数理解析研究所講究録
1185 巻 2001 年 18-27

Digital Halftoning: Formulation as a Combinatorial Optimization Problem and Approximation Algorithms Based on Network Flow

Tetsuo Asano[1], Naoki Fujikawa[1], Naoki Katoh[2], Tomomi Matsui[3], Hiroshi Nagamochi[4], Takeshi Tokuyama[5], and Nobuaki Usui[6]

[1]School of Information Science,
JAIST
Tatsunokuchi 923-1292, Japan
t-asano@jaist.ac.jp

[2] Graduate School of Engineering,
Kyoto University
Kyoto 606-8501, Japan
naoki@archi.kyoto-u.ac.jp

[3] Graduate School of Engineering,
University of Tokyo
Tokyo 113-8656, Japan
tomomi@misojiro.t.u-tokyo.ac.jp

[4] Dept. of Information and Computer Sciences,
Toyohashi University of Technology
Toyohashi 441-8580, Japan
naga@ics.tut.ac.jp

[5] Graduate School of Information Sciences,
Tohoku University
Sendai 989-3201, Japan
tokuyama@dais.is.tohoku.ac.jp

[6] Peripheral Systems Laboratories,
Fujitsu Laboratories Ltd.
Atsugi 243-0197, Japan
usui@flab.fujitsu.co.jp

Abstract: Digital halftoning is an important technique to convert an image having several bits for brightness levels into a binary image consisting of black and white dots which looks similar to an input image. The similarity between two images is measured by the total sum of differences in the weighted sums of brightness levels of pixels in a neighborhood surrounding each pixel. Then, the problem of producing an optimal halftoned image is a combinatorial optimization problem to find a binary image minimizing the measure for a given multiple-level image. Despite a negative result that it is NP-complete even for a simple neighborhood consisting of 2×2 pixels, we can rely on approximation algorithms mainly based on network flow algorithms.

Keywords: approximation algorithm, combinatorial optimization, discrepancy, 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 seems to be no great invention on the software side of the printing technology. What is required is a technique to convert an intensity image having several bits for brightness levels into a binary image consisting of black and white dots so that the binary image looks very similar to the input image. Theoretically speaking, the problem is how to approximate an input gray image by a binary image. Since this is one of the central techniques in computer vision and computer graphics, a great number of algorithms have been proposed (see, e.g., [12, 10, 6, 13, 14]). However, there have been very 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 of halftoning methods (instead of the "human eye's judgment"), and to handle the digital halftoning problem fully mathematically. Unfortunately, to the authors' knowledge no mathematical criterion has not been a main focus on digital halftoning even in the comprehensive studies in the Ph.D. Thesis by Ulichney [18].

So, one of the purposes of this paper is to settle some reasonable mathematical criterion for this problem based on a model for human visual perception. Although there are extensive studies on human visual perception (see, e.g., [16]), we use a simplified model to analyze computational complexity of the problem. Our criterion measures the "difference" between the average gray levels or the total sums of levels normalized between 0 and 1 in a neighborhood surrounding each pixel. Then, the problem of producing an optimal halftoned image is a combinatorial optimization problem to find a binary image minimizing the measure for a given multiple-level image. The complexity of the problem depends on the choice of the neighborhood family. In most cases it is NP-hard. Actually, we show that it is the case even if each neighborhood consists of 2×2 pixels.

Although the problem itself is defined in the two-dimensional plane, it is possible to define its one-dimensional version in which a sequence of real numbers is given. This problem itself is an interesting combinatorial problem, and we show that it can be solved in polynomial time by using negative cycle detection algorithms.

This implies that it is essentially necessary to consider approximation and/or heuristic algorithms. Even for the two-dimensional problem, if each neighborhood is one-dimensional along a space-filling curve, we can apply the algorithms for the one-dimensional case. The use of space-filling curves is popular in image processing, and we think this method (although it is merely a heuristic method since neighborhood should be truly two-dimensional in practice) is probably useful if we can generate space-filling curves in somewhat random manner[4]. We evaluate some popular error-propagating algorithms in digital halftoning in our optimization criterion.

2 Known Basic Algorithms

Given an $N \times N$ array A of real numbers between 0 and 1, we wish to construct 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 value by a fixed threshold, say 0.5. It is simplest, but the quality of the output image is worst since any uniform gray region becomes totally white or totally black. The most important is how to represent intermediate intensities. Among a number of algorithms for digital halftoning well-known are Ordered Dither and Error Diffusion. We briefly describe them.

2.1 Ordered Dither

Instead of using a fixed threshold over an entire image, this method uses different thresholds. A simple way of implementing this idea is as follows: We prepare a square submatrix of $M \times M$ entries which are integers ranging from 0 and $M^2 - 1$ and tile an input array by this submatrix. Then, for each entry (i, j) we have an input value $A(i, j)$ in A and an integer $D(i \text{ mod} M, j \text{ mod} M)$ in the submatrix. If $A(i, j) > D(i \text{ mod} M, j \text{ mod } M)/M^2$ then the corresponding output value $B(i, j)$ is determined to be 1, and otherwise it is 0. This submatrix is called a dither matrix. An example is shown below[12].

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

2.2 Error Diffusion

Ordered Dither is simple and efficient, but its output quality is not satisfactory in many cases. Another standard method uses a fixed threshold 0.5 but it diffuses the quantization error over unprocessed neighboring pixels according to some fixed ratios. The ratios suggested by Floyd and Steinberg in their paper [10] are $(7/16, 3/16, 5/16, 1/16)$ for the right, lower left, lower, and the lower right pixels, respectively. This method gives excellent results in many cases, but it tends to generate some visible patterns in an area of uniform intensity, which are caused by the fixed error-diffusing coefficients.

2.3 Related Combinatorial Problems

The quality of the ordered dither method heavily depends on the dither matrix. The matrix shown above is constructed according to a simple rule, which is known as Incremental Voronoi Insertion. That is, we start with an arbitrary entry. Since the submatrix tiles the entire plane, we have many copies of the entry. Then, we choose the entry farthest from the existing points (entries). Such a point must be a vertex of the Voronoi diagram. Thus, our strategy is to choose a Voronoi vertex that is farthest from the existing points. This selection is iterated until half of the entries are chosen. The remaining half is filled in symmetrically.

Now we have a natural question. Does this strategy optimize anything? Imagine a part of an image in which intensities gradually increase. Then, the number (or density) of white dots also increases according to the dither matrix starting from a dark part consisting of only 0-numbered entries until an entirely bright part. During the transition, white dots should be uniformly distributed. This means that for any k, $0 \leq k \leq M^2/2$ those entries having numbers greater than or equal to k must be as uniformly distributed as possible. The uniformity can be measured by the ratio of the minimum pairwise distance over the diameter of the maximum empty circle. A combinatorial problem related to this is to find a point sequence that minimizes the maximum of the ratios defined above. It is rather easy to see that the incremental Voronoi insertion is not optimal, but it is open to find such an optimal sequence.

An easier combinatorial problem is to distribute a given number of points within a unit square. This problem has been studied for many years in combinatorics under the name of Packing Problem. There was a break-through recently in this study which shows best packing patterns up to about 50 points and proved the optimality of their packing patterns consisting of up to 26 points. In our case the base plane is not continuous but discrete. If the size of the grid plane is small enough, say, 16×16, we can find an optimal solution by solving an integer programming on 256 variables.

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. This fact suggests us of a rotated dither matrix. Then, the problem is how to design such a rotated pattern consisting of M^2 elements. This is not so easy. In fact, to the authors' best knowledge only one method [17] is known. It is a rotation by a so-called rational angle defined by triangles of edges with integral lengths. We have developed a scheme for achieving rotation of a squared region approximately by any given angle. The detail will be described in the final report.

3 Why Raster Scan

Error Diffusion is one of the most popular methods for digital halftoning. One of the drawbacks of this method is that it tends to generate regular patternings, which are caused by fixed scan order and fixed coefficients for error diffusion. For better quality we need to incorporate some randomness. Thus, we come up with an idea to use random space-filling curves instead of raster scan. Recently, it has been observed that error diffusion along some space-filling curves such as Peano curves and Hilbert curves sometimes achieve better quality compared with the traditional error diffusion based on a raster scan. One drawback of the methods comes from the fact that such space-filling curves are usually defined recursively on a square grid plane. Thus, there is some difficulty when they are applied to rectangular images. Another drawback is found in its quality of an output image due to its recursive structure. Since it is recursively defined, each quarter of an image is completely separated and their boundaries are often visible in the resulting binary image.

The idea of using space-filling curves for digital halftoning is not new. Velho and Gomes [19] use space-filling curves for cluster-dot dithering. Zhang and Webber [20] give a parallel halftoning algorithm based on space-filling curves. Asano, Ranjan, and Roos [5] formulate digital halftoning as a mathematical optimization problem and obtain an approximation algorithm based on space-filling curves. So, the digital halftoning techniques based on space-filling curves seem to be promising. However, one of their serious disadvantages is the strong constraint on the image size, that is, the size of an input image must be something like a power of 2 since most of recursively defined space-filling curves are defined for square lattice planes of sizes being such.

We have developed a theory for random space-filling curves. One result[3] is a simple way of generating a random space-filling curves for a rectangular grid of even length in each side. It first constructs a random spanning tree on a smaller grid of the half side lengths and then generates a tour along the tree. It runs in linear time in the image size.

Another way of generating a random space-filling curve without any size constraint is also proposed[4]. It is based on the following fact: any space-filling curve is represented by regular arrangement of stones in such a way that white stone always lies to the right of the curve and black stone does to the left. Then, a space-filling curve without self intersection corresponds to an arrangement of stones in such a way that no cross of different colors is included and there are two connected components of those stones of the same color, where two stones are connected if they have the same color and they are adjacent horizontally or vertically. Then, as far as a condition called parity condition is satisfied, there is a space-filling curve with specified starting and exit entries. Moreover, defining an operation called flipping which changes the curve locally by exchanging two stone colors, we can guarantee that for any two space-filling curves one can be transformed to another by successive flippings. This suggests generation of random space-filling curves by successive random flippings.

4 A Mathematical Formulation of Digital Halftoning

The main task of this section is to define the digital halftoning problem as a combinatorial optimization problem by defining a reasonable concrete mathematical criterion. As we defined in the introduction, let $A = (a_{ij})_{i,j=1,\ldots,N}$, $0 \leq a_{ij} \leq 1$, be a matrix representing an image in the grid array G of size $N \times N$.

Imagine that we look at some pixel (i,j) of the gray-level image A. What happens is, we actually perceive an average of gray levels of some (small) neighborhood of that point. Using the same observation, the intensity around the pixel (i,j) of a binary image is proportional to the number of white points in the corresponding neighborhood. Therefore, density values should be roughly equal around *any* pixel between an output binary image of the digital halftoning and the input image A. The observation motivates us to give the following mathematical formulation:

For any region R in the grid array G (regarded as an $N \times N$ rectangle subdivided into N^2 pixels each of which corresponds to an array entry), we consider an error function $E^R(A, A')$ describing the difference between two pictures A and A' within the region R. A typical error function is the *difference of average density* $|A(R) - A'(R)|/|R|$, where $A(R)$ is the sum of entries of A located in R, and $|R|$ is the number of pixels in $|R|$.

A family $\mathcal{F}$ of regions in G is called a *neighborhood family* if there exists a map ϕ from G to $\mathcal{F}$ such that the region $\phi(p) \in \mathcal{F}$ contains the pixel p for each pixel $p \in G$. We call $\phi(p)$ the *neighborhood* of p. The map ϕ need not be surjective nor injective. Note that this is a quite weaker definition than the neighborhood system in usual geometry. For a neighborhood family $\mathcal{F}$, we define the L_∞ distance

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

between the images A and A' with respect to $\mathcal{F}$ and the error function E. This distance is also called the *maximum error* between A and A' (with respect to $\mathcal{F}$). We only consider the L_∞ distance in this paper for technical reason, although the L_p distance defined by $(\sum_{R \in \mathcal{F}} (E^R(A, A'))^p)^{1/p}$ might be a useful measure in practice.

We mainly consider the following family $\mathcal{F}$ consisting of (axis parallel) rectangular regions of a fixed shape $l \times k$. For an $l \times k$ rectangular region W in G, its center is the pixel $p(W)$ on the $\lceil l/2 \rceil$-th row and $\lceil k/2 \rceil$-th column counted from the north-west corner of W. Note that $p(W)$ has the center of gravity of W in the closure of the pixel. We denote W as $W(i,j)$ if $p(W) = (i,j)$. This defines the neighborhoods for the pixels (called *central pixels*) (i,j) satisfying $\lceil l/2 \rceil \leq i \leq N - \lceil (l-1)/2 \rceil$ and $\lceil k/2 \rceil \leq j \leq N - \lceil (k-1)/2 \rceil$. For each of other pixels (called near-boundary pixels), we define its neighborhood as the neighborhood of its nearest central pixel with respect to the Euclidean metric. This correspondence makes $\mathcal{F}$ to be a neighborhood family

Consider an input images A and its output binary image B in the digital halftoning. If the difference between the average gray level near each pixel image is small, the picture B is expected to look very similar to the original picture A; at least B is a very good approximation of A (without further knowledge of the contents of the image). Let us consider our neighborhood system. The number $|W(i,j)|$ of pixels in a neighborhood is independent of the choice of (i,j). With that, instead of the difference of average density, it is natural to use

$$E^{W(i,j)}(A,B) = |A(W(i,j)) - B(W(i,j))|.$$

to evaluate the visual difference near (i,j) between A and B, where $A(W(i,j))$ is the sum of elements of A within $W(i,j)$.

Note that we do not claim that our choice of the neighborhood family and the error function is the best for evaluating practical digital images: A broader neighborhood family is probably better, and the error function can be generalized in various ways adapting to real applications, e.g., by assigning larger weights to the pixels nearer to the center (i,j) in the neighborhood to take the summation; however, we consider the above simple model to investigate the complexity of its optimization. We discuss some broader choice of the neighborhood family in Section 5.3.

Our goal is to bring the local error $E^{W(i,j)}(A,B)$ close to 0 for any pixel (i,j); That is, to minimize the L_∞ distance $Dist_\infty^{\mathcal{F}}(A,B)$. Hence, we have the following formulation of the digital halftoning problem:

> Design a method to compute a binary image B such that $Dist_\infty^{\mathcal{F}}(A,B)$ is as small as possible for an input image A.

For simplicity, we write $Dist(A,B)$ for $Dist_\infty^{\mathcal{F}}(A,B)$ unless we want to emphasis the neighborhood system $\mathcal{F}$, error function, and/or the L_∞ measure. In particular, the binary image B minimizing $Dist(A,B)$ is called the *optimized digital approximation* of A (with respect to the neighborhood family $\mathcal{F}$). We investigate the time complexity of computing the optimized digital approximation.

5 One dimensional problem

If $l = 1$, each member in $\mathcal{F}$ is a horizontal strip of length k, and the problem can be solved on each row independently. Therefore, we first consider the one-dimensional version of the problem. Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ be a rational vector such that $0 \le a_j \le 1$ for all $j \in \{1, 2, \ldots, n\}$ and k be an integer with $1 \le k \le n$. We consider the neighborhood family $\mathcal{I}_k$ consisting of intervals of length k. If k is a constant, it is relatively easy to design a linear time algorithm in n by using dynamic programming (in precise, $O(2^k n)$ time using $O(2^k + n)$ space) to compute the optimized digital approximation $\boldsymbol{b}$. However, it is nontrivial to design an algorithm which is polynomial in both n and k.

5.1 Polynomial time algorithms

The one-dimensional optimized digital approximation of $\boldsymbol{a}$ is the binary vector $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ which is the solution of the following integer programming problem, where z corresponds to the distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ in our measure:

$$\begin{aligned}
&\text{minimize} \quad z \\
&\text{subject to} \quad -z \le \textstyle\sum_{j=i}^{i+k}(a_j - b_j) \le z && (1) \\
&\qquad (\forall i \in \{1, 2, \ldots, n-k\}), && (2) \\
&\qquad b_j \in \{0, 1\} \quad (\forall j \in \{1, 2, \ldots, n\}).
\end{aligned}$$

Since the variables $b_1, \ldots, b_n$ are 0-1 valued, we can replace the inequality constraints (1) by

$$\begin{aligned}
\lfloor z + \textstyle\sum_{j=i}^{i+k} a_j \rfloor &\ge \textstyle\sum_{j=i}^{i+k} b_j \\
&\ge \max\{\lceil -z + \textstyle\sum_{j=i}^{i+k} a_j \rceil, 0\} \\
&\qquad (\forall i \in \{1, 2, \ldots, n-k\}).
\end{aligned}$$

We introduce the variables $x_0, \ldots, x_n$ satisfying $x_i - x_0 = b_1 + \cdots + b_i$ for $i \in \{1, 2, \ldots, n\}$. Then the above problem is transformed to the following problem

$$\begin{aligned}
&\text{minimize} \quad z \\
&\text{subject to} \\
&\quad x_{i+k} - x_{i-1} \le \lfloor z + \textstyle\sum_{j=i}^{i+k} a_j \rfloor \\
&\qquad\qquad (\forall i \in \{1, 2, \ldots, n-k\}), \\
&\quad x_{i-1} - x_{i+k} \le -\max\{\lceil -z + \textstyle\sum_{j=i}^{i+k} a_j \rceil, 0\}, \\
&\qquad\qquad (\forall i \in \{1, 2, \ldots, n-k\}), \\
&\quad x_j - x_{j-1} \le 1, \qquad (\forall j \in \{1, 2, \ldots, n\}),
\end{aligned}$$

$x_{j-1} - x_j \le 0, \qquad (\forall j \in \{1, 2, \ldots, n\}),$
x_j is an integer, $\qquad (\forall j \in \{0, 1, 2, \ldots, n\}).$

The problem of checking the existence of the vector $(x_0, x_1, \ldots, x_n)$ satisfying the above constraints when the value z is fixed can be transformed to a negative cycle detection problem (see [2] for detail). The optimal value of z (i.e., smallest z causing no negative cycle) can be found by the ordinary binary search technique. Each edge weight is represented by a step function with respect to z. Thus, we only need to consider the break points of the step functions. If we define $q(s,i) = s + 0.5 + \sum_{j=i}^{i+k} a_j$ for integers s and i, the set $Q = \{q(s,i) | 1 \le i \le n-k, -k \le s \le k\}$ contains all the break points. By applying binary search technique (with some care), we can find the optimal value of z by executing the above negative cycle detecting algorithm $\mathrm{O}(\log nk) = \mathrm{O}(\log n)$ times with additional $O(n \log n)$ time for each search process. Thus we can find the optimal value of z in $\mathrm{O}(n^{1.5} \log^2 n)$ time in total.

Hence, we have the following theorem:

Theorem 5.1 *The 1-D optimized digital approximation can be computed in $O(n^{1.5} \log^2 n)$ time. The space requirement is $O(n)$.*

If k is small (say, smaller than $n^{1/4}$), we can obtain a faster algorithm.

Theorem 5.2 *The 1-D optimized digital approximation can be computed in $O(k^2 n \log n)$ time using $O(nk)$ space.*

5.2 Optimization Under Different Distances

The previous subsection considered optimization problem as follows:
Given a real vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ such that $0 \le a_i \le 1$ for all $i \in \{1, 2, \ldots, n\}$ and k is an integer with $1 \le k \le n$, find a binary vector $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ with $b_i = 0, 1$ for all i that minimizes the distance to $\boldsymbol{a}$, which is defined by

$Dist(\boldsymbol{a}, \boldsymbol{b}) = \max_{I \in \mathcal{I}_k} |\boldsymbol{a}(I) - \boldsymbol{b}(I)|,$

where $\mathcal{I}_k$ is the neighborhood family consisting of all intervals of length k.

One generalization of the problem is to use different distance. Another natural distance is L_2 distance, that is,

$Dist(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\Sigma_{I \in \mathcal{I}_k} (\boldsymbol{a}(I) - \boldsymbol{b}(I))^2}.$

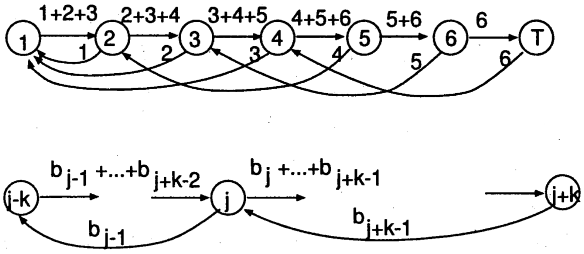

Figure 1: A network with quadratic costs.

This problem can be solved by reducing it to a network flow problem as follows: Define a set of nodes by $1, 2, \ldots, n-k+2$ and a set of arcs $A \cup B$, where
$A = \{(1,2), (2,3), \ldots, (n-k+1, n-k+2)\},$
$B = \{(2,1), (3,1), \ldots, (j-k+1, j+k-1),$
$\ldots, (n-k+2, n-k+1)\}.$
Here, each arc $(j, j+1)$ in A corresponds to an interval starting at j, that is, $[j, j+k-1]$ for all $j \in \{1, 2, \ldots, n-k+1\}$. We want the flow on the arc to be $\boldsymbol{a}([j, j+k-1])$. For the purpose each node j has two incoming arcs $(j-1, j)$ and $(j+k, j)$ and two outgoing arcs $(j, j+1)$ and $(j, j-k)$. The two A-type arcs $(j-1, j)$ and $(j, j+1)$ have capacity $[0, k]$ while the B-type arcs $(j+k, j)$ and $(j, j-k)$ have capacity $[0, 1]$. See Fig.1 for pictorial illustration.

Now, a flow defined by $\boldsymbol{a}([j, j+k-1])$ on each A-type arc $(j, j+1)$ and a_{i-1} on each B-type arc (j, i) is a feasible flow. Therefore, if we define the costs of the arcs to be 0 for B-type arcs and $(\boldsymbol{b}([j, j+k-1]) - \boldsymbol{a}([j, j+k-1]))^2$ for each A-type arc $(j, j+1)$, $j = 1, 2, \ldots, n$, then the problem is to find a flow to minimize the total cost. Fortunately, this type of the cost minimization problems can be solved in polynomial time both in n and k [1, 15].

5.3 Larger neighborhood family and modified error functions

A well-known method to perform the one-dimensional digital halftoning is "rounding with error-propagation". Starting from the first entry, it determines the value of $\boldsymbol{b}$ greedily. First, it simply round a_1 to obtain $b_1 = \lfloor a_1 \rfloor$. Suppose that if we have determined $b_1, .., b_t$, we consider the accumulated error $sum(t) = \sum_{i=1}^{t} (a_i - b_i)$. Then, b_{t+1} is determined as 0 if $a_{t+1} + sum(t) \le 1/2$ and otherwise 1. It is easy to obtain that

$-1/2 \leq sum(t) \leq 1/2$. Therefore, for any interval $I$, $|\boldsymbol{a}(I) - \boldsymbol{b}(I)| \leq 1$, where $a(I) = \sum_{i \in I} a_i$. This means that the algorithm gives an output sequence $\boldsymbol{b}$ with $dist(\boldsymbol{a}, \boldsymbol{b}) < 1$ for the family $\mathcal{I} = \cup_{k=1}^{n} \mathcal{I}_k$ of all intervals.

Our optimal solution described in previous subsections does not have the above property, since we have only considered the neighborhood family $\mathcal{I}_k$. However, we can consider a more generalized form in which we consider the set $\mathcal{I}$ of all intervals as the neighborhood family, define the error function $E^I(\boldsymbol{a}, \boldsymbol{b}) = |\boldsymbol{a}(I) - \boldsymbol{b}(I)|/z_I$, where z_I is a constant dependent on $I$ for each $I$. Similarly to the $O(n^{1.5} \log^2 n)$ time algorithm for $\mathcal{I}_k$, we can design an algorithm with a time complexity $O(n^{2.5} \log^2 n)$ for this generalized problem, provided that each z_I is a fractional number of a pair of $O(\log n)$ bit integers. In particular, if we set $z_I = 1$ for all intervals, the output satisfies that $|\boldsymbol{a}(I) - \boldsymbol{b}(I)| < 1$ for every interval $I \in \mathcal{I}$.

6 Matrix-rounding problem

6.1 Discrepancy problem

The following Baranyai's theorem is well-known (See [7, 8, 9]):

Proposition 6.1 *For any input* $[0,1]$*-valued matrix* $A = (a_{i,j})$, *there exists a binary matrix* $B$ *attaining that* $|\sum_{i=1}^{n}(a_{i,j} - b_{i,j})| < 1$ *for each* $j = 1, 2, \ldots, n$, $|\sum_{j=1}^{n}(a_{i,j} - b_{i,j})| < 1$ *for each* $i = 1, 2, \ldots, n$, *and* $|\sum_{j=1}^{n} \sum_{i=1}^{n}(a_{i,j} - b_{i,j})| < 1$.

This means that if we consider a region family $\mathcal{F}$ consisting of the whole matrix, all rows and all columns of the matrix, there always exists a rounding satisfying that $Dist_{\infty}^{\mathcal{F}}(A, B) < 1$. However, the rounding error is highly dependent on the region family $\mathcal{F}$.

The following theorem is well-known [9]:

Theorem 6.2 *The inhomogeneous discrepancy of a* $[0,1]$ *valued* $n \times n$ *matrix with respect to the family of all rectangular regions is* $O(\log^3 n)$ *and* $\Omega(\log n)$. *The same bounds hold for the inhomogeneous discrepancy for the family of all rectangular regions containing the left-upper corner entry of the matrix.*

We are interested in the matrix-rounding with respect to the set $\mathcal{F}_k$ of all $k \times k$ square regions. The following proposition is obtained in a straightforward manner from the theorem above:

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

We remark that a polynomial time algorithm for computing a rounding with an $O(\log^4 k)$ discrepancy can be designed based on the proof of Theorem 6.13 in [9]. This is theoretically better than the popular two-dimensional error diffusion algorithm, for which the rounding error can become $k$.

For the family $\mathcal{F}_2$ consisting of all 2×2 square regions, there exists an instance $A$ that the discrepancy is exactly 1. However, the authors do not know whether there exists an instance requiring $Dist_{\infty}^{\mathcal{F}_2}(A, B) > 1$ or not. It is easy to show that the inhomogeneous discrepancy with respect to $\mathcal{F}_2$ is at most 2; indeed, the checkerboard binary matrix $C$ satisfies $Dist_{\infty}^{\mathcal{F}_2}(A, B) \leq 2$ for any input matrix $A$ simultaneously. However, it is nontrivial to give a better upper bound; We can prove the following result (the proof is involved, and omitted in this version):

Theorem 6.4 *For any* $[0,1]$ *valued matrix* $A$, *there exists a binary matrix* $B$ *satisfying that* $Dist_{\infty}^{\mathcal{F}_2}(A, B) \leq 1.75$.

6.2 NP-completeness of computing an optimal matrix rounding

We showed the NP-completeness of the problem of computing an optimal matrix rounding.

Theorem 6.5 *For any* $\epsilon > 0$, *it is NP-complete to decide whether the optimal rounding error of a given matrix* $A$ *is greater than* $1 - \epsilon$ *or less than* $1/2 + \epsilon$ *with respect to the distance* $Dist_{\infty}^{\mathcal{F}_2}$.

We do not repeat the proof here. Refer to [2] for the detail. The idea is the reduction from the planar 3-SAT problem [11].

6.3 Approximation Algorithms

(i) Error Diffusion along Two Curves

Let us consider some approximation algorithms. One of them is an extension of the proposition 6.1 which guarantees the existence of "good" matrix rounding in the sense that the error at each row and each column are bounded by 1. Another observation is that the maximum error generated by the 1-D error diffusion along a space-filling curve is also bounded by 1. Can we combine the two results? That is, given two different space-filling curves C_1 and C_2, can we bound the maximum error by 1 simultaneously for the two curves? The answer is positive by the following network-flow algorithm.

A space-filling curve on a matrix can be regarded as a permutation of matrix elements. Let $(\sigma_\mu(1), \sigma_\mu(2), \ldots, \sigma_\mu(n^2))$ be a permutation corresponding to a space-filling $C_\mu, \mu = 1, 2$. Given an image matrix $\boldsymbol{a}[\ ]$, we wish to compute a binary matrix $\boldsymbol{b}[\ ]$ of the same size for which

$$\max\{|\sum_{j=1}^{i}(a[\sigma_\mu(j)] - b[\sigma_\mu(j)])|\} \leq 1$$

holds for every $i, 1 \leq i \leq n^2$ and $\mu = 1, 2$.

Construct a network as follows (see Fig. 3): we have two sets of $n^2 + 1$ nodes, $V_1 = \{v_1(1), v_1(2), \ldots, v_1(n^2+1)\}$ and $V_2 = \{v_2(1), \ldots, v_2(n^2+1)\}$. The first set V_1 of nodes are connected in the decreasing order while the second in the increasing order, that is, we have arcs $(v_1(i), v_1(i-1))$ for each $i = 2, 3, \ldots, n^2+1$ and $(v_2(i), v_2(i+1))$ for each $i = 1, 2, \ldots, n^2$. Extending σ_1 and σ_2 so that $\sigma_1(n^2+1) = \sigma_2(n^2+1) = n^2+1$, there is an obvious one-to-one correspondence between the two permutations σ_1 and σ_2. According to the correspondence we draw an arc $(v_1(i), v_2(j))$ whenever $\sigma_1(i) = \sigma_2(j)$. Finally, we add an arc $(v_2(n^2+1), v_1(n^2+1))$.

Arc capacity is defined as follows: Each edge $(v_1(i), v_1(i-1)), 2 \leq i \leq n^2+1$ is associated with the sum of input values $sum_1(i-1) = \sum_{j=1}^{i-1} a[\sigma_1(j)]$. The capacity of the edge is determined by the floor and ceiling of the sum, i.e., $[\lfloor sum_1(i-1)\rfloor, \lceil sum_1(i-1)\rceil]$. For the second set of nodes, each edge $(v_2(i), v_2(i+1)), 1 \leq i \leq n^2$ is associated with the sum of input values $sum_2(i) = \sum_{j=1}^{i} a[\sigma_2(j)]$. The capacity of the edge is determined by the floor and ceiling of the sum, i.e., $[\lfloor sum_2(i)\rfloor, \lceil sum_2(i)\rceil]$. The capacity of the edge $(v_2(n^2+1), v_1(n^2+1))$ is $[\lfloor\sum_{i=1}^{n^2} a[i]\rfloor, \lceil\sum_{i=1}^{n^2} a[i]\rceil]$. Any other crossing arc from $v_1(i)$ to $v_2(j)$ is associated with the input value $a[\sigma_1(i)] = a[\sigma_2(j)]$ and its capacity is $[0, 1]$.

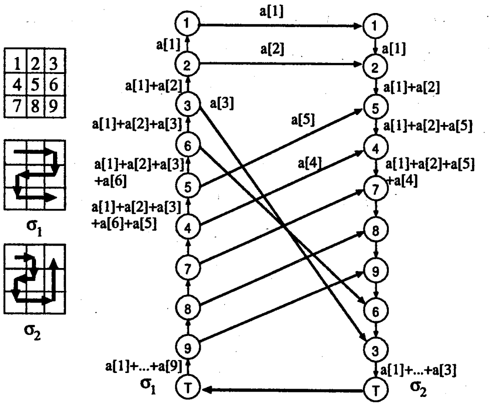

Figure 2: A network defined by two space-filling curves.

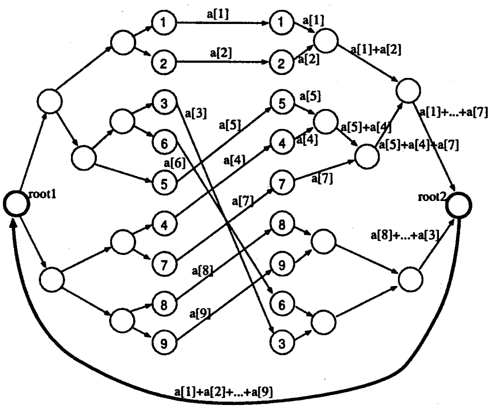

Figure 3: A network defined by two binary trees.

Now, a feasible flow is easily found since values associated with arcs defined above obviously satisfy the capacity conditions. Thus, the well-known integrality theorem implies the existence of a integer-valued feasible flow. Take any such flow, and the flow at crossing edges give a binary matrix we wanted.

(ii) Combining Two Binary Structures

The same trick works for a different structures. The network we constructed above has not hierarchical structure. Let us extend the structure by replacing each of the two simple paths with a binary tree structure (see Fig. 3). More concretely, for a permutation σ_μ we construct a binary tree T_μ with leaves $v_\mu(1), \ldots, v_\mu(n^2)$.

For the first set V_1 we orient the arcs in the binary tree T_1 downward, that is, from the root

$root_1$ down to the leaves. The arcs in the second binary tree T_2 are oppositely directed. We draw crossing arcs between the leaves of T_1 and T_2, just like previously. Each arc in the trees is associated with the sum of input values of its descendants (leaves). Similarly as above, the capacity of each arc is determined by the floor and ceiling values of the associated value. Then, those values associated with arcs define a feasible flow on the network. Thus, again by the integrality theorem an integer-valued flow does exist and it gives us an approximated solution.

(iii) Scaling Algorithm

We have shown a polynomial-time approximation algorithm for a matrix rounding problem. However, it does not suffice in practice because of high time complexity of the underlying flow algorithm. An idea to reduce the time complexity is to use the integrality of matrix elements. Although we have assumed that each input matrix element is a real number between 0 and 1, they are integers, say 8-bit integers, in an image matrix. Making use of the integrality we can devise an efficient algorithm.

Assume that each matrix element is a k-bit integer, 0 through 2^k-1. We use the same structure using two binary trees shown in Fig. 3. An idea is to repeat matrix rounding $k-1$ times until all the matrix elements become 0 or 1. Rounding even numbers cause no problem. For odd numbers we have to choose rounding up or rounding down. Our intention here is to balance rounding ups and rounding downs. If two odd integers are adjacent, we should apply different roundings to them, i.e., rounding up one and rounding down the other. This idea is generalized as follows: In the tree T_1, we mark all leaves having odd values and their incident arcs. We check each node at one level up whether the two arcs incident to it are both marked. If it is the case, we mark the node and stop. Otherwise, we clime up one more level in T_1 and repeat the same check. When we reach the root of T_1, all odd leaves are paired by paths in T_1 or connected to the root.

The same process is applied to T_2. Then, we add the crossing arcs connecting the same elements in the two sets of leaves. They altogether form (undirected) cycle(s). If we represent each such cycle by the integer values at the nodes along the cycle excluding the duplication, a sequence of odd numbers is obtained. Next, we divide each leaf value by 2. Here, those numbers in the cycles must be rounded to integers. We perform rounding up and rounding down iteratively along the sequence obtained above. Of course, there are two different ways of the rounding depending on whether we start with rounding up or down. Take any one of the two ways.

We iterate this process k times. Then, all the resulting leaf values become 0 or 1. These values naturally define a binary matrix. Although we have no space to prove it, the resulting matrix is one of the feasible flows on the network. This algorithm is efficient enough.

(iv) 2×2 neighborhood

In Section 6.2 we claimed that it is NP-hard to find an optimal rounding of a real-valued matrix even for the family of 2×2 neighborhoods. However, with little more constraints we have a polynomial-time algorithm. The idea is the following. We regard an image matrix as a checker board consisting of white and black cells. Then, we consider a family of 2×2 neighborhoods with white cells at their main diagonals. By the definition, each cell belongs to at most two (usually four except for boundary cells) neighborhoods. Decompose each 2×2 square into two horizontal pairs and combine them vertically. Then, 2×2 squares are combined horizontally, and the resulting 2×4 regions are combined vertically to have 4×4 regions, etc. Moreover, define costs of arcs appropriately, say by $(\boldsymbol{b}-\boldsymbol{a})^2$ where $\boldsymbol{b}$ is a flow on the arc and $\boldsymbol{a}$ is the sum of the values in the descendants. Then, we can find an optimal solution in the case of quadratic costs.

7 Concluding remarks

This paper gives an initial study on the optimization-based evaluation of digital halftoning algorithms. There are lot of open problems which are interesting from both viewpoints of theory and practice:

(1) For the neighborhood family consisting of $k \times k$ squares ($k \geq 2$), is it always possible to make $dist(A, B) < c$ where c is a constant strictly smaller than k?

(2) If the answer to (1) is yes, can we design a polynomial time algorithm to assure the condi-

tion? (Not seeking for the optimal solution).
One possible candidate is the use of random space filling curves [4], and perform the one-dimensional error propagation algorithm along curves. If we fix a space filling curve, we can always construct an instance so that $dist(A,B) > k - \epsilon$; however, it is difficult to construct an instance which performs badly for many randomly chosen filling curves simultaneously.
(3) Are there any good approximation algorithm which has a theoretical performance ratio?
(4) Find a very good neighborhood family and error function to evaluate the quality of digital halftoning in practice. If such a framework is firmly established, modern heuristic methods can be probably applied to have a good digital halftoning.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows, Theory Algorithms and Applications,* Princeton Hall,

[2] T. Asano, T. Matsui, and T. Tokuyama: "On the Complexities of the Optimal Rounding Problems of Sequences and Matrices," Proc. 7th Scandinavian Workshop on Algorithm Theory, pp.476-489, Bergen, Norway, July 2000.

[3] T. Asano: "Digital Halftoning Algorithm Based on Random Space-Filling Curve," IEICE Trans. on Fundamentals, Vol.E82-A, No.3, pp.553-556, March 1999.

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

[5] 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.

[6] B. E. Bayer: "An optimum method for two-level rendition of continuous-tone pictures," *Conference Record, IEEE International Conference on Communications,* 1 (1973), pp.(26-11)–(26-15).

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

[8] B. Bollobás, *Combinatorics*, Cambridge University Press, 1986.

[9] J. Beck and V. T. Sös, *Discrepancy Theory*, in *Handbook of Combinatorics* Volume II, (ed. R.Graham, M. Grötschel, and L Lovász) 1995, Elsevier.

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

[11] M. R. Garey and D. S. Johnson: *Computers and Intractability: A guide to theory of NP hardness,* Feeman and Company, 1979.

[12] D. E. Knuth: "Digital halftones by dot diffusion," ACM Trans. Graphics, 6-4 (1987), pp.245–273.

[13] J. O. Limb: "Design of dither waveforms for quantized visual signals," *Bell Syst. Tech. J.,* 48-7 (1969), pp. 2555–2582.

[14] B. Lippel and M. Kurland: "The effect of dither on luminance quantization of pictures," *IEEE Trans. Commun. Tech.,* COM-19 (1971), pp.879-888.

[15] M. Minoux: "Solving Integer Minimum Cost Flows with Separable Cost Objective Polynomially," Mathematical Programming Study 26 (1986) pp.237-239.

[16] M. Miyahara, K. Kotani, and V.R. Algazi: "Objective Picture Quality Scale (PQS) for Image Coding," IEEE Trans. on Communications, 46, 9, pp.1215-1226, Sept. 1998.

[17] V. Ostromoukhov, R.D. Hersh, and I. Amidror: "Rotated Dispersed Dither: a New Technique for Digital Halftoning," *Proc. of SIGGRAPH '94*, (1994) pp.123-130.

[18] R. Ulichney: *Digital halftoning*, MIT Press, 1987.

[19] L. Velho and de M. Gomes: "Digital Halftoning with Space Filling Curves," *Proc. of SIGGRAPH '91*, (1991) pp.81-90.

[20] Y. Zhang and R. E. Webber: "Space Diffusion: An Improved Parallel Halftoning Technique Using Space-Filling Curves," *Proc. of SIGGRAPH '93*, (1993) pp.305-312.