. The transpose of C is totally unimodular.
3. (C, I) is totally unimodular.
4. A matrix obtained by deleting a unit row (column) of C is totally unimodular.
5. A matrix obtained by multiplying a row (column) of C by -1 is totally unimodular.
6. A matrix obtained by interchanging two rows (columns) of C is totally unimodular.
7. A matrix obtained by duplicating columns (rows) of C is totally unimodular.
8. A matrix obtained by a pivot operation on C is totally unimodular.

An $m \times n$ $(0, 1)$ matrix is called an *interval matrix* if in each column the 1's appear consecutively; that is, if $c_{ij} = c_{kj} = 1$ and $k > i + 1$, then $c_{li} = 1$ for all l with $i < l < k$. When we have a family of intervals, then the associated incidence matrix is characterized by the *consecutive 1s' property* in rows, not in columns, it is totally unimodular since the transposition preserves the total unimodularity. Formally the following theorem is known [37], [47]:

Theorem 5.5: Interval matrices are totally unimodular.

This theorem gives us an answer to our question on why the optimization problem on one-dimensional arrays for a family of intervals is solved in polynomnial time. The incidence matrix associated with a family of intervals is characterized by consecutive 1's property on rows, and thus its transpose is an interval matrix. Thus, by the facts 2 and 3 in the Theorem 5.4 and Theorem 5.5 we can conclude that our incidence matrix is totally unimodular.

Here is a small example. Let $G_n = \{1, 2, 3\}$ and $\mathcal{F} = \{S_1 = \{1, 2\}, S_2 = \{1, 3\}, S_3 = \{2\}\}$. The incidence matrix finds totally unimodular since it satisfies the consecutive 1's property for a permutation $(3, 1, 2)$.

On the other hand, there is no permutation giving the consecutive 1's property for a family $\mathcal{F}' = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$. In fact, the determinant of the corresponding 3×3 matrix is -2, and thus the incidence matrix is not totally unimodular. This small example suggests that even one-dimensional version of the optimization problem may have no polynomial-time algorithm.

The bound (1) on the maximum rounding error is sharpened in a more general framework. A pair $(G_n, \mathcal{F})$ is considered as a hypergraph. Then, it is called unimodular if the incidence matrix associated with $(G_n, \mathcal{F})$ is totally unimodular. We have just seen that a hypergraph induced by a family of intervals is unimodular. As is stated above, the problem of minimizing the maximum rounding error can be formulated as an integer programming problem, and the unimodularity implies that its LP relaxation has an integral solution. A classical theorem of Ghouila-Houri [20] implies that totally unimodularity is a necessary and sufficient condition for the existence of a rounding with the maximum error less than 1. Moreover, the following sharpened result is given by Doerr [16]:

Theorem 5.6: If $(G_n, \mathcal{F})$ is a unimodular hypergraph, there exists a rounding $B = (b_j)$ of $A = (a_j)$ satisfying

$$\left|\sum_{j\in R} a_j - \sum_{j\in R} b_j\right| \leqq \min\left\{1 - \frac{1}{n+1}, 1 - \frac{1}{m}\right\} \quad (2)$$

for every $R \in \mathcal{F}$, where $m = |\mathcal{F}|$.

This theorem also holds for two-dimensional rounding, and this bound is sharp. Furthermore, such roundings can be computed efficiently [16].

5.6 Generalization on Metric

So far we have been interested in the objective function of the form $\max_{I_i\in\mathcal{F}} |y_i - c_i|$ or $\sum_{I_i\in\mathcal{F}} |y_i - c_i|$, which corresponds to L_∞ and L_1 metrics, respectively. A natural extension of the objective function is the following one based on L_p metric:

$$\left[\sum_{I_i\in\mathcal{F}} |y_i - c_i|^p\right]^{1/p},$$

where p is an integer $\geqq 1$. When p is finite, we can replace the objective function by

$$\sum_{I_i\in\mathcal{F}} |y_i - c_i|^p.$$

It is easy to see that this is a separable convex function. Now, we can apply the ideas of Minoux [32] and Hochbaum-Shanthiskumar [21] to replace $|y_i - c_i|^p$ by a piecewise linear convex continuous function $f_i(y_i)$ which is equal to $|y_i - c_i|^p$ for each integral value of y_i in $[0, |I_i|]$.

Polynomial-time solvability of our problem for L_p metric is based on the following theorem [21]:

Theorem 5.7: [Hochbaum and Shanthikumar 1990] Nonlinear separable convex optimization problem $\min\{\sum_{i=1}^n f_i(x_i) \mid A\boldsymbol{x} \geqq \boldsymbol{b}\}$ on linear constraints with a totally unimodular matrix A can be solved in polynomial time.

5.7 Algorithms Based on Network Flow

The observation in the previous subsection gives us a new light to our problem. However, those observations do not lead to lower-degree polynomial time algorithms. To have such algorithms, we reduce the optimization problem to that of finding a *minimum-cost network flow*.

To begin with, we shall consider a family of all intervals of length $k, k \geqq 2$, that is, $\mathcal{F} = \{I_1 = [1, k], I_2 = [2, k + 1], \ldots, I_{n-k+1} = [n - k + 1, n]\}$. We define a network associated with G_n and $\mathcal{F}$ as follows. For each interval $I_i, i = 1, \ldots, n-k+1$, we create a node v_i. We have two kinds of edges, forward edges and backward edges. For each $i = 1, \ldots, n-k$ we have a forward edge $e_i = (v_i, v_{i+1})$, which corresponds to the ith interval I_i. Each backward edge is associated with an element of an input array. The backward edge f_i associated with the ith element a_i is directed from v_a to v_b where I_a is the first interval containing the index i and I_b is the last one containing i. See Fig. 13.

For each forward edge e_i associated with the ith interval $[i, i + k - 1]$ we define

$$F(e_i) = \sum_{j=i}^{i+k-1} a_j,$$

and for each backward edge f_j associated with the jth element a_j we define

$$F(f_j) = a_j.$$

Using these values we define capacity $cap(e)$ and cost $cost(e)$ for each edge e:

$cap(e) = [\lfloor F(e) \rfloor, \lceil F(e) \rceil]$ and

$cost(e_i) = \lceil F(e_i) \rceil - \lfloor F(e_i) \rfloor$, for each forward edge e_i, and

$cost(f_j) = 0$, for each backward edge f_j.

Now, it is easy to find a feasible flow in this network. A flow defined by $F(e)$ on every edge e is a feasible flow since it trivially satisfies the capacity constraints and flow conservation law. Due to integrality of the capacity of each edge, we can guarantee the existence of an integer solution satisfying the capacity constraint. This feasible flow gives us a feasible solution to our rounding problem, that is, given an input array $A = (a_1, \ldots, a_n)$ and a family $\mathcal{F}$ of intervals on $[1, n]$, we can find a binary array B such that $|A(I) - B(I)| < 1$ for any interval $I \in \mathcal{F}$.

Moreover, we can also solve the optimization problem by taking the costs of edges into accounts. What is required is to find a minimum-cost network flow in the network associated with an input array and a family of intervals. Using the scaling algorithm by Edmonds and Karp [17], we can find an optimal rounding in time $O(|E| \log U(|E| + |V| \log |V|))$ for a network with node set V and edge set E and the largest integral capacity U. In our case, $|V|, |E|$ and U are all $O(n)$, and thus we have an $O(n^2 \log^2 n)$-time algorithm.

The idea of a minimum-cost network flow is extended to other families of intervals. An easy example is a family of intervals $\{[1, 2], [1, 3], [1, 4], \ldots, [1, n]\}$. An example of the corresponding network is shown in Fig. 14. It is easily shown that there is a feasible flow in the network as above. This guarantees the existence of a feasible rounding as the 1D-Error Diffusion finds one such rounding. Error Diffusion algorithm only guarantees that a solution obtained is always a feasible solution for rounding, but the framework we have just established gives us an optimal rounding in polynomial time in the array size.

We have also introduced a family $\mathcal{F}_{tree}$ of intervals with a tree structure. It is also easy to define a network for this family. The family is obtained by recursively partitioning an interval into two disjoint subintervals until we reach unit intervals, which form leaves of the tree. Tree edges are directed from the root to the leaves. Then, we create another node and draw an edge from every leaf. Finally, we draw an edge from the last node to the root of the tree. We associate each tree edge e_i for an interval I_i with the value $F(e_i) = \sum_{j \in I_i} a_j$ as before to define its capacity by $[\lfloor F(e_i) \rfloor, \lceil F(e_i) \rceil]$. For each edge e_j incident to a leaf $[j, j]$ or a_j, we set $F(e_j) = a_j$. Costs of tree edges are defined as before and costs of remaining edges are defined to be none. An example of the corresponding network is shown in Fig. 15.

5.8 Generalized Intervals

So far we have implicitly assumed a natural index ordering, that is, from 1 to n, and defined intervals on the ordering. However, there is no reason to fix the index order. We could consider any index order (or a permutation of $(1, \ldots, n)$). It is easy to see that the whole arguments so far in this section are not affected by this change. This observation is important because it suggests us to have more than one family of intervals. To begin with we shall consider two different index orderings $\Sigma_1 = (\sigma_1(1), \ldots, \sigma_1(n))$ and $\Sigma_2 = (\sigma_2(1), \ldots, \sigma_2(n))$. Then, for each ordering we can define a family of intervals $\mathcal{F}_1$ and $\mathcal{F}_2$. Surprisingly, we can combine two optimization problems for the two families.

Suppose that two families $\mathcal{F}_1$ and $\mathcal{F}_2$ are both de-

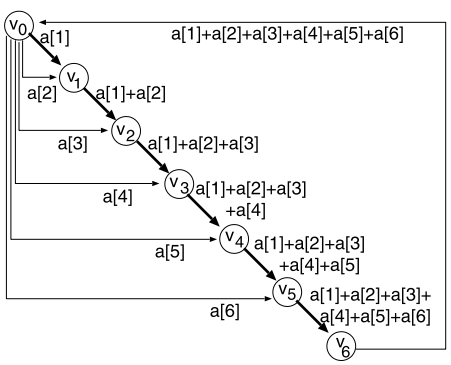


Fig. 14 A network for a family of intervals $\{[1, i], i = 1, \ldots, n\}$.

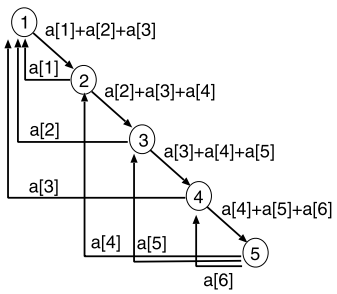


Fig. 13 A network for a family of all intervals of the same length.

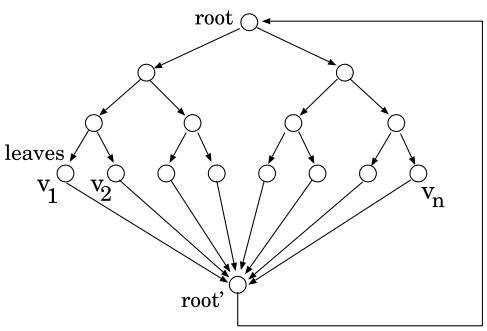


Fig. 15 A network for a family of intervals with a tree structure.

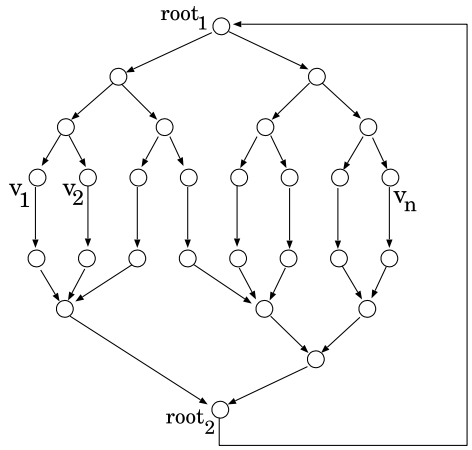


Fig. 16 A network for two tree structures.

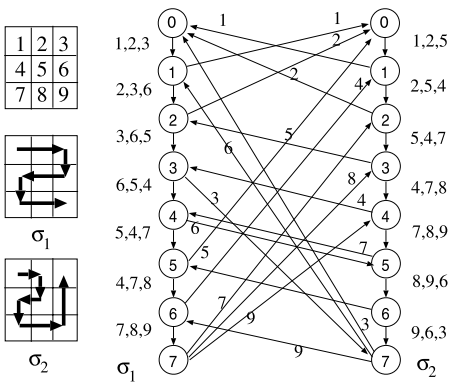


Fig. 17 A network for two space-filling curves.

fined by tree structures. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the trees associated with $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. We direct the edges in the two trees in an opposite way, that is, from root to leaves in $\mathcal{T}_1$ and from leaves to root in $\mathcal{T}_2$. Then, we connect the corresponding leaves with edges of no cost. Finally, we connect the two roots by an edge with no cost. The resulting network is shown in Fig. 16.

Now, it may be obvious that the same arguments hold for the network. This construction also suggests another important application to two-dimensional rounding. Although it is described in detail in the following section, the two-dimensional rounding is in general more difficult than the one-dimensional rounding. This is the reason why space-filling curves are applied to two dimensional rounding by converting the problem from 2D to 1D. As is shown in [7], a polynomial-time algorithm for rounding a one-dimensional array is applied to halftone a two-dimensional image. However, the framework just described above suggests us to optimize roundings along two space-filling curves simultaneously. See Fig. 17.

In the same manner we can combine two tree structures defined on different index orderings. We have seen a network defined for a family consisting of two different tree structures in Fig. 16. They are defined on the same index ordering, but this time we define two tree structures on different index orderings. Thus, the part connecting two trees are different. See Fig. 18.

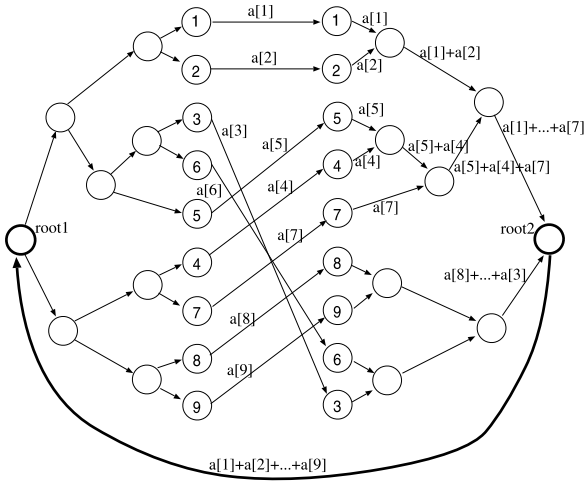


Fig. 18 A network for a family defined by two interval sets with tree structures on different index orderings.

6. Two-Dimensional Problem

6.1 Basic Observation

The main difference from the one-dimensional (1D) case is, of course, connectivity of elements. In the 1D case the connectivity of elements is defined by continuity in a given index order and a set of connected elements forms an interval on the ordering. On the other hand, in the 2D case, the index ordering is usually fixed and thus connectivity among elements is defined adjacency in the matrix (or plane). Given an $N \times N$ input real-valued matrix $A = (a_{ij})$, we define the index set G_n by

$$\begin{aligned} G_n &= \{(1,1),(1,2),\ldots,(N,N)\} \\ &= \{p_1, p_2, \ldots, p_n\}, \end{aligned}$$

where $n = N \times N$ denotes the total number of elements as before.

Then, a family $\mathcal{F}$ of regions over the $N \times N$ grid is specified as a subset of G_n. Given a family $\mathcal{F}$ on G_n, the incidence matrix $\mathcal{C}(G_n, \mathcal{F}) = (c_{ij})$ is defined by

$$c_{ij} = \begin{cases} 1 & \text{if } p_i \in R_j, \\ 0 & \text{otherwise} \end{cases}$$

The polynomial-time solvability of the optimization problem depends on total unimodularity of the incidence matrix. This is just the same as in the 1D case. Now we have a question: what family of regions is characterized by a totally unimodular incidence matrix? In the 1D case the consecutive 1's property is a good property to characterize totally unimodular matrices and that is why we have considered a family of

intervals. In the 2D case, if we ordered all the elements in a sequence, then we could define intervals on the sequence. An important difference here is that each element could be contained in many intervals in the one-dimensional case while it is hard in the 2D case. Fixed an index ordering, one element could be contained in many intervals. Fixing an index ordering means considering some space-filling curve on the image plane. If we have a raster order, we cannot have an interval with two-dimensional spread. We could have intervals corresponding to $k \times k$ regions for any $k \geqq 1$ by just arranging the pixels (matrix elements) so that the whole sequence is partitioned into disjoint subsequences, each forming a 2D region of area k^2. So, a family of intervals of the same length is meaningful in the 1D case but not in the 2D case because shifting one position in the 2D pixel sequence does not preserve 2D shapes of the corresponding regions. This is the difficulty in the 2D case.

6.2 Some Special Cases

To the author's knowledge the first result on matrix rounding is the following theorem [9], [11], [12]:

Theorem 6.1: [Baranyai 1974] Given a real-valued matrix $A = (a_{ij})$ and a family $\mathcal{F}$ of regions consisting of all rows, all columns and the whole matrix, there exists an integer-valued matrix $B = (b_{ij})$ such that $|\sum_{(i,j)\in R} a_{ij} - \sum_{(i,j)\in R} b_{ij}| < 1$ holds for every $R \in \mathcal{F}$.

Here is a simple example to explain the proposition. An input is a 3×3 matrix A:

$$A = \begin{pmatrix} 0.4 & 0.4 & 0.7 \\ 0.9 & 0.3 & 0.8 \\ 0.4 & 0.3 & 0.5 \end{pmatrix}.$$

For this matrix, we can define a network G as follows. G has two sets of nodes, one for all rows and the other for all columns. Let r_1, r_2, and r_3 be three nodes for rows 1, 2, and 3. Similarly, let c_1, c_2, and c_3 be those for all columns. In addition to them, it has two special nodes s and t. Then, for each row $i, i = 1, 2, 3$ we draw an edge from s to r_i and associate the corresponding row sum $\sum_{j=1}^{3} A_{ij}$ with the edge. Similarly, for each column $j, j = 1, 2, 3$ we draw an edge from c_j to t and associate the corresponding column sum $\sum_{i=1}^{3} Aij$ with the edge. We also connect all pairs of (r_i, c_j) and associate the matrix element A_{ij} with the edge. Finally, we draw an edge from t to s and associate the total sum $\sum_{i=1}^{3}\sum_{j=1}^{3} A_{ij}$ with the edge. See Fig. 19.

If we determine the capacity of each edge by the floor and ceiling values of its associated value, the graph defined above is a network flow graph. It is also easy to see that a set of associated values defines a feasible flow on the network. Then, the integrality property of the network flow, we can guarantee the existence of an integer flow that satisfies the capacity constraint. This integer flow gives us a binary matrix we want.

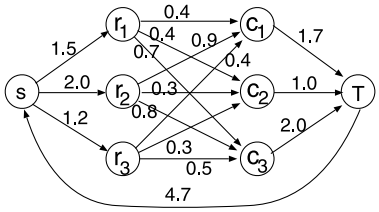


Fig. 19 A network corresponding to a matrix.

In general, the rounding error is highly dependent on the family $\mathcal{F}$ of regions.

The following theorem is well-known [12]:

Theorem 6.2: Given a $[0, 1]$-valued matrix of size $n = N \times N$ and a family of all rectangular regions over the matrix, for any binary matrix B of the same size the maximum rounding error $\max_{R\in\mathcal{F}} |A(R) - B(R)|$ is $O(\log^3 n)$ and $\Omega(\log n)$. The same bounds hold for the maximum rounding error for the family of all rectangular regions containing the left-upper corner entry of the matrix.

For a family $\mathcal{F}_k$ of all $k \times k$ square regions over the $N \times N$ matrix, we have the following theorem [6]:

Theorem 6.3: The maximum rounding error with respect to $\mathcal{F}_k$ is $O(\log^3 k)$ and $\Omega(\log k)$. Indeed, these bounds also hold for the union $\cup_{j=1}^{k} \mathcal{F}_j$.

It is combinatorially attractive to give better bounds for a small fixed constant k, and the problem seems to be highly nontrivial even for $k = 2$. A negative result is known [4].

Theorem 6.4 (Asano-Matui-Tokuyama 2000):
Given a $[0, 1]$-valued matrix A, a family $\mathcal{F}$ of all 2×2 regions over the matrix, and an arbitrary small number $\epsilon > 0$, the problem of determining whether there is a binary matrix B of the same size such that the maximum rounding error $\max_{R\in\mathcal{F}} |A(R) - B(R)|$ is greater than $1 - \epsilon$ or less than $1/2 + \epsilon$ is NP-hard.

A positive result is also known [6].

Theorem 6.5: [Asano-Tokuyama 2001] Given a $[0, 1]$-valued matrix A and a family $\mathcal{F}$ of all 2×2 regions over the matrix, there is a binary matrix B of the same size such that the maximum rounding error $\max_{R\in\mathcal{F}} |A(R) - B(R)|$ is bounded by 5/3.

Although no algorithm is given in the literature [6], it is not so hard to derive a strongly polynomial-time algorithm for finding such an approximate solution.

The above result may look rather easy. In fact, it is trivial to achieve the error bound 2 since error caused by the simple thresholding is 0.5 for each pixel and thus the total error in any 2×2 region is at most 2. Thus the improvement obtained in the theorem is only 1/3.

There is yet another observation concerning

bounds on the maximum rounding error. The maximum rounding error can be 0 since we may have an integer-valued matrix as an input. Then, how large can the maximum rounding error be? One non-trivial bound is 1. That it, there is an input matrix for which the maximum rounding error must be at least 1 for any possible rounding. Suppose we have a matrix consisting of 1/2 and 0 and a graph representing adjacency of non-zero elements has a simple cycle of odd length (see Fig. 20 for an example). To make the rounding error in each 2×2 square region containing two 1/2 elements less than 1, the corresponding region in an output matrix must have exactly one 1 in it. But it is impossible if the length of the cycle is odd. Thus, the rounding error must be at least 1 in this case, and we can achieve the bound.

These results can be compared with the performance of the error diffusion algorithm which propagates rounding error at a pixel (i, j) to unprocessed pixels in its neighborhood as follows:

$$\begin{array}{ccc} & \bullet & \alpha \\ \beta & \gamma & \delta \end{array}$$

Then, the performance of the algorithm is described by

Theorem 6.6: Given a $[0, 1]$-valued matrix A and a family $\mathcal{F}_k$ of all $k \times k$ regions over the matrix, there is a binary matrix B of the same size such that the maximum rounding error $\max_{R \in \mathcal{F}} |A(R) - B(R)|$ is bounded by $k + (k-1)(\gamma + \delta)$. Moreover, there is an instance A such that the maximum rounding error exceeds $k + (k-1)(\gamma + \delta) - \epsilon$ for any $\epsilon > 0$ if we apply the error diffusion algorithm.

When $k = 2$, the rounding error is $2 + \gamma + \delta$, which is 2.375 for the standard error diffusion algorithm with $\gamma = 5/16$ and $\delta = 1/16$. This implies that the error diffusion may be worse than the simple thresholding in the sense of the worst rounding error.

6.3 Optimization Algorithms

The Baranyai's theorem only guarantees the existence of the feasible solution to the rounding problem. However, applying the minimum-cost flow argument in the previous section, we can solve the optimization version

0	0	0	0	0	0	0	0
0	0	1/2	1/2	1/2	0	0	0
0	1/2	0	0	0	1/2	0	0
0	1/2	0	0	0	1/2	0	0
0	0	1/2	1/2	1/2	1/2	0	0
0	0	0	0	0	0	0	0

Fig. 20 A matrix containing an odd cycle.

of the problem in the theorem, that is, we can find in polynomial-time an integer-valued matrix that not only satisfies all the constraints in the theorem but also optimizes some criterion associated with rounding errors.

There is another interpretation of the theorem using the notion of total unimodularity. In this case the family of regions consists of all rows, all columns, and the whole matrix. To prove that the incidence matrix associated with this family is totally unimodular, we introduce *network matrices* which is defined as follows [47], [48]:

Let $D = (V, A)$ be a directed graph and let $T = (V, A_0)$ be a directed tree on V. Let M be the $A_0 \times A$-matrix defined by, for $a = (v, w) \in A$ and $a' \in A_0$:

$$M_{a',a} = \begin{cases} +1 & \text{if } P(v, w, T) \text{ passes through } a' \text{ forwardly;} \\ -1 & \text{if } P(v, w, T) \text{ passes through } a' \text{ backwardly;} \\ 0 & \text{if } P(v, w, T) \text{ does not pass through } a'. \end{cases}$$

Here, $P(v, w, T)$ denotes the unique $v - w$ path in T. Matrices arising in this way are called network matrices. It is known that network matrices are totally unimodular. More precisely see the following theorem:

Theorem 6.7: Network matrices have the following properties:

1. They are closed under row and column deletions and duplications.
2. They are closed under multiplication of a column by -1.
3. If A is a network matrix, then (A, I) is a network matrix.
4. They are closed under pivoting.
5. They are totally unimodular.

The incidence matrix associated with the family of all rows, all columns, and the whole matrix is in fact a network matrix. Given an $N \times N$ matrix A, we build a graph $D(A)$ by the complete bipartite graph $K_{N,N}$ associated with N rows and N columns with two special vertices s and t with edges connecting s to all row vertices and t to all column vertices. A small example is shown in Fig. 21. The, we build a tree $T(A)$ defined by the same vertex set and the edges from s to all column vertices and those from all row vertices to t. Then, the matrix M defined by D and T is a network matrix by the theorem. Now, delete the columns for the graph edges from s to row vertices and from column vertices to t. Then, the resulting matrix remains a network matrix, and it coincides with the incidence matrix associated with the family of all rows, all columns, and the whole matrix.

Another useful observation is the following:

Theorem 6.8: The node-edge incidence matrix of a bipartite graph is totally unimodular.

This theorem implies that a family of regions de-

fined by two different partitions of an image plane induces a totally unimodular incidence matrix. Hereafter, a family of regions is called a *unimodular family* if its associated incidence matrix is totally unimodular. Formally, a family $\mathcal{F} = \{R_1, R_2, \ldots, R_m\}$ is called a *partition family* (or a partition) of G_n if $\bigcup_{i=1}^{m} R_i = G_n$ and $R_i \cap R_j = \emptyset$ for any $R_i \neq R_j$ in $\mathcal{F}$. A *k-partition family* is a family of regions on a matrix which is a union of k different partitions of G_n.

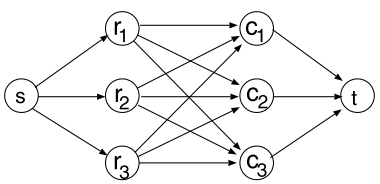
A family $\mathcal{F}$ of regions on a grid G_n is a *laminar family* if one of the following holds for any pair R_i and R_j in $\mathcal{F}$: (1) $R_i \cap R_j = \emptyset$, (2) $R_i \subset R_j$ and (3) $R_j \subset R_i$. The family is also called a laminar decomposition of the grid G_n. In general, a *k-laminar family* is a family of regions on a matrix which is a union of k different laminar families. Laminar families have a nice property [5]:

Theorem 6.9 (Asano-Katoh-Obokata-Tokuyama 2002): A 2-laminar family is unimodular.

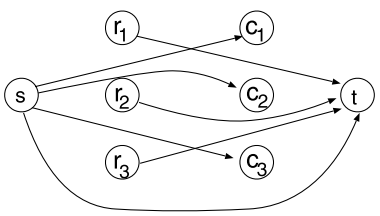
A connected region over a matrix is called a *tile* if we can cover the matrix by repeated use of the region without any overlap. A tile is represented by specifying its components in a general form. For example, a 2×2-tile with its upper left corner at even entry (an entry such that the sum of its row and column indices is even) over an $N \times N$ matrix is represented by $T_1 = \{(2i, 2j), (2i, 2j+1), (2i+1, 2j), (2i+1, 2j+1)\}_{i,j=0,\ldots,(N-1)/2}$.

Direct applications of Theorem 6.9 lead to various unimodular families of regions. The family of regions defined in Baranyai's theorem is a 2-laminar family. Also, take any 2−partition family consisting of 2×2 regions on a matrix. For example, take all 2×2 regions with their upper left corners located in even points (where the sums of their row and column indices are even). The set of all those regions defines two partition families $\mathcal{F}_{even}$ and $\mathcal{F}_{odd}$ where $\mathcal{F}_{even}$ (resp. $\mathcal{F}_{odd}$) consists of all 2×2 squares with their upper left corners lying at even (resp. odd) rows (see Fig. 22). Since any 2×2 region is partitioned into two 2×1 regions, with these 2×2 regions and 2×1 regions we have a 2-laminar family.

A 3−partition family is not unimodular in general. However, there are some families which are not 2-laminar but unimodular: For example, the set of all rectangular rigid submatrices of size 2 (i.e., domino tiles) is a 4-partition family, but it is unimodular.



(a) graph D.



(b) tree T.

Fig. 21 (a) A graph D associated with the family of regions in the Baranyai's theorem, and (b) a tree T with the same vertex set.

6.4 Approximation Algorithm

In this section we consider upper bounds for the generalized maximum rounding error of a real-valued matrix. Formally, we define a distance $Dist_p^{\mathcal{F}}(A, A')$ between two $[0,1]$-matrices A and A' of the same size and a positive integer p by

$$Dist_p^{\mathcal{F}}(A, A') = \left[\sum_{R \in \mathcal{F}} |A(R) - A'(R)|^p \right]^{1/p}.$$

What we are interested in is upper bounds on the L_p-Discrepancy bound.

***L_p*-Discrepancy Bound:**
Given a $[0,1]$-matrix $A \in \mathcal{A}$, a family $\mathcal{F}$ of subsets of G_n, and a positive integer p, investigate upper and lower bounds of

$$\mathcal{D}(G_n, \mathcal{F}, p) = \sup_{A \in \mathcal{A}} \min_{B \in \mathcal{B}} Dist_p^{\mathcal{F}}(A, B).$$

The pair $(G_n, \mathcal{F})$ defines a hypergraph on G_n, and $\mathcal{D}(G_n, \mathcal{F}, \infty)$ is called the *inhomogeneous discrepancy* of the hypergraph [12]. Abusing the notation, we call $\mathcal{D}(G_n, \mathcal{F}, p)$ the (inhomogeneous) L_p-discrepancy of the hypergraph, and also often call $Dist_p^{\mathcal{F}}(A, B)$ the L_p-discrepancy measure of (quality of) the output B with respect to $\mathcal{F}$.

A recent result for a unimodular family is the following [5].

Theorem 6.10: If $\mathcal{F}$ is unimodular and $p \leqq 3$, for any $A \in \mathcal{A}$ we have

$$\min_{B \in \mathcal{B}} Dist_p^{\mathcal{F}}(A, B) \leqq \frac{1}{2} |\mathcal{F}|^{1/p}.$$

For the case $p > 3$, we have the following [5]:

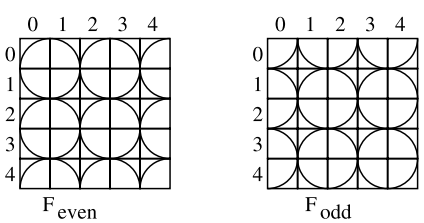


Fig. 22 2-partition family of 2×2 regions.

Theorem 6.11: If $\mathcal{F}$ is unimodular and $p > 3$, for any $A \in \mathcal{A}$ we have

$$\min_{B\in\mathcal{B}} Dist_p^{\mathcal{F}}(A,B) \leqq (p^p/(p+1)^{p+1}+2^p(p-1)/(p+1)^{p+1})^{1/p}|\mathcal{F}|^{1/p}.$$

The method described above does not work for a non-unimodular case. A simple but interesting family defining a non-unimodular hypergraph is the family of all 2×2 regions of A. The known upper bound is merely $\frac{5}{3}|\mathcal{F}|^{1/p}$ [6]. The following result is obtained [5].

Theorem 6.12: For any $A \in \mathcal{A}(G_n)$, and a family $\mathcal{F}$ of 2×2 regions of the matrix, (1) we have

$$\min_{B\in\mathcal{B}} Dist_1^{\mathcal{F}}(A,B) \leqq \frac{3}{4}|\mathcal{F}|,$$

and (2) we can find in polynomial time a $\{0,1\}$-valued matrix B such that

$$\sum_{R\in\mathcal{F}} |A(R)-B(R)| \leqq \sum_{R\in\mathcal{F}} |A(R)-B^*(R)| + \frac{9}{16}|\mathcal{F}|,$$

where B^* is the (unknown) optimal solution.

Fig. 23 Experimental results for "Wool." Output image by error diffusion algorithm (above) and that by the algorithm based on the minimum-cost network flow (below).

This is the first result concerning the performance of an approximation algorithm for the matrix rounding problem.

7. Experimental Evaluation

Most of the rounding algorithms described in this paper have been implemented. Among them, the matrix rounding algorithm based on the minimum-cost network flow is implemented using the library functions in LEDA [27] and compared with the error diffusion algorithm in the quality of output images. The data used for the experiments are *Standard high precision picture data* created by the Institute of Image Electronics Engineers of Japan, which include four standard pictures called, "Bride," "Harbor," "Wool," and "Bottles." They are color pictures of 8 bits each in RGB. Their original picture size is 4096×3072. In their experiments they are scaled down to 1024×768 in order to shorten the running time of the program. Figure 23 show experimental results for "Wool" to compare our algorithm with error diffusion. See Figs. 24–26 for other examples. The algorithm has been implemented using a 2-laminar family defined by two tiles

Fig. 24 Experimental results for "Harber."

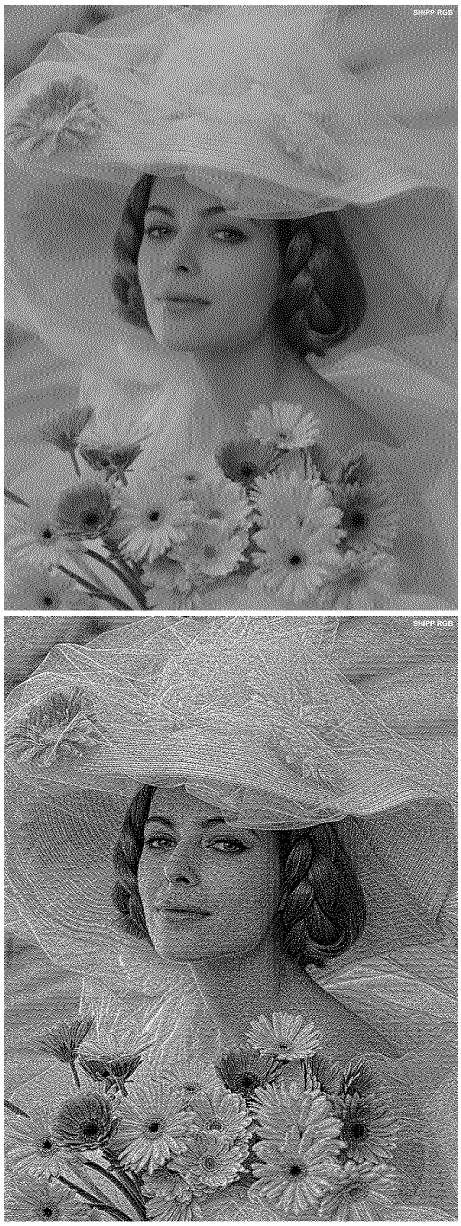
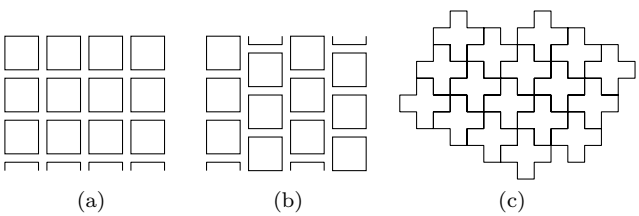


Fig. 25 Experimental results for "Bride."

Fig. 26 Experimental results for "Bottles."

among ones depicted in Fig. 27. Comparing the quality of output images between error diffusion and the algorithm, the algorithm produces images better in the sense of discrepancy. Although a low discrepancy image does not always mean a better image in human-eyes' criterion, the outputs are often favorite with humans if we use the L_p measure (in our experiment, $p = 1$). This is quite interesting, since it is difficult to have such a nice-looking output by using the L_∞ measure in the experiments.



(a) (b) (c)

Fig. 27 Three different partitions of the image plane (a) by 2×2 squares, (b) vertically shifted 2×2 squares, and (3) cross patterns consisting of 5 pixels.

8. Concluding Remarks

This paper surveys recent studies on the optimization-based evaluation of digital halftoning algorithms. There are a lot of open problems that are interesting from both of theory and practice. However, the most important open problem might be to find the best mathematical criterion which is faithful to the hu-

man visual system and simultaneously leads to some polynomial-time algorithm.

Acknowledgment

The author expresses his sincere thanks to Takeshi Tokuyama (Tohoku Univ.), Koji Obokata (JAIST), Naoki Fujikawa (former JAIST student), Tomomi Matsui (Tokyo Univ.), Naoki Katoh (Kyoto Uni.), Hisao Tamaki (Meiji Univ.), Hiroshi Nagamochi (Toyohashi Univ. Tech.), and Nobuaki Usui (Fujitsu Lab.) who are coauthors of the author's papers on digital halftoning. He also would like to thank Hiro Ito (Kyoto Univ.), David Mount (Maryland Univ., U.S.A.), Masashi Kiyomi (Tokyo Univ.), and Koji Nakano (JAIST).

The work has been partially supported by the Scientific Grant-in-Aid by the Ministry of Education, Culture, Sports, Science and Technology of Japan and the Kayamori Foundation of Informational Science and Advancement.

References

[1] M. Analoui and J.P. Allebach, "Model-based halftoning by direct binary search," Proc. SPIE/IS&T Symposium on Electronic Imaging Science and Technology, vol.1666, pp.96–108, 1992.

[2] T. Asano, "Digital halftoning algorithm based on random space-filling curve," IEICE Trans. Fundamentals, vol.E82-A, no.3, pp.553–556, March 1999.

[3] T. Asano, N. Katoh, H. Tamaki, and T. Tokuyama, "Convertibility among grid filling curves," Proc. ISAAC98, Springer LNCS 1533, pp.307–316, 1998.

[4] T. Asano, T. Matsui, and T. Tokuyama, "Optimal roundings of sequences and matrices," Nordic Journal of Computing, vol.7, no.3, pp.241–256, Fall 2000.

[5] T. Asano, K. Obokata, N. Katoh, and T. Tokuyama, "Matrix rounding under the L_p-discrepancy measure and its application to digital halftoning," Proc. ACM-SIAM Symposium on Discrete Algorithms, pp.896–904, San Francisco, 2002.

[6] T. Asano and T. Tokuyama, "How to color a checkerboard with a given distribution — Matrix rounding achieving low 2×2-discrepancy," Proc. ISAAC01, pp.636–648, Christchurch, 2001.

[7] T. Asano, D. Ranjan, and T. Roos, "Digital halftoning algorithms based on optimization criteria and their experimental evaluation," IEICE Trans. Fundamentals, vol.E79-A, no.4, pp.524–532, April 1996.

[8] B.E. Bayer, "An optimum method for two-level rendition of continuous-tone pictures," Conference Record, IEEE International Conference on Communications, vol.26, pp.11–15, 1973.

[9] Z. Baranyai, "On the factorization of the complete uniform hypergraphs," in Infinite and Finite Sets, ed. A. Hanaj, R. Rado and V.T. Sós, Colloq. Math. Soc. J'anos Bolyai 10, pp.91–108, 1974.

[10] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwartzkopf, Computational Geometry: Algorithms and Applications, Springer-Verlag, 1997.

[11] B. Bollobás, Combinatorics, Cambridge University Press, 1986.

[12] J. Beck and V.T. Sös, "Discrepancy theory," in Handbook of Combinatorics Volume II, ed. R.Graham, M. Grötschel, and L Lovász, Elsevier, 1995.

[13] L.G. Casado, I. Garcia, P.G. Szabó, and T. Csendes, "Packing equal circles in a square I—Problem setting and bounds for optimal solutions," in New Trends in Equilibrium Systems, pp.191–206, Kluwer Academic Publishers, Boston, 2000.

[14] L.G. Casado, I. Garcia, P.G. Szabó, and T. Csendes, "Packing equal circles in a square II—New results for up to 100 circles using the TAMSASS-PECS algorithm," in New Trends in Equilibrium Systems, pp.207–224, Kluwer Academic Publishers, Boston, 2000.

[15] B. Chazelle, The Discrepancy Method: Randomness and Complexity, Cambridge University Press, 2000.

[16] B. Doerr, "Lattice approximation and linear discrepancy of totally unimodular matrices — Extended abstract," SIAM-ACM Symposium on Discrete Algorithms, pp.119–125, 2001.

[17] J. Edmonds and R.M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," J. ACM, vol.19, pp.248–264, 1972.

[18] R.W. Floyd and L. Steinberg, "An adaptive algorithm for spatial gray scale," SID 75 Digest, Society for Information Display, pp.36–37, 1975.

[19] R. Geist, R. Reynolds, and D. Suggs, "Markovian framework for digital halftoning," ACM Trans. Graphics, vol.12, no.2, pp.136–159, 1993.

[20] A. Ghoulia-Houri, "Characterisation des matrices totalement unimodulaires," C.R. Acad/ Sci. Paris, vol.254, pp.1192–1194, 1962.

[21] D.S. Hochbaum and J.G. Shanthikumar, "Nonlinear separable optimization is not much harder than linear optimization," J. ACM, vol.37, no.4, pp.843–862, 1990.

[22] A.J. Hoffman and J.B. Kruskal, "Integral boundary points of convex polyhedra," in Linear Inequalities and Related Systems, ed. H.W. Kuhn and A.W. Tucker, pp.223–246, Princeton Univ. Press, Princeton, NJ, 1956.

[23] J.F. Jarvis, C.N. Judice, and W.H. Ninke, "A survey of techniques for the display of continuous-tone pictures on bilevel displays," Computer Graphics and Image Processing, vol.5, pp.13–40, 1976.

[24] D.E. Knuth, "Digital halftones by dot diffusion," ACM Trans. Graphics, vol.6, no.4, pp.245–273, 1987.

[25] Kodera ed., Practical Design and Evaluation of Halftoned Images, Trikepps, 2000.

[26] D.L. Lau and G.R. Arce, Modern Digital Halftoning, Marcel Dekker, New York, 2001.

[27] LEDA homepage: http://www.algorithmic-solutions.com/as_html/products/products.html.

[28] J.O. Limb, "Design of dither waveforms for quantized visual signals," Bell Syst. Tech. J., vol.48, no.7, pp.2555–2582, 1969.

[29] Q. Lin, "Halftone image quality analysis based on a human vision model," Proc. SPIE, vol.1913, pp.378–389, Feb. 1993.

[30] B. Lippel and M. Kurland, "The effect of dither on luminance quantization of pictures," IEEE Trans. Commun., vol.COM-6, pp.879–888, 1971.

[31] J. Matoušek, Geometric Discrepancy, Springer, 1991.

[32] M. Minoux, Solving Integer Minimum Cost Flows with Separable Cost Objective Polynomially, Mathematical Programming Study 26, pp.237–239, 1986.

[33] T. Mitsa and K.J. Parker, "Digital halftoning technique using a blue-noise mask," J. Opt. Soc. Am., A/vol.9, no.11, pp.1920–1929, 1992.

[34] M. Miyahara, K. Kotani, and V.R. Algazi, "Objective picture quality scale (PQS) for image coding," IEEE Trans. Commun., vol.46, no.9, pp.1215–1226, Sept. 1998.

[35] R. Morelli, "Pick's theorem and Todd class of a toric variety," Adv. Math., vol.100, pp.183–231, 1993.
[36] R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.
[37] G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, 1999.
[38] K.J. Nurmera and Östergård, "Packing up to 50 equal circles in a square," Discrete Comput. Geom., vol.18, pp.111–120, 1997.
[39] K.J. Nurmera and Östergård, "More optimal packings of equal circles in a square," Discrete Comput. Geom., vol.22, pp.439–457, 1999.
[40] V. Ostromoukhov, R.D. Hersh, and I. Amidror, "Rotated dispersed dither: A new technique for digital halftoning," Proc. SIGGRAPH '94, pp.123–130, 1994.
[41] R. Peikert, D. Würtx, M. Monagan, and C. de Groot, "Packing circles in a square: A review and new results," in System Modelling and Optimization (Proc. 15th IFIP Conf., Zürich, 1991), ed. P. Kall, Lecture Notes in Control and Information Sciences, vol.180, pp.45–54, Springer-Verlag, Berlin, 1992.
[42] F.P. Preparata and M.I. Shamos, Computational Geometry: An Introduction, Springer-Verlag, 1990.
[43] P. Raghavan, Randomized rounding and discrete ham-sandwitch theorems, Ph.D. Thesis, Univ. of California, Berkeley, 1986.
[44] P. Raghavan and C.D. Thompson, "Randomized rounding," Combinatorica, vol.7, pp.365–374, 1987.
[45] T. Roos, T. Asano, D. Ranjan, E. Welzl, and P. Widmayer, "Space filling curves and their use in the design of geometric data structures," Theoretical Computer Science, vol.181, pp.3–15, 1997.
[46] K. Sadakane, N. Takki-Chebihi, and T. Tokuyama, "Combinatorics and algorithms on low-discrepancy roundings of a real sequence," Proc. 28th ICALP, Springer LNCS2076, pp.166–177, 2001.
[47] A. Schrijver, Theory of Linear and Integer Programming, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, 1986.
[48] W.T. Tutte, "Lectures on matroids," J. Research of the National Bureau of Standards (B), vol.69, pp.1–47, 1965.
[49] R. Ulichney, Digital halftoning, MIT Press, 1987.
[50] R.A. Ulichney, "Dithering with blue noise," Proc. IEEE, vol.76, no.1, pp.56–79, 1988.
[51] R. Ulichney, "The void-and-cluster method for dither array generation," IS&T/SPIE Symposium on Electronic Imaging Science and Technology, Proc. Conf. Human Vision, Visual Processing and Digital Display IV, ed. Allebach, SPIE vol.1913, pp.332–343, John Wiley, 1993.
[52] N. Usui and T. Asano, "Binary-coding pattern creating method and apparatus," Patent no.9951700, 1998.
[53] L. Velho and de M. Gomes, "Digital halftoning with space filling curves," Proc. SIGGRAPH '91, pp.81–90, 1991.
[54] P.W. Wong, "A mixture distortion criterion for halftones," Proc. IS&T/OSA Optics and Imaging in the Information Age, pp.187–191, 1996.
[55] I.H. Witten and M. Neal, "Using peano curves for bilevel display of continuous-tone images," IEEE Comput. Graph. Appl., pp.47–52, 1982.
[56] M. Yao and K.J. Parker, "Modified approach to the construction of a blue noise mask," J. Electronic Imaging, vol.3, no.1, pp.92–97, 1994.
[57] Y. Zhang and R.E. Webber, "Space diffusion: An improved parallel halftoning technique using space-filling curves," Proc. SIGGRAPH '93, pp.305–312, 1993.

Tetsuo Asano received the B.E., and M.E., and Ph.D. degrees in Engineering from Osaka University in 1972, 1974, and 1977, respectively. He is currently a professor of JAIST (Japan Advanced Institute of Science and Technology). His research interest includes Computational Geometry, Discrete Algorithms, Combinatorial Optimization and their applications. Dr. Asano is a Fellow of ACM and a member of IEEE, SIAM, IPSJ, and ORS. He is a member of the editorial boards of Discrete and Computational Geometry, Computational Geometry: Theory and Applications, International Journal of Computational Geometry and Applications, etc.