

# Algorithms for linear time reconstruction by discrete tomography II

Matthew Ceko<sup>a</sup>, Silvia M.C. Pagani<sup>b,1,\*</sup>, Rob Tijdeman<sup>c</sup>

<sup>a</sup>*School of Physics and Astronomy, Monash University, Melbourne, Australia*

<sup>b</sup>*Dipartimento di Matematica e Fisica “N. Tartaglia”, Università Cattolica del Sacro Cuore, via Musei 41, 25121 Brescia, Italy*

<sup>c</sup>*Mathematical Institute, Leiden University, 2300 RA Leiden, P.O. Box 9512, The Netherlands*

---

## Abstract

The reconstruction of an unknown function  $f$  from its line sums is the aim of discrete tomography. However, two main aspects prevent reconstruction from being an easy task. In general, many solutions are allowed due to the presence of the switching functions. Even when uniqueness conditions are available, results about the NP-hardness of reconstruction algorithms make their implementation inefficient when the values of  $f$  are in certain sets. We show that this is not the case when  $f$  takes values in a field or a unique factorization domain, such as  $\mathbb{R}$  or  $\mathbb{Z}$ . We present a linear time reconstruction algorithm (in the number of directions and in the size of the grid), which outputs the original function values for all points outside of the switching domains. Freely chosen values are assigned to the other points, namely, those with ambiguities. Examples are provided.

*Keywords:* discrete tomography; ghost; lattice direction; reconstruction algorithm; switching function

---

## 1. Introduction

Tomography deals with the reconstruction of an object from the knowledge of its projections in a number of given directions. Radon [23] proved

---

\*Corresponding author

*Email addresses:* [matthew.ceko@monash.edu](mailto:matthew.ceko@monash.edu) (Matthew Ceko), [silvia.pagani@unicatt.it](mailto:silvia.pagani@unicatt.it) (Silvia M.C. Pagani), [tijdeman@math.leidenuniv.nl](mailto:tijdeman@math.leidenuniv.nl) (Rob Tijdeman)

<sup>1</sup>Research supported by D1 Research line of Università Cattolica del Sacro Cuore.

in 1917 that a differentiable function on  $\mathbb{R}^2$  can be determined explicitly by means of integrals over the lines in  $\mathbb{R}^2$ . By approximating this for a large number of projections and using filtered back projection, so-called computerized tomography provides a quick way to compute a very good representation of the object. This method has a wide range of applications, from scans in hospitals to archaeology, astrophysics and industrial environments. See e.g. [18, 20].

If the number of projection directions is small, discrete tomography may be advantageous compared to conventional back projection techniques. In this paper we consider a function  $f$  on a finite grid  $A$  of  $\mathbb{Z}^2$  representing the object. Projections become line sums, i.e. sums of the  $f$ -values at grid points on each line in finitely many given directions. Discrete tomography finds its origin in the fifties, mainly for only two directions, see e.g. [24]. In 1978, Katz [19] gave a necessary and sufficient condition for the presence of a nontrivial function with vanishing line sums, known as a switching function or ghost. The theory started to blossom in the nineties when it became relevant in the study of crystals. In 1991 Fishburn, Lagarias, Reeds and Shepp [12] gave necessary and sufficient conditions for uniqueness of reconstruction of functions  $f : A \rightarrow \{1, 2, \dots, N\}$  for some positive integer  $N$ .

An important distinction is whether the line sums are exact or may be inconsistent because of errors, termed noise, in the measurements. In case of noise the reconstruction can only be an approximation, see e.g. [2, 3, 22]. In what follows, we assume that the line sums are exact.

One of the main goals of discrete tomography is to ensure that the reconstructed function is equal to the function  $f$  from which the line sums originate. However, in general the problem is ill-posed. Therefore one investigates which additional constraints can be imposed in order to achieve uniqueness. For instance, one may use some known information about the shape of the domain of  $f$  such as convexity [13], the values  $f$  can attain (for the binary case see [4, 16], for the integer case see [6]), or the size of the domain of  $f$ , [4, 17]. In this paper we assume that the line sums come from some function  $f$  and are therefore consistent.

In 1999 Gardner, Gritzmann and Prangenberg [14] showed that the problem of reconstructing a function  $f : A \rightarrow \mathbb{N}$  from its line sums in  $d$  directions is solvable in polynomial time if  $d = 2$ , but it is NP-complete if  $d \geq 3$ . The NP-completeness concerns both consistency and uniqueness, as well as reconstruction. Moreover, a year later they showed that the three mentioned problems are NP-complete for two and more directions when more than five types of atoms are involved in the crystal [15].

We recall that the tomographic problem may be rephrased in terms of a linear system. If the function to be reconstructed has  $\mathbb{R}$  as codomain, then it is known that polynomial-time algorithms exist to solve the linear system (such as the Gauss elimination, see [1]). The crux of the NP-results in [14, 15] is therefore the requirement that the range of  $g$  is not closed under subtraction.

In 2001 Hajdu and Tijdeman [17] gave an algebraic representation of the complete set of solutions over the integers. Their result also holds for solutions over the reals or any unique factorization domain. They gave a polynomial expression for the nontrivial switching function with domain of minimal size, the so-called primitive switching polynomial, and showed that every switching polynomial is a multiple of the primitive switching polynomial. This implies that every switching function is a linear combination of domain shifts of the corresponding primitive switching function. Their result implies that arbitrary function values can be given to a certain set of points and that thereafter the function values of the other points of  $A$  are uniquely determined by the line sums. This was made explicit by Dulio and Pagani [11] and serves as a building block in this paper.

In 2015 Dulio, Frosini and Pagani [7, 8] showed that in the corners of  $A$  the function values are uniquely determined and can be computed in linear time if the number of directions  $d = 2$ . Later they proved conditional results for  $d = 3$  [9, 10]. Recently, Pagani and Tijdeman [21] generalized the result for any number of directions. In particular the object function can be reconstructed in linear time if there are no switching functions. Moreover, they showed that in general the part of  $A$  outside the convex hull of the union of all switching domains is uniquely determined and can be reconstructed in linear time. This result is another building block of our paper.

We prove that given the line sums of a function  $f : A \rightarrow \mathbb{R}$  in the directions of a set  $D$  we can compute a function  $g : A \rightarrow \mathbb{R}$  with the same line sums. Using the theory of [17] this implies that the complete set of such functions  $g$  can be explicitly presented.

Recently Ceko, Petersen, Svalbe and Tijdeman [5] constructed switching components called boundary ghosts, where the switching domain has the form of an annulus around a relatively large interior, see e.g. Figure 1. The values of  $f$  for points which do not lie on this annulus can be uniquely determined by their line sums. This paper introduces a method which makes it possible to compute these values in linear time.

The present paper relies heavily on [21], which was submitted before we started the research for the present paper. The above mentioned paper [5] did us realize that it is important to be able to compute quickly the

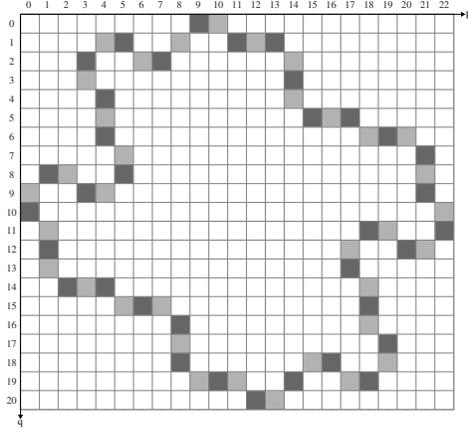


Figure 1: Boundary ghost. The grey pixels form a switching domain. The pixels inside the ghost have  $f$ -values which are uniquely determined by the line sums in the directions of  $D = \{(0, 1), (1, 0), (1, 1), (-1, 1), (-3, -1), (-1, -3), (5, -1), (7, 5), (-3, 7)\}$ .

function values at the points in the interior of the boundary ghost domain. To our surprise we discovered that a twofold extension of the method of [21] worked, even for arbitrary ghosts. We explain this twofold application in the present paper. For our method it suffices that the range of  $g$  is closed under subtraction. Further we provide a pseudo-code and a better justification of the linearity for the complexity than we did in [21].

In Section 2 we present notation and definitions, as well as information on switching functions. Section 3 shows how values of  $f$  in a corner region of  $A$  can be obtained from the line sums. The case without switching components is treated in Section 4, that with switching components in Section 5. A general algorithm to compute  $g$  can be found in Section 6. The justification of our linear time claim is given in Section 7. Conclusions are in Section 8.

## 2. Definitions and known results

We consider an  $m \times n$  rectangular grid of points

$$A = \{(p, q) \in \mathbb{Z}^2 : 0 \leq p < m, 0 \leq q < n\}.$$

In our figures the  $x$ -axis is oriented from left to right and the  $y$ -axis from top to bottom. The origin is therefore the upper-left corner point of  $A$ . For each point  $(p, q) \in \mathbb{Z}^2$  we consider the pixel  $\{(x, y) \in \mathbb{R}^2 : p \leq x < p+1, q \leq y < q+1\}$ . In figures the coordinates of a pixel are the coordinates of the attached point.

Primitive directions are pairs  $(a, b)$  of coprime integers. We agree to identify directions  $(a, b)$  and  $(-a, -b)$ . Since we only consider primitive directions, we simply call them directions. The horizontal and the vertical direction are given by  $(1, 0)$  and  $(0, 1)$ , respectively. We consider a finite set of directions  $D = \{(a_h, b_h) : h = 1, \dots, d\}$ . We say that  $D$  is valid for  $A$  if  $M := \sum_{h=1}^d a_h < m$  and  $N := \sum_{h=1}^d |b_h| < n$ , and nonvalid otherwise.

A lattice line  $L$  is a line containing at least two points in  $\mathbb{Z}^2$ . Let  $f : A \rightarrow \mathbb{R}$ . The line sum of  $f$  along the lattice line  $L(a, b, c) : ay = bx + c$  with direction  $(a, b)$  is defined as

$$\ell(a, b, c, f) = \sum_{aq=bp+c, (p,q) \in A} f(p, q).$$

A function  $F : A \rightarrow \mathbb{R}$  is called a *switching function* or *ghost* of  $(A, D)$  if all the line sums of  $F$  in all the directions of  $D$  are zero. Observe that then  $f$  and  $f + F$  have the same line sums in the directions of  $D$ . The support of a switching function is called a *switching domain*.

We say that something can be computed in linear time if the number of basic operations needed to compute it is  $\mathcal{O}(dmn)$ . Here a basic operation is an addition, subtraction, multiplication, division, decision about which of two quantities is larger or an assignment.

### 2.1. The location of switching domains

M. Katz [19] proved that  $f : A \rightarrow \mathbb{R}$  is uniquely determined by the line sums in the directions of  $D$  if and only if  $(A, D)$  is nonvalid. Fishburn et al. [12] showed that  $(p, q) \in A$  has a unique  $f$ -value if and only if  $(p, q)$  is not located in a switching domain. Hajdu and Tijdeman [17] associated to the function  $f : A \rightarrow \mathbb{R}$  the polynomial  $f^*(x, y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} f(i, j)x^i y^j$ . In this way every switching function corresponds with a switching polynomial. They defined

$$g_{(a,b)}^*(x, y) = \begin{cases} x^a y^b - 1 & \text{if } a > 0, b > 0, \\ x^a - y^{-b} & \text{if } a > 0, b < 0, \\ x - 1 & \text{if } a = 1, b = 0, \\ y - 1 & \text{if } a = 0, b = 1, \end{cases}$$

and

$$G_{i,j}^*(x, y) = x^i y^j \prod_{h=1}^d g_{(a_h, b_h)}^*(x, y)$$

for  $0 \leq i < m - M, 0 \leq j < n - N$ . They showed that  $G_{0,0}^*$  is a switching polynomial of minimal degree. We call the corresponding function a primitive switching function. Furthermore they proved the following result.

**Theorem 1** (Hajdu, Tijdeman [17], Theorem 1). *Suppose  $D$  is valid for  $A$ . Put  $M = \sum_{h=1}^d a_h, N = \sum_{h=1}^d |b_h|$ . Then for every switching function  $g : A \rightarrow \mathbb{R}$  its switching polynomial  $g^*$  can be uniquely written as*

$$g^* = \sum_{i=0}^{m-1-M} \sum_{j=0}^{n-1-N} c_{i,j} G_{i,j}^* \quad (1)$$

with  $c_{i,j} \in \mathbb{R}$  for all  $i, j$ . Conversely, every function  $g$  of which the switching polynomial is of the form (1) is a switching function.

This result is also valid if  $\mathbb{R}$  is replaced by  $\mathbb{Z}$  or any other field or unique factorization domain. A corollary of the theorem relevant for this paper is that the lexicographically lowest degree term of  $G_{i,j}^*$  is given by  $x^i y^{j+N_n}$  where  $N_n = \sum_{b_h < 0} -b_h$ . Thus we have free choice for the values of  $c_{i,j}$  for  $0 \leq i < m - M, N_n \leq j < N_n + n - N$  and by this choice the function  $g^*$  is uniquely determined. An illustration of Theorem 1 is given in Figure 2.

### 3. Uniqueness in the corner regions

Let again  $A = \{(p, q) \in \mathbb{Z}^2 : 0 \leq p < m, 0 \leq q < n\}$ . Let  $D$  be a set of directions  $(a_1, -b_1), \dots, (a_k, -b_k)$  with  $k \geq 2$  where  $a_1, \dots, a_k, b_1, \dots, b_k$  are positive integers ordered such that

$$\frac{b_1}{a_1} < \frac{b_2}{a_2} < \dots < \frac{b_k}{a_k}. \quad (2)$$

Note that by primitivity all the ratios are distinct. We call the points

$$\left( \sum_{h=1}^k a_h, 0 \right), \left( \sum_{h=2}^k a_h, b_1 \right), \left( \sum_{h=3}^k a_h, \sum_{h=1}^2 b_h \right), \dots, \left( 0, \sum_{h=1}^k b_h \right)$$

the border points of the upper left region  $(P_0, Q_0), (P_1, Q_1), \dots, (P_k, Q_k)$ , respectively. We denote the convex hull of the three points  $(0, 0), (P_{h-1}, Q_{h-1}), (P_h, Q_h)$  by  $V_h$  for  $h = 1, 2, \dots, k$  (see Figure 3). Let

$$V_{UL} = \bigcup_{h=1}^k V_h$$

be the upper left corner region. The other corner regions  $V_{UR}, V_{LL}, V_{LR}$  may be defined similarly (see Figure 2).

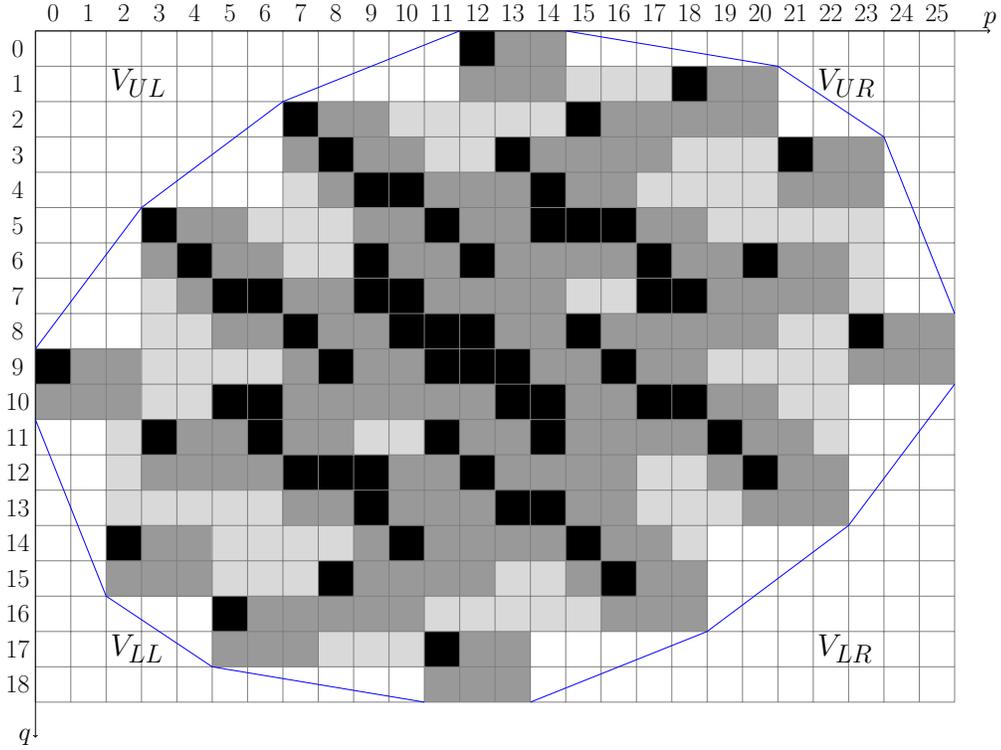


Figure 2: The situation for the 26 by 19 grid  $A$  and the set of directions  $D = \{(5, -2), (4, -3), (3, -4), (6, 1), (3, 2), (2, 5)\}$ . The dark grey and black pixels indicate the union of the switching domains. The black pixels represent the switching domain related to  $G_{0,0}^*$ . In the dark grey and black pixels the function  $f$  is not uniquely determined by its line sums in the directions of  $D$ . The  $f$ -values of the complement, the white and light grey pixels, are uniquely determined by these line sums. Since  $M = 23$ ,  $N = 17$ , there are six pixels where the choice is free, e.g. the pixels  $(0, 9)$ ,  $(0, 10)$ ,  $(1, 9)$ ,  $(1, 10)$ ,  $(2, 9)$ ,  $(2, 10)$ . Any other 3 by 2 block of dark grey and black pixels can be chosen instead. If the choice is made all the values of the unique solution satisfying the made choices are determined by the line sums in the directions of  $D$ . The white pixels form four corner regions. The broken line indicates the convex hull of the union of the switching components.

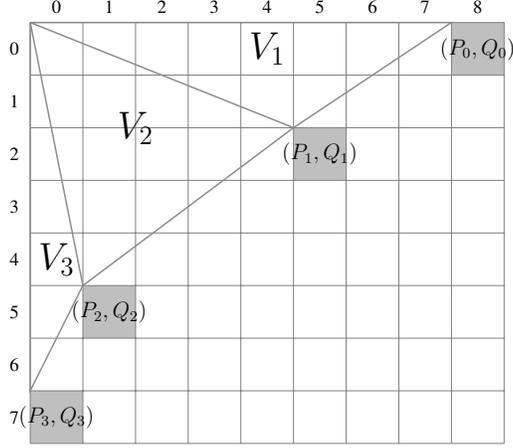


Figure 3: The triangles  $V_1, V_2, V_3$  for the set  $D = \{(3, -2), (4, -3), (1, -2)\}$ . The border points are  $(P_0, Q_0) = (8, 0)$ ,  $(P_1, Q_1) = (5, 2)$ ,  $(P_2, Q_2) = (1, 5)$ ,  $(P_3, Q_3) = (0, 7)$ . For every  $h$  the line through  $(P_{h-1}, Q_{h-1})$  and  $(P_h, Q_h)$  is a side of triangle  $V_h$ , and intersects each other triangle  $V_{\tilde{h}}$ , since the slopes increase with increasing  $\tilde{h}$  by the ordering in (2).

For a point  $(p, q) \in A$  we define its weight  $w(p, q)$  by

$$w(p, q) = \min_{h=1,2,\dots,k} \frac{b_h p + a_h q}{b_h P_h + a_h Q_h}.$$

The weight function in  $V_{UL}$  equals the quotient of the distance of the point  $(p, q)$  to the origin  $(0, 0)$  and the distance from the origin to the intersection  $(p', q')$  of the line through  $(0, 0)$  and  $(p, q)$  and the boundary of the convex hull. This weight has the property that every point  $(p, q)$  in  $V_{UL}$  has maximal weight among the integer points on the line  $\ell$  through  $(p, q)$  parallel to the line segment of the boundary of the convex hull through  $(p', q')$ .

The following lemma implies that if  $(p, q) \in V_{UL}$ , then the minimum in the definition of  $w(p, q)$  is reached for  $h$  such that  $(p, q) \in V_h$  (see Figure 4).

**Lemma 2** ([21], Lemma 2). *For  $(p, q) \in A$  the weight  $w(p, q)$  is reached for  $h$  such that*

$$\frac{Q_{h-1}}{P_{h-1}} \leq \frac{q}{p} \leq \frac{Q_h}{P_h}.$$

*and only for such  $h$ . The weight 1 is reached at the border points and not at other points of  $A$ .*

The next result states that the corner region  $V_{UL}$  except for the border points has unique  $f$ -values which can be computed in linear time.

	0	1	2	3	4	5	6	7	8
0	0	.125	.250	.375	.500	.625	.750	.875	1.000
1	1	2	4	7	11	15	20	26	$(P_0, Q_0)$
2	3	6	9	12	17	22	28		
3	5	10	14	19	25	$(P_1, Q_1)$			
4	8	16	21	27					
5	13	23	29						
6	18	$(P_2, Q_2)$							
7	24								
8	$(P_3, Q_3)$								

Figure 4: The weights (upper number inside each pixel) of the points for directions  $(3, -2), (4, -3), (1, -2)$ . The border points are  $(8, 0), (5, 2), (1, 5), (0, 7)$ . All the points with weight less than 1 are in the corner region and have uniquely determined  $f$ -values (Theorem 3). The lower numbers enumerate them with increasing weights. The (dark grey) border pixels are part of the switching domain and their  $f$ -values are therefore not uniquely determined. They have weight 1. All entirely white pixels have weight  $> 1$ .

**Theorem 3** ([21], Theorem 4 and Corollary 6). *Let  $A = \{(p, q) \in \mathbb{Z}^2 : 0 \leq p < m, 0 \leq q < n\}$ . Let  $D$  be a set of directions  $(a_1, -b_1), \dots, (a_k, -b_k)$  where  $a_1, \dots, a_k, b_1, \dots, b_k$  are positive integers ordered as in (2). Let the line sums of  $f : A \rightarrow \mathbb{R}$  in the directions of  $D$  be given. Then all the points  $(p, q)$  in  $V_{UL}$  except for the border points have uniquely determined  $f$ -values.*

**Corollary 4** ([21]). *The values of the points as in the previous theorem can be computed according to increasing weights and, if  $(p, q) \in V_h$ , by subtracting the sum of the  $f$ -values of the other points of  $A$  on the line through  $(p, q)$  in the direction of  $(a_h, b_h)$  from its line sum.*

**Corollary 5.** *Under the above conditions the  $f$ -values of the points  $(0, 0), (0, 1), \dots, (0, -1 + \sum_{h=1}^k b_h)$  can all be computed in linear time.*

#### 4. The nonvalid case

Suppose we are in the nonvalid case, then  $M \geq m$  or  $N \geq n$ . Without loss of generality assume that  $N \geq n$ . Then we apply Theorem 3 both to the upper corner region  $V_{UL}$  and to the lower corner region  $V_{LL}$ .

Let  $A$  be as above. Let

$$D = \{(a_1, -b_1), \dots, (a_k, -b_k), (a_{k+1}, b_{k+1}), \dots, (a_d, b_d), (0, 1)^*, (1, 0)^*\}$$

where  $a_1, \dots, a_d, b_1, \dots, b_d$  are positive integers ordered such that

$$\frac{b_1}{a_1} < \frac{b_2}{a_2} < \dots < \frac{b_k}{a_k}, \quad \frac{b_{k+1}}{a_{k+1}} > \frac{b_{k+2}}{a_{k+2}} > \dots > \frac{b_d}{a_d}$$

and the asterisk indicates that  $(0, 1)$  and/or  $(1, 0)$  may occur in  $D$ . Thus we assume that  $n \leq \sum_{h=1}^d b_h$  or  $(n = 1 + \sum_{h=1}^d b_h \text{ and } (0, 1) \in D)$ .

By Corollary 5 applied to  $V_{UL}$ , the  $f$ -values of the points  $(0, 0), (0, 1), \dots, (0, -1 + \sum_{h=1}^k b_k)$  can be computed. In a similar way we can apply the corollary to  $V_{LL}$  and the directions  $(a_{k+1}, b_{k+1}), \dots, (a_d, b_d)$  to conclude that the  $f$ -values of the points  $(0, n-1), (0, n-2), \dots, (0, n - \sum_{h=k+1}^d b_h)$  can be computed. It follows that the  $f$ -values of the points  $(0, 0), (0, 1), \dots, (0, n-1)$  can all be computed except when  $n = 1 + \sum_{h=1}^d b_h$  and  $(0, 1) \in D$ . In the latter case  $(p, q) = (0, \sum_{h=1}^k b_k)$  is the only point in the column  $p = 0$  with unknown  $f$ -value. However, this value can be found by subtracting from the line sum of the column  $p = 0$  the  $f$ -values of the other points in that column. In this way we have made our problem of computing the  $f$ -values one column smaller. We can repeat the procedure in order to find the  $f$  values of the next column. Continuing the process we arrive at the following conclusion.

**Theorem 6** ([21]). *Let  $A = \{(p, q) \in \mathbb{Z}^2 : 0 \leq p < m, 0 \leq q < n\}$ . Let  $D$  be a set of directions such that  $A$  is nonvalid for  $D$ . Let the line sums of  $f : A \rightarrow \mathbb{R}$  be given. Then the  $f$ -values of all points of  $A$  can be computed in linear time.*

In [21] algorithms are given for computing the  $f$ -values. These algorithms are more efficient than the procedure described above. The algorithm in Section 6 is as efficient as these algorithms.

## 5. The valid case

Let  $A = \{(p, q) \in \mathbb{Z}^2 : 0 \leq p < m, 0 \leq q < n\}$  and

$$D = \{(a_1, -b_1), \dots, (a_k, -b_k), (a_{k+1}, b_{k+1}), \dots, (a_d, b_d), (0, 1)^*, (1, 0)^*\}$$

where  $a_1, \dots, a_d, b_1, \dots, b_d$  are positive integers ordered such that

$$\frac{b_1}{a_1} < \frac{b_2}{a_2} < \dots < \frac{b_k}{a_k}, \quad \frac{b_{k+1}}{a_{k+1}} > \frac{b_{k+2}}{a_{k+2}} > \dots > \frac{b_d}{a_d}$$

and the asterisk indicates that  $(0, 1)$  and/or  $(1, 0)$  may occur in  $D$ . As observed in the previous section, by applying Corollary 5 to  $V_{UL}$  the  $g$ -values of the points  $(0, 0), (0, 1), \dots, (0, -1 + \sum_{h=1}^k b_h)$  can be computed. In a similar way we can apply the corollary to  $V_{LL}$  and the directions  $(a_{k+1}, b_{k+1}), \dots, (a_d, b_d)$  to conclude that the  $g$ -values of the points  $(0, n-1), (0, n-2), \dots, (0, n - \sum_{h=k+1}^d b_h)$  can be computed. In Section 2.1 it was observed that the  $g$ -values of the points  $(0, Q_k), (0, Q_k + 1), \dots, (0, Q_k + n - N - 1)$  can be freely chosen where  $g : A \rightarrow \mathbb{R}$  is a function satisfying the line sums. Combining these results we see that all the  $g$ -values of the points  $(0, 0), (0, 1), \dots, (0, n-1)$  can be computed or freely chosen, except for the case that  $(0, 1) \in D$  and  $n = 1 + \sum_{h=1}^d b_h$ . In the latter case only the  $g$ -value of  $(0, \sum_{h=1}^k b_h)$  is not determined, but this can be computed by subtracting the  $g$ -values of the other points in the leftmost column from the sum of that column. After this all  $g$ -values of the points in the leftmost column are fixed. In Section 2.1 it was further observed that the  $g$ -values of the points  $(p, Q_k), (p, Q_k + 1), \dots, (p, Q_k + n - N - 1)$  for  $p = 1, 2, \dots, m - M - 1$  can be freely chosen. Therefore we can repeat the above procedure successively for columns  $p = 1, 2, \dots, m - M - 1$ . Then  $M$  columns remain, for which the line sums in the directions of  $D$  are known. It is obvious that the computed  $g$ -values of the points which do not belong to a switching domain have the original  $f$ -value. We are left with a nonvalid case and we can apply an algorithm for that case to compute the remaining  $g$ -values.

We have shown that the following theorem holds.

**Theorem 7.** *Let  $A = \{(p, q) \in \mathbb{Z}^2 : 0 \leq p < m, 0 \leq q < n\}$  and  $D$  a set of primitive directions. Let  $f : A \rightarrow \mathbb{R}$  be an unknown function. Suppose all the line sums in the directions of  $D$  are known. Then we can compute a function  $g : A \rightarrow \mathbb{R}$  satisfying the line sums in linear time. The points which do not belong to any switching domain get their original  $f$ -value.*

**Example 8.** Consider the situation in Figure 2. We have  $m = 26, M = 23, n = 19, N = 17, Q_k = 9$ . We can freely choose the  $g$ -values of the points  $(0, 9)$  and  $(0, 10)$  and compute the  $g$ -values of the other points  $(0, q)$ . Next we do so for the columns  $p = 1$  and  $p = 2$ . We are left with a  $23$  by  $19$  rectangular grid. Since  $m = M = 23$ , this is a nonvalid case and we know that the remaining  $g$ -values can be computed in linear time. The found  $g$ -values of the white and light grey pixels are equal to the original  $f$ -values.

**Remark 9.** In this paper we assume that the line sums are correct and that there is no noise. It is easy to check whether this is true afterwards by

checking the line sums which have not been used for computing the  $g$ -values. In case the line sums are inconsistent, and it is better to use a method which treats the unused line sums in a similar way as the used line sums to obtain a good approximation of the original function.

**Remark 10.** Theorem 1 states that if  $g : A \rightarrow \mathbb{R}$  has the same line sums as  $f$ , then the associated polynomial  $g^*$  is of the form

$$f^* + \sum_{i=0}^{m-1-M} \sum_{j=0}^{n-1-N} c_{i,j} G_{i,j}^*$$

and each such function has the same line sums as  $f$ . It is possible to compute the coefficients  $c_{i,j}$  as follows. The point  $(0, Q_k)$  occurs only in the domain of  $G_{0,0}$  and therefore  $c_{0,0}$  can be found from the found value for  $(0, Q_k)$ . The point  $(0, Q_k + 1)$  occurs in  $G_{0,1}$  and maybe in  $G_{0,0}$ . Since  $c_{0,0}$  is already known,  $c_{0,1}$  can be computed. Considering the points with a free choice in the lexicographic order, each time a point occurs in only one new primitive switching domain and hence the corresponding coefficient can be computed.

## 6. An efficient algorithm

In this section we present an algorithm to find a function  $g : A \rightarrow \mathbb{R}$  which satisfies the given line sums of an unknown function  $f : A \rightarrow \mathbb{R}$ . This algorithm is based on ideas and results in [21]. Its complexity is studied in the next section.

In this algorithm it is not necessary to compute all weights as in Figure 4. Observe that after the  $g$ -values in  $V_{UL}$  and  $V_{LL}$  have been computed, the  $g$ -value of the next point on each row will be computed. For this the same direction will be used as used for the integer point immediately left of it, since everything shifts one place to the right. For the same reason the order in which the new points will be handled will be the same as for the points immediately left of them. This process will be continued. Thus, where in the example of Figure 4 initially on the row  $q = 1$  first the direction  $(1, -2)$ , next the direction  $(4, -3)$ , and finally the direction  $(3, -2)$  was used to compute the  $g$ -value, more to the right only the direction  $(3, -2)$  would be used. Suppose there would have been seven more columns  $p = -1, -2, \dots, -7$  with  $g$ -values equal to 0 in  $A$  (cf. Algorithm 1B of [21]). Then for  $p \geq 0$  we would have needed only the rightmost direction on each row and the direction needed to compute the  $g$ -values of points in the original  $A$  would only depend on their row. Therefore it suffices to follow the order in which



The above example illustrates the first seven steps of the algorithm. The above procedure is done for both  $V_{UL}$  and for  $V_{LL}$ . In between  $g$ -values 0 are substituted (or any other values) at the places where the switching domains offer free choice. Now the  $g$ -values in the first column are known and we can proceed with the next column and so on until we have treated  $m - M$  columns. An  $M$  by  $n$  grid remains to be handled, but this is a nonvalid case. This can be treated in a similar way, but starting from upper corner regions  $V_{UL}$  and  $V_{UR}$  and then going downwards. In the algorithm we have  $g(p, q) = f(p, q)$  for all the points  $(p, q)$  which are not in a switching domain of  $(A, D)$ .

The algorithm has 12 steps and is illustrated in Example 13 following it. For Steps 1-7 see Figure 6, for Steps 8-12 see Figure 7.

In Step 1 the directions are ordered in such a way that the corner regions become concave. Throughout the algorithm we write  $b_h$  instead of  $-b_h$  ( $h = 1, \dots, k$ ) so that  $b_h$  is always nonnegative. In Step 2 the border points of  $V_{UL}$  and  $V_{LL}$  are found. In Step 3 a function  $\delta$  is introduced indicating whether the  $g$ -value of a point has been computed. Step 4 provides a shortcut for nonvalid cases. If  $n \leq N$ , then this shortcut can be used after rows and columns have been interchanged. Step 5 serves to find the grid points left of the border line, measured by the sequence  $r$ . The weights of these points are computed as well, together with the sequence  $s$ , which indicates the direction of the line used to compute the  $g$ -value. In Step 6 the points found in the previous step are ordered after increasing weight by the sequence  $o$ . If weights are equal, the order is irrelevant. In Step 7 the  $g$ -values of the first  $m - M$  columns (and of some more points of  $A$ ) are computed.

Steps 8-12 are essentially equal to Steps 1-7, but with the columns  $p = 0, 1, \dots, m - M - 1$  omitted, as they have already been treated, and the roles of the rows and columns interchanged. In Step 8 the directions are reordered as now  $V_{UL}$  is mirrored and  $V_{UR}$  takes over the role of  $V_{LL}$ . A similar reordering of border points takes place in Step 9. The weights and the corresponding directions are found in Step 10. The order of the points found in the previous step are fixed in Step 11. Finally the remaining  $g$ -values are computed in Step 12 where the  $u$  is introduced to make the necessary shift because of the omitted  $m - M$  columns.

In the algorithm,  $\lambda(\mathbf{d}, p, q)$  denotes the line sum containing the point  $(p, q)$  in the direction  $\mathbf{d}$ , and  $\lceil r \rceil$  denotes the ceiling of  $r$ .

**Algorithm.**

**Input:** A set  $A = \{(p, q) : 0 \leq p < m, 0 \leq q < n\}$  with positive integers  $m, n$ , a finite set of (primitive) directions  $D$  and all the line sums in the

directions of  $D$  of a function  $f : A \rightarrow \mathbb{R}$ .

**Output:** Function  $g : A \rightarrow \mathbb{R}$  which satisfies the line sums.

**Step 1:** Initial values.

**for all**  $\mathbf{d} = (a, -b) \in D$  (with  $a > 0, b > 0$ ) **order the directions such that**

$$\frac{b_1}{a_1} < \frac{b_2}{a_2} < \dots < \frac{b_k}{a_k}.$$

**for all**  $\mathbf{d} = (a, b) \in D$  (with  $a > 0, b > 0$ ) **order the directions such that**

$$\frac{b_{k+1}}{a_{k+1}} > \frac{b_{k+2}}{a_{k+2}} > \dots > \frac{b_d}{a_d}.$$

$$M \leftarrow \sum_{h=1}^d a_h$$

$$N \leftarrow \sum_{h=1}^d b_h$$

**if**  $(1, 0) \in D$  **then**  $M \leftarrow M + 1$

**if**  $(0, 1) \in D$  **then**

$$N \leftarrow N + 1$$

$$\mathbf{d}_0 \leftarrow (0, 1)$$

**Step 2:** Border points.

$$(P_0, Q_0) \leftarrow \left( \sum_{h=1}^k a_h, 0 \right)$$

**for**  $h \leftarrow 1$  **to**  $k$  **do**

$$(P_h, Q_h) \leftarrow (P_{h-1} - a_h, Q_{h-1} + b_h)$$

$$(P_k^*, Q_k^*) \leftarrow \left( 0, n - 1 - \sum_{j=k+1}^d b_j \right)$$

$$(P_{k+1}, Q_{k+1}) \leftarrow (P_k^* + a_{k+1}, Q_k^* + b_{k+1})$$

**for**  $h \leftarrow k + 2$  **to**  $d$  **do**

$$(P_h, Q_h) \leftarrow (P_{h-1} + a_h, Q_{h-1} + b_h)$$

**Step 3:** Fixing switching functions.

**for**  $p \leftarrow 0$  **to**  $m - 1$  **do**

**for**  $q \leftarrow 0$  **to**  $n - 1$  **do**

$$\delta(p, q) \leftarrow 0$$

**for**  $p \leftarrow 0$  **to**  $m - M - 1$  **do**

**for**  $q \leftarrow Q_k$  **to**  $Q_k + n - N - 1$  **do**

$$g(p, q) \leftarrow 0$$

$$\delta(p, q) \leftarrow 1$$

**Step 4:** Nonvalid case.

**if**  $m \leq M$  **then** goto Step 8

**Step 5:** Choosing starting points  $r_h$ , weights  $w(r_h, h)$  and directions  $\mathbf{d}_{s(h)}$ .

**for**  $H \leftarrow 1$  **to**  $k$  **do**  
  **for**  $h \leftarrow Q_{H-1}$  **to**  $Q_H - 1$  **do**  
     $r_h \leftarrow \left\lceil \frac{(Q_H - h)P_{H-1} + (h - Q_{H-1})P_H}{Q_H - Q_{H-1}} - 1 \right\rceil$   
     $w(r_h, h) \leftarrow \frac{b_H r_h + a_H h}{b_H P_H + a_H Q_H}$   
     $s(h) \leftarrow H$   
  **for**  $h \leftarrow Q_k^* + 1$  **to**  $Q_{k+1}$  **do**  
     $r_h \leftarrow \left\lceil \frac{(h - Q_k^*)P_{k+1}}{Q_{k+1} - Q_k^*} - 1 \right\rceil$   
     $w(r_h, h) \leftarrow \frac{b_{k+1} r_h + a_{k+1}(n - 1 - h)}{b_{k+1} P_{k+1} + a_{k+1}(n - 1 - Q_{k+1})}$   
     $s(h) \leftarrow k + 1$   
  **for**  $H \leftarrow k + 2$  **to**  $d$  **do**  
    **for**  $h \leftarrow Q_{H-1} + 1$  **to**  $Q_H$  **do**  
       $r_h \leftarrow \left\lceil \frac{(Q_H - h)P_{H-1} + (h - Q_{H-1})P_H}{Q_H - Q_{H-1}} - 1 \right\rceil$   
       $w(r_h, h) \leftarrow \frac{b_H r_h + a_H(n - 1 - h)}{b_H P_H + a_H(n - 1 - Q_H)}$   
       $s(h) \leftarrow H$   
  **if**  $(0, 1) \in D$  **then**  $s(Q_k^*) \leftarrow 0$

**Step 6:** Ordering the points.

Order the points  $(r_h, h)$  for  $h \leftarrow 0, 1, \dots, Q_k - 1, Q_k^* + 1, Q_k^* + 2, \dots, n - 1$  after increasing values of  $w(r_h, h)$  and call these points in this order  $(p_0, q_0), (p_1, q_1), \dots, (p_{N-1}, q_{N-1})$ .  
**if**  $(0, 1) \in D$  **then**  $(p_{N-1}, q_{N-1}) \leftarrow (0, Q_k^*)$

**Step 7:** Assignment of  $f$ -values.

**for**  $t \leftarrow 1 - \max(P_0, P_d)$  **to**  $m - M - 1$  **do**  
  **for**  $h \leftarrow 0$  **to**  $N - 1$  **do**  
    **if**  $0 \leq p_h + t < m$  **and**  $\delta(p_h + t, q_h) = 0$  **then**  
       $g(p_h + t, q_h) \leftarrow \lambda(\mathbf{d}_{s(q_h)}, p_h + t, q_h)$   
      **for all**  $\mathbf{d} \in D$  **do**  
         $\lambda(\mathbf{d}, p_h + t, q_h) \leftarrow \lambda(\mathbf{d}, p_h + t, q_h) - g(p_h + t, q_h)$   
       $\delta(p_h + t, q_h) \leftarrow 1$

**Step 8:** Start nonvalid case, initial values, cf. Step 1.

$((a_1, b_1), \dots, (a_k, b_k)) \leftarrow ((a_k, b_k), \dots, (a_1, b_1))$

$((a_{k+1}, b_{k+1}), \dots, (a_d, b_d)) \leftarrow ((a_d, b_d), \dots, (a_{k+1}, b_{k+1}))$   
**if**  $(1, 0) \in D$  **then**  $\mathbf{d}_0 \leftarrow (1, 0)$

**Step 9:** Border points, cf. Step 2.

**if**  $M > m$  **then**  $M \leftarrow m$   
 $((P_0, Q_0), \dots, (P_k, Q_k)) \leftarrow ((P_k, Q_k), \dots, (P_0, Q_0))$   
 $((P_k^*, Q_k^*), (P_{k+1}, Q_{k+1}), \dots, (P_d, Q_d)) \leftarrow ((M - P_d - 1, n - Q_d - 1), \dots,$   
 $(M - P_{k+1} - 1, n - Q_{k+1} - 1), (M - P_k^* - 1, n - Q_k^* - 1))$

**Step 10:** Choosing starting points  $r_h$ , weights  $w(r_h, h)$  and directions  $\mathbf{d}_{s(h)}$ ,  
cf. Step 5.

**for**  $H \leftarrow 1$  **to**  $k$  **do**  
**for**  $h \leftarrow P_{H-1}$  **to**  $P_H - 1$  **do**  
 $r_h \leftarrow \left\lceil \frac{(P_H - h)Q_{H-1} + (h - P_{H-1})Q_H}{P_H - P_{H-1}} - 1 \right\rceil$   
 $w(h, r_h) \leftarrow \frac{a_H r_h + b_H h}{a_H Q_H + b_H P_H}$   
 $s(h) \leftarrow H$

**for**  $h \leftarrow P_k^* + 1$  **to**  $P_{k+1}$  **do**  
 $r_h \leftarrow \left\lceil \frac{(h - P_k^*)Q_{k+1}}{P_{k+1} - P_k^*} - 1 \right\rceil$   
 $w(h, r_h) \leftarrow \frac{M - 1 - h}{M - 1 - P_k^*}$   
 $s(h) \leftarrow k + 1$

**for**  $H \leftarrow k + 2$  **to**  $d$  **do**  
**for**  $h \leftarrow P_{H-1} + 1$  **to**  $P_H$  **do**  
 $r_h \leftarrow \left\lceil \frac{(P_H - h)Q_{H-1} + (h - P_{H-1})Q_H}{P_H - P_{H-1}} - 1 \right\rceil$   
 $w(h, r_h) \leftarrow \frac{a_H r_h + b_H(M - 1 - h)}{a_H Q_{H-1} + b_H(M - 1 - P_{H-1})}$   
 $s(h) \leftarrow H$

**if**  $(1, 0) \in D$  **then**  $s(P_k^*) \leftarrow 0$

**Step 11:** Ordering the points, cf. Step 6.

Order the points  $(r_h, h)$  for  $h \leftarrow 0, 1, \dots, P_k - 1, P_k^* + 1, P_k^* + 2, \dots, M - 1$  after increasing values of  $w(r_h, h)$  and call these points in this order  $(p_0, q_0), (p_1, q_1), \dots, (p_{M-1}, q_{M-1})$ .

**if**  $(1, 0) \in D$  **then**  $(p_{M-1}, q_{M-1}) \leftarrow (P_k^*, 0)$

**Step 12:** Assignment of  $f$ -values, cf. Step 7.

$u \leftarrow m - M$

```

for  $t \leftarrow 1 - \max(Q_0, Q_d)$  to  $n - 1$  do
  for  $h \leftarrow 0$  to  $M - 1$  do
    if  $0 \leq p_h + u < m$  and  $0 \leq q_h + t < n$  and  $\delta(p_h + u, q_h + t) = 0$ 
    then
       $g(p_h + u, q_h + t) \leftarrow \lambda(\mathbf{d}_{s(p_h)}, p_h + u, q_h + t)$ 
      for all  $\mathbf{d} \in D$  do
         $\lambda(\mathbf{d}, p_h + u, q_h + t) \leftarrow \lambda(\mathbf{d}, p_h + u, q_h + t) - g(p_h + u, q_h + t)$ 
       $\delta(p_h + u, q_h + t) \leftarrow 1$ 
    return  $g$ 

```

**Remark 12.** We are assuming that the line sums are exact. So, the fact that the output is consistent with the data can be easily checked by observing whether all the line sums are equal to zero (Steps 7 and 12 update the line sums by subtracting the value of each point).

**Example 13.** Let be given  $m = 21$ ,  $n = 16$ ,  $D = \{(0, 1), (1, 0), (1, 1), (-1, 1), (-3, -1), (-1, -3), (5, -1), (7, 5)\}$  and the line sums in the directions of  $D$  of some function  $f : A \rightarrow \mathbb{R}$  (the function itself is irrelevant for the example). The effect of the steps is the following.

**Step 1.** Initial values:  $k = 2$ ,  $d = 6$ ,  $\mathbf{d}_0 = (0, 1)$ ,  $(a_1, b_1) = (5, 1)$ ,  $(a_2, b_2) = (1, 1)$ ,  $(a_3, b_3) = (1, 3)$ ,  $(a_4, b_4) = (1, 1)$ ,  $(a_5, b_5) = (7, 5)$ ,  $(a_6, b_6) = (3, 1)$ .  $M = 19$ ,  $N = 13$ .

**Step 2.** Border points:  $(P_0, Q_0) = (6, 0)$ ,  $(P_1, Q_1) = (1, 1)$ ,  $(P_2, Q_2) = (0, 2)$ ,  $(P_2^*, Q_2^*) = (0, 5)$ ,  $(P_3, Q_3) = (1, 8)$ ,  $(P_4, Q_4) = (2, 9)$ ,  $(P_5, Q_5) = (9, 14)$ ,  $(P_6, Q_6) = (12, 15)$ .

**Step 3** Switching functions:  $\delta(p, q) = 1$ ,  $g(p, q) = 0$  for  $p = 0, 1$  and  $q = 2, 3, 4$ ;  $\delta(p, q) = 0$  for all other points  $(p, q)$  of  $A$ .

**Step 4** False: We are in a valid case, since  $m > M$ .

**Step 5.** Choice of starting points, weights and directions:  $r_0 = 5$ ,  $r_1 = 0$ ,  $r_6 = r_7 = r_8 = 0$ ,  $r_9 = 1$ ,  $r_{10} = 3$ ,  $r_{11} = 4$ ,  $r_{12} = 6$ ,  $r_{13} = 7$ ,  $r_{14} = 8$ ,  $r_{15} = 11$ ;  $w(5, 0) = .833$ ,  $w(0, 1) = .500$ ,  $w(0, 6) = .900$ ,  $w(0, 7) = .800$ ,  $w(0, 8) = .700$ ,  $w(1, 9) = .875$ ,  $w(3, 10) = .962$ ,  $w(4, 11) = .923$ ,  $w(6, 12) = .981$ ,  $w(7, 13) = .942$ ,  $w(8, 14) = .904$ ,  $w(11, 15) = .917$ ;  $s(0) = 1$ ,  $s(1) = 2$ ,  $s(6) = s(7) = s(8) = 3$ ,  $s(9) = 4$ ,  $s(10) = s(11) = s(12) = s(13) = s(14) = 5$ ,  $s(15) = 6$ ,  $s(5) = 0$  (See Figure 6).

**Step 6.** Ordering of the points:  $(p_0, q_0) = (0, 1)$ ,  $(p_1, q_1) = (0, 8)$ ,  $(p_2, q_2) = (0, 7)$ ,  $(p_3, q_3) = (5, 0)$ ,  $(p_4, q_4) = (1, 9)$ ,  $(p_5, q_5) = (0, 6)$ ,  $(p_6, q_6) = (8, 14)$ ,  $(p_7, q_7) = (11, 15)$ ,  $(p_8, q_8) = (4, 11)$ ,  $(p_9, q_9) = (7, 13)$ ,  $(p_{10}, q_{10}) = (3, 10)$ ,  $(p_{11}, q_{11}) = (6, 12)$ ,  $(p_{12}, q_{12}) = (0, 5)$ .

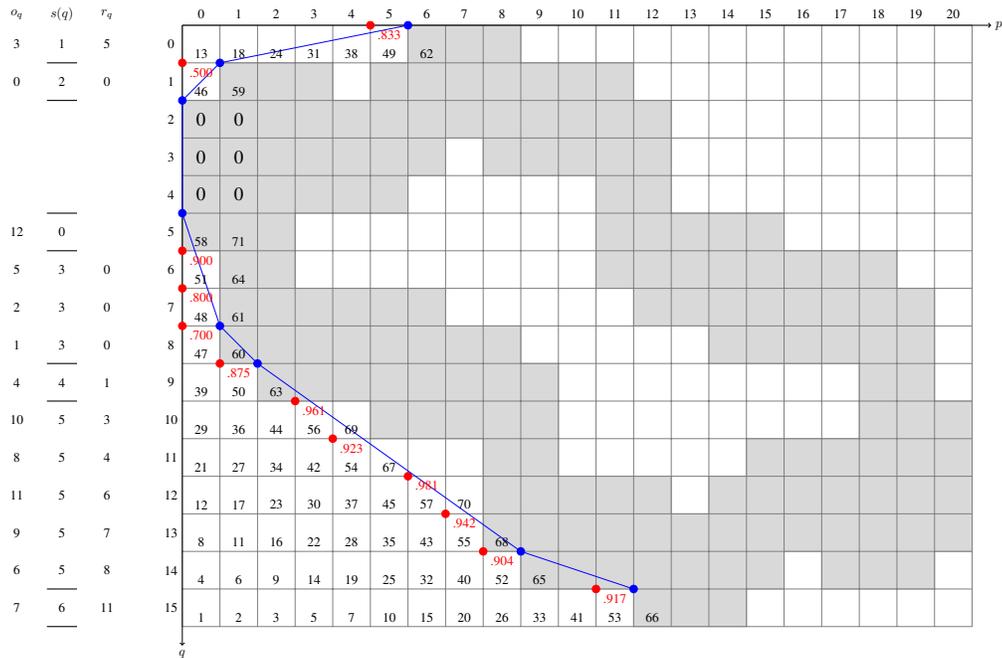


Figure 6: An illustration of Steps 1-7 of Example 13. The light grey pixels indicate the union of the switching domains. The white pixels indicate the pixels of which the  $f$ -values are unique, and therefore equal to the computed  $g$ -value. There are six primitive switching functions and their lexicographic smallest elements have a 0. In Step 3 their  $g$ -values are fixed as 0, but this may be replaced by any other values. Step 1 guarantees the concavity of the upper left corner region  $V_{UL}$  and the lower left corner region  $V_{LL}$ , left of the broken line. The border points for  $V_{UL}$  and  $V_{LL}$  are indicated by the dots along the broken line. They are found in Step 2. Step 4 provides a shortcut in case of a nonvalid case; if  $m \leq M$ , then the coordinates can be switched. For each row the grid point just left of the border line is computed in Step 5. The weights of these points are indicated in the upper numbers inside the pixels. The (highlighted) points immediately left of the broken line are ordered after size as indicated in the column  $o_q$  (see Step 6). The function  $s$  indicates the directions which are used for the grid points in that row. Finally, in Step 7, the  $g$ -values are computed for the first  $m - M$  columns and some more pixels. The order in which they are calculated is given in black.

**Step 7.** Assignment. See Figure 6 for the order in which the  $f$ -values are computed, indicated by the black numbers. After this step the  $f$ -values of the first  $m - M = 2$  columns are known and  $M = 19$  columns are left.

**Step 8.** We are now in a nonvalid case and apply a switch of coordinate axes. The new initial values are:  $\mathbf{d}_0 = (1, 0)$ ,  $(a_1, b_1) = (1, 1)$ ,  $(a_2, b_2) = (5, 1)$ ,  $(a_3, b_3) = (3, 1)$ ,  $(a_4, b_4) = (7, 5)$ ,  $(a_5, b_5) = (1, 1)$ ,  $(a_6, b_6) = (1, 3)$ .

**Step 9.** Border points:  $(P_0, Q_0) = (0, 2)$ ,  $(P_1, Q_1) = (1, 1)$ ,  $(P_2, Q_2) = (6, 0)$ ,  $(P_2^*, Q_2^*) = (6, 0)$ ,  $(P_3, Q_3) = (9, 1)$ ,  $(P_4, Q_4) = (16, 6)$ ,  $(P_5, Q_5) = (17, 7)$ ,  $(P_6, Q_6) = (18, 10)$ .

**Step 10.** Choice of starting points, weights and directions:  $r_0 = 1, r_1 = r_2 = \dots = r_5 = 0, r_7 = r_8 = r_9 = 0, r_{10} = 1, r_{11} = 2, r_{12} = r_{13} = 3, r_{14} = 4, r_{15} = r_{16} = 5, r_{17} = 6, r_{18} = 9$ ;  $w(0, 1) = .500, w(1, 0) = .167, w(2, 0) = .333, w(3, 0) = .500, w(4, 0) = .667, w(5, 0) = .833, w(7, 0) = .917, w(8, 0) = .833, w(9, 0) = .750, w(10, 1) = .904, w(11, 2) = .942, w(12, 3) = .981, w(13, 3) = .885, w(14, 4) = .923, w(15, 6) = .962, w(16, 5) = .865, w(17, 6) = .875, w(18, 9) = .900$ ;  $s(0) = 1, s(1) = s(2) = \dots = s(5) = 2, s(7) = s(8) = s(9) = 3, s(10) = s(11) = \dots = s(16) = 4, s(17) = 5, s(18) = 6, s(6) = 0$ .

**Step 11** Ordering of the points:  $(p_0, q_0) = (1, 0), (p_1, q_1) = (2, 0), (p_2, q_2) = (0, 1), (p_3, q_3) = (3, 0), (p_4, q_4) = (4, 0), (p_5, q_5) = (9, 0), (p_6, q_6) = (5, 0), (p_7, q_7) = (8, 0), (p_8, q_8) = (16, 5), (p_9, q_9) = (17, 6), (p_{10}, q_{10}) = (13, 3), (p_{11}, q_{11}) = (18, 9), (p_{12}, q_{12}) = (10, 1), (p_{13}, q_{13}) = (7, 0), (p_{14}, q_{14}) = (14, 4), (p_{15}, q_{15}) = (11, 2), (p_{16}, q_{16}) = (15, 5), (p_{17}, q_{17}) = (12, 3), (p_{18}, q_{18}) = (6, 0)$ .

**Step 12.** Assignment. See Figure 7 for the order in which the  $g$ -values are computed, indicated by the black numbers. This has to be continued in the obvious way. When this step has been completed all the  $g$ -values are known.

## 7. Complexity

We state the complexity of each step of the algorithm, where we count every addition, subtraction, multiplication, division and determining of the larger of two explicit quantities as one operation.

- Step 1:  $\mathcal{O}(d \log d)$ ;
- Step 2:  $\mathcal{O}(d)$ ;
- Step 3:  $\mathcal{O}(mn)$ ;
- Step 4:  $\mathcal{O}(1)$ ;

$o_h$	2	0	1	3	4	6	18	13	7	5	12	15	17	10	14	16	8	9	11
$s(h)$	1	2	2	2	2	2	0	3	3	3	4	4	4	4	4	4	4	5	6
$r_h$	1	0	0	0	0	0	0	0	0	0	1	2	3	3	4	5	5	6	9

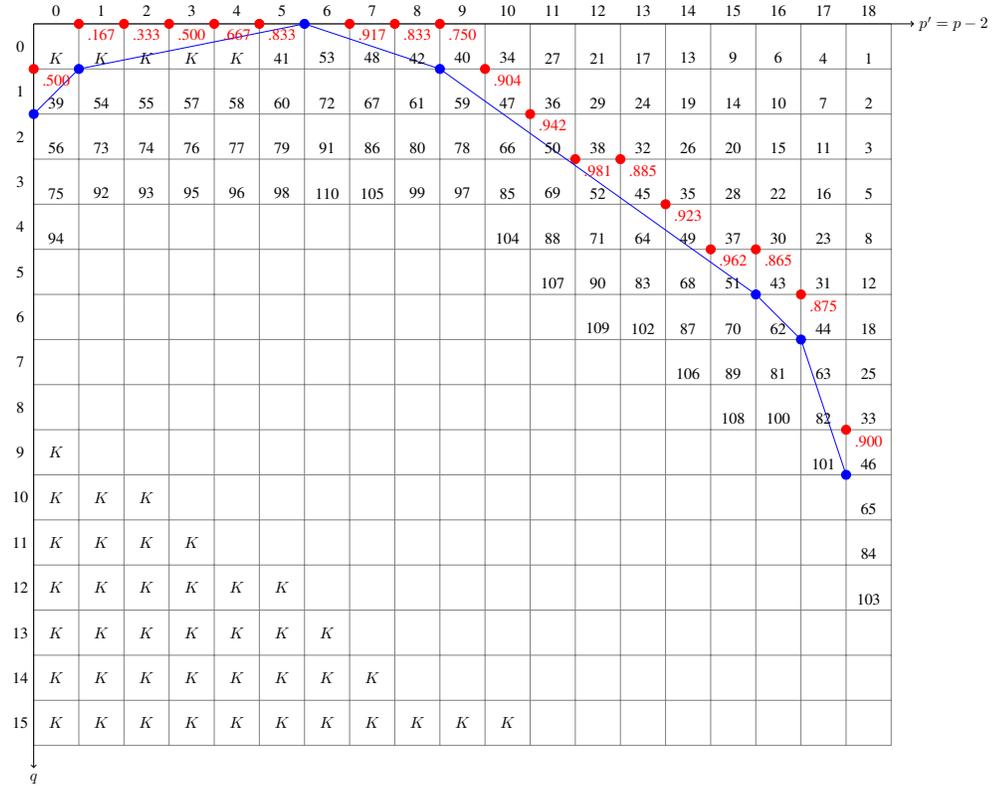


Figure 7: This figure illustrates Steps 8-12 of Example 13. We have omitted the first  $m - M = 2$  columns so that the column numbers indicate  $p - 2$ . The  $K$ 's indicate the pixels of which the  $g$ -values have already been calculated in Step 7. In this stage the roles of rows and columns are interchanged. Step 8 serves to adjust the order of the directions. In Step 9 the border points in  $V_{UL}$  are reordered, those in  $V_{UR}$  are those of  $V_{LL}$  mirrored. Again the broken line connects them. This time the points which determine the direction to be used are the integer points immediately *above* the border points. Their  $Q$ -values, their weights (the above number inside the pixel) and the sequence  $s$  are computed in Step 10. They are ordered in Step 11 and the order is indicated in row  $o_h$ . Finally in Step 12 the remaining  $g$ -values are computed where by using  $u$  the original  $p$ -values are used instead of  $p - 2$ . The order of the way the  $g$ -values are found is indicated by the lower numbers inside the pixels. This has to be completed downwards to find all  $g$ -values.

- Steps 5 and 10:  $\mathcal{O}(n)$  and  $\mathcal{O}(m)$ , respectively;
- Steps 6 and 11:  $\mathcal{O}(n \log n)$  and  $\mathcal{O}(m \log m)$ , respectively;
- Steps 7+12:  $\mathcal{O}(dmn)$ .

Without loss of generality we may assume  $m \leq n$ . It follows that the complexity of the algorithm is  $\mathcal{O}(dmn)$  unless  $mn = \mathcal{O}(\log d)$  or  $dm = \mathcal{O}(\log n)$  or  $dn = \mathcal{O}(\log m)$ . The latter case can not occur since  $m \leq n$ . If  $m = \mathcal{O}(\log d)$ , then we recall that  $a_h \geq 1$  with at most one exception. So we can delete  $d - m - 1$  directions and still have a nonvalid case. After deletion we have, denoting by  $d$  the new number of directions,  $m = d - 1$  and complexity  $\mathcal{O}(dmn)$ .

It remains to consider  $dm = \mathcal{O}(\log n)$ . In this case only the complexity of Step 6 has to be adjusted. This can be achieved by remembering in Step 5 for every  $H$  the location of the point  $(r_h, h)$  with  $s(h) = H$  and minimal weight for direction  $(a_H, b_H)$ . This has complexity  $\mathcal{O}(dn)$ . Now Step 6 proceeds as follows. For each direction  $(a_H, b_H)$  let  $(r_h, h)$  be the point with  $s(h) = H$  and minimal weight. For  $H \leq k$  we start with the vectors

$$(r_h, h, b_H(P_H - 1) + a_H Q_H, b_H P_H + a_H Q_H, R_H, S_H, T_H, U_H),$$

where  $(R_H, S_H) = (Q_{H-1}, Q_H - 1)$  and  $(T_H, U_H)$  is the unique pair satisfying  $0 \leq T_H < a_H$  and  $b_H T_H + a_H U_H = 1$ . For  $k < H \leq d$  we start with the vectors

$$(r_h, h, b_H(P_H - 1) + a_H(n - 1 - Q_H), b_H P_H + a_H(n - 1 - Q_H), R_H, S_H, T_H, U_H),$$

where  $(R_H, S_H) = (Q_{H-1} + 1, Q_H)$  and  $(T_H, U_H)$  is the unique pair satisfying  $0 \leq T_H < a_H$  and  $b_H T_H - a_H U_H = 1$ . Observe that in each case the quotient of the third and fourth entry is the weight. We order such vectors on the top line according to increasing weight. At each step we increase by one the third entry of the first (leftmost) vector. If the third entry now is still smaller than the fourth entry, we replace the first two entries  $(r_h, h)$  by  $(r_h + T_H, h + U_H)$  or  $(r_h + T_H - b_H, h + U_H - a_H)$  such that the second entry is in  $[R_H, S_H]$ . If the third entry becomes equal to the fourth one, we neglect the vector in the sequel. In any case we order the remaining vectors on the line again after increasing weight. This procedure runs until there is no vector left. At every step the two leftmost entries of the leftmost vector give the next value  $(p_h, q_h)$ . The computation of the vectors  $(T_H, U_H)$  has complexity  $\mathcal{O}(d \log n)$ , the computation of each row  $\mathcal{O}(d \log d)$  and there are  $n$  rows. Therefore the total complexity is  $\mathcal{O}(nd \log d)$ . This is  $\mathcal{O}(dmn)$ ,

unless  $m = \mathcal{O}(\log d)$ . We have already remarked that in this case we can delete  $d - m - 1$  directions, still have a nonvalid case, and have complexity  $\mathcal{O}(dmn)$ .

**Example 14.** [Continuation of Example 13]. The described procedure yields as the first row the vectors

$$\begin{aligned} &(0, 1, 1, 2, 1, 1, 0, 1), \quad (0, 8, 7, 10, 6, 8, 0, -1), \quad (5, 0, 5, 6, 0, 0, 1, 0), \\ &(1, 9, 7, 8, 9, 9, 1, 0), \quad (8, 14, 47, 52, 10, 14, 3, 2), \quad (11, 15, 11, 12, 15, 15, 1, 0). \end{aligned}$$

Since the last four entries do not change, we do not mention them in the table. Then the table becomes as follows (each row represents one step in the procedure).

$$\begin{array}{cccccc} (0,1,1,2) & (0,8,7,10) & (5,0,5,6) & (1,9,7,8) & (8,14,47,52) & (11,15,11,12) \\ (0,8,7,10) & (5,0,5,6) & (1,9,7,8) & (8,14,47,52) & (11,15,11,12) & \\ (0,7,8,10) & (5,0,5,6) & (1,9,7,8) & (8,14,47,52) & (11,15,11,12) & \\ (5,0,5,6) & (1,9,7,8) & (0,6,9,10) & (8,14,47,52) & (11,15,11,12) & \\ (1,9,7,8) & (0,6,9,10) & (8,14,47,52) & (11,15,11,12) & & \\ (0,6,9,10) & (8,14,47,52) & (11,15,11,12) & & & \\ (8,14,47,52) & (11,15,11,12) & & & & \\ (11,15,11,12) & (4,11,48,52) & & & & \\ (4,11,48,52) & & & & & \\ (7,13,49,52) & & & & & \\ (3,10,50,52) & & & & & \\ (6,12,51,52) & & & & & \end{array}$$

The sequence  $(p_0, q_0) = (0, 1)$ ,  $(p_1, q_1) = (0, 8)$ ,  $(p_2, q_2) = (0, 7)$ ,  $\dots$ ,  $(p_{11}, q_{11}) = (6, 12)$  can be read from the leftmost two entries. At the end  $(p_{12}, q_{12}) = (0, 5)$  has to be added.

## 8. Conclusions

In this paper we have addressed the tomographic reconstruction problem for functions with values in a unique factorization domain or field, such as integers and reals. A key argument is that one may ask for the point values even when many solutions are admissible, since the values of points outside the switching domains are common to all functions satisfying the problem.

Starting from the characterization of the switching functions in [17] and the results in [21], we have shown that all points with uniquely determined value, namely, not belonging to switching domains, can be recovered once we give an arbitrary value to  $(m - M)(n - N)$  points, where  $(m - M)(n - N)$  is the number of linearly independent switching functions. We have provided an algorithm which computes the point values systematically and runs in

time linear in  $dmn$ , where  $d$  is the number of directions. By the result in [17] our algorithm provides the complete set of solutions with values in the unique factorization domain or in the field.

The proposed approach works when line sums are supposed to be exact and therefore not all projections are necessary to recover a solution. It underlines the structural difference between unique factorization domains and other kinds of sets, such as  $\{0, 1\}$  (leading to binary images), since in the latter case the reconstruction problem has proven to be NP-hard [14, 15].

Two questions arise by this paper. Firstly, does there exist a similar algorithm for higher dimensions? Secondly, given a system of inconsistent line sums, is there a fast way to find a best approximation of consistent line sums so that the algorithm of this paper can be applied to construct the most likely set of solutions (over  $\mathbb{Z}$  or  $\mathbb{R}$ )?

## References

### References

- [1] K.A. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons, Inc., New York, 2nd edition, 1989.
- [2] K.J. Batenburg and L. Plantagie. Fast approximation of algebraic reconstruction methods for tomography. *IEEE Trans. Image Process.*, 21(8):3648–3658, 2012.
- [3] K.J. Batenburg and J. Sijbers. DART: a practical reconstruction algorithm for discrete tomography. *IEEE Trans. Image Process.*, 20(9):2542–2553, 2011.
- [4] S. Brunetti, P. Dulio, and C. Peri. Discrete tomography determination of bounded lattice sets from four X-rays. *Discrete Appl. Math.*, 161(15):2281–2292, 2013.
- [5] M. Ceko, T. Petersen, I. Svalbe, and R. Tijdeman. Boundary ghosts for discrete tomography. *J. Math. Imaging Vision*, pages 1–13, 2021.
- [6] B. van Dalen, L. Hajdu, and R. Tijdeman. Bounds for discrete tomography solutions. *Indag. Math.*, 24(2):391–402, 2013.
- [7] P. Dulio, A. Frosini, and S.M.C. Pagani. Uniqueness regions under sets of generic projections in discrete tomography. In E. Barucci, A. Frosini,

- and S. Rinaldi, editors, *Discrete geometry for computer imagery*, volume 8668 of *Lecture Notes in Comput. Sci.*, pages 285–296. Springer, 2014.
- [8] P. Dulio, A. Frosini, and S.M.C. Pagani. A geometrical characterization of regions of uniqueness and applications to discrete tomography. *Inverse Problems*, 31(12):125011, 2015.
- [9] P. Dulio, A. Frosini, and S.M.C. Pagani. Geometrical characterization of the uniqueness regions under special sets of three directions in discrete tomography. In N. Normand, J. Guédon, and F. Autrusseau, editors, *Discrete Geometry for Computer Imagery*, volume 9647 of *Lecture Notes in Comput. Sci.*, pages 105–116. Springer, Cham, 2016.
- [10] P. Dulio, A. Frosini, and S.M.C. Pagani. Regions of uniqueness quickly reconstructed by three directions in discrete tomography. *Fund. Inform.*, 155(4):407–423, 2017.
- [11] P. Dulio and S.M.C. Pagani. A rounding theorem for unique binary tomographic reconstruction. *Discrete Appl. Math.*, 268:54–69, 2019.
- [12] P.C. Fishburn, J.C. Lagarias, J.A. Reeds, and L.A. Shepp. Sets uniquely determined by projections on axes. II. Discrete case. *Discrete Math.*, 91(2):149–159, 1991.
- [13] R.J. Gardner and P. Gritzmann. Discrete tomography: determination of finite sets by X-rays. *Trans. Amer. Math. Soc.*, 349(6):2271–2295, 1997.
- [14] R.J. Gardner, P. Gritzmann, and D. Prangenberg. On the computational complexity of reconstructing lattice sets from their X-rays. *Discrete Math.*, 202(1-3):45–71, 1999.
- [15] R.J. Gardner, P. Gritzmann, and D. Prangenberg. On the computational complexity of determining polyatomic structures by X-rays. *Theoret. Comput. Sci.*, 233(1-2):91–106, 2000.
- [16] L. Hajdu. Unique reconstruction of bounded sets in discrete tomography. In *Proceedings of the Workshop on Discrete Tomography and its Applications*, volume 20 of *Electron. Notes Discrete Math.*, pages 15–25. Elsevier, Amsterdam, 2005.
- [17] L. Hajdu and R. Tijdeman. Algebraic aspects of discrete tomography. *J. Reine Angew. Math.*, 534:119–128, 2001.

- [18] G.T. Herman. *Fundamentals of computerized tomography: image reconstruction from projections*. Springer, 2nd edition, 2009.
- [19] M.B. Katz. *Questions of uniqueness and resolution in reconstruction from projections*. Lecture Notes in Biomath. Springer-Verlag, 1978.
- [20] F. Natterer. *The mathematics of computerized tomography*. SIAM, 2001.
- [21] S.M.C. Pagani and R. Tijdeman. Algorithms for linear time reconstruction by discrete tomography. *Discrete Appl. Math.*, 271:152 – 170, 2019.
- [22] D.M. Pelt and K.J. Batenburg. Fast tomographic reconstruction from limited data using artificial neural networks. *IEEE Trans. Image Process.*, 22(12):5238–5251, 2013.
- [23] J. Radon. Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten. *Ber. Verh. Sächs. Akad. Wiss. Leipzig Math.-Phys. Kl.*, (69):262–277, 1917.
- [24] H.J. Ryser. Combinatorial properties of matrices of zeros and ones. *Canad. J. Math.*, 9:371–377, 1957.