数理解析研究所講究録
1120 巻 1999 年 140-150

計算複雑度から見たディジタルハーフトーニング

Computational Complexity of Digital Halftoning

浅野 哲夫 [1] 徳山 豪 [2] 松井 知己 [3]
Tetsuo Asano Takeshi Tokuyama Tomomi Matsui

[1] 北陸先端科学技術大学院大学情報科学研究科 〒 923-1292 石川県能美郡辰口町旭台
t-asano@jaist.ac.jp
[2] 東北大学大学院情報科学研究科 〒 980-0812 仙台市青葉区片平 2-1-1
tokuyama@dais.is.tohoku.ac.jp
[3] 東京大学大学院工学研究科計数工学専攻 〒 113-8656 東京都文京区本郷 7-3-1
tomomi@misojiro.t.u-tokyo.ac.jp

ABSTRACT: Digital Halftoning is a technique to convert a gray image into a binary image. This paper discusses the computational complexity of the problem of obtaining a binary image which is optimal in the sense that it looks most similar to the input gray image. It is shown that it is NP-complete. computational complexity of the one-dimensional version of the problem is also discussed.

Keywords: digital halftoning, NP-complete, Planar satisfibility

1 Introduction

One of the most fundamental techniques in image processing is digital halftoning. Mathematically, the input of the digital halftoning problem is a two-dimensional $N \times N$ array $A = (a_{i,j})$ $1 \leq i, j \leq N$ of real numbers satisfying $0 \leq a_{i,j} \leq 1$, and its output is a binary $N \times N$ array $B = (b_{i,j})$ (i.e. $b_{i,j} = 0$ or 1) "approximating" A. The value $a_{i,j}$ corresponds to the brightness level (gray level) of the (i, j) pixel of the pixel grid. The intention of this method is to convert a given image which consists of several bits for brightness levels into a binary image having only black and white pixels. This kind of technique is indispensable when dealing with images which have to be printed on output devices producing black dots only, such as facsimiles and laser printers.

Up to now, a large number of methods and algorithms for digital halftoning have been proposed (see, e.g., [7, 4, 2, 8, 9]). A comprehensive summary of the results obtained in the literature can be found in the Ph. D. Thesis by Ulichney [10]. However, there have been very few studies discussing reasonable criteria for evaluating the quality of output images; maybe because the problem itself is very practical. Actually, the most common criterion of almost all existing studies on digital halftoning is to judge the quality of the output picture by human eyes. It is desirable to establish a good evaluation system of halftoning methods (instead of the "human eye's judgement"), and to handle the digital halftoning problem fully mathematically. Unfortunately, to the authors' knowledge, no concrete mathematical criterion is considered in the literature.

The aim of this paper is to propose such a fully mathematical framework. Investigating the digital halftoning problem as a combinatorial optimization problem, we give a mathematical criterion of "goodness" of the approximation of the array A with B, and discuss the complexity of the problem for some fundamental cases.

Let $\mathcal{A}$ be a set of matrices whose entries are real values in $[0, 1]$, and let $\mathcal{B}$ be its subsetset consisting of all binary matrices. In the information theory, a natural way to approximate an element A of $\mathcal{A}$ with an element in $\mathcal{B}$ is to define a distance among $\mathcal{A}$, and select the element in $\mathcal{B}$ nearest to A with respect to the distance. We adopt this idea, and define the distance by using the differnce of the entries of two image matrices in a neighborhood surrounding each pixel (using a suitable *neighborhood family*). In the formulation, given the output B, it is easy to compute the distance from the input A. However, if we want to compute the output minimizing the distance, we encounter a combinatorial optimization problem, and it is shown that the complexity of the problem depends on the choice of the neighborhood family.

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 (Section 3). However, if the neighborhood family is essentially two-dimensional, it is difficult to design a polynomial-time algorithm. Indeed, even if each neighborhood is a 2×2 square, we can show that the problem is NP hard (Section 4).

This implies that it is essentially necessary to consider approximation and/or heuristic algoirthms. 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 truely two-dimensional in practice) is probably useful if we can generate space-filling curves in somewhat random manner[1]. We evaluate some popular error-propagating algorithms in digital halftoning in our optimization criterion. We have not yet succeeded to outperform the error-propagating algorithms, and therefore, the design of theoretically better algoirthm is posed as an important open problem.

2 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 black 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 $\mathcal{W}$ 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 $\mathcal{W}$ 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 3.2.

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 $Dist_\infty^{\mathcal{W}}(A,B)$. Hence, we have the following formulation of the digital halftoning problem:

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

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

3 One dimensional problem

If $l = 1$, each member in $\mathcal{W}$ 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 $0 \leq a_j \leq 1$ for all $j \in \{1, 2, \ldots, n\}$ and k be an integer with $1 \leq k \leq n$. 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 with respect to n by using dynamic programming (in precise, $O(2^k n)$ time using $O(2^k + n)$ space) to compute the optimized digital approximation $\boldsymbol{b}$ (we omit it in this version). However, it is nontrivial to design an algorithm which is polynomial on both n and k.

3.1 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:

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

Since the variables $b_1, \dots, b_n$ are 0-1 valued, we can replace the inequality constraints (2) by

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

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

$$\begin{array}{rll} \text{minimize} & z & \\ \text{subject to} & x_{i+k} - x_{i-1} \le \lfloor z + \sum_{j=i}^{i+k} a_j \rfloor \quad (\forall i \in \{1,2,\dots,n-k\}), & (2)\\ & x_{i-1} - x_{i+k} \le -\max\{\lceil -z + \sum_{j=i}^{i+k} a_j \rceil, 0\} \quad (\forall i \in \{1,2,\dots,n-k\}), & (3)\\ & x_j - x_{j-1} \le 1 \quad (\forall j \in \{1,2,\dots,n\}), & (4)\\ & x_{j-1} - x_j \le 0 \quad (\forall j \in \{1,2,\dots,n\}), & (5)\\ & x_j \text{ is an integer } \quad (\forall j \in \{0,1,2,\dots,n\}). & \end{array}$$

Let us consider the problem for checking the existence of the vector $(x_0, x_1, \dots, x_n)$ satisfying the above constraints when the value z is fixed. We show that the problem is transformed to a negative cycle detection problem. The method to find the optimal value of z is discussed later. Let $H = (N, E)$ be a directed graph with a vertex set $N = \{0,1,2,\dots,n\}$ and an arc set $E = E_1 \cup E_2 \cup E_3 \cup E_4$ where

$$\begin{aligned} E_1 &= \{(0,k+1),(1,k+2),\dots,(n-k-1,n)\},\\ E_2 &= \{(k+1,0),(k+2,1),\dots,(n,n-k-1)\},\\ E_3 &= \{(0,1),(1,2),\dots,(n-1,n)\},\\ E_4 &= \{(1,0),(2,1),\dots,(n,n-1)\}. \end{aligned}$$

The arc weight w_{ij} of an arc $(i,j) \in E$ is defined by

$$w_{ij} = \begin{cases} \lfloor z + \sum_{j=i}^{i+k} a_j \rfloor & (\text{ if } (i,j) \in E_1),\\ -\max\{\lceil -z + \sum_{j=i}^{i+k} a_j \rceil, 0\} & (\text{ if } (i,j) \in E_2),\\ 1 & (\text{ if } (i,j) \in E_3),\\ 0 & (\text{ if } (i,j) \in E_4), \end{cases}$$

where the variable z is fixed. The arc sets E_1, E_2, E_3, E_4 correspond to the constraints (2), (3), (4), (5), respectively. Note that the graph has $O(n)$ arcs.

A negative cycle is an elementary directed cycle C satisfying that the sum total of the arc weights in C is negative. The detection of a negative cycle in H can be done in $O(n^{1.5}\log(n\Gamma))$ time by using Gabow-Tarjan's scaling algorithm, where Γ is the maximum weight, and hence $\log(n\Gamma) = O(\log n)$ in our graph.

If there exists a negative cycle, then the inequality system (2), (3), (4), (5) is infeasible. Indeed, let $C = ((i_0,i_1),(i_1,i_2),\dots,(i_{p-1},i_p),(i_p,i_0))$ be a negative cycle, i.e., $\sum_{(i,j)\in C} w_{ij} < 0$. Then by adding the inequalities corresponding C we have the following inequalities

$$0 = (x_{i_0} - x_{i_1}) + (x_{i_1} - x_{i_2}) + \cdots + (x_{i_p} - x_{i_0}) \le \sum_{(u,v)\in M'} w_{uv} < 0$$

and it is a contradiction.

On the other hand, if the graph contains no negative cycle, we can find an integer vector $(x_0, \dots, x_n)$ which satisfies the inequalities (2), (3), (4), (5) as follows: If a strongly connected graph does not have any negative cycle, for any pair of vertices (i, j), the shortest path length from i to j is well defined. Here we denote the length of the shortest path from the vertex i to n by x_i^*. Then the optimality principle implies that the solution satisfies the inequalities (2), (3), (4), (5). Since the arc weights are integer valued, the shortest path length x_i^* is also integer valued for each vertex i. The path lengths x_i^* $(i = 1, 2, ..., n-1)$ can be also computed by using Gabow-Tarjan's algorithm.

Now, we discuss the method to find the optimal value of z (i.e., smallest z causing no negative cycle). We employ 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 $Q = \{q(s, i) | 1 \leq i \leq n-k, -k \leq s \leq k\}$ 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 $O(\log nk) = O(\log n)$ times with additional $O(n \log n)$ time for each search process. Thus we can find the optimal value of z in $O(n^{1.5} \log^2 n)$ time in total.

Hence, we have the following theorem:

Theorem 3.1 *The one-dimensional 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 3.2 *The one-dimensional optimized digital approximation can be computed in $O(k^2 n \log n)$ time using $O(nk)$ space.*

Proof We give an $O(k^2 n)$ time algorithm for checking the existence of negative cycles of H. For each index $p \in \{0, 1, 2, \dots, n\}$, the subgraph of H induced by the vertices $\{0, 1, \dots, p\}$ is denoted by H_p. It is clear that H_p is strongly connected. For each triplet of vertices (i, j, p) satisfying $i, j \in \{p-k, \dots, p\}$ and $k \leq p$, $d(i, j, p)$ denotes the shortest path length from i to j in the graph H_p when H_p does not contain any negative cycle. We first find a negative cycle in H_k, if it exists. If the graph H_k does not contain any negative cycle, we calculate the values $\{d(i, j, k) \mid i, j \in \{0, \dots, k\}\}$ by using an all-pairs shortest path algoirthm. It takes $O(k^{2.5} \log n)$ time so far by using methods given in the proof of the previous theorem.

Suppose that H_p has no negative cycle, and we have already computed $\{d(i, j, m) \mid i, j \in \{0, \dots, m\}\}$ for all $m \leq p$. If the graph H_{p+1} has no negative cycle. Then we can calculate the values $\{d(i, j, p+1) \mid i, j \in \{p-k+1, \dots, p+1\}\}$ from the values $\{d(i, j, p) \mid i, j \in \{p-k, \dots, p\}\}$ easily. It is because, a shortest path P from i to j in H_{p+1} satisfies one of the following two conditions; (1) P is contained in H_p, or (2) P is partitioned into a sequence of three subpaths (P_1, P_2, P_3) such that P_1, P_3 are contained in H_p and P_2 is a path consists of two arcs a_1, a_2 where a_1 enters to the vertex $p+1$ and a_2 leaves from $p+1$. Since both the in-degree and out-degree of each vertex is bounded by 4, we can calculate the values $\{d(i, j, p+1) \mid i, j \in \{p-k+1, \dots, p+1\}\}$ from the values $\{d(i, j, p) \mid i, j \in \{p-k, \dots, p\}\}$ in $O(k^2)$ when we maintain the values $\{d(i, j, p) \mid, j \in \{p-k, \dots, p\}\}$ by an $(k+1) \times (k+1)$ array.

On the other hand, if H_{p+1} has a negative cycle C^*, C^* must contain the vertex $p+1$. Since we have the shortest length in H_p between each (ordered) pair of vertices in $\{p-k, p-k+1, \dots, p\}$ and each vertex in H_{p+1} which is adjacent to $p+1$ is contained in $\{p-k, p-k+1, \dots, p\}$, we can check the existence of a negative cycle easily. Since the degree of each vertex is bounded by a constant, we can check the existence of a negative cycle in constant time. From the above, we can check the existence of the negative cycle of H in $O(k^2 n)$ time.

When the variable z is fixed and the graph H does not have any negative cycle, we can find the shortest path lengths x_i^* for $i = 1, 2, \dots, n-1$ in $O(k^2 n)$ time by using a similar argument.

By replacing the $O(n^{1.5} \log n)$ time algorithm of Gabow and Tarjan with the above algorithm, we have an $O(k^2 n \log n)$ time algorithm for our one-dimensional digital halftoning problem □

3.2 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 $a_{t+1} + sum(t) \leq 1/2$ 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 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 $I \in \mathcal{I}$.

4 Two-dimensional problem

4.1 Easy cases

We have seen that the problem is polynomial time solvable if $l = 1$. Of course, if the neighborhood family is the set of intervals of a space filling curve of the grid, it is essentially one-dimensional, and can be solved. Also, if we consider a neighborhood family consisting of regions of size 2 (domino shapes located vertical, horizontal, or diagonal), the problem is polynomial-time solvable, since it is formulated as a 2SAT problem (we omit details in this version). It is, however, not known whether the problem is polynomial-time solvable or not for the neighborhood family consisting of all size-3 dominos each of which is located vertical or horizontal.

4.2 NP-hard case

Let us consider the case where $k = l = 2$. In this case, the neighborhood of (i, j) is the 2×2 square region consisting of $(i, j), (i+1, j), (i, j+1)$ and $(i+1, j+1)$. For this neighborhood family, we show the NP-hardness of a problem of determining whether given an input image A there is a binary matrix B for which the maximum error is bounded by some given constant. More concretely, we prove the following:

Theorem 4.1 *For the above neighborhood family, it is NP hard to decide whether the optimized approximation B of an input A satisfies $Dist(A, B) \geq 1$ or $Dist(A, B) \leq 1/2$.*

Our input matrix A is obtained as a matrix with half-integral entries (i.e., $0, 1/2$, or 1). In order to prove the theorem, we prepare some useful definitions and a lemma. A zero-entry of A is called an *absolute-zero* entry if it is contained in a 2×2 square such that its all entries are zeros. A pair of two $1/2$ entries is called a *good pair* if there exists a 2×2 square region consisting of the pair and two absolute-zero entries. The following lemma is immediate:

Lemma 4.2 *If $Dist(A, B) \leq 1/2$, each absolute-zero entry must become 0 in B. Moreover, each good pair must become a pair of a 0 entry and a 1 entry.*

We prove the theorem by using a reduction from the planar 3-SAT problem [6]. The instance of planar 3-SAT is a Boolean expression $E = E_1 \wedge E_2 \wedge \ldots \wedge E_m$ where each clause E_j contains at most three literals which are variables or their negations and a planar graph is defined by the vertex set $\{E_1, E_2, \dots, E_m, u_1, u_2, \dots, u_q\}$ and the edge set $\{(E_i, u_j)|E_j$ contains u_j or $\overline{u_j}\}$. The nodes E_i ($i = 1, 2, \dots, m$) are called the *clause nodes*, while the nodes u_j ($j = 1, 2, \dots, q$) are called the *literal nodes*. Then, the problem is to decide whether there exists an assignment $F \subseteq \{u_1, \overline{u_1}, u_2, \overline{u_2}, \dots, u_q, \overline{u_q}\}$ making the expression E true.

A polynomial time reduction from a planar 3-SAT problem to the corresponding optimal halftoning is established as follows: Suppose that we are given a Boolean expression E of the above form together with a planar graph defined above. We embed an input graph on a pixel grid of size polynomial in the total number of clauses and variables. Each literal node can be replaced with a tree of branching degree 3 to give the same SAT instance.

Two pixels (i, j) and (i', j') in the grid is called adjacent if and only if the neighbor $W(i, j)$ contains (i', j') or $W(i', j')$ contains (i, j). In our neighborhood family, (i, j) has eight adjacent pixels. A planar graph of degree 3 can be embedded into the grid so that each edge is represented by a series of adjacent nodes, in such a way that no pair of pixels in two different edges are not adjacent to each other except their terminal pixels, which are either literal nodes, branching nodes or clause nodes. Also, we can assume that there are enough grid space between each pair of nodes. In the following, we further modify the embedding such that the SAT assignment information is represented by using a image matrix, and define an input image matrix A so that if E is satisfiable then $Dist(A, B) \leq 1/2$, otherwise $Dist(A, B) = 1$.

For each variable node (i, j) associated to u_s, we set $a_{i,j} = 1/2$. If we assign 0 to the valuable u_s, then $b_{i,j} = 0$, else $b_{i,j} = 1$. An edge between two nodes is a path consisting of adjacent pixels each containing an 1/2. Moreover, each pair of adjacent pixels must form a good pair. Figure 1 shows an edge between two nodes X and Y. For convenience' sake, we omit to write 0 entries in the figures, and also each 1/2 entry is represented by an h (meaning "half"). The nodes X and Y also have values 1/2. Note that the direction of the edge can be bent to any of four possible slopes if we have a sufficient open space.

¿From Lemma 4.2, if there exists an approximation B of A with distance (at most) 1/2, there are only two possible assignments. One is shown in Figure 1, and the other is its opposite. This means that the value of b_Y of B at Y is uniquely determined by b_X. In the case of Figure 1, $b_Y = b_X$, However, we can define another path shown in Figure 2 forcing $b_Y = \overline{b_X}$; thus, it creates the negation of a variable. Indeed, we can make both an odd-length path and an even-length path between two nodes to control the assignment of the literal in each clause.

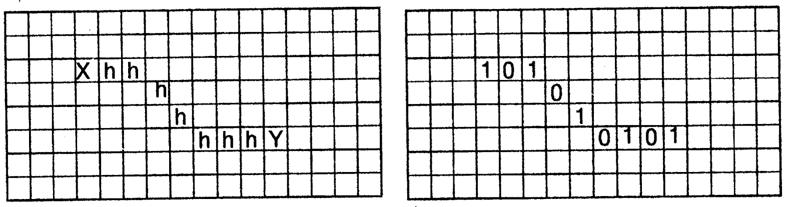

Figure 1: A gadget for an edge and its possible assignment.

A branching point (of degree 3) of is illustrated in Figure 3. Note that all zero entries of the input matrix (left-side drawing) are absolute-zero entries.

The rest to show is that we can simulate a clause node in the planar 3-SAT instance. Let the clause be $x \vee y \vee z$, where x, y and z are literals or their negations. First, we make a weaker gadget, which corresponds to $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$ (not-all-equal-3SAT clause). See Figure 4. Its upper drawing gives the assignment of 1/2 in the input matrix A. The 0-1 value of B at X, Y, Z is determined by the value of x, y, and z, respectively.

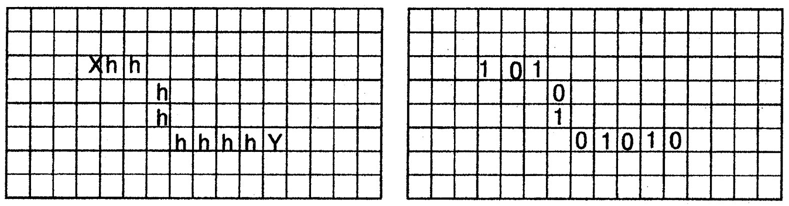

Figure 2: An edge giving the negation of the previous figure

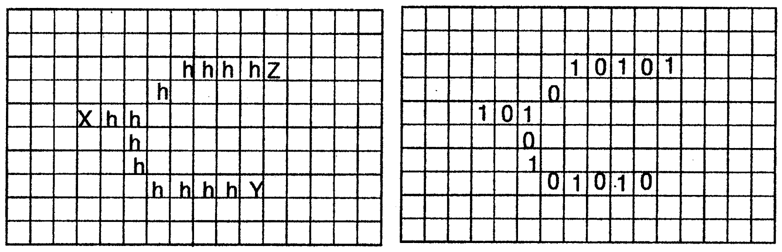

Figure 3: A gadget for a branching point.

If $dist(A, B) < 1$, once we fix the values at X, Y, and Z in B, all values except the h at the center crossing are uniquely determined Suppose that $X = Y = Z = 0$. then, we have B as the lower drawing. Then, we can see that there is no possible assignment at the center crossing p (pixel with the ? mark), since the 2×2 matrix containing p as its south-east corner has the entry sum $3/2$ in A, and hence its entry sum in B must be either 1 or 2. Hence, the value at p must be 0 in B. However, the 2×2 matrix contains p as its north-west corner also has the entry sum $3/2$ in A, and hence the value at p must be 1 in B, the contradiction. We can see that $X = Y = Z = 1$ is another impossible assignment to make $dist(A, B) < 1$; on the other hand, all other assignments are OK.

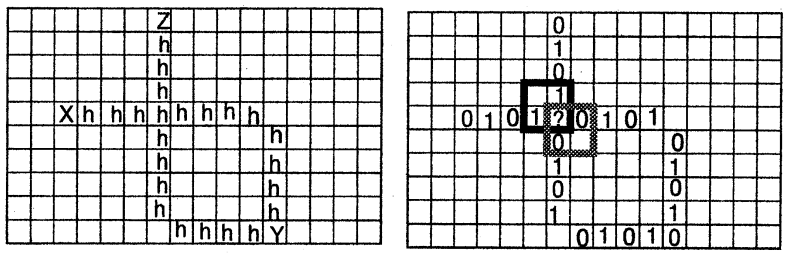

Figure 4: A gadget for a not-all-equal-3SAT clause.

Next, we modify it to the matrix in Figure 5. It is easy to see that if $X = Y = Z = 0$, there is no B such that $dist(A, B) < 1$. However, if $X = Y = Z = 1$, the right-hand side drawing shows that it is possible to make $dist(A, B) = 1/2$. A key difference is that we have a 1 entry in A, which is permitted to become 0 in B without violating the distance condition.

We have thus obtained the following proposition:

Proposition 4.3 *The planar 3SAT instance is satisfiable if and only if there exists a Boolean matrix*

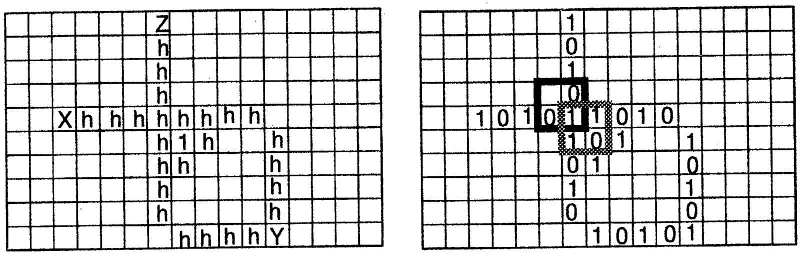

Figure 5: A gadget for a 3SAT clause.

B satisfying dist$(A, B) < 1$. Moreover, if such B exists, dist$(A, B) \leq 1/2$.

Therefore, we have the NP-hardness. We can also consider the neighborhood family consisting of all 2×2 matrices and all 1×1 matrices with the error function $|A(R) - B(R)|$. Indeed, we replace each 1 entry of A in the gadget of Figure5 with a $3/4$ entry to obtain NP hardness for the family, where $dist(A, B) \geq 1$ and $dist(A, B) \leq 3/4$ cannot be distinguished from each other. The NP-hard result can be generalized as follows (we omit the proof in this version):

Theorem 4.4 *For the neighborhood family consisting of all $2k \times 2k$ squares, it is NP hard to decide whether the optimized approximation B of an input A satisfies $Dist(A, B) \geq 2/2k$ or $Dist(A, B) \leq 1/2k$. Also, it is NP-hard to compute the optimal approximation for the neighborhood family consisting of all 3×3 squares.*

5 Worst case examples, and performance of propagating algorithms

5.1 Bad sequences in the one-dimensional case

It is important to investigate what is the (asymptotically) worst case example for the digital halftoning with respect to our criterion. We can construct such an example for the one-dimensional problem if $k = o(n)$; in other words, the sequence $\boldsymbol{a}$ maximizing the solution z in Section 3. We have already seen that z is less than 1 for any input sequence, since the propagating algorithm can attain it; hence, it suffices to construct a sequence which forces z to become 1 asymptotically.

Proposition 5.1 *There exists an input sequence which makes $z = 1 - O(k/n)$.*

Proof For readability, we only give a proof for the case where $k = 2$, i.e., each neighborhood is an interval of size 2. It is not difficult to modify it for a general k. Consider a sequence $\boldsymbol{a} = (a_i)$ $i = 1, 2, .., 2p$ defined by $0, 1/(4p-1), (4p-3)/(4p-1), 3/(4p-1), (4p-5)/(4p-1), \ldots (2p-2)/(4p-1), 2p/(4p-1)$,which satisfies $a_i + a_{i+1} = (4p-2)/(4p-1)$ and $a_{i+1} + a_{i+2} = 4p/(4p-1)$ if $i \geq 1$ is odd. There is a unique $\boldsymbol{b}$ approximating $\boldsymbol{a}$ with distance less than $(4p-2)/(4p-1)$. Indeed, $\boldsymbol{b} = 0, 0, 1, 0, 1, 0, 1, 0, \ldots, 1, 0, 1$, which is the alternating sequence except the first two zeros. Let $\boldsymbol{a}'$ be the reverse sequence of $\boldsymbol{a}$, and consider the concatenation of $\boldsymbol{a}$ and $\boldsymbol{a}'$. Naturally, the possible approximation must be the concatenation of $\boldsymbol{b}$ and its reverse $\boldsymbol{b}'$. However, in the middle part, the input is $2p/(4p-1), 2p/(4p-1)$, and the output is $1, 1$. Thus, the difference of the entry sum in this neighborhood is $2 - 4p/(4p-1) = (4p-2)/(4p-1)$. Hence, it is impossible to approximate within the distance of $(4p-2)/(4p-1)$, which proves the proposition. □

5.2 Performance of two-dimensional propagating algorithm

For two dimensional case, as seen before in the NP-hardness proof, there exists an instance A (for 2×2 square neighborhoods) that requires $dist(A, B) \geq 1$. However, the authors do not know whether there exists an instance requiring $dist(A, B) > 1$ or not.

A popular practical method in digital halftoning is the error propagation algorithm (we have already seen its one-dimensional version): Scan the grid array in the scan-line order (i.e., scan row-wise from top row to bottom row, from left to right on each row), and greedily round the entries of A into binaries propagating the remaining error at each visited pixel to its unvisited neighbor pixels. The outline of the algorithm is as follows: We use four parameters α, β, γ, and δ satisfying $\alpha + \beta + \gamma + \delta = 1$. At first, $error(i, j)$ is set to be 0 for each pixel (i, j) in the grid. When we visit the pixel (i, j), $b_{i,j}$ is obtained by rounding $a_{i,j} + error(i, j)$ to the nearer binary value. Now, compute $rem(i, j) = a_{i,j} + error(i, j) - b_{i,j}$, which is the error remained at (i, j), and update the error values of its unvisited neighbors such as $error(i, j+1) := error(i, j+1) + \alpha\ rem(i, j)$, $error(i+1, j) := error(i+1, j) + \beta\ rem(i, j)$, $error(i+1, j+1) := error(i+1, j+1) + \gamma\ rem(i, j)$, and $error(i+1, j-1) := error(i+1, j-1) + \delta\ rem(i, j)$. We need some care for pixels near the boundary of G, but we ignore it here for simplicity.

Proposition 5.2 *Suppose that B is the output of the error propagating algorithm for an input A. Then, for the neighborhood family consisting of $k \times k$ squares, $dist(A, B) \leq k + (k-1)(\gamma + \delta)$. Moreover, there is an instance A such that the distance exceeds $k + (k-1)(\gamma + \delta) - \epsilon$ for any $\epsilon > 0$ if we apply the error propagating algorithm.*

Proof It is observed that $-0.5 \leq rem(i, j) < 0.5$ holds (if we round 0.5 to 1 in the algorithm). Fix a $k \times k$ square R in G. Let S_{in} be the set of pixels outside R each of which has at least one pixel in R to which error is propagated. Also, let S_{out} be the set of pixels in R each of which has at least one pixel of $G - R$ to which error is propagated. Consider the total propagating value coming into R and also the total propagating value going out of R, we have $|A(R) - B(R)| \leq k + (k-1)(\gamma + \delta)$. On the other hand, we can manage to create the input attaining $rem(i, j) = -1/2$ for each $(i, j) \in S_{in}$, and $rem(i', j') > 1/2 - \epsilon/k$ for each $(i', j') \in S_{out}$. This gives the lower bound. □

Indeed, if $\gamma = \delta = 0$, $dist(A, B) \leq k$ holds and it is tight.

6 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 condition? (Not seeking for the optimal solution).
One possible candidate is the use of random space filling curves [1], and perform the one-dimensional 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 bad 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] T. Asano, N. Katoh, H. Tamaki, and T. Tokuyama: "Convertibility among grid filling curves," *Proc. ISAAC98, Springer LNCS 1533* (1998), pp.307–316.

[2] 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).

[3] Y. Crama, P. Hansen and B. Jaumard: "The basic algorithm for pseudo-boolean programming revisited," *Discrete Applied Mathematics,* 29 (1990), pp.171–185, 1990.

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

[5] H. N. Gabow and R. E. Tarjan: "Faster scaling algorthms for netrowk problems," *SIAM J. Comp.,* 18 (1989), pp.1013–1036.

[6] M. R. Garey and D. S. Jhonson: *Computers and Interactability: A guide to theory of NP hardness,* Feeman and Company, 1979.

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

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

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

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