Reconstruction, Enumeration, and Examination of Binary Images with Prior Information

Doctoral Thesis

by

Norbert Hantos

Supervisor

Péter Balázs

PhD School in Computer Science

Department of Image Processing and Computer Graphics

University of Szeged, Szeged, Hungary

2015

Acknowledgement

First of all, I would like to address my acknowledgement to my supervisor, Péter Balázs, who gave me all the help, advice and guidance I needed for my thesis. Also, I would like to express my gratitude to my friends Erika Griechisch and Gábor Németh for proofreading the manuscript of my thesis and giving additional suggestions and helpful critics, among with the list of typos and minor mistakes. In addition, I would like to thank László Varga for providing me the basic LaTeX files. Last but not least, I am thankful to my parents and my brother for for their encouragement, and my fiancée Zsófia Horváth for her love and support.

The research was carried out at the Department of Image Processing and Computer Graphics of the University of Szeged. The work was partially supported by the TÁMOP-4.2.2/08/1/2008-0008 and TÁMOP-4.2.1/B-09/1/KONV-2010-0005 programs of the Hungarian National Development Agency and by the OTKA grant K112998 "Discrete Tomography from Incomplete and Uncertain Data: Modelling, Algorithms, and Applications".

Norbert Hantos, September 2015.

Notation and Abbreviation

A, B, …, F, G, …	Binary matrices with a_{ij}, b_{ij}, …, f_{ij}, g_{ij}, … elements
p, q, …	Points of $\mathbb{Z}^2$ with coordinates (i_p, j_p), (i_q, j_q), …
$N_4(p)$	The set of 4-adjacent points to p
$\|A\|$	The number of 1-s in the binary matrix A
$w(A)$	The width (the number of columns) of the binary matrix A
$A \subseteq B$	The binary matrix A is a subset of the binary matrix B with equal size, i.e., $\forall i \forall j (a_{ij} = 1 \rightarrow b_{ij} = 1)$
$\mathcal{H}(A) = (h_1, \ldots, h_m)$	The horizontal projection of A, containing the number of 1-s in each row of A
$\mathcal{V}(A) = (v_1, \ldots, v_n)$	The vertical projection of A, containing the number of 1-s in each column of A
$\oplus$; $\ominus$; $\oplus_k$	Morphological dilation; morphological erosion; iterative morphological dilation with iteration number k
$\mathcal{S}(F, Y)$; $\mathcal{S}_k(F, Y)$	The morphological skeleton of the binary image F with the structuring element Y; the k-th skeletal subset
κ_p	The skeletal label of point p
$\|\|.\|\|_2$	Euclidean norm of a vector
$\lfloor . \rfloor$	The floor function
Mfa	Minimum Flip Augmentation Problem
SA	Simulated Annealing

List of Figures

List of Tables

Contents

Introduction

The main task of tomography is to reconstruct images representing two-dimensional cross-sections of three-dimensional objects from their projections. Undoubtedly, the main applications of tomography arise from the field of medicine, but it is a very useful imaging tool also in physics, chemistry, biology, and industry. An important subfield is binary tomography [48], which aims to reconstruct binary images. In the most common applications of this field, e.g., electron tomography [1, 12] and non-destructive testing [13], usually just few projections of the object can be measured, since the acquisition of the projection data can be expensive or damage the object. Moreover, the physical limitations of the imaging devices make it sometimes impossible to take projections from numerous angles. Owing to the small number of projections the binary reconstruction can be extremely ambiguous.

The presence of certain binary patterns in the image can violate the unique reconstruction of the image, especially from small number of projections. Therefore, analysis of binary patterns plays a vital role in binary tomography. If uniqueness is guaranteed, then the binary image can be stored in a (lossless) compressed form by its projections. Nevertheless, even if the image contains specific patterns causing the projections not being able to determine the image in a unique way, there is still a chance to reconstruct the original image uniquely, if properly chosen elements of that are stored as well [25].

Besides, a common way to reduce the number of solutions of the reconstruction task is to assume that the image to be reconstructed satisfies certain geometrical properties. In many applications of binary tomography the image itself is naturally unknown in advance. However, when one thinks of encoding binary images by their projections for data security or image compression reasons, it is clear that the original image is available. Then, in the decoding phase it becomes important to know whether the image, possibly with some additional prior information, can be uniquely revealed from the projection data [51].

Assuming different prior information about the original image, several theoretical results are known, regarding the efficient reconstruction of binary images and the number of solutions, using just the horizontal and vertical projections. In [9, 21], the authors use certain structured images for lossless data compression, which again shows the usefulness of binary reconstruction in the field of image encoding. The reconstruction complexity and the number of solutions are well-studied in numerous classes of images when two projections are available [10, 27, 30]. Furthermore, in the reconstruction process the prior knowledge is often incorporated into an energy function, thus the reconstruction task becomes equivalent to a function minimization problem. There are various methods to solve that kind of problems [35, 63, 66].

This thesis is a summary of the Author's research in the field of binary tomography. The central focus of this work is to examine additional prior information for the reconstruction task from at most two projections, expand the theoretical background of complexity, determine the number of solutions for certain classes of binary images, and develop new algorithms for binary tomography.

The structure of the dissertation is the following. First, Chapter 1 gives the necessary preliminaries. This chapter does not contain any new contribution (with the exception of Lemma 1.4.1). We summarize previous results of the field, and provide the mathematical backgrounds for understanding the results of the thesis. Then, Chapters 2, 3, and 4 give a detailed description of our results.

Chapter 2 deals with binary patterns which can ensure uniqueness of the reconstruction. In binary images, the reconstrucion can be unique if these patterns are fixed in a preprocessing step. However, finding the minimal number of patterns to fix is generally hard, thus the existence of a deterministic, polynomial-time algorithm for finding the global optimum in general case is questionable. In this chapter, we show how to reduce the searching space drastically, without losing the global optimum, and give deterministic, polyominal-time heuristics based on our theoretical results. We compare those algorithms to another well-known methods in the literature, on a wide set of random binary matrices, and also a real-life dataset of presence-absence matrices. We conclude that our algorithms perform better than the previous ones, both in the number of pattern alteration and running time, especially on sparse matrices.

The reconstruction of specific binary structures from few projections is an extensively studied problem in discrete tomography. Several algorithms exist to solve this task from two projections. For testing the efficiency of those (and of more general reconstruction) algorithms in the average case, enumeration and random generation of these images according to several parameters is an important issue. In Chapter 3, we study the reconstrucion only from the horizontal projection, and provide fast algorithms to reconstruct special binary images with a prescribed horizontal projection. With certain modifications we also can generate such images from a uniform distribution. Furthermore, formulas for the number of solutions with minimal and with any given number of columns are provided as well.

In Chapter 4 we investigate a new kind of prior information to aim the reconstruction, the so-called morphological skeleton – a region-based shape descriptor – which represents the general form of binary objects [38]. We prove that the reconstrucion is generally still hard in the term of complexity, however, a rough reconstruction is always possible in a short time and a small number of iterations. We propose some variants of a method based on a stochastic function minimizer algorithm to reconstruct binary images from their horizontal and vertical projections and the given prior information. With additional restrictions the result will be smoother, however, the convergency of the method becomes slower. A related issue is the uniqueness of the reconstruction. Regardless of the complexity of the reconstruction, the possible number of the solutions can be exponential in the size of the image [30, 65]. We study the uniqueness of the reconstruction of certain type of images, using two projections and the shape descriptor prior.

Finally, in Chapter 5 we give a conclusion of the thesis. For each chapter we summarize our results, and discuss possible further extensions.

Chapter 1

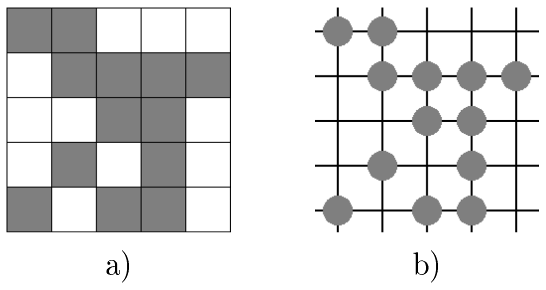
Preliminaries

1.1 The Binary Reconstruction Problem

In *binary tomography* the task is to reconstruct a two-dimensional binary image from a set of projections. The image can be represented by a binary matrix[1] $A = (a_{ij})$, or can be defined as a finite subset of $\mathbb{Z}^2$ (definition is up to translation), where the size of the image is defined by the size of its minimal bounding discrete rectangle. If a point $p = (i_p, j_p)$ of $\mathbb{Z}^2$ is in the given subset, then it is called an *object point*, and indicated by $a_{ij} = 1$ in the binary matrix representation. Otherwise, it is called a *background point*, and then $a_{ij} = 0$ in the corresponding position of the binary matrix. Figure 1.1 shows three different representations of the same binary image.

Generally, in the continuous setting the binary reconstruction task is to recover an unknown binary function $f(x, y) : \mathbb{R}^2 \to \{0, 1\}$ (the image) from a bunch of its

[1]In the literature, a binary matrix is often denoted by A, B, etc., while a binary image is usually denoted by F, G, etc. In this thesis those definitons are equivalent, and we use both notations.



$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

c)

Figure 1.1: Different representations of the same binary image: (a) a set of pixels using two colors; (b) a representation of a subset of $\mathbb{Z}^2$; (c) a binary matrix.

line integrals, the so-called projections given by the Radon-transform

$$[Rf](s,\theta) = \int_{-\infty}^{\infty} f(s\cos\theta - u\sin\theta, s\sin\theta + u\cos\theta)du \tag{1.1}$$

for certain fixed θ angles. Here s and u denote the variables of the coordinate system rotated by the angle θ and $[Rf](s,\theta)$ is called the θ-angle *parallel projection* of f (see Fig. 1.2 for example). In some applications of this field usually just few projections of the object can be measured, since the acquisition of the projection data can be expensive or damage the object. Moreover, the physical limitations of the imaging devices make it sometimes impossible to take projections from numerous angles. In binary tomography we assume the image to be reconstructed is homogeneous, and for the reconstruction usually only a few number of projections are given. In case of only two projections, the *horizontal* and *vertical projection* of a binary image is defined as the vector of the row and column sums, respectively, of the image matrix. Formally, given a binary image A of size of $m \times n$, the horizontal and vertical projection is defined by the vector $\mathcal{H}(A) = (h_1, \ldots, h_m)$ and $\mathcal{V}(A) = (v_1, \ldots, v_n)$, respectively, where

$$h_i = \sum_{j=1}^{n} a_{ij}, \quad i = 1, \ldots, m \ , \tag{1.2}$$

and

$$v_j = \sum_{i=1}^{m} a_{ij}, \quad j = 1, \ldots, n \ . \tag{1.3}$$

The task is to reconstruct the binary image A from its horizontal and vertical projections. The basic question about the general reconstruction problem is the following:

Problem. Binary Reconstruction from Two Projections
Instance. H and V finite vectors of non-negative integers, possible additional prior information.
Question. Is there a binary image A such that $\mathcal{H}(A) = H$, $\mathcal{V}(A) = V$, and A matches with the given prior information?

However, one can consider the following problems related to binary reconstruction as well.

- How many solutions are there?
- Is there a way to give the n-th solution with an acceptable amount of computation (hence, without listing the first $n-1$ solution)?
- How hard is to find a single solution? Is there a polynomial-time algorithm

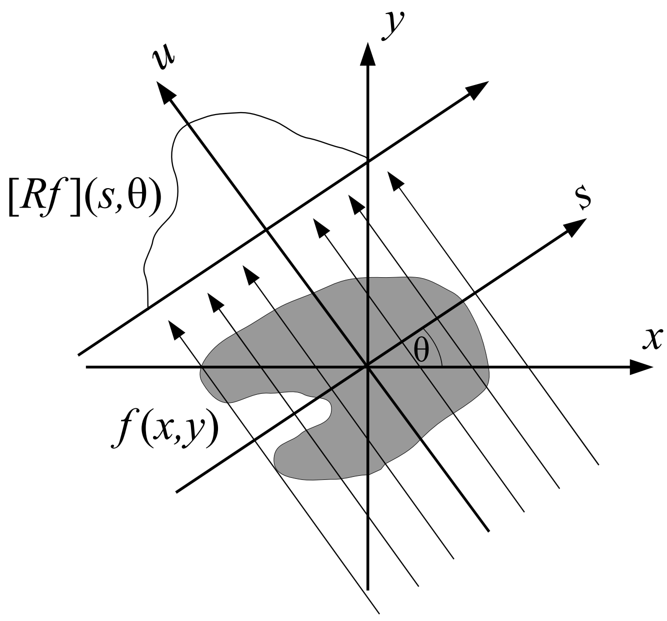


Figure 1.2: The continuous setting of the binary reconstruction task.

for the given task?

- What kind of heruristics can speed up the search for a solution?
- What is considered as a solution? Can it be an image satisfying just approximately the projection vectors, and/or the additional prior information?

Similar questions can be asked in case of more than two projections. In this thesis we only focus on the above questions if only two projections are known, at most. Generally, if only the horizontal and vertical projections are given, the number of solutions can be exponential [27]. A common way to reduce the number of solutions of the reconstruction task is to assume that the image to be reconstructed satisfies certain geometrical properties (e.g., convexity and/or connectedness, as in [23]).

The first method to answer the question of the binary reconstruction problem from two projections was published in [65]. In the same work it was also showed that the solution is not always uniquely determined. Furthermore, in practical applications noisy projection data also complicates the reconstruction. To overcome those problems one can transform the original task to a function minimization problem (assuming the size of the image is $m \times n$)

$$f(\mathbf{x}) = ||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + \alpha \cdot g(\mathbf{x}) \quad \to \quad \min \ , \tag{1.4}$$

where $\mathbf{x}$ is an $(mn) \times 1$ binary vector representing the unknown image in a vector form using row-by-row traversal; $\mathbf{b} = \big(\mathcal{H}(F), \mathcal{V}(F)\big)^T$ is an $(m+n) \times 1$ vector con-

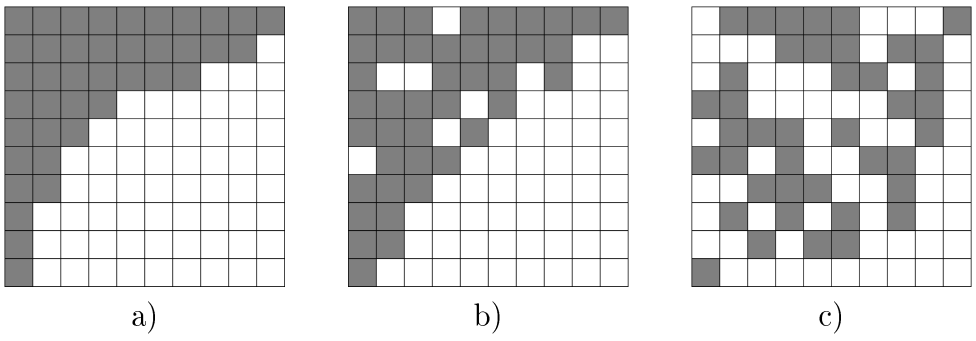


Figure 1.3: Three matrices with rearranged columns and rows to form a triangluar (nested) shape as close as possible. a) a fully nested (switching component free) matrix; b) a nearly fully nested matrix with low number of flips; c) a uniform random matrix.

taining the projections, and $\mathbf{A}$ is an $(m+n)\times(mn)$ binary matrix, with $a_{ij}=1$ if and only if the pixel x_i is hit by the j-th projection ray, 0 otherwise. The function $g(\mathbf{x})$ handles additional information of the image, such as shape, connectivity, perimeter, etc. (as in [23]). The lower value it takes the closer the reconstructed image is to the expected one. $g(\mathbf{x})$ is multiplied by the weighting parameter $\alpha > 0$. In this thesis, especially in Chapter 4 we use the function $g(\mathbf{x})$ to add prior information about the image to the minimization task.

1.2 The Minimum Flip Augmentation Problem

In the field of biogeography and ecology, nestedness is an important measurment of presence-absence binary matrices, where rows can represent species, while columns can represent certain locations. The nestedness of the matrix describes the connection and dependency between species in different location. Similarly, data mining and intelligent data analysis can use nestedness – or nestedness-like measurements – to show the connection between the data represented by rows and columns of a binary matrix [58]. A binary matrix is called *fully nested*, if and only if its rows form a chain of subsets; that is, any two rows are ordered by the subset relation, where we view each row as a subset of the columns indicated by the 1-entries. If a matrix is fully nested, it is possible to permute the rows and columns in a way that the resulted binary image forms a triangular shape, as Fig. 1.3a shows.

Nestedness is strongly connected to the concept of switching components. A *switching component* is a 2×2 submatrix of a given binary matrix such that the diagonal of the submatrix contains 1-s, and the anti-diagonal contains 0-s, or vice versa. Formally, the indices i_1, j_1, i_2, j_2 form a switching component in the binary matrix $A=(a_{ij})$, if either $a_{i_1j_1}=a_{i_2j_2}=1$ and $a_{i_1j_2}=a_{i_2j_1}=0$, or $a_{i_1j_1}=a_{i_2j_2}=0$ and $a_{i_1j_2}=a_{i_2j_1}=1$. A binary matrix is switching component free, if it does not

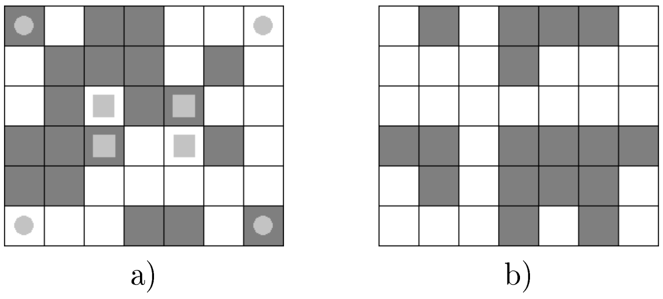


Figure 1.4: Examples of switching components and a switching component free matrix: a) a binary image with two examples of switching components marked with gray squares and disks; b) a switching component free binary image.

contain any switching components. For examples, see Fig. 1.4.

A matrix is fully nested if and only if it is switching component free [58]. If the matrix contains switching components, no permutation on the rows and columns could lead to a triangular shaped image. See Fig. 1.3b and Fig. 1.3c for examples.

Let us define *0-1 flips* as an operation that changes a 0 element of a given binary matrix to 1. The Minimum Flip Augmentation Problem (Mfa) plays an important role in determining the nestedness of a binary matrix.

Problem. Minimum Flip Augmentation (Mfa)
Instance. A binary matrix A.
Question. Constructing a switching component free binary matrix A^* from A using 0-1 flips, what is the minimum number of flips?

Unfortunately, as it was proven in [58], determining the minimal number of 0-1 flips, thus to achieve uniqueness of the reconstruction from two projections is generally an NP-hard problem. The goal of Chapter 2 is to give heuristics for this problem.

1.3 Chang's Algorithm

Chang's algorithm [25] is a polyominal-time binary image reconstruction algorithm for unique matrices. A binary matrix is *unique* if and only if there is exactly one solution for the reconstruction task that satisfies the given horizontal and vertical projections. The algorithm requires the absence of switching components in the matrix, which is the necessary and sufficient condition of non-uniqueness [65]. However, the algorithm can deal with a set of so-called forbidden positions, i.e., positions which has a fixed value in the matrix. Algorithm 1 shows the pseudo-code of Chang's Algorithm.

In Chapter 2 we show how to compress (not necessarily unique) binary images

Algorithm 1 Chang's Algorithm

```
Require: Vector of column and rows sums $C$ and $R$, respectively; set of forbidden
   positions $Q$
Ensure: Binary matrix $A$ satisfying $C$, $R$, and $Q$
  Let $A$ be a matrix full of free elements
  Fix every position of $Q$ in $A$; update $C$ and $R$ by substracting the fixed 1-s
  while there are free elements in $A$ do
    Find a row (column) where all the free elements must be either 0 or 1 to fulfil
    the corresponding row sum (column sum)
    if there is no such a row (column) then
      The solution does not exist or is non-unique
      return $\emptyset$
    else
      Fix the free elements in the found row or column to 0-s (or 1-s) in $A$, and
      update $C$ and $R$
    end if
  end while
  return $A$
```

in a lossless way using Chang's algorithm as the decoding algorithm.

1.4 The Morphological Skeleton

The skeleton is a region-based shape descriptor which represents the general form of binary objects [17]. One way of defining the skeleton of a 2-dimensional continuous object is as the set of the centers of all maximal inscribed (open) disks [36]. A disk is maximal inscribed if it is included in an object, but it is not contained by any other inscribed disk. Discrete skeletons are analogues of the continuous skeleton, but they can be calculated only approximately. There are three frequently used techniques for extracting skeletons from discrete binary images: thinning, Voronoi-based, and distance-based methods [69].

Special skeletons of discrete binary images can be expressed via morphological operations [38], where the continuous disks are approximated by successive dilations of the selected structuring element that is to represent the unit disk. We know that those dilated structuring elements cannot provide "good" approximations of Euclidean disks [67]. Consequently, the heuristic morphological skeletons do not yield "reasonable" skeletons. Despite of this drawback, the original binary image can be exactly reconstructed from the skeletal subsets of arbitrary morphological skeletons.

The *morphological dilation* of a binary image F with the structuring element

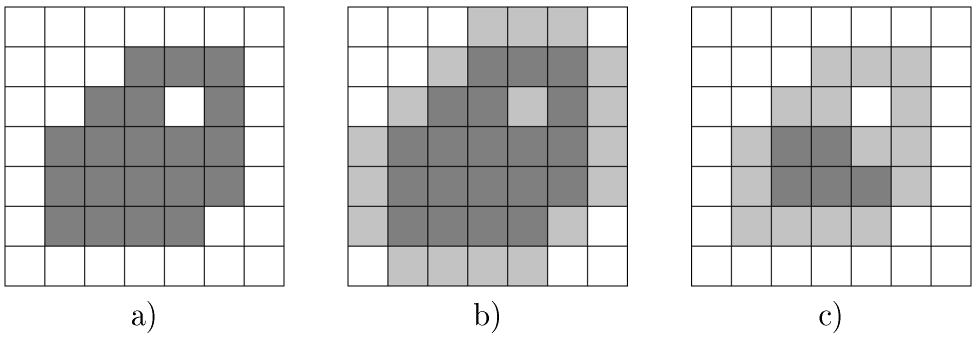


Figure 1.5: An example of the basic morphological operations: (a) original image; (b) the result of morphological dilation; (c) the result of morphological erosion. The structuring element is the origin and its 4-neighbors. Dark gray pixels indicate unaltered object points, light gray pixels indicate altered points.

$Y \subset \mathbb{Z}^2$ is defined by

$$F \oplus Y = \Big\{ p \in \mathbb{Z}^2 \mid \big(F \cap (\hat{Y})_p\big) \subseteq F \Big\}, \tag{1.5}$$

where $\hat{Y}$ denotes the reflection of Y through the origin and $(X)_p$ denotes X translated to the point p.

The definition of the *morphological erosion* is analogous,

$$F \ominus Y = \big\{ p \in \mathbb{Z}^2 \mid (Y)_p \subseteq F \big\}. \tag{1.6}$$

Figure 1.5 shows an example of morphological dilation and erosion.

The iterative morphological dilation is a morphological dilation applied k times ($k \in \mathbb{N}_0$), i.e.,

$$F \oplus_k Y = \begin{cases} F & \text{if } k = 0, \\ (F \oplus_{k-1} Y) \oplus Y & \text{if } k > 0. \end{cases} \tag{1.7}$$

The definition of the iterative morpological erosion $F \ominus_k Y$ is similar to (1.7). The *morphological skeleton* [38, 67] $\mathcal{S}(F, Y)$ of a binary image F determined by a structuring element $Y \subset \mathbb{Z}^2$ is defined by

$$\mathcal{S}(F, Y) = \bigcup_{k=0}^{K} \mathcal{S}_k(F, Y), \tag{1.8}$$

where

$$\mathcal{S}_k(F, Y) = (F \ominus_k Y) \setminus \big[(F \ominus_{k+1} Y) \oplus Y\big] , \tag{1.9}$$

and K is the radius of the largest inscribed disk. In other words,

$$K = \max\{\ k \mid F \ominus_k Y \neq \emptyset\ \} . \tag{1.10}$$

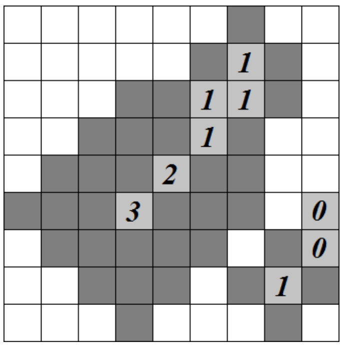


Figure 1.6: An example of the morphological skeleton $\mathcal{S}(F, Y)$ of a binary image F (dark and light gray pixels). Light gray pixels indicate the skeletal points with their corresponding labels.

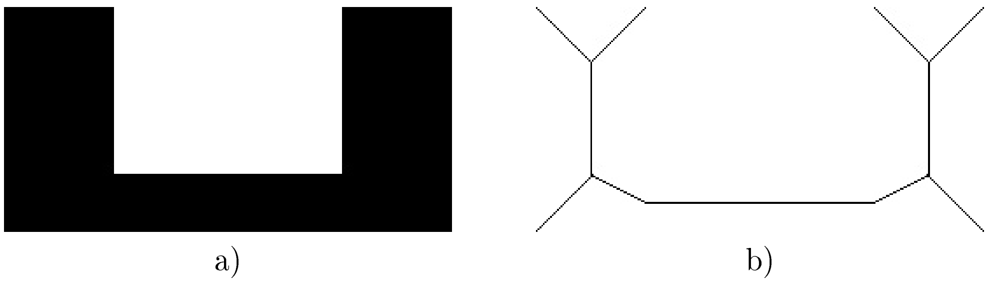


Figure 1.7: An example of the morphological skeleton: (a) original image F; (b) the morphological skeleton $\mathcal{S}(F, Y)$, which represents the shape of the original image.

A point $p \in F$ is called a *skeletal point* if $p \in \mathcal{S}(F, Y)$ for a fixed structuring element Y.

An important property of the morphological skeleton is that the image F can be exactly reconstructed from the skeletal subsets and the structuring element:

$$F = \bigcup_{k=0}^{K} \left[\mathcal{S}_k(F, Y) \oplus_k Y\right] = \bigcup_{p \in \mathcal{S}(F,Y)} \left(p \oplus_{\kappa_p} Y\right), \tag{1.11}$$

where κ_p denotes the *skeletal label* of p such that $p \in \mathcal{S}_{\kappa_p}(F, Y)$. Since the skeletal subsets are disjoint, the labels are unique and well-defined.

From now we assume that the structuring element Y corresponds to the 4-neighbors of the origin and the origin itself:

$$Y = \{\ (-1, 0),\ (0, -1),\ (0, 0),\ (0, 1),\ (1, 0)\ \}\ . \tag{1.12}$$

Figure 1.6 shows an example of the morphological skeleton and the skeletal labels. Figure 1.7 shows another example of the morphological skeleton, while Fig. 1.8 shows how to create the morphological skeleton step by step.

The following lemma provides an important property of the labels.

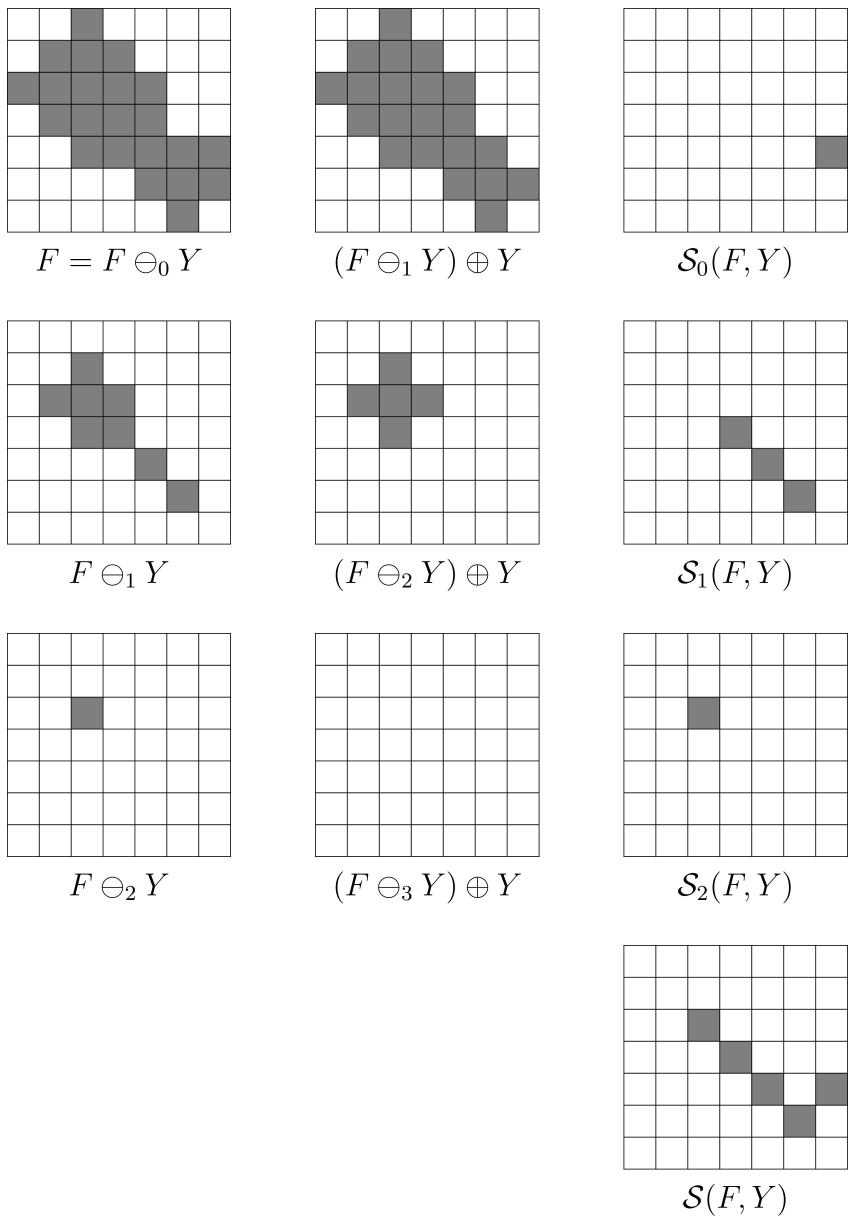


Figure 1.8: Generating the morphological subsets $\mathcal{S}_k(F, Y)$ $(k = 0, 1, 2)$ according to (1.9). The top left image equals the original image F. In each row, the image in the third column is the difference of the previous two images. Note that $\mathcal{S}_k(F, Y) = \emptyset$ for $k \geq 3$, thus the final morphological skeleton is $\mathcal{S}(F, Y) = \mathcal{S}_0(F, Y) \cup \mathcal{S}_1(F, Y) \cup \mathcal{S}_2(F, Y)$.

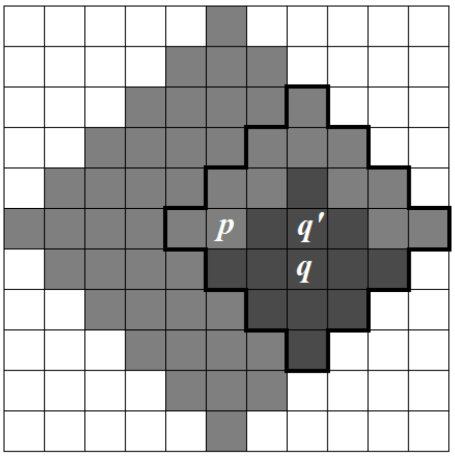


Figure 1.9: An example of the assumption in Lemma 1.4.1. Gray pixels indicate $\widehat{P}$, dark gray pixels indicate $\widehat{Q}$, where $\widehat{Q} \subsetneq \widehat{P}$. Here, $d_1(p,q) = 3$, $\kappa_p = 5$ and $\kappa_q = 2$. Note that $(q' \oplus_{\kappa_q+1} Y) \subset \widehat{P}$, enclosed with thick lines.

Lemma 1.4.1 *Let $p \in \mathcal{S}_{\kappa_p}(F,Y)$ and $q \in \mathcal{S}_{\kappa_q}(F,Y)$ be two distinct skeletal points of a binary image F and the structuring element Y defined by (1.12). The Manhattan distance of the skeletal points is greater than the difference on their labels, i.e.,*

$$d_1(p,q) > |\kappa_p - \kappa_q| \ . \tag{1.13}$$

Proof

Assume to the contrary that p and q are skeletal points of F with $d_1(p,q) \le |\kappa_p - \kappa_q|$. Without loss of generality, let $\kappa_p \ge \kappa_q$, therefore our assumption is

$$d_1(p,q) + \kappa_q \le \kappa_p \ . \tag{1.14}$$

Let $\widehat{P} = (p \oplus_{\kappa_p} Y)$ and $\widehat{Q} = (q \oplus_{\kappa_q} Y)$. From (1.11) we know that $\widehat{P}, \widehat{Q} \subset F$. Also because of the attributes of Y defined by (1.12) and the morphological dilatation,

$$u \in \widehat{P} \iff d_1(p,u) \le \kappa_p \tag{1.15}$$

$$v \in \widehat{Q} \iff d_1(q,v) \le \kappa_q \tag{1.16}$$

for any $u, v \in \mathbb{Z}^2$.

Therefore, for every point $v \in \widehat{Q}$, using (1.14) and (1.16),

$$d_1(p,v) \le d_1(p,q) + d_1(q,v) \le d_1(p,q) + \kappa_q \le \kappa_p \ . \tag{1.17}$$

Overall with (1.15), $(v \in \widehat{Q}) \to (v \in \widehat{P})$, and since $p \ne q$, it follows that $\widehat{Q} \subsetneq \widehat{P}$. Hence, there exists a point q' such that q and q' are 4-neighbors and $(q' \oplus_{\kappa_q+1} Y) \subset \widehat{P} \subset F$, so $q' \in (F \ominus_{\kappa_q+1} Y)$ (see Fig. 1.9 for an example). Consequently, $q \in \big[(F \ominus_{\kappa_q+1} Y) \oplus Y\big]$, which means $q \notin \mathcal{S}_{\kappa_q}(F,Y)$ by (1.9), which is a contradiction. $\square$

The morphological skeleton plays a great role in Chapter 4, where it is used as additional information for the binary reconstruction task.

1.5 Simulated Annealing

Simulated Annealing (SA) was intruduced in [61], and later independently described in [53] and [24]. SA is a generic probabilistic method for global optimization problems, i.e., finding the global minimum of functions whose characteristics are usually unknown and/or extremely complex. It only assumes that evaluation of the function is possible at arbitrary points. Unlike gradient descent methods, with SA the probability of finding global optimal solution tends to 1.

SA, as its name suggests, simulates the physical phenomenon of annealing, where the goal is to find the state with the minimal internal energy of a system through controlled cooling, regardless the initial state. During the cooling process, the energy of the state might increase due to the thermal noise, hence can avoid local optimum. Overall, if the cooling process is approached carefully, the process terminates in a state having (nearly) minimal energy. In terms of SA, the system is the function to be minimized, and the states are simply the values of the function. See Algorithm 2 for the pseudo-code for SA.

Algorithm 2 Simulated Annealing

```
Require: $f$ function, $x$ initial solution, $T$ initial temperature
Ensure: $x^*$ solution for minimizing $f$
  repeat
    $x' \leftarrow x$,  modify $x'$ randomly
    if $f(x') < f(x)$ then
      $x \leftarrow x'$
    else
      $x \leftarrow x'$ with a probability of $e^{\frac{f(x')-f(x)}{T}}$
    end if
    Decrease $T$
  until the termination criterion is satisfied
  return $x^* \leftarrow x$
```

SA is simple to implement, robust, and flexible in the sense of controlling its parameters. However, one serious drawback of the method is that one has to fine-tune many parameters to achieve an acceptable approximation of the global minimum of f, including random modification of the actual solution, the annealing schedule, and the selection of the starting parameters. Furthermore, in many practical applications it is not trivial to describe the optimization problem as a function minimization problem, which can bring more parameters into the process.

In the literature there are many studies where SA have been applied to solve an image reconstruction problem. In [52], the authors found that SA ensured slightly better reconstruction quality than the iterative method for single photon emission computed tomography (SPECT). In [60] a variation of SA was proposed to solve the electrical impedance tomography (EIT) reconstruction problem. In [57], the authors used SA for reconstructing discrete images on a triangular grid from six projections. The authors of [56] studied the performance of various implementations of the SA algorithm when applied to binary functions, especially in binary reconstruction.

We use SA in Chapter 4 for solving the reconstruction problem, where one or two of the projections are given, as well as the morphological skeleton (see Section 1.4).

Chapter 2

Eliminating Switching Components in Binary Matrices

2.1 Introduction

Studying the structure of binary matrices plays a vital role in numerous applications of computer science. Binary matrices can describe the connection between the data represented in rows and the data represented in columns; they can contain binary patterns in a natural way; or can represent a whole digital image. Therefore, analyzing binary matrices is an important task of intelligent data analysis [15], data mining [71], low-level image processing [38], and machine learning [62], among others. One commonly performed task is to localize and enumerate special subpatterns in the binary matrix. Such a basic and essential subpattern of a binary matrix is the so-called switching component, which is a 2×2 submatrix with exactly two 1-s in the diagonal and two 0-s in the antidiagonal, or vice versa (for a formal definition, see Section 1.2). The importance of searching switching components – if they exist – in a binary matrix comes from the fact that the absence of these patterns is a necessary and sufficient condition for the unique reconstruction of the matrix from the vectors of its row and column sums [65]. Therefore, if uniqueness is guaranteed then the binary image represented by the binary matrix can be stored in a (lossless) compressed form by those two vectors. The presence of even one switching component makes the solution non-unique when no additional prior information is available about the binary matrix to be reconstructed. Nevertheless, even if the matrix contains switching components, then there is still a chance to reconstruct the matrix uniquely [25], if properly chosen elements of the original matrix are stored as well, using them as prior information. One can store, e.g., the positions of 0-s which need to be inverted to 1-s (by so-called 0-1 flips) in order to make the matrix switching component free. The aim is then to find the minimal number of 0-1 flips needed to achieve uniqueness. Apart from image reconstruction, the number and the position of switching components also play an important role in the field of biogeography and ecology. There, binary matrices can represent the presence or absence of certain species (rows) on certain locations (columns), which is also strongly connected to the theory of 0-1 flips (see [58], and the references given there). Then, the so-called nestedness is a relevant measure of the matrix, which describes how strongly the species depend on each other and their locations. Similarly, data mining and intelligent data analysis can also use nestedness – or nestedness-like measurements – to show the connection between the data represented by a binary matrix (see again references in [58]).

Unfortunately, as it was proven in [58], determining the minimal number of 0-1 flips to achieve uniqueness is generally an NP-hard problem. Besides, one can consider 1-0 flips instead of 0-1 flips. In that case, the minimal number of flips can differ for a certain matrix, but the problem is still generally NP-hard due to the symmetrical roles of 1-s and 0-s. Moreover, if both 0-1 and 1-0 flips are allowed, finding the minimal number of flips is considered to be NP-hard, although it has not been proven yet. In this chapter, we focus only on the 0-1 flips. We show that the minimal number of 0-1 flips can be found by determining the proper ordering of the columns of the matrix according to a certain filling function, instead of searching through matrix elements and switching components. Based on theoretical results, we develop two deterministic, polynomial-time heuristics to find the minimal number of 0-1 flips. We compare those methods to another well-known methods in the literature, on a wide set of random binary matrices, and also a real-life dataset of presence-absence matrices. We conclude that the algorithms searching for proper column permutations perform better, regarding both the number of 0-1 flips and running time, especially on sparse matrices. Moreover, we show how to use these algorithms in a simple way for general binary image compression with Chang's algorithm, previously described in Section 1.3.

2.2 Problem Setting and Theoretical Results

Our goal is to answer the Minimum Flip Augmentation problem, described in Section 1.2. The aim is to determine the minimal number of 0-s needed to change into 1-s of a binary matrix in order to make the matrix switching component free. Changing each 0 to 1 in a given A binary matrix would yield a binary matrix with no switching component, therefore such a switching component free binary matrix A^* always exists. On the other hand, in [58] the following lemma is proven.

Lemma 2.2.1 Mfa *is NP-complete.*

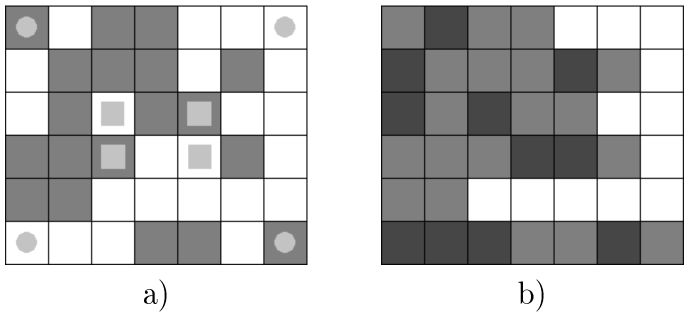


Figure 2.1: An example of the canonical expansion function: a) a binary image, some of the switching components are marked with squares and disks, (b) the same image after applying the canonical expansion function. Dark gray pixels indicate the 0-1 flips.

A naive approach to find a (not necessary optimal) solution may include a searching through the switching components of A, and eliminate them by changing 0 values into 1-s, in a sequential order. There can be $O(m^2n^2)$ switching components, and a certain 0 can belong to $O(mn)$ switching components. For example, in the identity matrix any 0 element under the main diagonal forms a switching component with any 0 element above the main diagonal with the corresponding 1-s. The aim is to identify a minimal sized sequence of those 0-s, thus an exhaustive search for the optimal solution may require $O((mn)!)$ steps. We show how to speed up the searching process through special operations in order to gain much faster approximate solutions. In that case, the exhaustive search will require at most $O((\min\{m,n\})! \cdot mn)$ steps. Before describing the heuristics, we provide some theoretical results. First, we give the definition of the canonical expansion function.

The *canonical expansion* of the binary matrix A is a binary matrix ψA of the same size as A, with elements defined by

$$\psi a_{ij} = \begin{cases} 0 & \text{if } a_{ij'} = 0 \text{ for every } j' \geq j, \\ 1 & \text{otherwise.} \end{cases}$$

Figure 2.1 shows an example of a canonical expansion. Since ψ performs only 0-1 flips in A, thus $A \subseteq \psi A$. If $\psi A = A$ for a binary matrix A, then A is called a *canonical matrix*[1]. Note that $\psi\psi A = \psi A$, therefore the canonical expansion of any binary matrix is a canonical matrix.

Besides, given the binary matrix A of size $m \times n$ and a permutation π of order n, let πA denote the binary matrix which consists of the columns of A according to π. The following lemmas show important properties of canonical matrices.

Lemma 2.2.2 *Any canonical matrix is switching component free.*

[1]Not to be confused with canonical *hv*-convex images described in Chapter 3.

Proof
The presence of a switching component requires a row in the binary matrix containing a 0 followed by a 1 somewhere in the same row (not necessarily on an adjacent position). Since $\psi A = A$ for a canonical matrix A, A clearly has no such rows. □

Lemma 2.2.3 *Let A be a switching component free binary matrix with non-increasing column sums. Then A is a canonical matrix.*

Proof
Assume to the contrary that A is a switching component free binary matrix with non-increasing column sums, and A is not a canonical matrix. Since A is not a canonical matrix, there exists a row i such that $a_{ij} = 0$ and $a_{ij'} = 1$ for some $j < j'$ columns. But the j-th column contains at least as much 1-s as the j'-th column, and therefore there must be a row i' such that $a_{i'j} = 1$ and $a_{i'j'} = 0$. Then, a_{ij}, $a_{ij'}$, $a_{i'j}$ and $a_{i'j'}$ form a switching component, which is a contradiction. □

The next lemma describes a property of the canonical expansion.

Lemma 2.2.4 *Let A and B be two binary matrices of the same size $m \times n$. If $A \subseteq B$ then $\psi A \subseteq \psi B$.*

Proof
Let $i \in \{1, \dots, m\}$ be an arbitrary row index. Moreover, let $j_{l(A)}$ denote the position of the last 1 in the i-th row of A, i.e., $a_{ij_{l(A)}} = 1$ and $a_{ij} = 0$ for $j > j_{l(A)}$. Similarly, let $j_{l(B)}$ denote the position of the last 1 in the i-th row of B, hence $b_{ij_{l(B)}} = 1$ and $b_{ij} = 0$ for $j > j_{l(B)}$. From $A \subseteq B$ it follows that $j_{l(A)} \leq j_{l(B)}$.

By the definition of the canonical expansion, $\psi a_j = 1$ if and only if $1 \leq j \leq j_{l(A)}$. Similarly, $\psi b_j = 1$ if and only if $1 \leq j \leq j_{l(B)}$. Since $j_{l(A)} \leq j_{l(B)}$, it follows that $\psi b_k = 1$ whenever $\psi a_k = 1$ for $k = 1, \dots, n$. The row index i was chosen arbitrarily, thus we get $\psi A \subseteq \psi B$. □

Finally, the following theorem reveals the connection between canonical expansions and the solutions of the MFA problem.

Theorem 2.1 *Let A be a binary matrix of size $m \times n$, and let A^* denote a solution of* MFA*(A). Then there is a column permutation π of order n such that $\pi^{-1}\psi\pi A = A^*$.*

Proof
Let π be a (not necessarily unique) permutation such that πA^* is a binary matrix with non-increasing column sums. Trivially, $A \subseteq A^*$, and by the definition of the column permutation and the subset relation, $\pi A \subseteq \pi A^*$. A column permutation has no effect on the existence of switching components, hence πA^* is still switching component free. But then, by Lemma 2.2.3, πA^* is canonical and therefore $\psi\pi A^* = \pi A^*$. Since $\pi A \subseteq \pi A^*$, by Lemma 2.2.4 we get $\psi\pi A \subseteq \psi\pi A^* = \pi A^*$. Therefore, $\pi^{-1}\psi\pi A \subseteq A^*$.

On the other hand, by the definition of canonical expansion, $\pi A \subseteq \psi\pi A$, and therefore, $A \subseteq \pi^{-1}\psi\pi A$. Moreover, on the basis of Lemma 2.2.2, $\psi\pi A$ is switching component free, thus $\pi^{-1}\psi\pi A$ is also switching component free. Furthermore, from the arguments of the previous paragraph it follows that $|\pi^{-1}\psi\pi A| \leq |A^*|$, and therefore $\big(|\pi^{-1}\psi\pi A| - |A|\big) \leq \big(|A^*| - |A|\big)$. Since A^* is a solution of the $\textsc{Mfa}(A)$ problem, the right hand side of above inequality is minimal. Therefore the left hand side must be also minimal, thus $\pi^{-1}\psi\pi A$ must be a solution of the $\textsc{Mfa}(A)$ problem. We have that $|\pi^{-1}\psi\pi A| = |A^*|$ which together with $\pi^{-1}\psi\pi A \subseteq A^*$ yields $\pi^{-1}\psi\pi A = A^*$. □

Figure 2.2 illustrates Theorem 2.1. Unfortunately, the proof of the theorem defines π as a function of the solution A^*, and due to Lemma 2.2.1 finding the proper column permutation is generally NP-complete. Nevertheless, the number of possible column permutations is much smaller than the number of possible sequences of switching components, in general.

Corollary 2.1 *To find a solution of the* $\textsc{Mfa}$*(A) problem, it is sufficient to search for the corresponding column permutation π. The number of such permutations is $O(n!)$, or considering the transposed matrix, $O((\min\{m,n\})!)$.*

Constructing the canonical expansion of a matrix can be done in $O(mn)$ time, and thus an exhaustive search for the optimal column permutation requires $O((\min\{m,n\})! \cdot mn)$ time, in the worst case.

2.3 Heuristics

We now describe four different heuristics for the $\textsc{Mfa}$ problem, which try to minimize the number of 0-s needed to be flipped to 1-s in order to make the matrix switching component free. All of them are deterministic methods and have a polynomial running time. We note that the algorithms work for 1-0 flips as well, if one inverts the elements of the input matrix, runs the algorithm, and inverts again the elements of the resulted matrix. Unfortunately, the difference between the number of necessary 0-1 flips and 1-0 flips can be arbitrary large. For example, consider the binary matrix I of size $n \times n$ with 1-s in the main diagonal and 0-s elsewhere. It is easy to see that any column permutation π leads to an optimal solution, since $\pi^{-1}\psi\pi I$ results the same number of 0-1 flips, which is $n(n-1)/2$. On the other hand, considering 1-0 flips the optimal solution is to change I into an empty matrix, which has only n number of 1-0 flips. Since the number of necessary 0-1 and 1-0 flips can differ, using the ones (or even a mixture of them) which provide smaller number of flips is considerable; although for the tests we only considered 0-1 flips.

Algorithms $\textsc{Switch}$ (Algorithm 3) and $\textsc{Columns}$ (Algorithm 4) are taken from [58] for comparison. $\textsc{Switch}$ is a switching component searching algorithm,

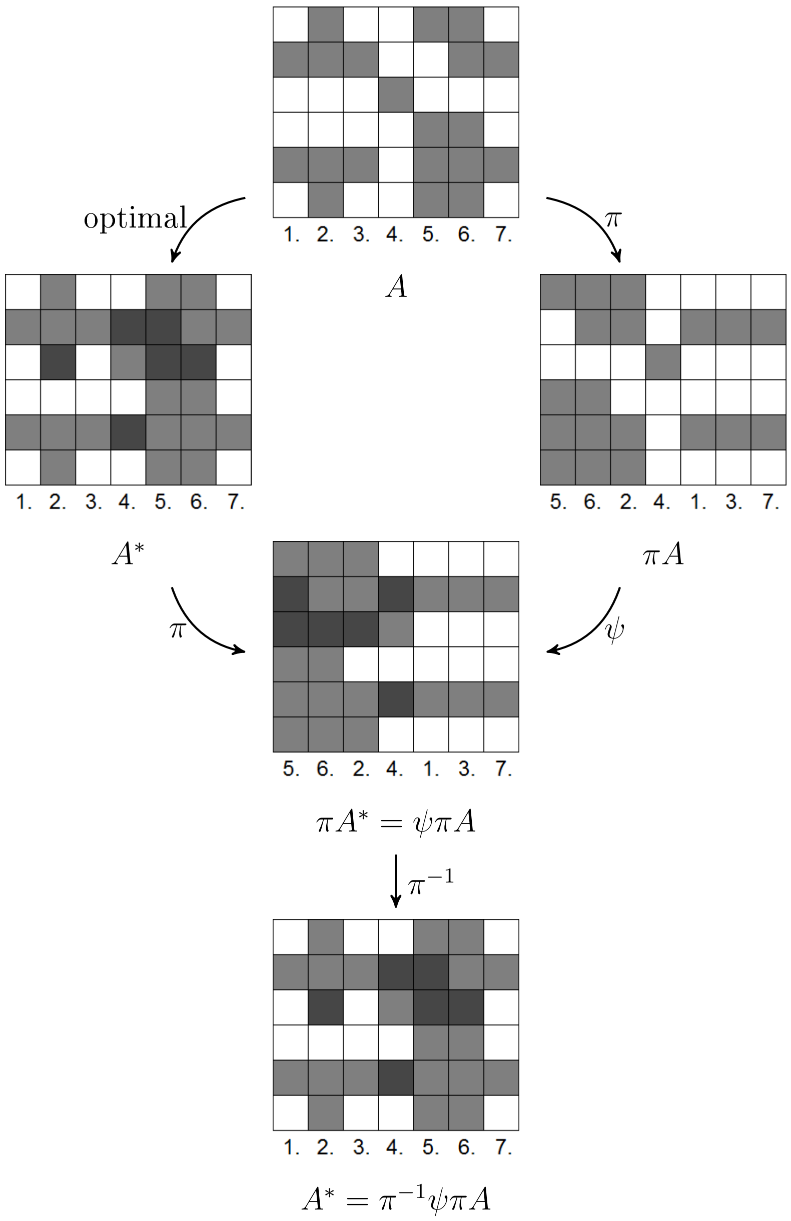


Figure 2.2: Illustration of Theorem 2.1. One of the optimal solutions (A^*) found by exhaustive search (left image of the second row), π defined as a column permutation in a way that πA^* has non-increasing column sums (third row). Numbers indicate the original column indices.

while Columns works with column permutations. Although [58] does not use the concept of canonical expansions, for technical convenience, we give the pseudo code of Columns to our terms. Our own methods ColPerm1 (Algorithm 5) and ColPerm2 (Algorithm 6) are based on Theorem 2.1 and Corollary 2.1. All algorithms require a binary matrix A with the size of $m \times n$ as input, and provide a binary matrix A' such that $A \subseteq A'$ and A' is switching component free. The worst-case time complexity of Switch is $O(m^2 n^2 \log(mn))$ according to [58]. Examining the pseudo-codes, it is easy to see that the time complexity for Columns is $O(n \log n + m)$, for ColPerm1 is $O(mn^2)$, and for ColPerm2 is $O(mn^3)$. Note that m and n are interchangeable considering the transpose of the matrices.

Algorithm 3 Switch

```
$C \leftarrow$ zero matrix with a size of $m \times n$
$A' \leftarrow A$
for each row index $i$ and column index $j$ do
    Let $c_{ij}$ be the number of switching components including $a_{ij} = 0$
end for
while $A'$ is not switching component free do
    $(i, j) \leftarrow \arg\max\{c_{ij}\}$
    $a'_{ij} \leftarrow 1$
    Update $C$
end while
return $A'$
```

Algorithm 4 Columns

```
Let $\pi$ be a column permutation such that $\pi A$ contains the columns of $A$ in a
non-increasing order by the sum of their elements
return $A' \leftarrow \pi^{-1} \psi \pi A$
```

Algorithm 5 ColPerm1

```
Let $\pi$ be the identical permutation
for each column index $i$ do
    Let $j > i$ be the column index for which the column permutation $\pi_{ij}$ yields
    the biggest decrease in the number of 0-1 flips when applying the operator $\psi$
    Swap columns $i$ and $j$ by $\pi_{ij}$
    $\pi \leftarrow \pi \cdot \pi_{ij}$
end for
return $A' \leftarrow \pi^{-1} \psi \pi A$
```

Since Mfa is NP-complete by Lemma 2.2.1, a polyominal-time algorithm cannot necessary find the optimal solution in all cases (unless P = NP). Figure 2.3 shows small counter-examples demonstrating that the given algorithms can fail in finding the real optimal solution.

Algorithm 6 ColPerm2

```
while true do
    Let $i$ and $j$ be column indices for which the column permutation $\pi_{ij}$ yields the
    biggest decrease in the number of 0-1 flips when applying the operator $\psi$
    if there are such $i$ and $j$ indices then
        Swap columns $i$ and $j$ by $\pi_{ij}$
        $\pi \leftarrow \pi \cdot \pi_{ij}$
    else
        Break loop
    end if
end while
return $A' \leftarrow \pi^{-1}\psi\pi A$
```

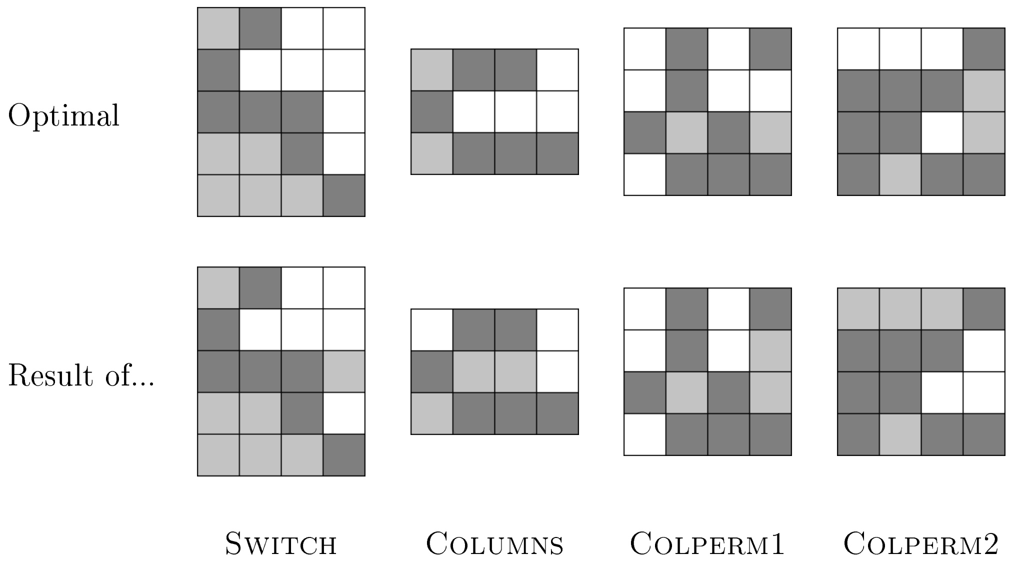


Figure 2.3: Example images to show the non-optimality of the heuristics. Dark gray pixels indicate the 1-s in the original matrix, while light gray pixels indicate 0-1 flips. The resulted image is always switching component free, but not necessarily minimal in the number of 0-1 flips.

2.4 Numerical Results

We studied the performance of the algorithms described in Section 2.3 on random binary matrices and on an existing database containing real-life data. We implemented the algorithms in MATLAB 7.13.0.564. The test was performed under Windows 7 on one core of an Intel Core i5-2410M of 2.3 GHz PC with 4GB of RAM.

2.4.1 Artificial Dataset

Our test set contained matrices of size 20×20, 40×40, 60×60, 80×80, and 100×100 and with exactly 10%, 20%, ..., 90% number of 1-s related to the total number of the matrix entries, thus, providing matrices with different densities.

Table 2.1: Average number of 0-1 flips calculated by the algorithms SWITCH (SWI), COLUMNS (COL), COLPERM1 (CP1), and COLPERM2 (CP2).

20×20				40×40				60×60			
SWI	COL	CP1	CP2	SWI	COL	CP1	CP2	SWI	COL	CP1	CP2
121.8	133.5	120.7	**114.2**	831.9	851.3	768.3	**729.4**	2341.5	2325.7	2088.0	**2024.9**
172.1	191.7	170.9	**163.7**	986.2	1008.6	928.1	**900.3**	2481.4	2475.8	2316.7	**2269.9**
175.4	200.6	180.0	**173.2**	929.5	965.3	900.8	**879.8**	2297.6	2283.2	2181.7	**2139.2**
161.6	183.5	168.4	162.0	823.4	854.7	811.1	**797.1**	2003.8	2012.5	1940.0	**1915.2**
145.7	162.6	151.3	147.8	695.0	729.1	698.9	**687.8**	1684.4	1702.1	1652.4	**1634.3**
121.8	136.0	127.2	125.0	**563.0**	592.8	570.8	564.0	1343.2	1372.8	1338.3	**1327.5**
91.3	102.3	95.7	93.9	**423.0**	447.7	433.4	427.8	**1000.6**	1036.2	1011.4	1005.4
62.6	69.3	66.1	64.1	**283.5**	298.6	290.9	287.3	**668.4**	691.7	678.2	673.7
30.5	33.4	32.0	31.4	**141.9**	149.3	145.1	143.9	**334.5**	345.7	339.6	337.3

80×80				100×100			
SWI	COL	CP1	CP2	SWI	COL	CP1	CP2
4596.0	4554.4	4139.5	**4039.0**	7587.4	7460.6	6892.0	**6764.5**
4636.3	4585.2	4352.7	**4282.3**	7459.6	7358.1	7055.4	**6951.2**
4210.8	4183.2	4024.6	**3970.5**	6706.9	6631.8	6441.8	**6377.5**
3675.5	3653.4	3550.4	**3517.1**	5819.4	5766.7	5632.9	**5592.6**
3069.4	3071.0	3000.4	**2977.6**	4874.9	4846.1	4752.6	**4724.3**
2456.0	2473.6	2425.1	**2410.6**	3899.1	3895.1	3834.3	**3815.0**
1828.2	1864.4	1832.3	**1822.1**	2915.3	2932.5	2890.5	**2879.8**
1215.2	1244.6	1224.6	1220.5	**1922.9**	1957.6	1934.5	1926.4
607.1	622.2	613.1	611.1	**960.0**	978.5	968.1	965.1

With each size and density we generated 50 binary matrices from uniform random distribution. Thus, our test set contained a total of 2250 matrices.

Table 2.1 shows the results for the number of 0-1 flips provided by each algorithm. From top to bottom the rows represent the density of the 1-s in the matrices, from 10% to 90%. The numerical entries are the averaged result on the 50 matrices for the given size and density. The smallest numbers are typeset in bold. In a similar way, Table 2.2 shows the average running time of the algorithms SWITCH and COLPERM2 which provided the best values from the viewpoint of optimality. We did not provide the exact running time values for COLUMNS and COLPERM1 due to their high speed: COLUMNS processed the result in less than 0.002 seconds for all matrices, while COLPERM1 had a running time of 0.344 seconds in the slowest case.

From the tables we deduce that searching through column permutations yields a result much sooner than searching through switching components, as SWITCH does. Furthermore, COLPERM2 gives better results for the number of 0-1 flips, especially when the matrix is big and sparse. We deduce that SWITCH performs better if the number of switching components is small, which occurs if the matrix is small and/or dense.

2.4.2 Real Dataset

As we described in Section 1.2, nestedness is an important measurement of presence-absence binary matrices in the field of biogeography and ecology. Here, we

Table 2.2: Average running time of the algorithms Switch (SWI) and ColPerm2 (CP2) in seconds.

20×20		40×40		60×60		80×80		100×100	
SWI	CP2	SWI	CP2	SWI	CP2	SWI	CP2	SWI	CP2
0.094	0.084	2.361	0.516	15.111	2.921	52.522	9.383	132.550	24.700
0.145	0.045	3.082	0.483	17.112	2.445	55.919	7.850	139.931	20.639
0.156	0.041	3.194	0.460	17.103	2.188	55.075	7.305	139.507	17.371
0.149	0.038	3.080	0.369	16.010	1.877	51.955	5.951	128.205	15.130
0.159	0.033	2.776	0.332	15.147	1.623	46.227	4.882	114.348	12.227
0.136	0.027	2.389	0.257	12.329	1.267	39.843	3.812	98.638	10.162
0.116	0.026	1.832	0.227	9.817	0.998	31.580	3.212	79.335	7.664
0.080	0.018	1.324	0.169	6.994	0.710	22.027	2.203	53.989	5.330
0.041	0.015	0.684	0.098	3.778	0.438	11.550	1.288	28.745	3.053

tested our algorithms on a real-life dataset containing information about the relation between two datasets. The majority of the dataset were assembled for a meta-analysis of nested subset distribution patterns and the metrics used to evaluate them[2]. The dataset contains binary matrices describing 150 archipelagos, to identify poorly represented taxa (many invertebrate groups), life-zones (especially aquatic and marine systems), or geographic locations (e.g., tropical systems). The database contains 289 matrices overall. One representative example of the dataset is seen in Fig. 2.4. Furthermore, Figures 2.5 and 2.6 show two examples on the performance of the heuristics.

We classified the matrices into 9 groups according to their densities. Figures 2.7 and 2.8 show two examples of the result in the number of flips. The other groups showed similar results.

We compared the algorithms by the number of flips, and counted how many times they provided the best or the worst result out of four. Table 2.3 shows these results. The first coulmn indicates the densities of the matrices, the second column indicates the number of matrices with the given density in the database. The number of wins (respectively, losses) show how many times the given algorithm provided the best (repectively, worst) result, including ties. Table 2.4 shows the same results, but excluding ties (the given algorithm was strictly the best / worst). Best results are shown in bold.

The results of the real dataset are highy correlated to the results of the artificial dataset, namely, ColPerm2 provided usually the best results (most wins and least losses), while Switch gave usually good results, especially on dense matrices. ColPerm2 was moderate in the number of flips, and Columns was usually the worst, however, the last two heuristics were much faster than the first two. ColPerm2 and Switch took several seconds on large matrices, while ColPerm

[2]http://aics-research.com/nestedness/tempcalc.html available in February 2015, also used in [58].

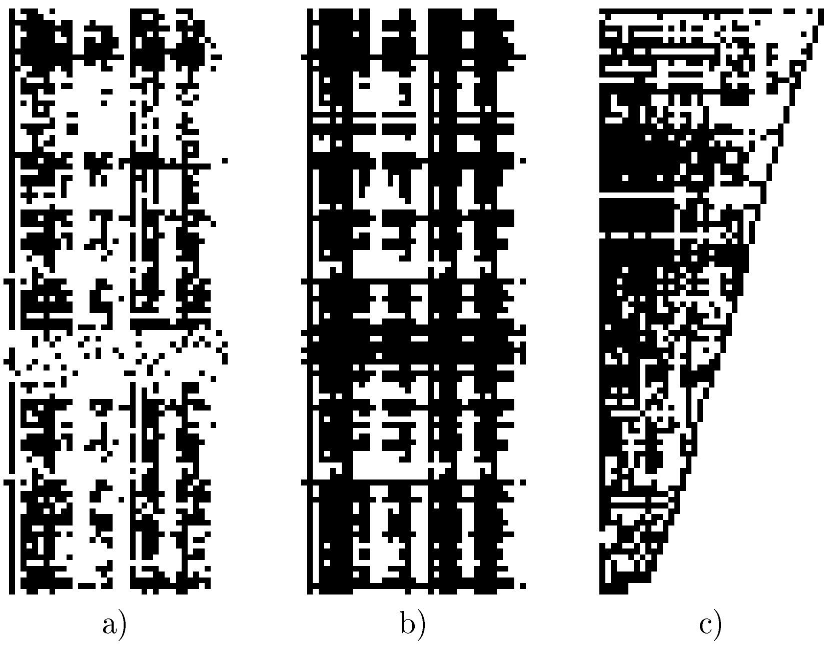


Figure 2.4: The Fullglas matrix of the real dataset. a) the original matrix contains the presence of goldenrods, milkweeds, and legumes in 102 prairie fragments in Iowa and Minnesota; b) the fully nested matrix provided by ColPerm2; c) the original matrix with rearranged rows and columns according to the row sums and coulmn sums of the second matrix. Black pixels indicate 1-s (presence), white pixels indicate 0-s (absence).

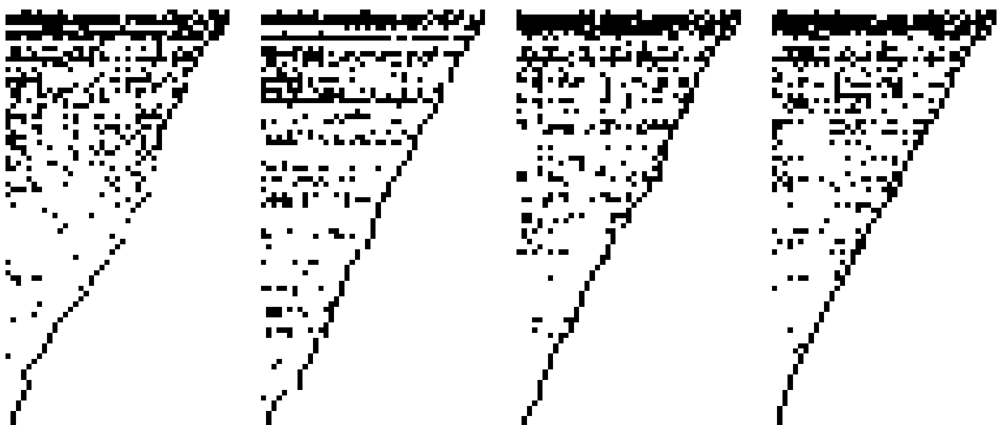


Figure 2.5: The Artiherb matrix: Understory herbs in planted stands of trees in the Alexandria Moraine, Minnesota (isolates). The rows and columns are permuted according to the heuristics: Switch, Columns, ColPerm1, ColPerm2 (left to right, respectively).

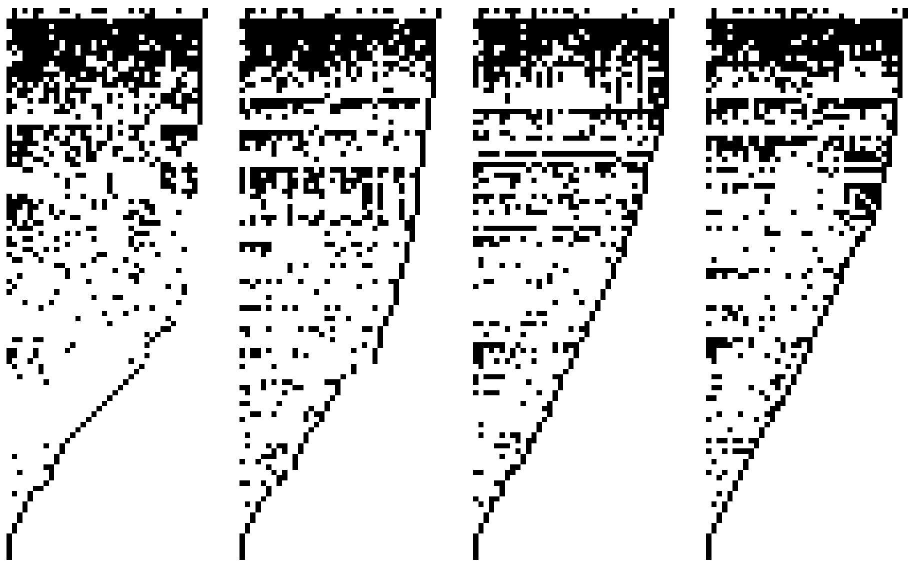


Figure 2.6: The TANGANYO matrix: Ostracods in Lake Tanganyika, Africa; areas not isolated from each other. The rows and columns are permuted according to the heuristics: SWITCH, COLUMNS, COLPERM1, COLPERM2 (left to respectively).

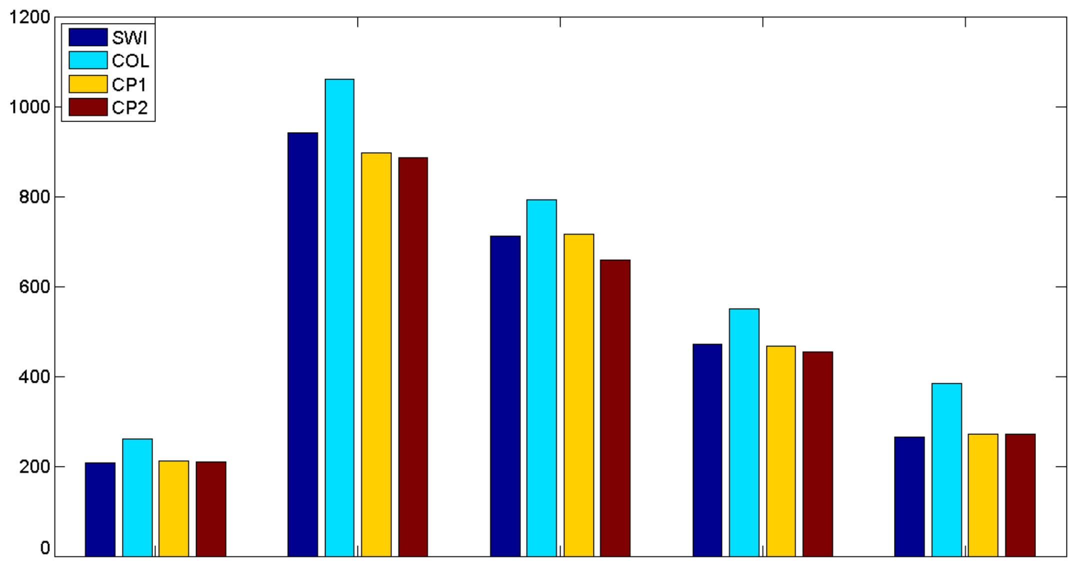


Figure 2.7: Numerical results for the group with density between 0% and 10% (5 matrices). Vertical axis indicates the number of 0-1 flips.

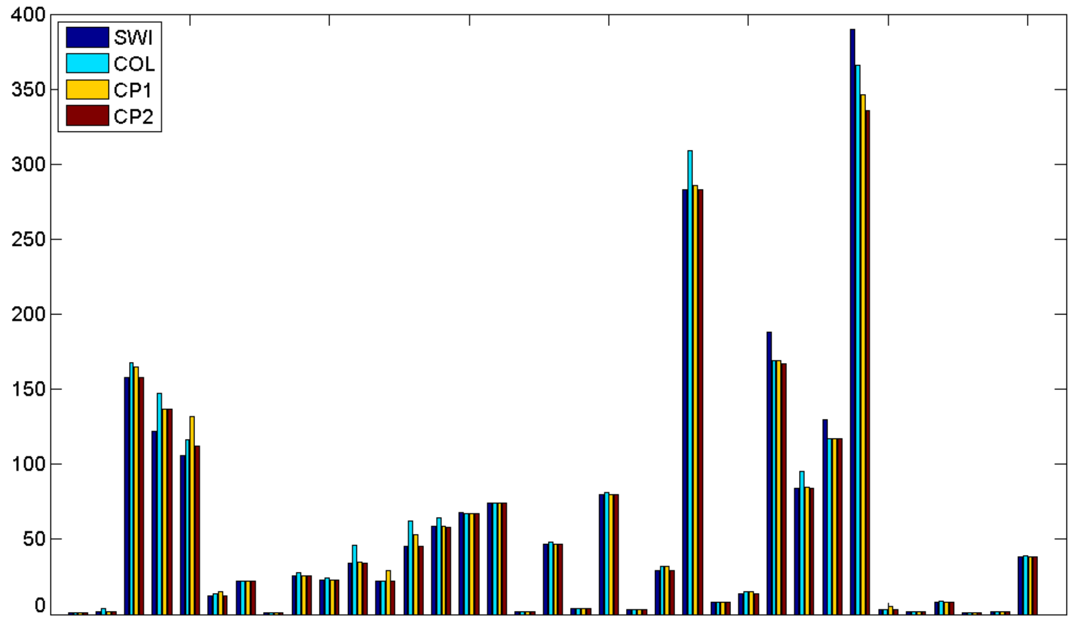


Figure 2.8: Numerical results for the group with density between 50% and 60% (35 matrices). Vertical axis indicates the number of 0-1 flips.

Table 2.3: The number of wins and losses on the real dataset inluding ties.

Density (%)	Num. of matrices	Num. of wins				Num. of losses			
		SWI	COL	CP1	CP2	SWI	COL	CP1	CP2
0 – 10	5	2	0	0	**3**	**0**	5	**0**	**0**
10 – 20	26	7	1	2	**26**	4	16	7	**0**
20 – 30	66	36	15	25	**58**	25	44	16	**9**
30 – 40	76	42	23	36	**69**	28	42	33	**11**
40 – 50	50	40	16	34	**46**	18	40	22	**15**
50 – 60	35	30	15	20	**33**	15	27	17	**11**
60 – 70	20	**18**	12	15	**18**	**12**	19	**12**	**12**
70 – 80	7	**7**	4	4	6	**2**	5	5	**2**
80 – 90	4	**4**	3	**4**	**4**	**3**	4	**3**	**3**

Table 2.4: The number of wins and losses on the real dataset excluding ties.

Density (%)	Num. of matrices	Num. of wins				Num. of losses			
		SWI	COL	CP1	CP2	SWI	COL	CP1	CP2
0 – 10	5	2	0	0	**3**	**0**	5	**0**	**0**
10 – 20	26	0	0	0	**18**	4	15	6	**0**
20 – 30	66	8	0	0	**20**	15	36	6	**0**
30 – 40	76	6	1	0	**17**	14	28	19	**0**
40 – 50	50	1	0	1	**6**	3	25	7	**0**
50 – 60	35	2	0	0	**3**	4	14	4	**0**
60 – 70	20	**2**	0	0	1	1	7	**0**	**0**
70 – 80	7	**1**	0	0	0	**0**	2	2	**0**
80 – 90	4	**0**	**0**	**0**	**0**	**0**	1	**0**	**0**

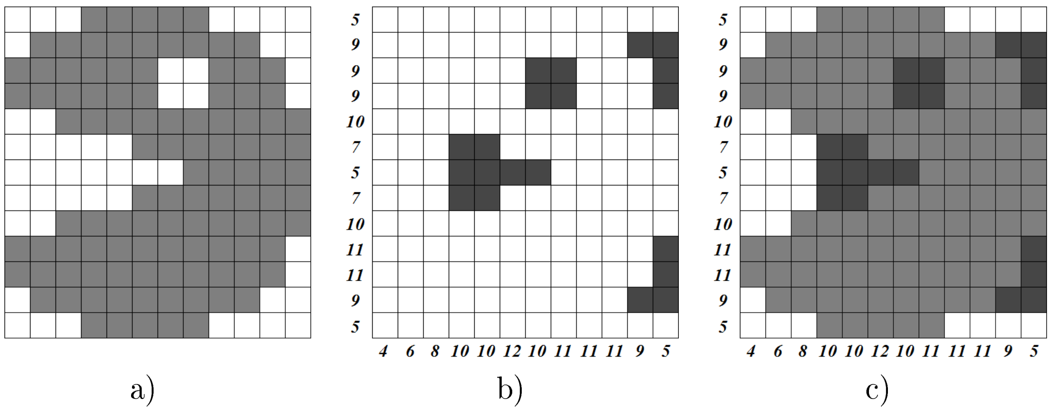


Figure 2.9: An example of image compression. a) original image; b) the stored data of the projections and the forbidden positions (0-1 flips); c) Chang's algorithm found the unique solution of the reconstruction.

and Columns always gave their output within a fraction of a second.

2.5 Data Compression

Chang's algorithm (see the pseudo code in Section 1.3) is a polyominal-time algorithm for reconstructing unique matices from the horizontal and vertical projections, i.e., when the number of solutions is exactly one. The algorithm requires the absence of switching components, which is the necessary and sufficient condition of non-uniqueness [65]. However, the algorithm can deal with a set of so-called forbidden positions, i.e., positions which have fixed values (0 or 1) in the matrix.

As a consequence, any binary matrix can be reconstructed from their projections uniquely if the forbidden positions are the positions of the 0-1 flips, which make the matrix switching component free. Consequently, an arbitrary matrix can be stored – and reconstructed – through its projections with the additional information of the positions of the 0-1 flips. The less number of 0-1 flips is needed in order to make the matrix switching component free, the less data should be stored for a unique reconstruction. Thus, finding fast heuristics providing low number of 0-1 flips is essential for this type of data compression. Figure 2.9 shows an example of such a reconstruction.

For an image with the size of $n \times n$, one has to store the bits of the two projections plus the bits of the forbidden positions. Thus the overall number of bits to be stored is at most

$$2n\lceil \log n\rceil + |Q| \cdot 2\lceil \log n\rceil = (2|Q| + 2n) \cdot \lceil \log n\rceil,$$

where $|Q|$ is the number of forbidden positions (the number of 0-1 flips). The overall data compression ratio is

$$\frac{n^2}{(2|Q| + 2n) \cdot \lceil \log n \rceil} \ .$$

As a comparison, one can determine the morphological skeleton of the image (see Section 1.4 for definition). The image is uniquely reconstructable in polyominal-time if the positions of the morphological skeletal points and their labels are given, hence the number of bits to be stored is at most

$$|S| \cdot 2n\lceil \log n \rceil + |S| \cdot \lceil \log n \rceil = 3|S| \cdot \lceil \log n \rceil,$$

where $|S|$ is the number of skeletal points. In this case the data compression ratio is

$$\frac{n^2}{3|S| \cdot \lceil \log n \rceil} \ .$$

2.6 Summary

Switching components are special patterns in binary matrices that play an essential role in many image processing and pattern analysis tasks. The minimal number of 0-1 flips needed to make the binary matrix switching component free is an important measurement on the binary matrix. However, determining the minimal number of 0-1 flips is generally NP-complete. In this chapter we studied this problem, and proved that the task is equivalent to finding a proper permutation of the columns. Based on that result, we designed heuristic algorithms, and compared them with previously published algorithms both on an artificial and a real dataset. We found that the column based heuristics performed significantly faster and gave better results in the average case than switching component based heuristics. We showed how to use Chang's algorithm with the provided heuristics for binary image compression.

The findings of this research have been published in a conference proceeding [42] and are accepted for publication in a journal [41].

Chapter 3

Reconstruction and Random Generation of *hv*-Convex Images from the Horizontal Projection

3.1 Introduction

Projections of binary images, as described in Section 1.1, are fundamental shape descriptors widely used in tasks of pattern recognition, image processing, binary tomography, etc. In binary tomography [48, 49] they are used to reconstruct binary images from them. In the last 20 years, many of the subclasses of binary images had been studied, where the image to be reconstructed has to meet some special properties. One of the most frequently studied classes of binary images is that of *hv*-convex 4-connected binary images (with other term, *hv*-convex polyominoes). Another deeply studied type of binary images is the class of canonical *hv*-convex images[1] and its subclass of *hv*-convex 8-connected but not 4-connected images [6]. In [9, 21], the authors use *hv*-convex polyominoes for lossless data compression. The authors of [14] use *hv*-convexity in deciding when a form of local consistency called path consistency is suffcient to guarantee that a network is both minimal and globally consistent. In [31] it had been showed that *hv*-convex polyominoes are closed under composition, intersection, and transposition; establishing that path consistency over these constraints produces a minimal and decomposable network.

Many studies have been made about the reconstruction complexity and the number of solutions in the class of *hv*-convex images when the horizontal and vertical projections are available [10, 27, 30]. In [8] a detailed complexity analysis of the methods for reconstructing *hv*-convex polyominoes from horizontal and vertical

[1]Not to be confused with canonical matrices described in Chapter 2. The definitions are different, both used by the literature. We kept the terminology, since the two definitions should not interfere.

projections is given, both for the worst and for the average cases. Average time complexity is often measured empirically by generating inputs for an algorithm from a uniform random distribution. However, for that purpose, a uniform generator of the studied class of binary images is needed, which is usually not easy to develop. For *hv*-convex polyominoes such generator algorithms were described in [22] and in [50]. In [28] and [29] a general method has been also presented for the recursive enumeration and generation of several types of polyominoes.

Although the reconstruction of *hv*-convex polyominoes from the horizontal and vertical projections along with the identification of the number of possible solutions have been extensively studied [8, 10, 23, 26, 27, 30], those problems have been surprisingly not yet investigated if just one projection is given. Moreover, even if the exact number of solutions are known for reconstructing certain type of images, it is not necessarily true that an enumeration of the solutions – and thus, the uniform random generation – would be a trivial task. In this chapter, we fill this gap by describing a linear-time reconstruction algorithm and providing formulas for the number of solutions with minimal and with any given number of columns. Later on, we extend the above results by giving an elementary enumeration algorithm which provides a method for generating *hv*-convex polyominoes with given horizontal projection from a uniform random distribution, in quadratic time.

Despite the fact that the reconstruction from two projections is NP-hard in the general class of binary images [70], various polynomial-time algorithms are known for reconstructing canonical and 8-connected but not 4-connected images [7, 8, 11, 23, 26, 54]. In the end of this chapter, we complement the previous results by showing that reconstructing an *hv*-convex canonical image with minimal width from the horizontal projection is possible in linear in time of size of the horizontal projection. We propose an algorithm which not just reconstructs such an image but results always in an 8-connected image. Furthermore, the algorithm can be easily extended to reconstruct general *hv*-convex images with arbitrary width in the same running time. Thus, we deduce that reconstructing a 4-connected, an 8-connected, or even a general *hv*-convex binary image from one projection are similarly easy problems, from the viewpoint of computational complexity. This is in contrast to the case of two projections, where reconstructing a 4-connected or 8-connected *hv*-convex binary image is possible in polyominal time, while the reconstruction of general *hv*-convex binary images is NP-complete.

3.2 Definitions

Two positions $p = (i_p, j_p)$ and $q = (i_q, j_q)$ in a binary image are said to be *4-adjacent* if $|i_p - i_q| + |j_p - j_q| = 1$. p and q is *8-adjacent* if they are 4-adjacent

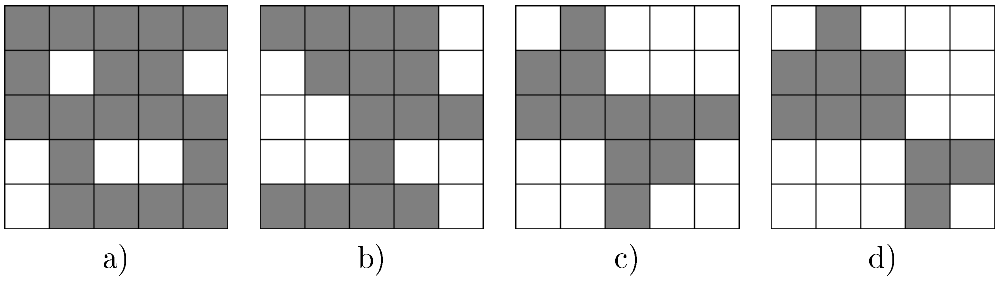


Figure 3.1: Binary images of size 5×5 with different properties: (a) a general polyomino with holes; (b) an *h*-convex but not *v*-convex polyomino; (c) an *hv*-convex polyomino; (d) an *hv*-convex 8-connected but not 4-connected binary image. Note that the last one is not a polyomino.

or $|i_p - i_q| = |j_p - j_q| = 1$. The positions p and q are *4-connected* if there is a sequence of distinct black pixels $p_0 = p, \ldots, p_k = q$ in the binary image such that p_l is 4-adjacent to p_{l-1}, respectively, for each $l = 1, \ldots, k$. A binary image F is *4-connected* if any two different object points in F are 4-connected. The 4-connected binary images are also called *polyominoes* [37]. The definition of 8-connectedness can be given analogously with 8-connected object points. The binary image F is *horizontally* and *vertically convex*, or shortly *hv*-convex if the black pixels are consecutive in each row and column of the image. See Fig. 3.1 for examples of binary images with different properties.

For the definition of canonical images, we have to define 4-components. A maximal 4-connected set of black pixels of a binary image F is called a *4-component* (shortly, component) of F. Every binary image F can be partitioned into components F_1, F_2, $\ldots$, F_k $(k \geq 1)$ in a uniquely determined way. Let us denote by $[i_l, i'_l] \times [j_l, j'_l]$ the minimal bounding rectangle of the l-th component of F $(l = 1, \ldots, k)$. An *hv*-convex image is called *canonical* [2], if it consists of a single 4-connected component or the smallest containing rectangles of the 4-connected components are 8-connected to each other with their bottom-right and upper left corners. Figure 3.2 shows an example of a canonical *hv*-convex image. In case of a canonical *hv*-convex image F with k components, we can assume for the components – without loss of generality – that $i_1 = 1, i_l = i'_{l-1} + 1$ $(l = 2, \ldots, k)$, and $i'_k = m$, where m is the number of rows in F.

Upper stack polyominoes are special *hv*-convex polyominoes which contain the two bottom corners of their minimal bounding rectangles. Similarly, *lower stack polyominoes* are *hv*-convex polyominoes that contain the two top corners of their minimal bounding rectangles. Finally, *parallelogram polyominoes* are *hv*-convex polyominoes that contain both their top left and bottom right, or both their top right and bottom left corners of their minimal bounding rectangles. Any *hv*-convex

[2]Again, not to be confused with canonical matrices described in Chapter 2.

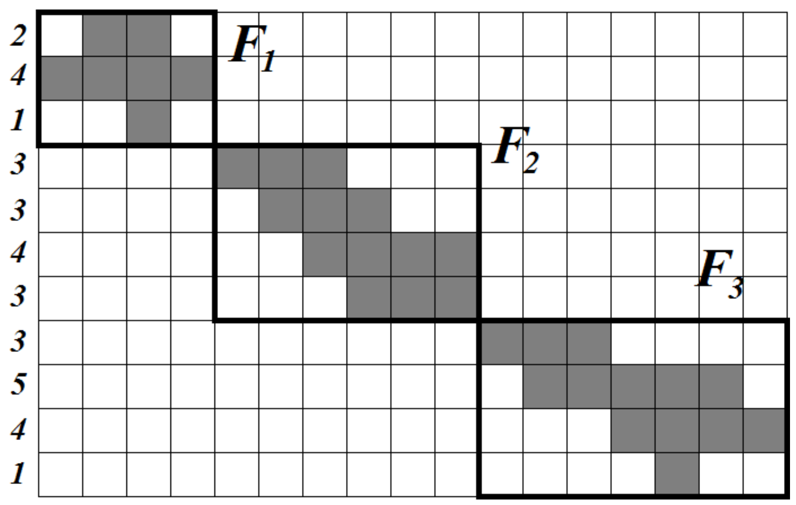


Figure 3.2: A canonical hv-convex image F with horizontal projection $(2, 4, 1, 3, 3, 4, 3, 3, 5, 4, 1)$, containing three 4-connected components F_1, F_2, and F_3. Thick lines indicate the smallest containing rectangles of the components. The width of the image is $w(F) = 17$.

polyomino can be constructed (not necessarily uniquely) from an upper stack, a parallelogram, and a lower stack polyomino [50]. Figure 3.3 shows examples of the special types of polyominoes, and such a construction.

In the followings, we call the consecutive 1-s in the i-th row of an hv-convex polyomino as the i-th *strip*. Through the chapter, we assume that hv-convex binary images are encoded with the starting positions of their strips.

3.3 Reconstructing hv-Convex Polyominoes

Let $H = (h_1, \ldots, h_m) \in \mathbb{N}^m$ be a vector of size m. We give an algorithm called *GreedyRec* which constructs an F hv-convex polyomino with m rows and the minimal possible number of columns. Algorithm 7 gives the pseudo code. For positioning the i-th strip, see Fig. 3.4 for examples. Finally, Fig. 3.5a shows an example result of the algorithm.

Theorem 3.1 *GreedyRec constructs an hv-convex polyomino satisfying the horizontal projection with minimal number of columns, in $O(m)$ time.*

Proof
It is clear that the resulted image is an hv-convex polyomino with the required horizontal projection. We prove by induction that no solution exists with less number of columns.

Let $n_o^{(k)}$ be the number of columns in a minimal-column solution of the problem (i.e., an hv-convex polyomino satisfying the projections with minimal number of columns), considering only the first k components of the input $(h_1, \ldots, h_k)$, where

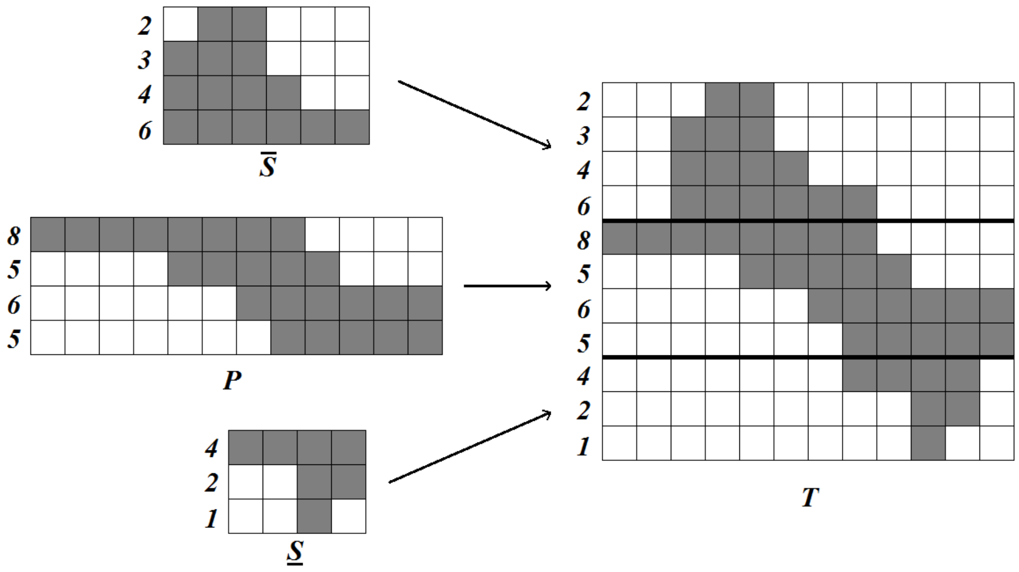


Figure 3.3: An *hv*-convex polyomino T composed of an upper stack $\overline{S}$, a parallelogram P, and a lower stack $\underline{S}$ polyomino.

Algorithm 7 *GreedyRec*

```
Require: Projection values $h_1 \leq \cdots \leq h_m$
Ensure: Strip positions $s_1, \ldots, s_m$
  $s_1 \leftarrow 1$
  for $i = 2 \rightarrow m$ do
    if $h_i = h_{i-1}$ then
      Let the $i$-th strip $s_i$ be aligned just below the $(i-1)$-st strip
    end if
    if $h_i < h_{i-1}$ then
      Let the $i$-th strip $s_i$ be aligned to the right of the $(i-1)$-st strip
    end if
    if $h_i > h_{i-1}$ then
      Let the $i$-th strip $s_i$ be aligned to the left of the $(i-1)$-st strip
    end if
  end for
  return $s_1, \ldots, s_m$
```

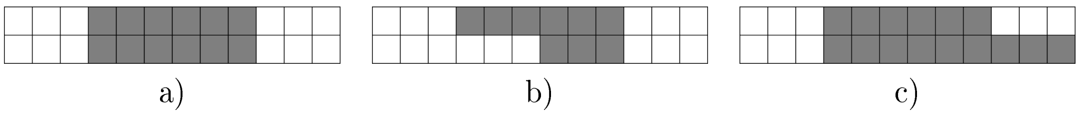


Figure 3.4: Steps of *GreedyRec* with the $(i-1)$-st and the i-th rows. Cases: (a) $h_i = h_{i-1}$; (b) $h_i < h_{i-1}$; (c) $h_i > h_{i-1}$.

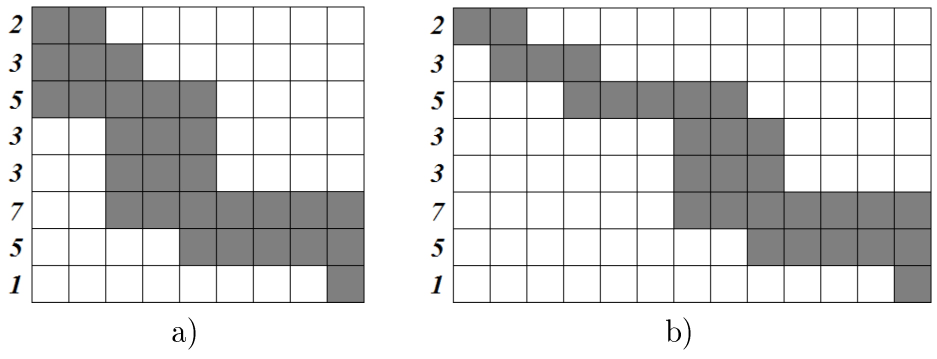


Figure 3.5: Example results of a reconstruction. (a) the minimum-size output of *GreedyRec* for $H = (2, 3, 5, 3, 3, 7, 5, 1)$ with 9 columns; (b) another solution with 13 columns.

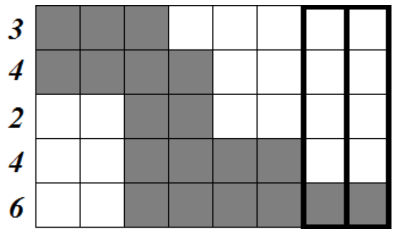


Figure 3.6: Examples of k-simple columns. Here, $h_4 = 4$ and $h_5 = 6$. Thick lines indicate 5-simple columns. The maximal number of 5-simple columns are $h_5 - h_4 = 2$.

$k \leq m$. Similarly, let $n_g^{(k)}$ be the number of columns in the result of *GreedyRec* for the first k components of the input.

For $k = 1$, $n_g^{(1)} = n_o^{(1)} = h_1$, so *GreedyRec* is optimal. For $k > 1$ assume that $n_g^{(k-1)} = n_o^{(k-1)}$.

If $h_k \leq h_{k-1}$, then $n_g^{(k)} = n_g^{(k-1)}$ (Cases 2(a) and 2(b) of *GreedyRec*), therefore the number of columns does not change. Since $n_o^{(k)} \geq n_o^{(k-1)}$, therefore $n_g^{(k)} = n_o^{(k)}$, and *GreedyRec* is still optimal.

If $h_k > h_{k-1}$, then $n_g^{(k)} = n_g^{(k-1)} + h_k - h_{k-1}$ (Case 2(c) of *GreedyRec*). Assume to the contrary that an arbitrary optimal algorithm provides a better result, hence $n_o^{(k)} < n_o^{(k-1)} + h_k - h_{k-1}$.

For a further analysis, let us call a column *k-simple* if its $(k-1)$-st element is 0 and its k-th element is 1 (see Fig. 3.6). The number of k-simple columns is at least $h_k - h_{k-1}$, and due to vertical convexity, in a k-simple column there is no 1-s above the k-th row. Therefore, the first $k - 1$ number of strips must fit into $n_o^{(k)} - (h_k - h_{k-1})$ number of non-k-simple columns at most. Due to h-convexity and connectivity, non-k-simple-columns must be successive. Therefore, the first $k - 1$ number of strips fit into a matrix with a column number of $n_o^{(k)} - (h_k - h_{k-1}) < n_o^{(k-1)} + h_k - h_{k-1} - (h_k - h_{k-1}) = n_o^{(k-1)}$, which is a contradiction to the minimality of $n_o^{(k-1)}$. Hence, *GreedyRec* is still optimal.

The complexity of the algorithm is straightforward, if the polyomino is represented by the first positions of its strips. □

One can easily modify the output of *GreedyRec* to expand it to have a predefined number of columns (if possible) by moving the k-th, $(k+1)$-st, $\ldots$, m-th strips further to the right, if the previous strip allows it (i.e., when the image remains hv-convex and 4-connected). The smallest possible number of columns, provided by *GreedyRec*, is $N_{\min} = N_m$, where

$$N_i = \begin{cases} h_i & \text{if } i = 1 \ , \\ N_{i-1} & \text{if } h_i \leq h_{i-1} \ , \\ N_{i-1} + h_i - h_{i-1} & \text{if } h_i > h_{i-1} \ . \end{cases} \tag{3.1}$$

This formula can be easily derived from the steps of the algorithm. The largest possible number of columns is

$$N_{\max} = \sum_{i=1}^{m} h_i - m + 1 \ , \tag{3.2}$$

where every strip is connected to the previous and the next strips through only one element. The modified *GreedyRec* can construct any solution between $N_{\min}$ and $N_{\max}$ in linear time. An example result of the modified algorithm is given in Fig. 3.5b.

3.4 Enumerating hv-Convex Polyominoes

Enumeration of polyominoes according to several parameters (area, perimeter, size of the bounding rectangle, etc.) is an extensively studied field of combinatorial geometry. Regarding the number of hv-convex polyominoes satisfying two projections, several results have been published. The authors of [30] determined an upper and lower bound to the maximum number of convex polyominoes having the same orthogonal projections, and also proved that under some conditions, the ambiguity of the solution can be exponential. A method was given in [28] for the enumeration of hv-convex polyominoes from two projections. The authors also determined the generating function of convex polyominoes. In [34] a method was proposed to determine the number of hv-convex polyominoes that fit into a discrete rectangle of given size. In this section, we provide formulas to enumerate hv-convex polyominoes satisfying the given horizontal projection.

3.4.1 Arbitrary Number of Columns

We first give a formula to calculate the number of hv-convex polyominoes with a given horizontal projection $H = (h_1, \ldots, h_m)$, if there is no restriction on the number of colums of the resulted image.

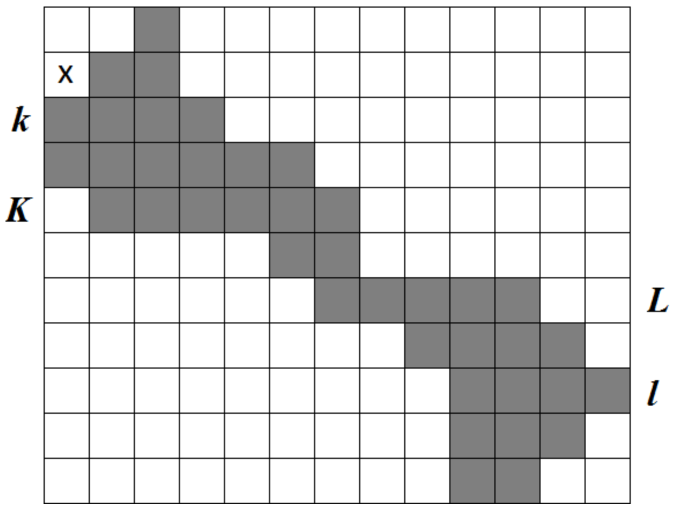


Figure 3.7: An hv-convex polyomino with $H = (1, 2, 4, 6, 6, 2, 5, 4, 4, 3, 2)$, where $K = 5$, and $L = 7$. The smallest left anchor position is $k = 3$, the greatest right anchor position is $l = 9$. The $(k-1)$-st strip can be placed on the top of the k-th strip in 2 different ways, and cannot occupy the position marked by x, since the k-th strip must be the leftmost strip.

Given an hv-convex polyomino F, the smallest integer k for which $f_{k1} = 1$ is called the *smallest left anchor position*. Similarly, the *greatest right anchor position* is the greatest integer l for which $f_{ln} = 1$. Furthermore, let K denote the greatest integer for which $h_1 \leq h_2 \leq \cdots \leq h_K$. Similarly, let L be the smallest integer for which $h_L \geq h_{L+1} \geq \cdots \geq h_m$. Figure 3.7 illustrates these definitions.

First, assume that $K < L$. Then, $K < k, l < L$ cannot hold, due to v-convexity. Also note that for every $k < l$ solution, a vertically mirrored image is also a solution with $l < k$, and vice versa. For this reason, we only count the cases with $k < l$ (i.e., $1 \leq k \leq K$ and $L \leq l \leq m$), and multiply the result by 2.

Let $\overline{S}_k(H)$ denote the number of upper stack polyominoes having the horizontal projection $(h_1, \ldots, h_k)$. Similarly, let $\underline{S}_l(H)$ denote the number of lower stack polyominoes having the horizontal projection $(h_l, \ldots, h_m)$. Furthermore, let $P_{k,l}(H)$ denote the number of parallelogram polyominoes with the horizontal projection $(h_k, \ldots, h_l)$, having the smallest left anchor position k and the greatest right anchor position l.

Lemma 3.4.1 *$\overline{S}_1(H) = 1$, and $\overline{S}_k(H) = \prod_{i=2}^{k}(h_i - h_{i-1} + 1)$ $(k \geq 2)$. $\underline{S}_m(H) = 1$, and $\underline{S}_l(H) = \prod_{i=l}^{m-1}(h_i - h_{i+1} + 1)$ $(l < m)$.*

Proof
The formula $\overline{S}_1(H) = 1$ is trivial. If $k \geq 2$, then the $(k-1)$-st strip can be placed on the top of the k-th strip in $h_k - h_{k-1} + 1$ different ways. Similarly, the $(k-2)$-nd strip can be placed on the top of the $(k-1)$-st strip, in $h_{k-1} - h_{k-2} + 1$ different ways. And so on. Finally, the first strip can be placed in $h_2 - h_1 + 1$ ways on the

top of the second strip. The formula for the lower stack polyominoes can be proven analogously. □

Lemma 3.4.2 $P_{k,l}(H) = \prod_{i=k}^{l-1} \min\{h_i, h_{i+1}\}$.

Proof
The k-th strip is fixed (it is in the leftmost position), and we can place the $(k+1)$-st strip under the k-th strip in $\min\{h_k, h_{k+1}\}$ ways. The $(k+2)$-nd strip can be placed under the $(k+1)$-st strip in $\min\{h_{k+1}, h_{k+2}\}$ ways. And so on. Finally the l-th strip can be placed under the $(l-1)$-st strip in $\min\{h_{l-1}, h_l\}$ ways. □

In the sequel we use the convention that empty (non-defined) factors of a product are always equal to 1.

Theorem 3.2 *Let $H \in \mathbb{N}^m$. If $K < L$ then the number of hv-convex polyominoes with the horizontal projection H is*

$$P_{K<L}(H) = 2 \cdot \sum_{k=1}^{K} \sum_{l=L}^{m} \left(\overline{S}_{k-1}(H) \cdot (h_k - h_{k-1}) \cdot P_{k,l}(H) \cdot (h_l - h_{l+1}) \cdot \underline{S}_{l+1}(H) \right) . \quad (3.3)$$

If $K \geq L$, then the number of solutions is

$$P_{K \geq L}(H) = P_{K<L}(H) - \overline{S}_L(H) \cdot \underline{S}_K(H) . \quad (3.4)$$

Proof
We observe that an hv-convex polyomino with the smallest left anchor position k and the greatest right anchor position l can be uniquely decomposed into a (possibly empty) upper stack polyomino consisting of the first $k-1$ rows, a (possibly empty) lower stack polyomino of consisting of the last rows from $l+1$ to m, and a parallelogram polyomino consisiting of the k-th, $(k+1)$-st, ..., l-th rows. If k is the smallest left anchor position, then the $(k-1)$-st strip (the bottom strip of the upper stack polyomino) cannot reach the leftmost position (see the position marked by x in Fig. 3.7), therefore the upper stack can be connected to the parallelogram in $(h_k - h_{k-1})$ ways. With a similar argument, the lower stack can be connected to the bottom row of the parallelogram in $(h_l - h_{l+1})$ ways. Thus, using lemmas 3.4.1 and 3.4.2, for fixed k and l the number of possible solutions is $\overline{S}_{k-1}(H) \cdot (h_k - h_{k-1}) \cdot P_{k,l}(H) \cdot (h_l - h_{l+1}) \cdot \underline{S}_{l+1}(H)$. Including also the mirrored cases we get (3.3).

If $K \geq L$, then the same formula as in (3.3) can be applied. However, in this case, it counts some of the solutions twice through symmetry (where the parallelogram poliominoes are rectangular). Note that the longest strips in H are $h_L = h_{L+1} = \cdots = h_K$, and (3.3) counts all the cases twice when these strips are right under each other. Regarding that the L-th strip is the bottom of the upper stack polyomino, and the K-th strip is the uppermost row of the lower stack polyomino, the number of cases counted twice is $\overline{S}_L(H) \cdot \underline{S}_K(H)$, using Lemma 3.4.1. □

3.4.2 Fixed Number of Columns

Now, we give a recursive formula to calculate the number $P_n(H)$ of hv-convex polyominoes having the horizontal projection $H = (h_1, \ldots, h_m)$, when the number of columns is fixed to n. First, assume again that $K < L$. Let $r \geq 1$ and $P(p_1, \ldots, p_r, n)$ denote the number of parallelogram polyominoes with n columns, having the horizontal projection $(p_1, \ldots, p_r)$.

Lemma 3.4.3 *$P(p_1, n) = 1$ if $p_1 = n$. $P(p_1, n) = 0$ if $p_1 \neq n$. Furthermore, for $r > 1$ we have the following recursion*

$$P(p_1, \ldots, p_r, n) = \begin{cases} \sum_{i=1}^{p_1} P(p_2, \ldots, p_r, n - i + 1) & \text{if } p_1 \leq p_2 \ , \\ \\ \sum_{i=1}^{p_2} P(p_2, \ldots, p_r, n - (p_1 - p_2) - i + 1) & \text{if } p_1 > p_2 \ . \end{cases}$$

Proof
If $r = 1$, then either the strip itself of length p_1 occupies n number of columns (and should be counted as a solution) or not. If $r > 1$ and $p_1 \leq p_2$, then we count recursively every possible solution where the second strip is shifted to the right under the first strip, and the number of remaining columns decreases proportionately. If $r > 1$ and $p_1 > p_2$, then additionally, we have to substract the difference from the number of required columns, since the second strip must be shifted with at least $p_1 - p_2$ positions to the right, relatively to the first position of the first strip. □

Therefore, including the possible stack polyominoes and the mirrored cases, the number of solutions for a fixed n is

$$P_n(H) = 2 \cdot \sum_{k=1}^{K} \sum_{l=L}^{m} \left(\overline{S}_{k-1}(H) \cdot (h_k - h_{k-1}) \cdot P(h_k, \ldots, h_l, n) \cdot (h_l - h_{l+1}) \cdot \underline{S}_{l+1}(H) \right) ,$$

where $P(h_k, \ldots, h_l, n) = 0$ if $k > l$.

If $K \geq L$ then we have to substract some of the solutions in the same way as in (3.4). Note that this concerns only $P_{N_{\min}}(H)$ (where n is minimal), since for every other case a mirrored solution is truly a different solution.

$P_n(H)$ also provides a different formula for calculating the number of solutions, if the size of the polyomino can be arbitrary, namely

$$\sum_{n=N_{\min}}^{N_{\max}} P_n(H) ,$$

where $N_{\min}$ and $N_{\max}$ is given by (3.1) and (3.2), respectively.

3.5 Random Generation of *hv*-Convex Polyominoes

The authors of [5] collected several statistics on *hv*-convex binary images, including the number of solutions of the reconstruction from given horizontal and vertical projections. In [8] three algorithms had been compared for reconstructing *hv*-convex polyominoes and *hv*-convex 8-connected binary images from two projections. In the study the algorithms listed all the possible solutions for the given problem. In [4], the author described a method to generate some special *hv*-convex binary images from a uniform random distribution.

In this section we provide an elementary enumeration algorithm based on how we counted the number of possible solutions in Section 3.4. The algorithm ensures a method for generating *hv*-convex polyominoes with given horizontal projection from a uniform random distribution. We prove that the algorithm has a quadratic running time in the length of the horizontal projection.

3.5.1 Arbitrary Number of Columns

In this subsection we provide an algorithm for generating *hv*-convex polyominoes with arbitrary number of columns. For a given $H = (h_1, \ldots, h_m)$ and a non-negative integer b ($0 \leq b \leq P - 1$) the algorithm constructs the b-th *hv*-convex polyomino with horizontal projection H with respect to a certain ordering, where P is the number of solutions (see (3.3) and (3.4)). Note that there is always at least one *hv*-convex polyomino satisfying the given horizontal projection.

Let us encode the *hv*-convex polyomino F with $(s_1, \ldots, s_m)$ where s_i denotes the starting position of the i-th strip ($1 \leq i \leq m$). Then, for a fixed row number i, $f_{ij} = 1$ if and only if $j \in \{s_i, s_i + 1, \ldots, s_i + h_i - 1\}$, and $f_{ij} = 0$ otherwise. First, we give generating algorithms for special polyominoes. The generation of upper stack polyominoes is given by Algorithm 8. In the pseudo-code, b mod d is the remainder of the integer division b/d. Figure 3.8 shows an example of generating upper stack polyominoes.

Algorithm 8 *GenUpperStack*

```
Require: Projection values $h_1 \leq \cdots \leq h_r$ and ordinal number $b$
Ensure: Strip positions $s_1, \ldots, s_r$
  $s_r \leftarrow 1, \quad s \leftarrow 1$
  for $i = r - 1 \rightarrow 1$ do
    $d \leftarrow h_{i+1} - h_i + 1$
    $s_i \leftarrow s + (b \mod d)$
    $s \leftarrow s_i$
    $b \leftarrow \lfloor b/d \rfloor$
  end for
  return $s_1, \ldots, s_r$
```

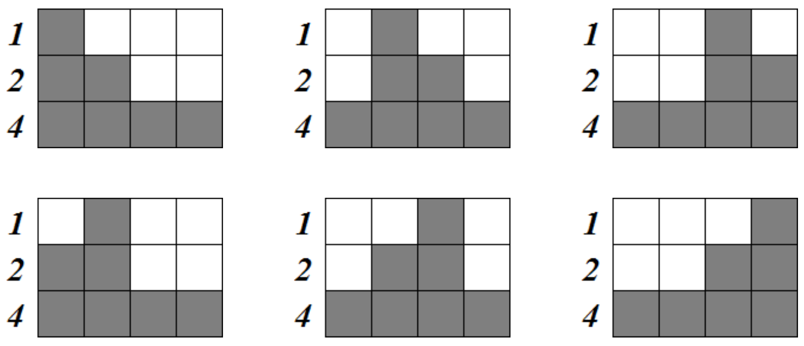


Figure 3.8: Example of generating upper stack polyominoes with the horizontal projection $H = (1, 2, 4)$ for $b = 0, 1, \ldots, 5$, respectively.

Lemma 3.5.1 *For a given vector $H = (h_1, \ldots, h_r)$ $(h_1 \le \cdots \le h_r)$* GenUpperStack *generates different upper stack polyominoes for different values $0 \le b \le \overline{S}_r(H) - 1$, where $\overline{S}_r(H)$ is the number of possible upper stack polyominoes with horizontal projection H.*

Proof
The condition $h_1 \le \cdots \le h_r$ is necessary for an upper stack polyomino. The last strip must be in the first position, i.e., $s_r = 1$. The position of the i-th strip $(i = r-1, r-2, \ldots, 1)$ depends on the current ordinal number b and the previous relative position s, where $s = 1$ in the beginning. The i-th strip can be placed on the top of the $(i+1)$-st strip in $d = h_{i+1} - h_i + 1$ different ways, i.e., $s_i \in \{s, s+1, \ldots, s+d-1\}$. The algorithm chooses one of these cases depending on the remainder of the integer division b/d, which can be $0, 1, \ldots, d-1$. Namely, let $s_i = s + (b \bmod d)$. Then the algorithm modifies s for the next strip, i.e., $s = s_i$, and also divides b by d for further positioning. Finally, the algorithm returns with the strip positions. Since these positions are calculated by a sequence of integer divisions, the algorithm generates different solutions for different values of b $(0 \le b \le \overline{S}_r(H) - 1)$. □

In a smiliar way, one can provide *GenLowerStack* for generating lower stack polyominoes. The next algorithm generates parallelogram polyominoes with respect to the ordinal number b, where the first strip is a left anchor, and the last strtip is a right anchor (see Algorithm 9). Note that if $r = 1$, then the condition of the for-loop of the algorithm cannot be satisfied, and therefore the operations of the cycle are never performed. Figure 3.9 shows an example of generating parallelogram polyominoes.

Lemma 3.5.2 *For a given vector $H = (h_1, \ldots, h_r)$* GenPara *generates different parallelogram polyominoes (with left anchor position 1 and right anchor position r) for different values $0 \le b \le P_{1,r}(H) - 1$, where $P_{1,r}(H)$ is the number of such polyominoes with horizontal projection H.*

Algorithm 9 *GenPara*

```
Require: Projection values $h_1, \ldots, h_r$ and ordinal number $b$
Ensure: Strip positions $s_1, \ldots, s_r$
  $s_1 \leftarrow 1, \quad s \leftarrow 1$
  for $i = 2 \rightarrow r$ do
    if $h_i < h_{i-1}$ then
      $s \leftarrow s + h_{i-1} - h_i$
    end if
    $d \leftarrow \min\{h_{i-1}, h_i\}$
    $s_i \leftarrow s + (b \mod d)$
    $s \leftarrow s_i$
    $b \leftarrow \lfloor b/d \rfloor$
  end for
  return $s_1, \ldots, s_r$
```

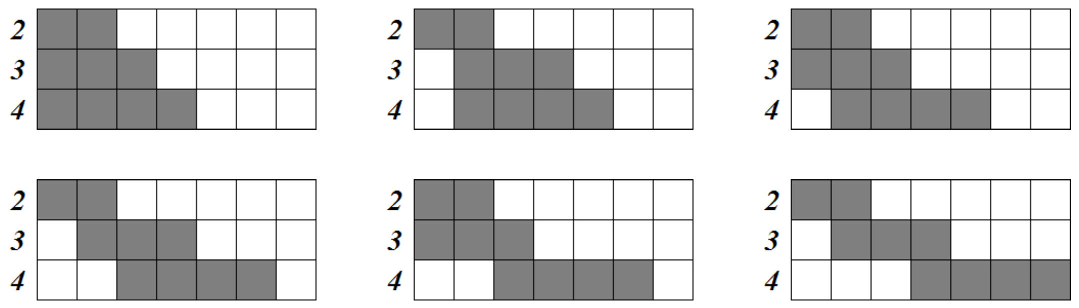


Figure 3.9: Example of generating parallelogram polyominoes with the horizontal projection $H = (2, 3, 4)$ for $b = 0, 1, \ldots, 5$, respectively.

Proof

The first strip is in the first position ($s_1 = 1$), and the i-th strip can be put under the $(i-1)$-st strip in $\min\{h_{i-1}, h_i\}$ different ways ($i = 2, \ldots, r$). If $h_i < h_{i-1}$, then we have to increase the possible first position s with $h_{i-1} - h_i$ due to v-convexity and the right anchor property of the last strip. The rest of the algorithm is similar to *GenUpperStack*. □

The following lemma describes an important property of Algorithm *GenPara*.

Lemma 3.5.3 GenPara *provides a parallelogram polyomino with minimal number of columns if and only if $b = 0$.*

Proof

If $b = 0$, then *GenPara* is equivalent to *GreedyRec*, that provides a solution with minimal number of columns (see again Algorithm 7 of Section 3.3, and Theorem 3.1). If $b > 0$, then a position t ($1 < t \leq r$) exists, such that the t-th strip is shifted further to the right, relative to the solution provided by *GreedyRec*. Hence all the t-th, $(t+1)$-st, …, r-th strips are shifted to the right, and the number of columns cannot be minimal. □

Algorithm *GenAnchor* (see Algorithm 10) generates *hv*-convex polyominoes with fixed left and right anchor positions.

Algorithm 10 *GenAnchor*

```
Require: Projection values $h_1, \dots, h_m$, first left anchor position $k$, last right anchor position $l$ $(k \le l)$, and ordinal number $b$
Ensure: Strip positions $s_1, \dots, s_m$
  if $k \ge 2$ then
    $\overline{b} \leftarrow b \bmod \overline{S}_{k-1}(H)$,    $b \leftarrow \lfloor b/\overline{S}_{k-1}(H) \rfloor$
    $(s_1, \dots, s_{k-1}) \leftarrow GenUpperStack(h_1, \dots, h_{k-1}, \overline{b})$
    $\overline{d} \leftarrow h_k - h_{k-1}$
    $\overline{R} \leftarrow (b \bmod \overline{d}) + 1$,    $b \leftarrow \lfloor b/\overline{d} \rfloor$
    $s_1 \leftarrow s_1 + \overline{R}$,  $\dots$,  $s_{k-1} \leftarrow s_{k-1} + \overline{R}$
  end if
  if $l \le m-1$ then
    $\underline{b} \leftarrow b \bmod \underline{S}_{l+1}(H)$,    $b \leftarrow \lfloor b/\underline{S}_{l+1}(H) \rfloor$
    $(s_{l+1}, \dots, s_m) \leftarrow GenLowerStack(h_{l+1}, \dots, h_m, \underline{b})$
    $\underline{d} \leftarrow h_l - h_{l+1}$
    $\underline{R} \leftarrow (b \bmod \underline{d}) - 1$,    $b \leftarrow \lfloor b/\underline{d} \rfloor$
  end if
  $(s_k, \dots, s_l) \leftarrow GenPara(h_k, \dots, h_l, b)$
  if $l \le m-1$ then
    $\underline{R} \leftarrow \underline{R} + s_l$
    $s_{l+1} \leftarrow s_{l+1} + \underline{R}$,  $\dots$,  $s_m \leftarrow s_m + \underline{R}$
  end if
  return $s_1, \dots, s_m$
```

Lemma 3.5.4 *For a given vector $H = (h_1, \dots, h_m)$* GenAnchor *generates different hv-convex polyominoes with fixed smallest left anchor position k, and fixed greatest right anchor position l $(k \le l)$ for different values $0 \le b \le \widehat{P} - 1$, where $\widehat{P} = \overline{S}_{k-1}(H) \cdot (h_k - h_{k-1}) \cdot P_{k,l}(H) \cdot (h_l - h_{l+1}) \cdot \underline{S}_{l+1}(H)$ is the number of possible such polyominoes with horizontal projection H.*

Proof

The algorithm generates the upper stack, the lower stack, and the parallelogram polyominoes separately. Since k is the position of the smallest left anchor, the upper stack polyomino is determined by the first $k-1$ strips (if $k = 1$, there is no upper stack polyomino). *GenUpperStack* provides the $\overline{b}$-th solution from the possible upper stack polyominoes, where $0 \le \overline{b} < \overline{S}_{k-1}(H)$ is a valid input of the algorithm. The ordinal number b is divided by the number of existing solutions for the further parts of the construction.

The $(k-1)$-st strip (the last strip of the upper stack polyomino) can only be in the 2nd, 3rd, ..., $(h_k - h_{k-1} + 1)$-st position. Therefore, the algorithm has to shift the first $k-1$ strip further to the right with $\overline{R} \in \{1, 2, \dots, h_k - h_{k-1}\}$. The choice of R depends on the current ordinal number, which is decreased further after the choice of $\overline{R}$.

The generation of the lower stack polyomino is similar. The algorithm also has to shift the positions at least to the position of the l-th strip. Note that the size of the shift (see $\underline{R}$) can be calculated after the parallelogram stack polyomino is generated by *GenPara* due to the relative positions to s_l. We have to decrease b first, so *GenPara* will be the last part of our algorithm but we have to call *GenPara* first, because we need s_l in order to calculate the relative shifting $\underline{R}$ of the lower stack polyomino. That is the reason why the final $\underline{R}$ is calculated in two parts, unlike $\overline{R}$.

It is easy to see that *GenAnchor* provides different solutions for different ordinal numbers, if $0 \le b \le \widehat{P} - 1$. □

Algorithm *GenAnchor* has an ordering described in the following lemma.

Lemma 3.5.5 *In* GenAnchor *the ordinal number of any solution with minimal number of columns is less than the ordinal number of any solution with non-minimal number of columns. In other words,* GenAnchor *orders the solutions by starting with all the smallest sized solutions.*

Proof
Using our defnition of decomposition of an *hv*-convex polyomino into an upper stack, a lower stack, and a parallelogram polyomino, the width of the *hv*-convex polyomino is equal to the width of its parallelogram part (see again Fig. 3.3, for example). *GenAnchor* builds the stack polyominoes and their relative shifts first and the parallelogram polyomino last. Therefore, in the algorithm the function *GenPara*$(h_k, \ldots, h_l, 0)$ is called if and only if the non-negative ordinal number b is less than $\overline{P} \cdot \underline{P}$, where $\overline{P}$ (respectively, $\underline{P}$) denotes the number of different ways the upper (respectively, lower) stack polyominoes can be composed with the parallelogram polyominoes. Both $\overline{P}$ and $\underline{P}$ are independent from the choice of b. As a consequence of Lemma 3.5.3, the resulted *hv*-convex polyomino is minimal in the number of columns if and only if $0 \le b < \overline{P} \cdot \underline{P}$. □

Before describing the next algorithm, we recall a definition from [26]. An *hv*-convex polyomino is called a *centered polyomino* if it has a left anchor position that is also a right anchor position (see Fig. 3.10, for example). The following properties are important for the rest of the chapter:

- if F is a centered polyomino, then $K \ge L$, and $h_L = h_{L+1} = \cdots = h_K$,
- if F is a centered polyomino, then $s_L = s_{L+1} = \cdots = s_K = 1$ (which also yields that F can be decomposed such that its parallelogram polyomino part is a single rectangle),
- the set of centered polyominoes with the same horizontal projection is closed with respect to vertical mirroring,

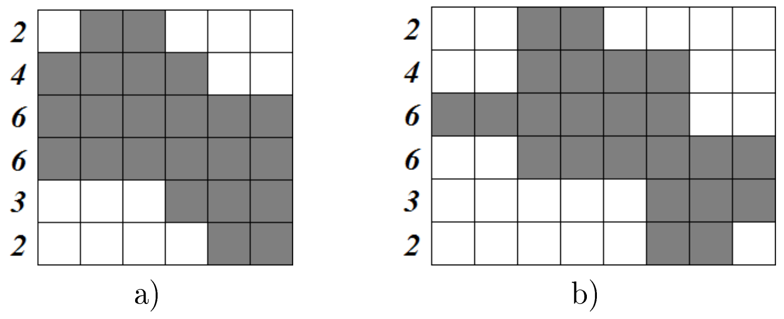


Figure 3.10: Examples of centered and non-centered polyominoes. (a) a centered polyomino; (b) a non-centered polyomino with the same horizontal projection. Note that $4 = K \geq L = 3$, and the first image is minimal in the number of columns, while the second one is not.

- if $K \geq L$ and F is an hv-convex polyomino with minimal number of columns then F is a centered polyomino.

Also note that for a non-centered polyomino if $k < l$ (where k and l is the first left anchor and the last right anchor positions, respectively), a vertically mirrored non-centered hv-convex polyomino exists with the same horizontal projection and with $k > l$.

The following algorithm called *GenCentered* generates centered polyominoes (see Algorithm 11). Figure 3.11 shows an example of generating centered polyominoes.

Algorithm 11 *GenCentered*

```
Require: Projection values $h_1, \ldots, h_m$ with $K \geq L$, and ordinal number $b$
Ensure: Strip positions $s_1, \ldots, s_m$
  $\overline{b} \leftarrow b \bmod \overline{S}_L(H)$,        $b \leftarrow \lfloor b/\overline{S}_L(H) \rfloor$
  $(s_1, \ldots, s_L) \leftarrow GenUpperStack(h_1, \ldots, h_L, \overline{b})$
  $(s_K, \ldots, s_m) \leftarrow GenLowerStack(h_K, \ldots, h_m, b)$
  $s_{L+1} \leftarrow 1, \quad \ldots, \quad s_{K-1} \leftarrow 1$
  return $s_1, \ldots, s_m$
```

Lemma 3.5.6 *For a given vector* $H = (h_1, \ldots, h_m)$ GenCentered *generates different centered polyominoes for different values* $0 \leq b \leq \overline{S}_L(H) \cdot \underline{S}_K(H) - 1$, *where* $\overline{S}_L(H) \cdot \underline{S}_K(H)$ *is the number of possible centered polyominoes with horizontal projection* H.

Proof
The proof is trivial due to the properties of the centered polyominoes. □

Finally, we describe *GenAll* to generate the b-th solution out of all the possible hv-convex polyominoes with the given horizontal projection (see Algorithm 12). In the algorithm,

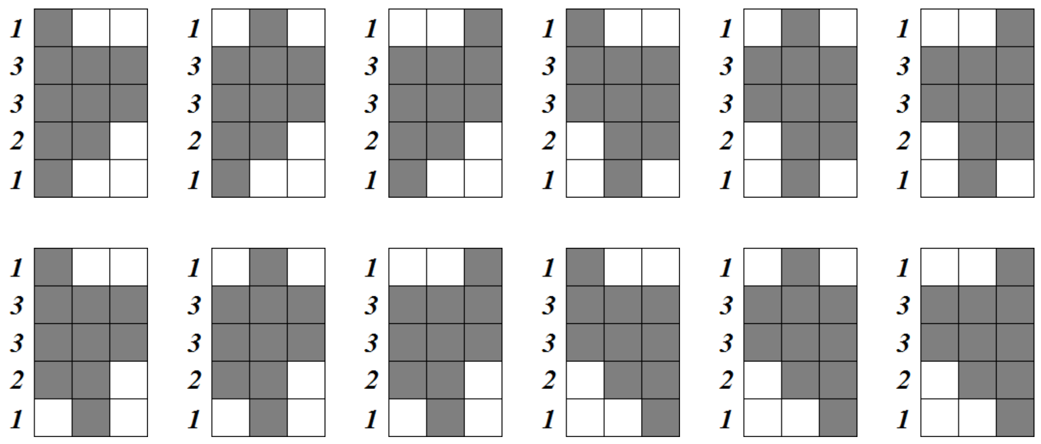


Figure 3.11: Example of generating centered polyominoes from the horizontal projection $H = (1, 3, 3, 2, 1)$. The values of b are going from 0 to 11, respectively. Note that $k = 2$ and $k = 3$ are both left and right anchors. Also note that vertical mirroring does not bring any new solutions.

- the number of centered solutions is 0 if $K < L$ (see the properties of centered polyominoes), and $\overline{S}_L(H) \cdot \underline{S}_K(H)$ otherwise,
- the number of non-centered solutions for fixed $k < l$ is $\overline{S}_{k-1}(H) \cdot (h_k - h_{k-1}) \cdot P_{k,l}(H) \cdot (h_l - h_{l+1}) \cdot \underline{S}_{l+1}(H)$ if $K < L$, and $\overline{S}_{k-1}(H) \cdot (h_k - h_{k-1}) \cdot \big(P_{k,l}(H) - 1\big) \cdot (h_l - h_{l+1}) \cdot \underline{S}_{l+1}(H)$ otherwise,
- the number of centered solutions for fixed $k < l$ is 0 if $K < L$ (again, see the properties of centered polyominoes), and $\overline{S}_{k-1}(H) \cdot (h_k - h_{k-1}) \cdot (h_l - h_{l+1}) \cdot \underline{S}_{l+1}(H)$ otherwise,
- *FlipMatrix* flips the positions as a vertical matrix flip, i.e., $s_i = n - s_i - h_i + 2$ for $i = 1, \dots, m$, where n is the number of columns, $n = \max_i \{s_i + h_i - 1\}$.

Theorem 3.3 *For a given vector $H = (h_1, \dots, h_m)$* GenAll *generates different hv-convex polyominoes for different values $0 \leq b \leq P - 1$, where P is the number of possible hv-convex polyominoes with horizontal projection H. The algorithm runs in $O(m^2)$ time.*

Proof

The algorithm *GenAll* first enumerates all the centered polyominoes (if there is any), then enumerates all the non-centered polyominoes with $k < l$ and vertically mirrors the solution if necessary, depending on the parity of the ordinal number b. The separation of centered and non-centered cases is necessary since the set of centered polyominoes is closed with respect to vertical mirroring.

The number of centered polyominoes is either $\overline{S}_L(H) \cdot \underline{S}_K(H)$ or 0 (depending on the relation of K and L). If the ordinal number b is less than the number of centered polyominoes then the algorithm returns with the b-th centered polyomino.

Algorithm 12 *GenAll*

```
Require: Projection values $h_1, \dots, h_m$ and ordinal number $b$
Ensure: Strip positions $s_1, \dots, s_m$
  if $b <$ #centered soultions then
    $(s_1, \dots, s_m) \leftarrow$ GenCentered$(h_1, \dots, h_m, b)$
    return $s_1, \dots, s_m$
  else
    $b \leftarrow b -$ #centered soultions
  end if
  isflipped $\leftarrow b \pmod 2$,        $b \leftarrow \lfloor b/2 \rfloor$
  for $k = 1 \to K$ do
    for $l = L \to m$ do
      if $b <$ #non-centered solutions for fixed $k$ and $l$ then
        $b \leftarrow b +$ #centered solutions for fixed $k$ and $l$
        $(s_1, \dots, s_m) \leftarrow$ GenAnchor$(h_1, \dots, h_m, k, l, b)$
        if isflipped $= 1$ then
          $(s_1, \dots, s_m) \leftarrow$ FlipMatrix$(s_1, \dots, s_m)$
        end if
        return $s_1, \dots, s_m$
      else
        $b \leftarrow b -$ #non-centered solutions for fixed $k$ and $l$
      end if
    end for
  end for
```

Otherwise (if b is equal or greater than that value) the algorithm decreases b with the number of centered polyominoes.

The rest of the algorithm deals with non-centered polyominoes with $k < l$, therefore it divides the (already decreased) ordinal number b by 2, and it will vertically mirror the solution depending of the remainder of this division (it correspods to the case $k > l$).

For every k and l, the algorithm checks if the ordinal number b is less than the number of non-centered cases with those k and l. If b is less than that value, then the algorithm found the anchor positions k and l, and *EnumAnchor* generates the b-th solution. Otherwise, b is decreased and the iteration goes on. Note that according to Lemma 3.5.5 and the properties of the centered polyominoes, the solutions of *EnumAnchor* for small ordinal numbers are centered polyominoes (if there is any). Therefore, the ordinal number has to be increased by the number of centered poliominoes in order to skip them. Finally, the algorithm vertically flips the solution if necessary (see the argument above).

It is easy to see that the runtime of *GenUpperStack*, *GenLowerStack*, *GenPara*, *GenAnchor*, and *GenCentered* is $O(m)$. The number of centered and non-centered solutions for all k and l can be calculated in advance in $O(m^2)$ time. The procedure *GenAnchor* (possibly together with *Flipmatrix*) is only called once during the process (as the last step of the generation). This means that the worst case time

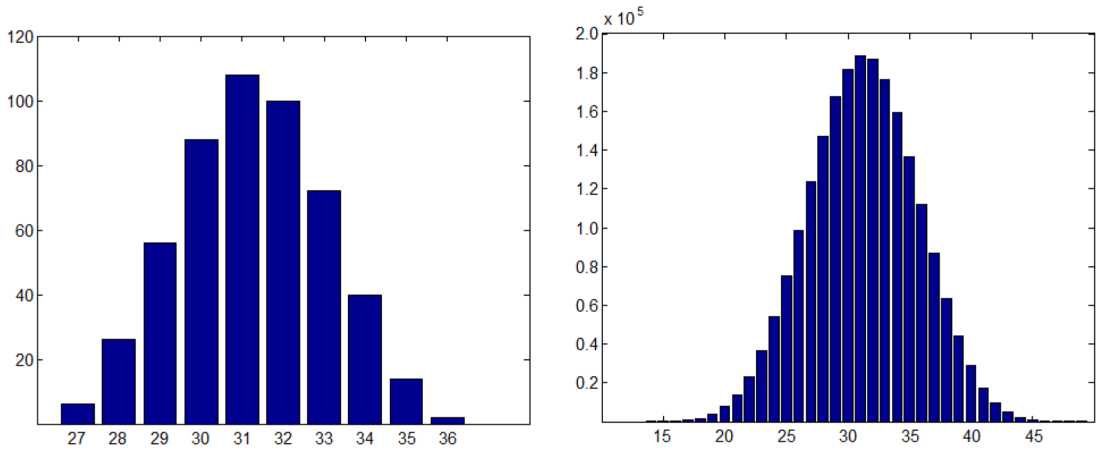


Figure 3.12: Examples of the distribution of the solutions. Horizontal axis shows the width of the solutions (number of columns), vertical axis shows the number of solutions. The projections are $(2, 4, 2, 9, 1, 10, 4, 1, 4, 8)$ with 512 solutions in total, and $(7, 3, 3, 7, 6, 5, 7, 8, 6, 6)$ with 2 154 600 solutions in total, respectively.

complexity of *GenAll* is $O(m^2)$. $\square$

3.5.2 Fixed Number of Columns

The Algorithm *GenAll* can be easily modified to generate polyominoes with fixed number of columns. The width of an hv-convex polyomino is determined by its parallelogram polyomino, since for fixed anchors the change of the upper and lower stack polyominoes has no effect on the number of columns. Such a modified algorithm could use a different version of *GenPara* in *GenAnchor* for generating parallelogram polyomines with fixed number of columns. Lemma 3.4.3 gives a recursive formula to calculate the number $P(p_1, \ldots, p_r, n)$ of parallelogram polyominoes with n columns, and having the horizontal projection $(p_1, \ldots, p_r)$. Using that recurrence one can provide a dynamic programming algorithm for enumerating all the solutions in a similar way as seen in *GenPara* and *GenAll*. Note that due to the minimality property of centered polyominoes and the fixed number of columns, Lemma 3.5.3 is not needed in the modifed version of *GenPara*.

We also studied how the number of possible hv-convex polyominoes varies depending on the number of the prescribed columns. Figure 3.12 shows just two examples of our general observation that the histogram of the number of solutions follows generally a normal-like distribution. This information could also be exploited in the future, to design more sophisticated reconstruction algorithms for the class of hv-convex polyominoes.

3.6 Reconstructing Canonical hv-Convex Images

The reconstruction of hv-convex 8-connected but not 4-connected images, as well as canonical hv-convex images from two projections was previously deeply studied. In [11], the authors examined the problem of reconstructing a discrete binary image satisfying some convexity conditions from its two orthogonal projections. They also showed that determining the existence of an h-convex (v-convex) polyomino or arbitrary binary image with connected rows (columns) having assigned orthogonal projections is an NP-complete problem. The authors of [26] showed that reconstructing a special class of centered hv-convex polyominoes, which contains a row whose length equals the total width of the object, can be solved in linear time, if the horizontal and vertical projections are given. In [54] the reconstruction algorithms and complexity results are summarized in the case of hv-convex images, hv-convex polyominoes, hv-convex 8-connected images, and directed h-convex images. Moreover, it is shown that certain h-convex images are uniquely reconstructible with respect to the row and column sum vectors. In [8], the authors compared three reconstruction algorithms used for reconstructing hv-convex 8-connected binary images from two projections. The authors of [23] proposed an algorithm for reconstructing a binary image satisfying some convexity conditions from two projections, with a worst case complexity of $O\big(mn\log(mn)\min\{m^2,n^2\}\big)$ for an image with size of $m\times n$. In some special cases they gave an upper bound for the complexity of the algorithm as $O\big(mn\log(mn)\big)$. The authors of [7] introduced an algorithm for reconstructing 8-connected but not 4-connected hv-convex binary images with worst case complexity of $O(mn\min\{m,n\})$.

In this section, we show that reconstructing an hv-convex canonical image with minimal width from the horizontal projection is possible in $O(m)$ time, where m is the size of the horizontal projection vector. First, we propose an algorithm that reconstructs such an image, and show that the resulted image is always 8-connected. This algorithm can be easily extended to reconstruct general hv-convex images with arbitrary width in the same running time.

3.6.1 Properties of the Reconstruction

We set the following reconstruction problem:

Reconstruction-Can-hv(H,k)

Instance: *A vector* $H=(h_1,\dots,h_m)\in\mathbb{N}^m$*, and an integer* k $(1\le k\le m)$.

Task: *Construct a canonical hv-convex image* F *with exactly* k *number of 4-connected components such that* $\mathcal{H}(F)=H$*, and* $w(F)$ *is minimal.*

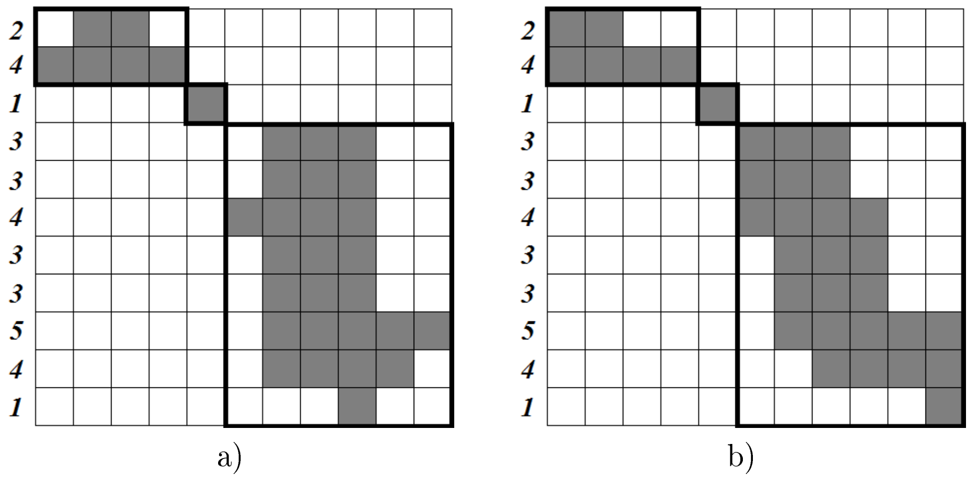


Figure 3.13: (a) A minimal width canonical hv-convex image F with the same projection and component number as of Fig. 3.2; (b) replacing each of the components by the result of a *GreedyRec* reconstruction, results in F^*. Here, $w(F) = w(F^*) = 11$. Note that F^* is 8-connected.

Figure 3.13a shows an example of such a reconstructed image.

It is easy to see that if H contains only positive integers, then a solution of the above problem, and thus one with a minimal width also exists. We will show that finding a solution can be done in $O(m)$ time, and that the solution is always 8-connected and unique.

First, the following lemma provides an important property of the components of a canonical hv-convex image.

Lemma 3.6.1 *Let F be a solution of the* Reconstruction-Can-hv(H,k) *problem with the components $F_1, \ldots, F_k$. Let G_i be the result of the* GreedyRec *algorithm (see again Algorithm 7 in Section 3.3) from the input vector $\mathcal{H}(F_i)$, where $1 \le i \le k$. Then $w(F_i) = w(G_i)$, and replacing F_i with G_i in F for some i yields also a (not necessarily different) solution F^* of the* Reconstruction-Can-hv(H,k) *problem.*

Proof
Clearly, the number of rows of F_i equals the number of rows of G_i, and both images are hv-convex polyominoes. Due to Theorem 3.1, $w(G_i) \le w(F_i)$. Replacing F_i with G_i in F yields an hv-convex image F^* with k 4-connected components, too. Since F is of minimal width, $w(F_i) \le w(G_i)$ (there are no zero-columns in F). Hence, F^* is also of a minimal width canonical hv-convex image with k components and with the same horizontal projection, thus it is a solution of the Reconstruction-Can-hv(H,k) problem. Figure 3.13 shows an example of such an exchange.□

As a consequence, it is enough to search for a solution F^* of the Reconstruction-Can-hv(H,k) problem, where each of the components F_i^* is provided by the

GreedyRec algorithm from the corresponding part of H. The next lemma shows that F^* is 8-connected.

Lemma 3.6.2 *A binary image G obtained by* GreedyRec *has a black pixel both in its top left and bottom right corners, i.e., if the size of G is $m \times n$, then $g_{1,1} = 1$ and $g_{m,n} = 1$.*

Proof
The black color of the pixel in the top left corner is a consequence of Step 1 of *GreedyRec*. The black color of the pixel in the bottom right corner is a consequence of Step 2 of the algorithm (see previous Fig. 3.4). □

As a direct consequence, the solution F^* of the RECONSTRUCTION-CAN-HV(H,k) problem (where each component is provided by *GreedyRec*) is 8-connected (again, see Fig. 3.13b for example).

Before the following lemma, we introduce the concept of *breaking*.

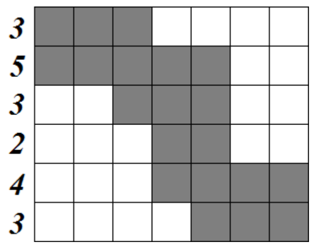
Definition 3.1 *Let G be a binary image with the horizontal projection $H = (h_1, \ldots, h_m)$, and let ${}_cG$ denote the binary image that is derived from G by shifting the $(c+1)$-st, $(c+2)$-nd, $\ldots$, m-th strip to the right with* $\min\{h_c, h_{c+1}\}$ *positions (where $1 \le c < m$). In that case we say that we got ${}_cG$ by* breaking *G at the* breakpoint *c. We say that the c is* minimal *if* $c = \arg\min_{1 \le j < m} \{ \min\{h_j, h_{j+1}\} \}$.

Note that the minimal breakpoint is not necessarily unique. Figure 3.14 shows an example of breaking.

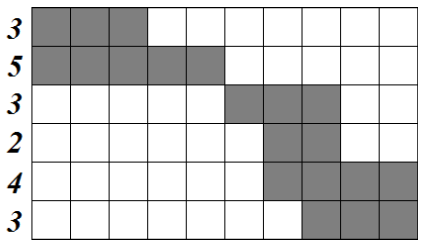
Generally breaking a binary image does not necessarily modify the number of the 4-connected components. The following lemma describes a property of breaking special hv-convex binary images.

Lemma 3.6.3 *Let F^* be an (8-connected) canonical hv-convex image with horizontal projection H, and with exactly two components G_1 and G_2, where both components are obtained by the* GreedyRec *algorithm from the corresponding part of H. Let r denote the number of rows of G_1. Then $F^* = {}_rG$, where G is the result of the* GreedyRec *algorithm from H, and* $w(F^*) = w({}_rG) = w(G) + \min\{h_r, h_{r+1}\}$.

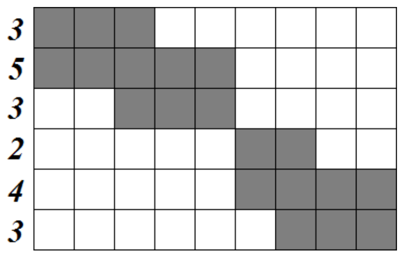
Proof
Let G be the result of the *GreedyRec* algorithm from H. Then ${}_rG$ can be decomposed into two sub-images, one consisting of the pixels of ${}_rG$ (and their smallest containing rectangle) of the first r rows, and another one consisting of the pixels of the last $m - r$ rows (and their smallest containing rectangle), where m is the number of elements in H. Since in *GreedyRec* the position of the i-th strip only depends on the position of the $(i-1)$-st strip (and the corresponding values of H), it follows that both sub-images are the result of the *GreedyRec* algorithm from the corresponding part of H. By the definition of breaking, the second sub-image is shifted to the right with $\min\{h_r, h_{r+1}\}$ positions. We show that the sub-images are



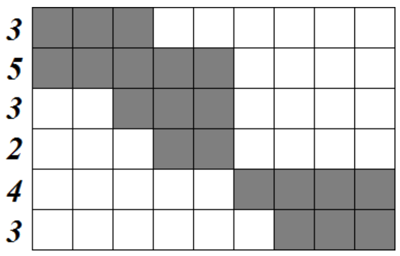
(a) G



(b) ${}_2G$



(c) ${}_3G$



(d) ${}_4G$

Figure 3.14: Examples of breaking: (a) an obtained image G of *GreedyRec*; (b) G is broken at the breakpoint $c = 2$; (c) G is broken at the minimal breakpoint $c = 3$; (d) G is broken at another minimal breakpoint $c = 4$.

actually 4-connected components, which are 8-connected but not 4-connected to each other.

If $h_r \leq h_{r+1}$, then the position of the $(r+1)$-st strip is the same as the position of the r-th strip in G (as described in *GreedyRec)*. In ${}_rG$, the $(r+1)$-st strip is shifted to the right with h_r positions. Therefore, the r-th and the $(r+1)$-st strips are 8-connected but not 4-connected.

If $h_r > h_{r+1}$, then the $(r+1)$-st strip is shifted to the right relative to the r-th strip with $h_r - h_{r+1}$ positions in G. In ${}_rG$, it is shifted further with h_{r+1} positions. The overall amount of shifting is h_r, and again, the r-th and the $(r+1)$-st strips are 8-connected but not 4-connected.

As a consequence, ${}_rG$ is an 8-connected canonical hv-convex image with two components, furthermore, $F^* = {}_rG$, and due to the shifting, $w({}_rG) = w(G) + \min\{h_r, h_{r+1}\}$. □

The following consequence of Lemma 3.6.3 plays an important role in our proposed algorithm.

Corollary 3.1 *For each canonical hv-convex F^* image with k component, there exists a series of $k-1$ number of breakpoints in G, such that the obtained image is equivalent to F^*, where G is the result of the* GreedyRec *algorithm from $\mathcal{H}(F^*)$.*

3.6.2 Algorithm for Reconstructing Canonical hv-Convex Images

We are now ready to provide an algorithm, *CanonicalRec* which gives an 8-connected solution of the RECONSTRUCTION-CAN-HV(H,k) problem in $O(m)$ time (see Algorithm 13). Note that the algorithm always provides a solution if H contains only positive integers, and $1 \leq k \leq m$. Furthermore, if $k = 1$ then the whole image is a single 4-connected component, while for $k = m$, each of the strips itself is a 4-connected component.

Algorithm 13 *CanonicalRec*

```
Require: A vector of projection values $H = (h_1, \ldots, h_m)$ and a number $k$ of
   components $(1 \leq k \leq m)$
Ensure: A minimal-width 8-connected canonical $hv$-convex image $F^*$ with $k$ components
   1) Find $c_1, c_2, \ldots, c_{k-1}$ $(1 \leq c_i < m$, and $c_i \neq c_j$ if $i \neq j)$ such that
   $\sum_{c_i} \min\{h_{c_i}, h_{c_i+1}\}$ is minimal
   2) Create image $G$ with GreedyRec from $H$
   3) Create image $F^*$ by using the series of $c_i$ as breakpoints in $G$
   return $F^*$
```

Theorem 3.4 CanonicalRec *is correct and has a running time of* $O(m)$.

Proof

If $k = 1$, then *CanonicalRec* is equivalent to *GreedyRec*, which constructs an hv-convex image containing one 4-connected component with minimal width, therefore *CanonicalRec* is correct. For the rest of the proof assume $k > 1$, thus the series of c_i breakpoints in Step 1 is non-empty. Without loss of generality, we can suppose that the c_i breakpoints in Step 1 are ordered in a way that the series of $\min\{h_{c_i}, h_{c_i+1}\}$ values is non-decreasing.

According to Lemma 3.6.3, the obtained F^* is a canonical hv-convex image with k number of components and with the horizontal projection $\mathcal{H}(F^*) = H$. Furthermore, F^* is 8-connected according to the consequence of Lemma 3.6.2. We only have to prove that F^* is of minimal width.

Assume to the contrary that there exist an F' solution such that $w(F') < w(F^*)$. Note that according to Corollary 3.1, such an F' can also be given with a series of breakpoints $c'_1, c'_2, \ldots, c'_{k-1}$ in G. Again, let the series of $\min\{h_{c'_i}, h_{c'_i+1}\}$ values be ordered in a non-decreasing way.

Since F' cannot be a result of the *CanonicalRec* algorithm, $\sum_{c'_i} \min\{h_{c'_i}, h_{c'_i+1}\}$ is not minimal. Since both the $\min\{h_{c_i}, h_{c_i+1}\}$ and the $\min\{h_{c'_i}, h_{c'_i+1}\}$ series are ordered by magnitude, there exists a smallest j index for which $\min\{h_{c'_j}, h_{c'_j+1}\} > \min\{h_{c_j}, h_{c_j+1}\}$.

Let $\widehat{F}$ be the image which is constructed from F' such that the (c'_j+1)-st, (c'_j+2)-nd, …, m-th strip is shifted to the left with $\min\{h_{c'_j}, h_{c'_j+1}\}$ positions. According to Lemma 3.6.3, $\widehat{F}$ is canonical, hv-convex with $k-1$ components, and $w(\widehat{F}) = w(F') - \min\{h_{c'_j}, h_{c'_j+1}\}$. Now, using c_j as a breakpoint in $\widehat{F}$, the obtained $\widetilde{F}$ image is canonical, hv-convex with k components, and $w(\widetilde{F}) = w(\widehat{F}) + \min\{h_{c_j}, h_{c_j+1}\} = w(F') - \min\{h_{c'_j}, h_{c'_j+1}\} + \min\{h_{c_j}, h_{c_j+1}\}$. Since $\min\{h_{c'_j}, h_{c'_j+1}\} > \min\{h_{c_j}, h_{c_j+1}\}$, it follows that $w(\widetilde{F}) < w(F')$, therefore $\widetilde{F}$ is a solution of the RECONSTRUCTION-CAN-HV(H,k) problem with less number of columns, which is a contradiction to the width-minimality of F'. Therefore, the result of *CanonicalRec* is correct.

As of Step 1, finding the breakpoints $c_1, c_2, \ldots, c_{k-1}$ is equivalent to selecting the $k-1$ smallest elements from the series $\min\{h_1, h_2\}$, $\min\{h_2, h_3\}$, $\ldots$, $\min\{h_{m-1}, h_m\}$ in any order. The $(k-1)$-st smallest element can be found with the *Median of medians* algorithm in $O(m)$ time [18], and after that the first $k-1$ smallest elements can be identified simply by scanning at most two times the series of possible breakpoints. Overall, finding the breakpoints has a run-time of $O(m)$. Step 2 and Step 3 can be done with $O(m)$ moves according to Theorem 3.1. The overall run-time of the *CanonicalRec* algorithm is $O(m)$. □

3.6.3 Reconstructing General hv-Convex Images

Similarly to the modified version of *GreedyRec* described in the end of Section 3.3, one can modify the *CanonicalRec* algorithm in a way that the obtained image has n number of columns, where n can be any value between the minimal value provided by *CanonicalRec* and the maximal value of $\left(\sum_{i=1}^{m} h_i\right) - m + k$. Moreover, the horizontal relative order of the 4-connected components of the result can be arbitrarily changed, still providing hv-convex binary images with the same horizontal projection. Hence, if there are at least two 4-connected components, it is possible to give also a solution which is hv-convex but not connected (see Fig. 3.15 for example).

3.7 Summary

The reconstruction of hv-convex polyominoes from few projections is an extensively studied problem in discrete tomography. Several algorithms exist to solve this task from two projections. From the viewpoint of testing the efficacy of those (and of more general reconstruction) algorithms in the average case, the reconstruction, enumeration and random generation of hv-convex polyominoes according to several parameters is an important issue.

In this chapter, we showed how to reconstruct hv-convex polyominoes from a given horizontal projection with minimal number of columns in linear time. This algorithm can easily be extended to give a solution with any required number of columns, if such a solution exists. We also gave formulas for counting all possi-

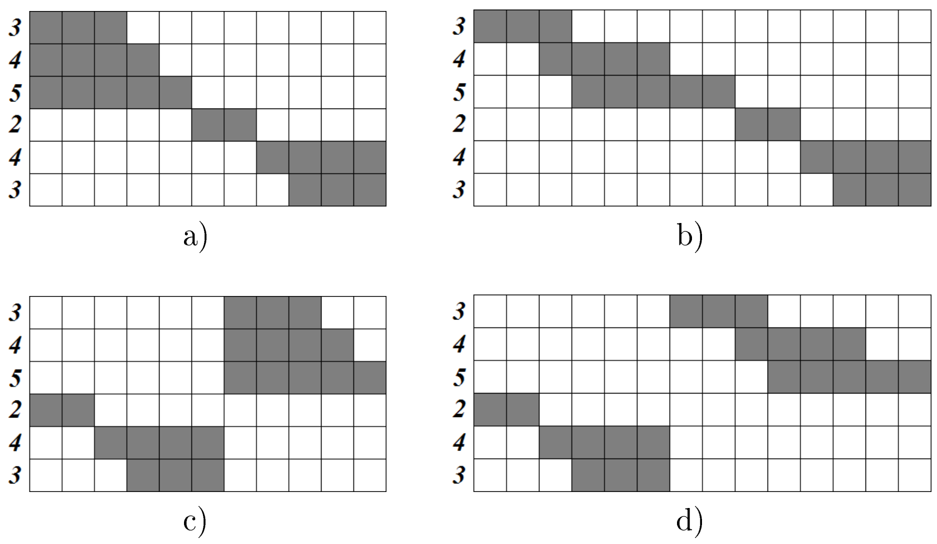


Figure 3.15: (a) The reconstruction result of the *CanonicalRec* algorithm with $k = 3$, where $w(F_1) = 11$; (b) the modified result with $w(F_2) = 14$; (c-d) modified results which are not 8-connected.

ble solutions, one for any number of columns, and another one for fixed number of columns. Furthermore, we provided an algorithm that generates an *hv*-convex polyomino with a prescribed horizontal projection from a uniform distribution. In the end of this chapter we provided a fast polynomial-time algorithm that can reconstruct canonical *hv*-convex images with certain number of 4-connected components and with minimal number of columns from a given horizontal projection in $O(m)$ time. We also showed that the results of the proposed algorithm are always 8-connected. Furthermore, the algorithm can be extended in a straightforward way to give a solution with any required number of columns, if such a solution exists. Even more important is that a further extension of the algorithm yields a polynomial-time reconstruction algorithm for the general class of *hv*-convex binary images, when only one projection is given. When both the horizontal and the vertical projections are given the problem is known to be NP-complete.

The findings of this research have been published in two conference proceedings [39, 44], and one journal paper [43].

Chapter 4

Morphological Skeleton as Additional Information for the Reconstruction

4.1 Introduction

In binary tomography, as described in Section 1.1, due to the small number of available projections the reconstruction is usually very underdetermined. Therefore, additional information is needed to reduce the number of possible solutions. However, in certain cases the reconstruction can be NP-hard. Determining the computational complexity of different variants of the main problem is essential, however, the complexity highly depends on the additional requirements the image to be reconstructed must satisfy.

In the most simple case only two projections are availble[1]. Without further restrictions, reconstruction from those projections can be performed in $O(mn + n \log n)$ time for an image with size of $m \times n$, although the number of solutions can be extremely high [64]. Using additional information such as h-convexity or v-complexity (or even both) can cause the reconstruction problem to be NP-complete in general [70]. 4-connectedness alone also yields NP-completeness [70], even with h- or v-complexity together [11]. However, in [11] and [26], it is shown that there is a polynomial-time reconstruction algorithm if the image to be reconstructed satisfies both hv-convexity and 4-connectedness. On the other hand, using more than two projections can, again, make the problem NP-complete in general [32]. Chapter 3 gives additional references about the previous results in field.

In this chapter we study the reconstruction from an additional shape descriptor, the so-called morphological skeleton. The skeleton is a region-based shape descrip-

[1]In extreme cases, as we have shown in Chapter 3, only one.

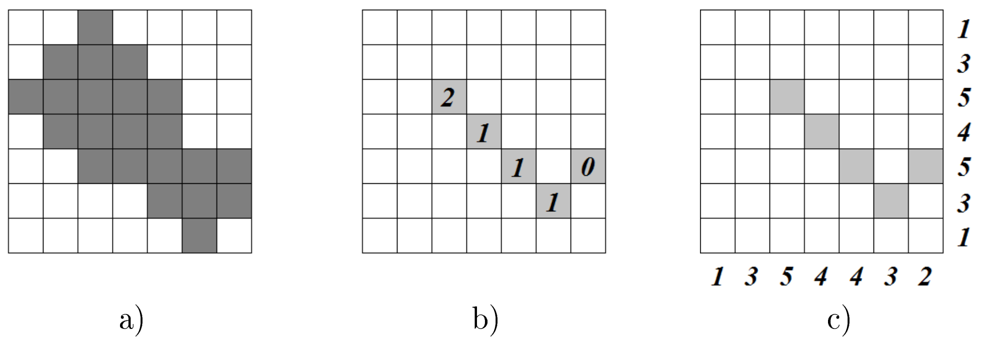


Figure 4.1: Examples of two kinds of reconstruction problems: (a) the image F to be reconstruced; (b) if the skeletal label is known for each $p \in \mathcal{S}(F, Y)$, F is uniquely reconstructable by (1.11) of Section 1.4; (c) the considered problem is to reconstruct F from $\mathcal{S}(F, Y)$ and the two projections.

tor which represents the general form of binary objects. We provide a detailed explanation of the morphological skeleton in Section 1.4. In this chapter, we deal with the reconstruction problem in which the entire morphological skeleton (instead of the individual skeletal subsets) and one or two projections of the original image are known. Figure 4.1 gives an example of the reconstruction problem. A practical application of morphological skeleton and image reconstruction can be used in the field of data compressing, in a similar way of [59].

First, we show that the reconstruction of 4-connected images (polyominoes) from the horizontal and vertical projections is still NP-complete, even if the morphological skeleton is given. Moreover, we show that the solution is not necessarily unique. Despite the theoretical drawback, under some circumstances an acceptable image quality can be achieved. In the reconstruction process a priori knowledge is often incorporated into an energy function, thus the reconstruction task becomes equivalent to a function minimization problem. There are various methods to solve that kind of problems [35, 63, 66]. In this chapter we show how to use Simulated Annealing (SA) for the binary reconstruction problem using morphological skeleton and two projections in one case, and only one projection in the other. We propose three variants of a method to solve the above problem, based on parametric SA reconstruction.

A related issue, as mentioned in Chapter 3, is the uniqueness of the reconstruction. In the end of this chapter we study the uniqueness of the reconstruction of certain type of 4-connected hv-convex images, using two projections and the morphological skeleton. We show that the unique reconstructability of a certain parametric subclass of hv-convex binary images is strongly connected to its parameters.

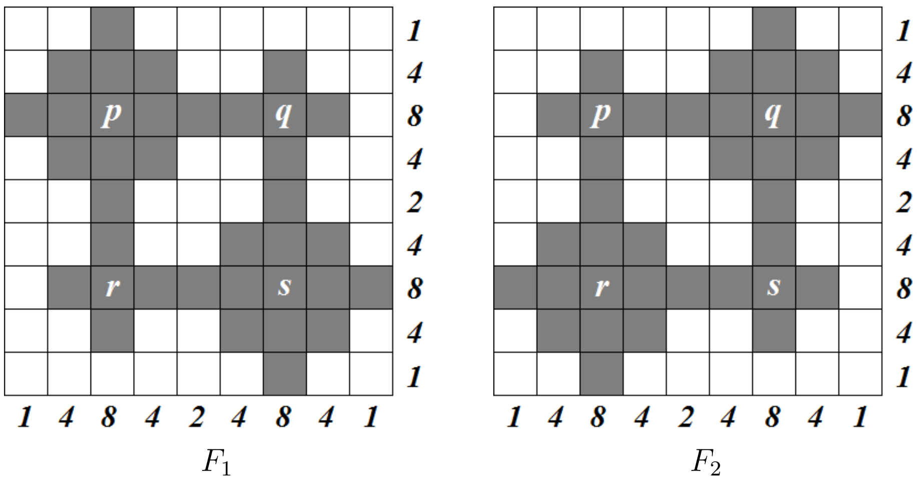


Figure 4.2: Two different images F_1 and F_2 having the same projections and morphological skeleton, where $\mathcal{S}(F_1, Y) = \mathcal{S}(F_2, Y) = \{p, q, r, s\}$.

4.2 Reconstructing Polyominoes is NP-complete

In this section we prove the NP-completeness of the problem where the horizontal and the vertical projections and the morphological skeleton of the polyomino to be reconstructed are given. Recalling the definitions from Chapter 3, two positions $p = (i_p, j_p)$ and $q = (i_q, j_q)$ in a binary image are said to be 4-adjacent if $|i_p - i_q| + |j_p - j_q| = 1$. The positions p and q are 4-connected if there is a sequence of distinct black pixels $p_0 = p, \ldots, p_k = q$ in the binary image such that p_l is 4-adjacent to p_{l-1}, respectively, for each $l = 1, \ldots, k$. A binary image F is 4-connected and is called a polyomino if any two different object points in F are 4-connected. Note that in our definition holes are allowed inside a polyomino.

Now we can give the formal definition of our problem.

Problem. Skel Rec Poly
Instance. $H \in \mathbb{N}^m$, $V \in \mathbb{N}^n$ vectors and $S \subset \mathbb{Z}^2$ binary image.
Question. Does there exist a polyomino F of size $m \times n$ such that $H = \mathcal{H}(F)$, $V = \mathcal{V}(F)$ and $S = \mathcal{S}(F, Y)$, where Y is given by (1.12)?

Figure 4.2 shows that there can be more than one solution. Furthermore, we prove that finding even one solution, if exists, is generally NP-complete. Our proof is based on [70], where the reconstruction of polyominoes from vertical and horizontal projections was examined. Our reduction is done from the following version of the NP-complete problem published in [33]. The following problem is known to be NP-complete.

Problem. Three Partition
Instance. Positive integers $a_1, \ldots, a_{3k}$ that are encoded in unary and that fulfil

the condition[2] $\sum_{i=1}^{3k} a_i = k(2B+1)$ for some integer B.

Question. Does there exists a partitioning of $a_1, \ldots, a_{3k}$ into k triples such that the elements of every triple add up to exactly $2B+1$?

For example, if $B = 6$, $k = 3$ and the positive a integers are 2, 3, 4, 4, 4, 5, 5, 6, 6, the answer is yes, since the partition $\mathcal{I} = \big\{\{1,6,8\},\{2,3,9\},\{4,5,7\}\big\}$ gives us a solution $2+5+6 = 3+4+6 = 4+4+5 = (2B+1) = 13$.

We will give the transformation from a Three Partition instance to a Skel Rec Poly instance and declare all the necessery lemmas for the proof. Frow now, k and B are fixed integers for a particular Three Partition instance $a_1, \ldots, a_{3k}$.

The main idea behind the transformation is to describe a Three Partition instance as a 4-connected image, parts of which resemble a permutation matrix with i number of columns. Each column in the permutation matrix corresponds to a_i number of contiguous elements of the image, whose positions describe which partition contains that particular a_i in the Three Partition problem. If the Three Partition instance has a solution, then a valid permutaton matrix, thus a polyomino satisfying the morphological skeleton and the projections can be easily constructed. On the other hand, if there is a polyomino with the given morphological skeleton and the projections – hence, Skel Rec Poly has a solution – then that image must encode a valid permutation matrix, which provides a solution to the original Three Partition instance. Figure 4.3 shows the core idea of the transformation. For a more detailed example, see Fig. 4.5 and 4.6. We explain the construction and the function of those images later.

For two arbitrary vectors $\vec{a} \in \mathbb{N}^n$ and $\vec{b} \in \mathbb{N}^m$ let "$\diamond$" denote the concatenation:

$$\vec{a} \diamond \vec{b} = (a_1, \ldots, a_n, b_1, \ldots, b_m) \ . \tag{4.1}$$

For any $s \in \mathbb{N}$ let the exponentiation of a vector denote the iterative concatenation:

$$\begin{aligned} \vec{a}^s &= \vec{a} && \text{if } s = 1, \\ \vec{a}^s &= \vec{a}^{s-1} \diamond \vec{a} && \text{if } s > 1. \end{aligned} \tag{4.2}$$

Let $\vec{u}$ be a fixed vector of size $1 \times 36k$:

$$\vec{u} = \Big((0)^{11} \diamond (1)\Big)^{3k} = (0,0,0,0,0,0,0,0,0,0,0,1)^{3k} \ . \tag{4.3}$$

Let $\mathcal{C}^V$, $\mathcal{C}^{00}$, $\mathcal{C}^{01}$, $\mathcal{C}^{10}$ and $\mathcal{C}^{11}$ be fixed binary images of size 5×12 as shown in Fig. 4.4. $\mathcal{C}^V$ is called a *skeletal cell*, the others are called *object cells*. Let $\mathcal{R} = \{1, \ldots, 2B{+}1\} \times \{1, \ldots, 3k\} \times \{1, \ldots, k\}$. Finally, let $\varphi : \mathcal{R} \to \{\mathcal{C}^V, \mathcal{C}^{00}, \mathcal{C}^{01}, \mathcal{C}^{10}, \mathcal{C}^{11}\}$

[2]In the original version of Three Partition, each a_i satisfies $\frac{2B+1}{4} < a_i < \frac{2B+1}{2}$. This version is more general and thus is still NP-hard.

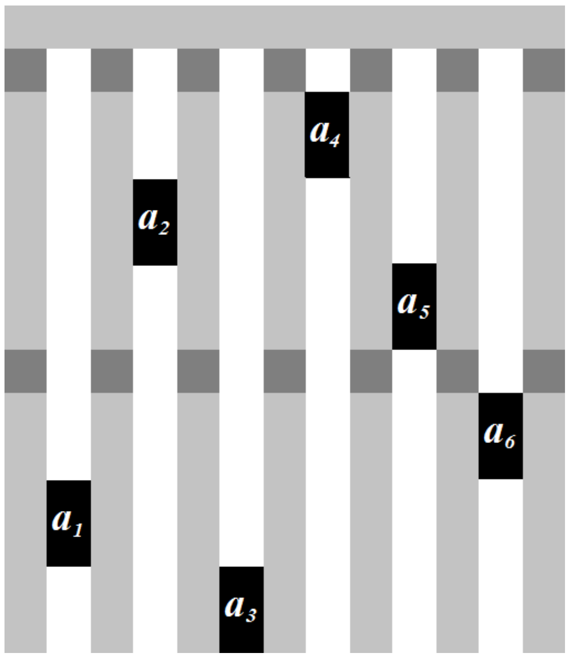


Figure 4.3: The core idea of the transformation for $k = 2$. Black pixels indicate the elements corresponding to the a_i numbers. Gray pixels indicate the parts responsible for the 4-connectedness, while dark gray pixels denote the borders between the partitions. Here, the first partition in the solution of the Three Partition instance contains (a_4, a_2, a_5), the second one is (a_6, a_1, a_3).

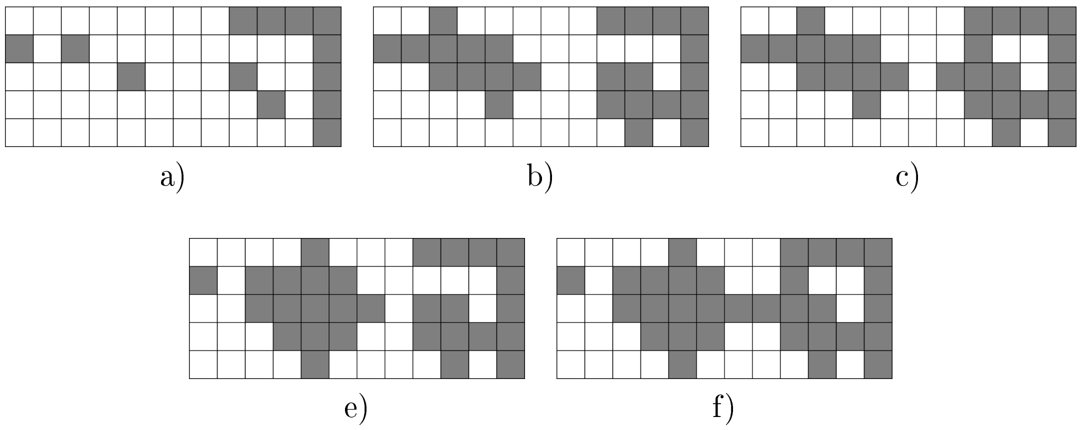


Figure 4.4: Different type of cells: (a) a skeletal cell $\mathcal{C}^V$; (b) object cell $\mathcal{C}^{00}$; (c) object cell $\mathcal{C}^{01}$; (d) object cell $\mathcal{C}^{10}$; (e) and object cell $\mathcal{C}^{11}$ (e).

and let G_φ be the following binary matrix of size $\big(k(5(2B+1)+1)+1\big) \times \big(36k+1\big)$:

$$G_\varphi = \left(\begin{array}{c|cccc}
\multicolumn{5}{c}{\vec{1}} \\ \hline
 & \multicolumn{4}{c}{\vec{u}} \\ \cline{2-5}
 & (1,1,1)\varphi & (1,2,1)\varphi & \dots & (1,3k,1)\varphi \\
 & (2,1,1)\varphi & (2,2,1)\varphi & \dots & (2,3k,1)\varphi \\
 & \multicolumn{4}{c}{\vdots} \\
 & (2B{+}1,1,1)\varphi & (2B{+}1,2,1)\varphi & \dots & (2B{+}1,3k,1)\varphi \\ \cline{2-5}
 & \multicolumn{4}{c}{\vec{u}} \\ \cline{2-5}
 & (1,1,2)\varphi & (1,2,2)\varphi & \dots & (1,3k,2)\varphi \\
 & (2,1,2)\varphi & (2,2,2)\varphi & \dots & (2,3k,2)\varphi \\
 & \multicolumn{4}{c}{\vdots} \\
 & (2B{+}1,1,2)\varphi & (2B{+}1,2,2)\varphi & \dots & (2B{+}1,3k,2)\varphi \\ \cline{2-5}
\vec{1}^T & \multicolumn{4}{c}{\vdots} \\ \cline{2-5}
 & \multicolumn{4}{c}{\vec{u}} \\ \cline{2-5}
 & (1,1,k)\varphi & (1,2,k)\varphi & \dots & (1,3k,k)\varphi \\
 & (2,1,k)\varphi & (2,2,k)\varphi & \dots & (2,3k,k)\varphi \\
 & \multicolumn{4}{c}{\vdots} \\
 & (2B{+}1,1,k)\varphi & (2B{+}1,2,k)\varphi & \dots & (2B{+}1,3k,k)\varphi
\end{array}\right), \tag{4.4}$$

where $(r,i,j)\varphi$ is called a *cell* of G_φ. We say that $(r,i,j)\varphi$ and $(r',i',j')\varphi$ are in the same *cell block* if $j = j'$, in the same *cell column* if $i = i'$ and in the same *cell row* if $r = r'$, and also $j = j'$. $(r,i,j)\varphi$ is in the j-th cell block, in the i-th cell column and in the r-th cell row of the j-th cell block.

Now we give the transformation from a THREE PARTITION instance to a SKEL REC POLY instance.

Let $m = k(5(2B+1)+1)+1$, $n = 36k+1$ and $H \in \mathbb{N}^m$ be the following vector:

$$H = (n) \diamond \Big((3k+1) \diamond H_1 \diamond H_2^{B-1} \diamond H_1 \diamond H_2^{B-1} \diamond H_1\Big)^k , \tag{4.5}$$

where

$$H_1 = (1)^5 + \mathcal{H}(\mathcal{C}^{11}) + (3k-1) \cdot \mathcal{H}(\mathcal{C}^{00}) \tag{4.6}$$

and

$$H_2 = (1)^5 + \mathcal{H}(\mathcal{C}^{10}) + (3k-1) \cdot \mathcal{H}(\mathcal{C}^{00}) \tag{4.7}$$

Moreover, let $V \in \mathbb{N}^n$ be the following vector:

$$V = (m) \diamond V_1 \diamond V_2 \diamond \cdots \diamond V_{3k} , \tag{4.8}$$

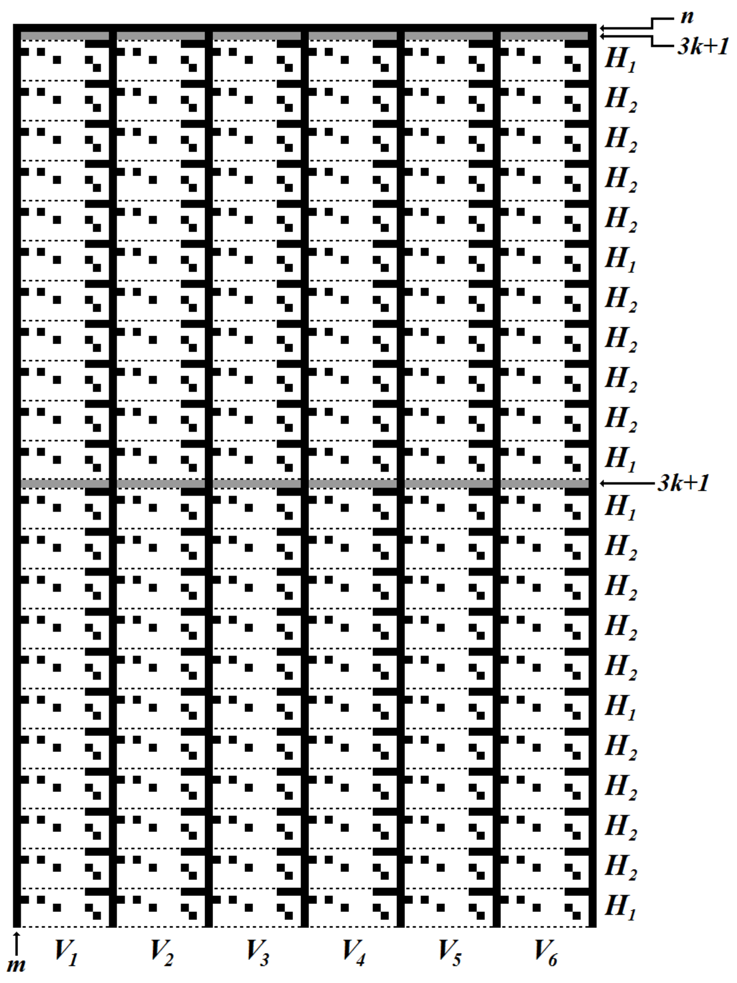


Figure 4.5: The skeletal set S and the required projections given by the transformation if $k = 2$ and $B = 5$. Black pixels denote object points. Gray pixels indicate the background points of vectors $\vec{u}$. Broken lines indicate the border of the cells. The size of S is 113×73.

where

$$\begin{aligned} V_i = (1)^{12} \quad &+ \quad k \cdot \big(\ (0)^{11} \diamond (1)\ \big) + \mathcal{V}(\mathcal{C}^{11}) + (a_i - 1) \cdot \mathcal{V}(\mathcal{C}^{10}) + \\ &+ \quad (k(2B+1) - a_i) \cdot \mathcal{V}(\mathcal{C}^{00}) \end{aligned} \tag{4.9}$$

for $i = 1, \ldots, 3k$.

Finally, let the skeletal set $S = G_{\varphi^v}$ of size $m \times n$, where $\varphi^v : \mathcal{R} \to \{\mathcal{C}^V\}$, $(r, i, j)\varphi^v = \mathcal{C}^V$ for every (r, i, j) in $\mathcal{R}$. Figure 4.5 shows an example of S and the required projections if $k = 2$ and $B = 5$.

Lemma 4.2.1 *If the* Three Partition *instance has a solution, then there exists a polyomino F of size $m \times n$ with horizontal projection H, vertical projection V, and skeletal set S.*

Proof
Let a_{j_1}, a_{j_2} and a_{j_3} be the elements in the j-th triple in the solution of the Three

Partition instance. Let $F = G_{\varphi'}$, where $\varphi' : \mathcal{R} \to \{\mathcal{C}^{11}, \mathcal{C}^{10}, \mathcal{C}^{00}\}$, and the j-th cell block has the following object cells in the j_1-th cell column:

$$(r, j_1, j)\varphi' = \begin{cases} \mathcal{C}^{11} & \text{if } r = 1 \ , \\ \mathcal{C}^{10} & \text{if } r \in \{2, \dots, a_{j_1}\} \ , \\ \mathcal{C}^{00} & \text{otherwise,} \end{cases} \tag{4.10}$$

in the j_2-th cell column:

$$(r, j_2, j)\varphi' = \begin{cases} \mathcal{C}^{11} & \text{if } r = B + 1 \ , \\ \mathcal{C}^{10} & \text{if } r \in \{a_{j_1} + 1, \dots, a_{j_1} + a_{j_2}\} \setminus \{B + 1\} \ , \\ \mathcal{C}^{00} & \text{otherwise,} \end{cases} \tag{4.11}$$

and finally in the j_3-th cell column:

$$(r, j_3, j)\varphi' = \begin{cases} \mathcal{C}^{11} & \text{if } r = 2B + 1 \ , \\ \mathcal{C}^{10} & \text{if } r \in \{a_{j_1} + a_{j_2} + 1, \dots, a_{j_1} + a_{j_2} + a_{j_3} - 1\} \ , \\ \mathcal{C}^{00} & \text{otherwise.} \end{cases} \tag{4.12}$$

Figure 4.6 shows an example, where the instance of the Three Partition problem is $a_1 = 3$, $a_2 = 3$, $a_3 = 3$, $a_4 = 4$, $a_5 = 4$, $a_6 = 5$ $(k = 2, B = 5)$.

It is easy to verify that $\mathcal{S}(\mathcal{C}, Y) = \mathcal{C}^V$ for any object cell $\mathcal{C}$. Moreover, due to the construction of G_φ, and since $(p \oplus Y) \cap S = \emptyset$ for any $p \in \mathbb{Z}^2$, it follows that $\mathcal{S}(G_\varphi, Y) = \mathcal{S}(S, Y) = S$ for any $\varphi : \mathcal{R} \to \{\mathcal{C}^V, \mathcal{C}^{00}, \mathcal{C}^{01}, \mathcal{C}^{10}, \mathcal{C}^{11}\}$. Therefore, $\mathcal{S}(F, Y) = S$.

Note that the number of $\mathcal{C}^{10}$ cells in the i-th cell column is $a_i - 1$, the number of $\mathcal{C}^{11}$ cells is one and every other cell is $\mathcal{C}^{00}$. With the first row and the first column and with the $\vec{u}$ vectors, it is easy to verify that $V = \mathcal{V}(F)$ according to (4.8).

Similarly, each cell row contains exactly one $\mathcal{C}^{10}$ or $\mathcal{C}^{11}$ – depending on the corresponding horizontal projections –, and every other cell is $\mathcal{C}^{00}$, therefore with the additional elements in F we have $H = \mathcal{H}(F)$ according to (4.5).

The 4-connectedness of $G_{\varphi'}$ is a consequence of the attributes of $G_{\varphi'}$ and the object cells: every object point in a $\mathcal{C}^{10}$ cell is 4-connected to the object points of a $\mathcal{C}^{11}$ cell (possibly through other $\mathcal{C}^{10}$ cells in the same cell column), and every object point in a $\mathcal{C}^{11}$ cell is 4-connected to the first row of F through the $\vec{u}$ vectors and other $\mathcal{C}^{11}$ cells. The object points in the $\mathcal{C}^{00}$ cells are 4-connected to the first and last columns of other cells or the first column of F. Finally, the first column of F is trivially 4-connected to the first row of F. Again, see Fig. 4.6 for example. □

Lemma 4.2.2 *If there exists a polyomino F of size $m \times n$ with horizontal projection H, vertical projection V, and skeletal set S, then a $\varphi : \mathcal{R} \to \{\mathcal{C}^{00}, \mathcal{C}^{01}, \mathcal{C}^{10}, \mathcal{C}^{11}\}$*

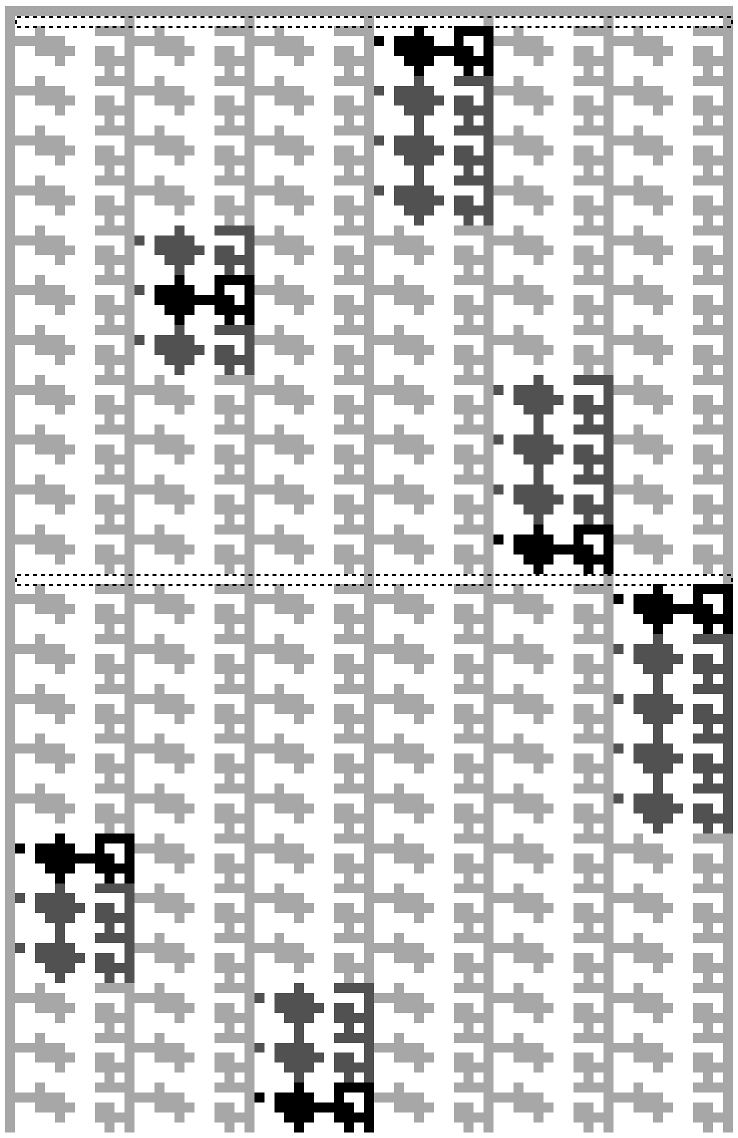


Figure 4.6: An instance of Skel Rec Poly transformed from an instance of Three Partition, with $a_1 = 3$, $a_2 = 3$, $a_3 = 3$, $a_4 = 4$, $a_5 = 4$, $a_6 = 5$. Black pixels denote the object points in cells $\mathcal{C}^{11}$, dark gray pixels denote the object points in cells $\mathcal{C}^{10}$. Light gray pixels indicate the remaining object points ($\mathcal{C}^{00}$ cells, $\vec{u}$ vectors, the first row and the first column of F). Broken lines indicate the $\vec{u}$ vectors. It is clearly visible that the first triple of the solution is (a_4, a_2, a_5), the second one is (a_6, a_1, a_3), and the sum of the triples is 11.

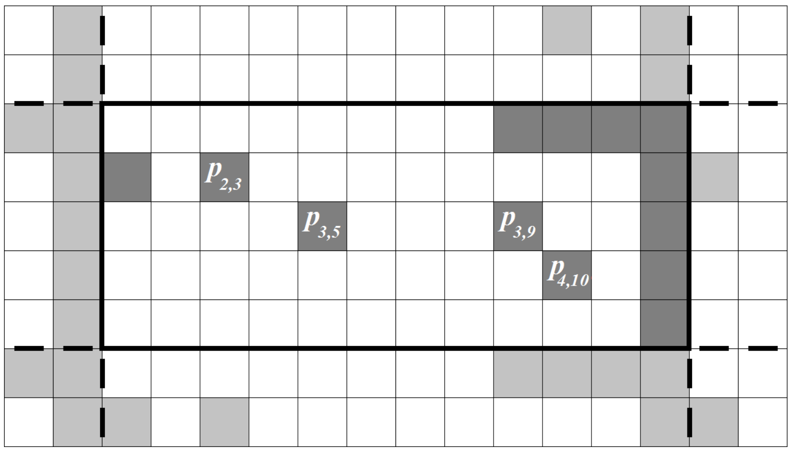


Figure 4.7: The notation of the skeletal points in $\mathcal{C}$, enclosed with thick black lines. Every skeletal point other than $p_{2,3},\ p_{3,5},\ p_{3,9},$ and $p_{4,10}$ has a skeletal label 0 due to the 4-connectedness. Broken lines indicate parts of neighbor cells.

exists such that $F = G_\varphi$.

Proof

According to (1.11), in order to determine F, we only have to find the labels of the skeletal points in S. Since $S = G_{\varphi^v}$, S is built up from cells. Let $\mathcal{C}$ be an arbitrary submatrix in F corresponding to a skeletal cell in S. Let $p_{i,j}$ be the point in the i-th row and the j-th column in $\mathcal{C}$ $(i = 1,\ldots,5,\ j = 1,\ldots 12)$.

A special case of Lemma 1.4.1 is $\kappa_p = \kappa_q$, if p and q are 4-neighbors. Note that there are lots of 4-connected skeletal points in S, see previous example of Fig. 4.5, therefore their labels are all the same. Due to the projections, it is easy to verify that those labels must be 0.

The skeletal points with the unknown labels are $p_{2,3},\ p_{3,5},\ p_{3,9},$ and $p_{4,10}$, since those points are not 4-connected to the rest of the skeletal points (see Fig. 4.7). We can establish an upper bound for each label using Lemma 1.4.1. For example, $2 = d_1(p_{2,3}, p_{2,1}) > |\kappa_{p_{2,3}} - \kappa_{p_{2,1}}|$, then, since $\kappa_{p_{2,1}} = 0$, it follows that $\kappa_{p_{2,3}} \le 1$. The determination of the other bounds is similar. To summarize:

$$\begin{array}{lcll} \kappa_{p_{2,3}} & \le & 1 & \text{because of } p_{2,1}\ , \\ \kappa_{p_{3,5}} & \le & 3 & \text{because of } p_{2,3}\ , \\ \kappa_{p_{3,9}} & \le & 1 & \text{because of } p_{1,9}\ , \\ \kappa_{p_{4,10}} & \le & 1 & \text{because of } p_{4,12}\ . \end{array} \tag{4.13}$$

Let us now investigate the vertical projection value of F in the 11-th column of $\mathcal{C}$. According to (4.9), it is equal to

$$1 + k\cdot 0 + 2 + (a_i - 1)\cdot 2 + (k(2B+1) - a_i)\cdot 2 = 2k(2B+1) + 1 \tag{4.14}$$

for any $i = 1,\ldots,3k$. There is one object point in the first row of F and in all the $p_{1,11}$ points of the $k(2B+1)$ number of $\mathcal{C}$ submatrices in the same column. Therefore, there has to be other $k(2B+1)$ object points in this column in order

Table 4.1: Possible labels for the non-trivial skeletal points in submatrix $\mathcal{C}$ within their bounds. "*" denotes an arbitrary non-negative integer.

$\kappa_{p_{2,3}} \leq 1$	$\kappa_{p_{3,5}} \leq 3$	$\kappa_{p_{3,9}} \leq 1$	$\kappa_{p_{4,10}} \leq 1$	Note
*	*	*	0	Contradicts the projection value of the 11-th column
0	{0,1}	*	*	Contradicts the 4-connectedness
0	2	0	1	Equals to $\mathcal{C}^{10}$
0	2	1	1	Equals to $\mathcal{C}^{11}$
0	3	*	*	Contradicts Lemma 1.4.1
1	0	*	*	Contradicts the 4-connectedness
1	1	0	1	Equals to $\mathcal{C}^{00}$
1	1	1	1	Equals to $\mathcal{C}^{01}$
1	{2,3}	*	*	Contradicts the projection value of the 1-st row

to fulfil the vertical projections. Those object points cannot belong to any of the $\vec{u}$ vectors (since the corresponding horizontal projections are already satisfied).

Assume that $\kappa_{p_{4,10}} = 0$ for some submatrix $\mathcal{C}$. Thus, there can be no other object points in the 11-th column of this particluar submatrix. Overall, there has to be $k(2B+1)$ object points in the 11-th column of $k(2B+1)-1$ submatrices, which is a contridaction, due to the upper bounds of the skeletal labels. Therefore, $\kappa_{p_{4,10}} = 1$ for every submatrix $\mathcal{C}$.

Let us take a look at $p_{2,3}$ and $p_{3,5}$. If $\kappa_{p_{2,3}} = 0$, then $\kappa_{p_{3,5}} \geq 2$, otherwise F cannot be 4-connected, due to the upper bound of the remaining skeletal labels. Also, if $\kappa_{p_{2,3}} = 0$, then $\kappa_{p_{3,5}} < 3$ because of Lemma 1.4.1. Therefore, if $\kappa_{p_{2,3}} = 0$, then $\kappa_{p_{3,5}} = 2$.

Finally, if $\kappa_{p_{2,3}} = 1$, then $\kappa_{p_{3,5}} \geq 1$ because of the 4-connectedness of F. According to (4.6) and (4.7), the horizontal projection value of F corresponding to the 1-st row of submatrix $\mathcal{C}$ is

$$1 + 5 + (3k-1) \cdot 5 = 15k + 1 \tag{4.15}$$

in both of the rows H_1 and H_2. There is one object point in the first column of F and $p_{1,9}$, $p_{1,10}$, $p_{1,11}$ and $p_{1,12}$ points in the first row of $3k$ number of $\mathcal{C}$ submatrices in the same row. Therefore, there is $3k$ further object points in $3k$ submatrices, in order to fulfil the horizontal projections. Similarly to the previous case, if $\kappa_{p_{2,3}} = 1$, the only solution satisfying the requirements is $\kappa_{p_{3,5}} = 1$.

Table 4.1 shows all the possible cases for the labels of an arbitrary submatrix $\mathcal{C}$. It is easy to verify that using valid labels yields the submatrix $\mathcal{C}$ to be equal to one of the object cells, hence F can be described as G_φ with some φ. □

Lemma 4.2.3 *If there exists a polyomino F of size $m \times n$ with horizontal projection H, vertical projection V, and skeletal set S, then the number of $\mathcal{C}^{11}$ cells is one, the number of $\mathcal{C}^{10}$ cells is $a_i - 1$, and the number of $\mathcal{C}^{00}$ cells is $k(2B+1) - a_i$ in*

Table 4.2: The number of the cells in the i-th cell column.

	$\#\mathcal{C}^{00}$	$\#\mathcal{C}^{01}$	$\#\mathcal{C}^{10}$	$\#\mathcal{C}^{11}$
Case 1	$k(2B+1)-a_i$	0	a_i-1	1
Case 2	$k(2B+1)-a_i-1$	1	a_i	0

the i-th cell column of $F = G_\varphi$. Moreover, F does not contain any $\mathcal{C}^{01}$ cells with a certain $\varphi : \mathcal{R} \to \{\mathcal{C}^{00}, \mathcal{C}^{01}, \mathcal{C}^{10}, \mathcal{C}^{11}\}$.

Proof

From Lemma 4.2.2 we know that F can be described with a G_φ matrix, hence F is built up from cells. According to (4.8), the vertical projection value corresponding to the 8-th column of the i-th cell column is equal to

$$1 + k \cdot 0 + 1 + (a_i - 1) \cdot 0 + (k(2B+1) - a_i) \cdot 0 = 2. \qquad (4.16)$$

Since one object point is in the first row of F, and only $\mathcal{C}^{01}$ and $\mathcal{C}^{11}$ contain object points in the 8-th column, the sum of the number of $\mathcal{C}^{01}$ and $\mathcal{C}^{11}$ cells are one for all i.

The vertical projection value corresponding to the 7-th column of the i-th cell column is equal to

$$1 + k \cdot 0 + 1 + (a_i - 1) \cdot 1 + (k(2B+1) - a_i) \cdot 0 = a_i + 1. \qquad (4.17)$$

Again, one object point is in the first row of F, and only $\mathcal{C}^{10}$ and $\mathcal{C}^{11}$ contain object points in the 7-th column, therefore the sum of the number of those cells is a_i for all i.

Similarly, the vertical projection value corresponding to the 2-nd column of the i-th cell column is equal to

$$1 + k \cdot 0 + 0 + (a_i - 1) \cdot 0 + (k(2B+1) - a_i) \cdot 1 = k(2B+1) - a_i + 1. \qquad (4.18)$$

Only $\mathcal{C}^{00}$ and $\mathcal{C}^{01}$ contain object points in the 2-nd column, therefore the sum of the number of those cells is $k(2B+1) - a_i$ for all i.

Overall, we have 3 equations for 4 non-negative integer variables (for the number of cells in the i-th cell column), and since the sum of two of those variebles is one, it is easy to verify that we have only two solutions, as Table 4.2 shows. Both cases satisfy the requirements of the vertical projections of F.

Note that in an arbitrary $\mathcal{C}^{10}$ cell the $(p_{3,5} \oplus_2 Y)$ component is not necessary 4-connected to a skeletal point labelled trivially by 0, but it can be 4-connected to other components of $\mathcal{C}^{10}$ or $\mathcal{C}^{11}$ cells in the same cell column (through skeletal points $p_{1,5}$ and $p_{5,5}$, again, see Fig. 4.6 for example). If there is no $\mathcal{C}^{11}$ cell in a cell column, then there is at least one $\mathcal{C}^{10}$ in that cell column, and its $(p_{3,5} \oplus_2 Y)$ component cannot be 4-connected to the other parts of F. Therefore, the second case in Table 4.2 is not possible. □

Lemma 4.2.4 *If there exists a polyomino F of size $m \times n$ with horizontal projection H, vertical projection V, and skeletal set S, then the number of $\mathcal{C}^{11}$ cells is one in the cell row corresponding to H_1 in (4.5), the number of $\mathcal{C}^{10}$ cells is one in the cell row corresponding to H_2 in (4.5), the number of $\mathcal{C}^{00}$ cells is $3k-1$ in any cell row, and F has no $\mathcal{C}^{01}$ cells.*

Proof
The horizontal projection value of F corresponding to the 5-th row of any cell row is equal to

$$1 + 3 + (3k-1) \cdot 2 = 6k + 2, \tag{4.19}$$

according to (4.6) and (4.7). There is one object point in the first column of F and at least two object points in every cell in a cell row (skeletal points $p_{5,10}$ and $p_{5,12}$). There are $3k$ number of cells in a cell row, which yields $6k+1$ object points overall. Only cell $\mathcal{C}^{10}$ and cell $\mathcal{C}^{11}$ contain additional object points in the 5-th row, and there is no $\mathcal{C}^{01}$ in F according to Lemma 4.2.3. Therefore the number of cells $\mathcal{C}^{00}$ is $3k-1$. Similarly, due to the 2-nd row of any cell row, it is easy to verify that the number of cells $\mathcal{C}^{11}$ is one in the cell row corresponding to H_1 in (4.5), and 0 otherwise. Moreover, the number of cells $\mathcal{C}^{10}$ is one in the cell row corresponding to H_2 in (4.5), and 0 otherwise. □

Lemma 4.2.5 *If there exists a polyomino F of size $m \times n$ with horizontal projection H, vertical projection V, and skeletal set S, then the* Three Partition *instance has a solution.*

Proof
According to Lemma 4.2.3, the number of $\mathcal{C}^x$ cells is a_i in the i-th cell column in F, where $x \in \{10, 11\}$. Those cells have to form a contiguous interval of cells in a cell column due to the 4-connectedness of F, previously mentioned in Lemma 4.2.3 (there is exactly one $\mathcal{C}^{11}$ cell in every cell column). Moreover, owing to the $\vec{u}$ vectors, they all have to be in one cell block.

Let $\mathcal{P}$ be a partitioning of the numbers a_i into k parts: Number a_i belongs to partition j if and only if the a_i number of $\mathcal{C}^x$ cells in i-th cell column are in the j-th cell block.

According to Lemma 4.2.4 and (4.5), the number of $\mathcal{C}^{11}$ cells in a cell block is 3, therefore exactly 3 contiguous intervals of $\mathcal{C}^x$ belong to a cell block, which means 3 of the a_i numbers belong to the same partition in $\mathcal{P}$. Since the number of the $\mathcal{C}^{10}$ cells is $2B-2$, the sum of the $\mathcal{C}^x$ cells in a cell block is $2B+1$, which means that the sum of the corresponding a_i numbers in a partition of $\mathcal{P}$ is $2B+1$. Therefore, $\mathcal{P}$ is a solution to the Three Partition instance. □

We now have all the facts needed to prove our main theorem.

Theorem 4.1 Skel Rec Poly *is NP-complete.*

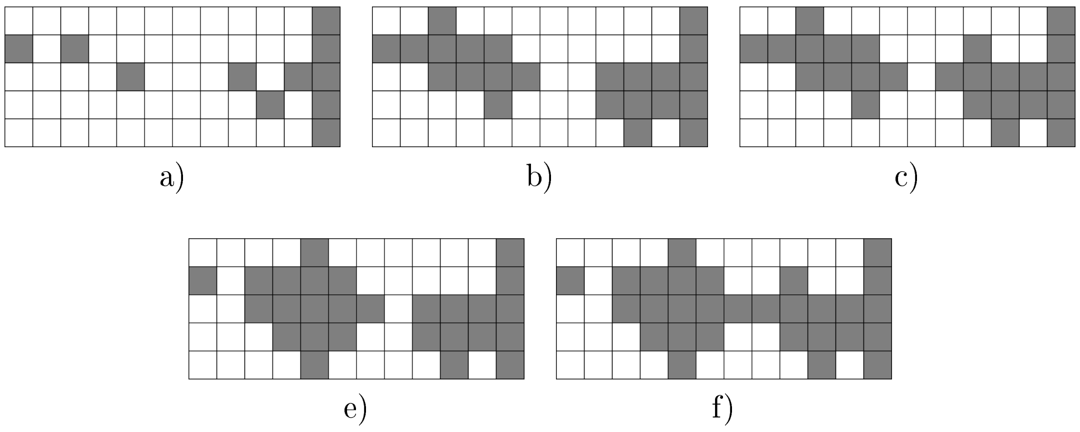


Figure 4.8: Different type of cells in the proof of the NP-competeness if the image has to be a simple connected polyomino: (a) the skeletal cell $\mathcal{C}^V$; (b) object cell $\mathcal{C}^{00}$; (c) object cell $\mathcal{C}^{01}$; (d) object cell $\mathcal{C}^{10}$; (e) object cell $\mathcal{C}^{11}$.

Proof
To prove that Skel Rec Poly is in NP, we need to validate in polynomial time whenever a binary image F of size $m \times n$ satisfies the requirements of Skel Rec Poly. As for the horizontal and vertical projections, the validation can be done in $O(nm)$. To generate the skeletal set $\mathcal{S}(F, Y)$, $O(nm \cdot \max\{n, m\})$ time is sufficient [59]. Finally, the 4-connectedness can be checked in polynomial time with, e.g., connected-component labeling algorithms [68].

The transformation from a Three Partition instance to a Skel Rec Poly instance is polynomial. The NP-hardness of Skel Rec Poly is a direct consequence of Lemma 4.2.1, Lemma 4.2.5 and the NP-hardness of Three Partition.□

Although our definition of the polyominoes allows holes, the NP-completeness still holds for simply connected binary images. If we replace the skeletal cell and the object cells in Fig. 4.4 with the images shown in Fig. 4.8, and modify the required skeletal set S, and the vectors H and V according to the new cells, then the proof works similarly as that of Theorem 4.1. Note that in that case, in (4.13) $\kappa_{p_{3,9}} \leq 1$ owing to $p_{3,11}$.

In the next section we show that without even requiring 4-connectedness on the image the problem is still generally NP-complete.

4.3 Reconstructing Binary Images is NP-complete

In this section in a similar way of Section 4.2 we prove that the reconstruction is NP-complete even if there is no connectivity constraints on the image to reconstruct. First, we define our introduced problem as a decision problem and name it as Skel Rec. The idea behind the reduction is similar to the one in [70].

Problem. Skel Rec

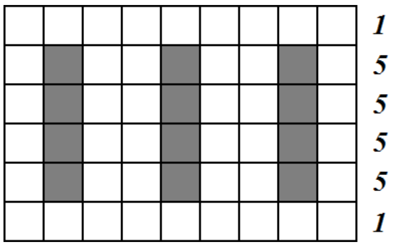


Figure 4.9: The $(3,4)$-widget $W_{3,4}$.

Instance. $H \in \mathbb{N}^m$, $V \in \mathbb{N}^n$ vectors and $S \subset \mathbb{Z}^2$ binary image.
Question. Does there exist a binary image F of size $m \times n$ such that $H = \mathcal{H}(F)$, $V = \mathcal{V}(F)$ and $S = \mathcal{S}(F, Y)$, where Y is given by (1.12)?

In order to prove the NP-completeness, first we prove that Skel Rec is NP-hard through reduction. Once again, we use the Three Partition problem described in the previous section, where an instance contains the non-negaive integers $a_1, \ldots, a_{3k}$, and the question is, if there exists a partitioning of $a_1, \ldots, a_{3k}$ into k triples such that the elements of every triple add up to exactly $2B+1$. We give a (logspace) reduction from Three Partition to Skel Rec, showing NP-hardness of the latter. To achieve this, we define a *widget*, that is, a parametrized skeleton image equipped with horizontal projection values: given k, $a > 0$, let the (k,a)-widget $W_{k,a}$ be a skeleton image of height $a+2$, width $3k$ with (i,j) being the skeletal pixel if and only if $j \equiv 2 \pmod 3$ and $1 < i < a+2$ (i.e., starting with the second column, every third column is all-one except for the first and the last rows), equipped with the following horizontal projection vector: $1, k+2, k+2, \ldots, k+2, 1$, where $k+2$ is repeated a times. See Fig. 4.9 for an example.

Note that if a widget occurs as a subpattern in a Skel Rec instance, then the skeletal label of each skeletal point in the widget must be either 0 or 1. Indeed, the skeletal points belonging to the same vertical lines are 4-connected, thus they have to have the same label according to Theorem 1.4.1 (this common label will also be called the label of the line). Should this label be at least 2, the horizontal projection in each non-border row would exceed $k+2$, even if all the other vertical lines have label 0: the others would contribute $k-1$ object points and the one having at least label 2 would contribute at least 5 object points, yielding a horizontal projection of at least $k+4$.

By an analogous argument one can see that at most one vertical line can have a label of 1, and all the others 0, since a line with label 1 contribute 3 object points, two of them would contribute 6, and along with the other $k-2$ lines a horizontal projection of at least $k+4$ would appear. Observe that the six object pixels could not overlap in this case, that is the point of the two-column gap between the lines.

Now we are ready to define the reduction. Given an instance B and non-negative integers $a_1, \ldots, a_{3k}$ of THREE PARTITION, we construct the following instance of SKEL REC:

- the skeletal image equipped with horizontal projections is $W_{k,a_1} \diamond W_{k,a_2} \diamond \cdots \diamond W_{k,a_{3k}}$, where $\diamond$ stands for the vertical composition of skeletal images and horizontal projections,
- the vector of the vertical projections is $\big(2B+1,\ k(2B+1)+6,\ 2B+1\big)$ repeated k times.

The result of the reduction applied to the input of the previous example is shown in Fig. 4.10.

Lemma 4.3.1 *If the instance of the* THREE PARTITION *problem has a solution, then the instance of the* SKEL REC *resulted by the reduction also has a solution.*

Proof
Suppose $\mathcal{I} = \{I_1, \ldots, I_k\}$ is a solution of the THREE PARTITION instance, i.e., $|I_j| = 3$ and $\displaystyle\sum_{i \in I_j} a_i = 2B+1$ for each $1 \le j \le k$, and $\displaystyle\bigcup_{j=1}^{k} I_j = \{1, \ldots, 3k\}$. Then for each $1 \le i \le 3k$, if $i \in I_j$, then let the label of the j-th vertical line be 1 in W_{k,a_i} corresponding to the i-th widget; let the label of the other lines 0. A straightforward calculation shows that all the horizontal and vertical projections are satisfied, and that the original skeleton image is indeed the skeleton of this image. Thus, the SKEL REC instance is consistent. $\square$

Lemma 4.3.2 *If the instance of the* SKEL REC *resulted by the reduction has a solution, then the instance of the original* THREE PARTITION *problem also has a solution.*

Proof
Let P a solution image of the SKEL REC problem. Since the labels of the skeletal points – thus, the labels of the lines in the skeletal image – define P uniquely, we will talk about labels of lines with respect to P. The horizontal projections require that in each W_{k,a_i} widget there is exactly one vertical line with label 1 and all others have label 0. We claim that the partition $\mathcal{I} = \{I_j : 1 \le j \le k\}$ with $i \in I_j$ if and only if the j-th vertical skeletal line (which is in column $3j-1$) of the widget W_{k,a_i} has label 1, is a solution to the THREE PARTITION instance.

From the previous argument we have that $\mathcal{I}$ is indeed a partition of $\{1, \ldots, 3k\}$. For any $1 \le j \le k$, we have to show that $|I_j| = 3$ and $\sum_{i \in I_j} a_i = 2B+1$. The vertical projection constraint on column $3j-1$ is $k(2B+1)+6$. If each vertical skeletal line in that column would have label 0, then by construction the projection would be exactly $k(2B+1)$. When we decide to increase the label of a line to 1, the projection value increases by two (again, see Fig. 4.10 for example). Observe that these freshly added object points cannot overlap, since there are two empty rows between the vertical skeletal lines. Thus, there are exactly three indices i such that the j-th line in the i-th widget has label 1, hence $|I_j| = 3$.

For the sum, consider the constraint on column $3j$, which is $2B + 1$. If each vertical skeletal line in column $3j - 1$ would have label 0, the projection of column $3j$ would be 0 (note that since the maximal label is 1 for every line by the horizontal constraints, no label of any other line can affect the projection of column $3j$). If we set the label of the j-th line in the i-th widget to 1, we increase the projection by a_i. Hence, $\sum_{i \in I_j} a_i = 2B + 1$.

Overall, $\mathcal{I}$ is a solution to the THREE PARTITION instance. □

Theorem 4.2 *The decision problem* SKEL REC *is NP-complete.*

Proof
The reduction from THREE PARTITION to SKEL REC is clearly logspace. From Lemma 4.3.1 and Lemma 4.3.2, we know that SKEL REC is at least as hard as THREE PARTITION. The NP-hardness of the latter proves the NP-hardness of the former.

It is also easy to prove that SKEL REC is in NP. Considering a possible solution F with size of $m \times n$ to the problem, we need to validate its correctness in polynomial time. The validation of the projections can be done in $O(mn)$. The morphological set $\mathcal{S}(F, Y)$ can be constructed in $O(mn \cdot \max\{m, n\})$ [59], and comparing $\mathcal{S}(F, Y)$ to S is also polynomial. Thus, SKEL REC is in NP, and overall it is NP-complete. □

If only the horizontal projection and the morphological skeleton is given, the problem is still NP-complete, considering the structuring element Y of (1.12). This can be proven similarly to Theorem 4.2. Figure 4.11 shows an example of the reduction.

4.4 Reconstruction as Optimization Problem

Although the reconstruction from two projections and morphological skeleton is generally NP-hard, under some circumstances an acceptable image quality can be achieved. We transform the problem into an energy minimization (or function minimization) task, where finding a minimum of the given function is equivalent to finding an optimal solution to the reconstruction problem. Moreover, a solution to the reconstruction is acceptable if the value of the corresponding energy function is close to its minimum. There are various methods to solve that kind of problems. In [35], the authors used a memetic algorithm for reconstructing binary images from horizontal, vertical, diagonal and anti-diagonal projections. In [63] a fan-beam projection model is implemented and used in systematic experiments in order to determine the optimal parameter values for reconstruction with Simulated Annealing. The authors of [66] presented a method that uses DC programming,

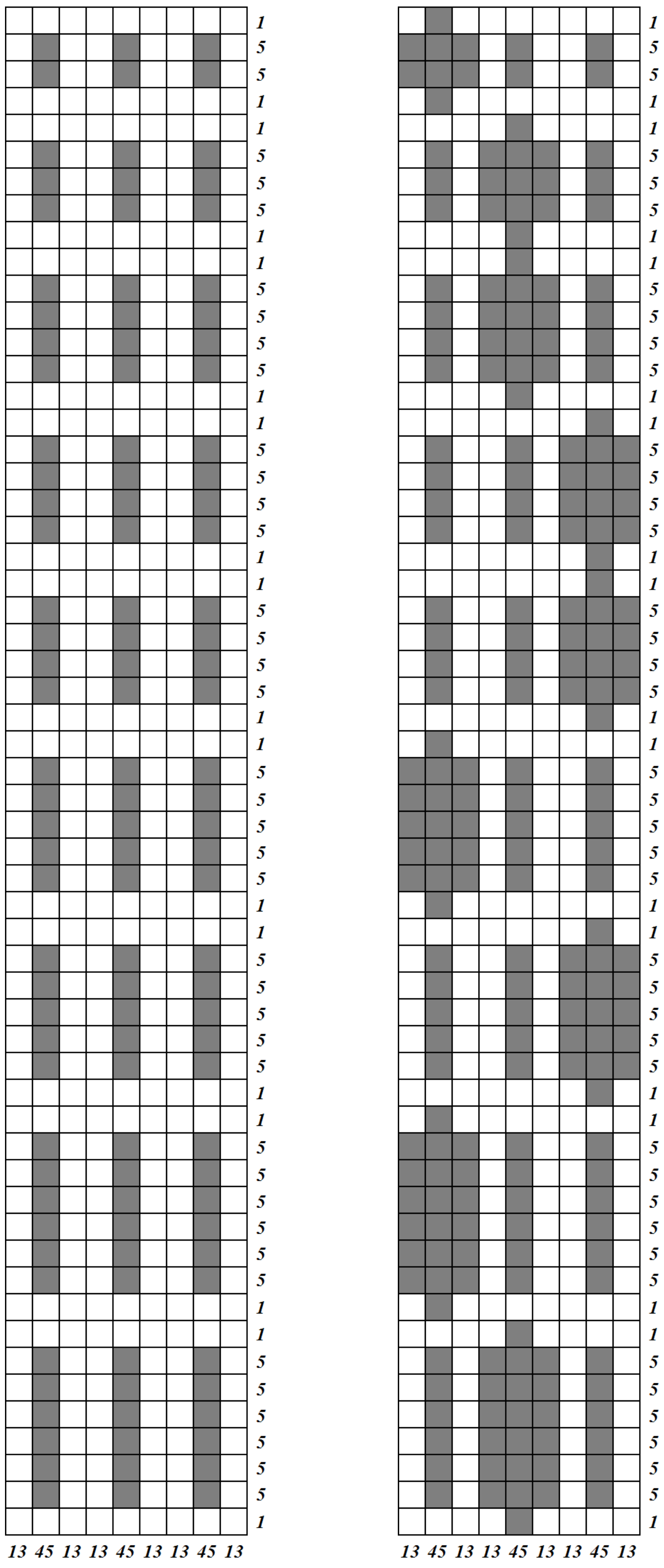


Figure 4.10: An example of the reduction, where $2B + 1 = 16$, $k = 3$ and the non-negative integers are 2, 3, 4, 4, 4, 5, 5, 6, 6. Left: the Skel Rec instance after the reduction. Right: a reconstruction corresponding to the solution $\mathcal{I} = \{\{1, 6, 8\}, \{2, 3, 9\}, \{4, 5, 7\}\}$.

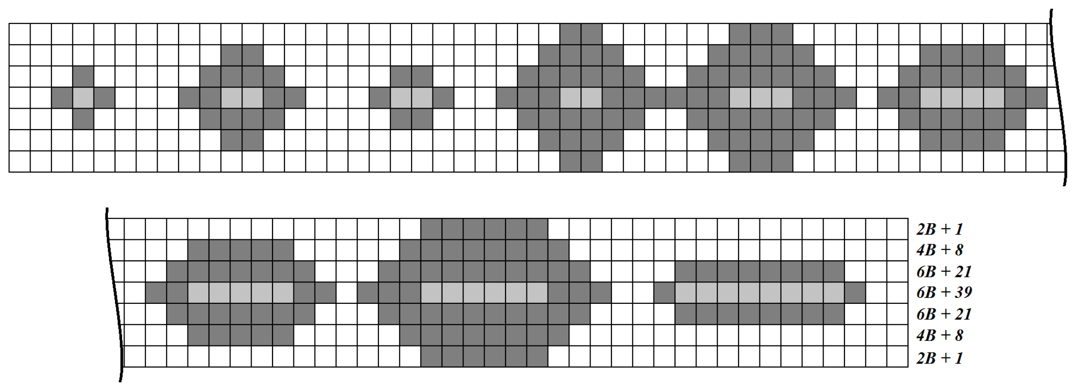


Figure 4.11: An example of the reduction if only the horizontal projection and the morphological skeleton is known. The image is cut in half for the better visibility. The Three Partition instance has non-negative integers 1, 2, 2, 2, 3, 4, 5, 6, 8 (with $k = 3$ and $B = 5$), with the solution $\mathcal{I} = \big\{\{1,3,9\},\{2,6,7\},\{4,5,8\}\big\}$. Light gray pixels indicate the skeleton.

a non-convex optimization technique, to solve the reconstruction of binary objects from few projection directions.

We choose Simulated Annealing (SA) as the optimization method for our problem owing to its simplicity, robustness, and flexibility in the control parameters. Section 1.5 gives a detailed description of the general method. Perhaps the most important advantage of SA over the competitive methods is that it can guarantee a near optimal solution in a reasonable time. SA also provides a natural way to encode the requirements of the projections, as well as the morphological skeleton, into an energy function. We propose three variants of a method to solve the reconstruction problem for two projections, and also for only the horizontal projection, based on parametric SA reconstruction.

First, we need to define our reconstruction problem as an energy function. Let $H \in \mathbb{N}_0^n$ and $V \in \mathbb{N}_0^n$ be two vectors, and $S \subset \mathbb{Z}^2$ be a finite set of points. Our task is to reconstruct an image F for which $\mathcal{S}(F,Y) = S$, and which (at least approximately) satisfies $\mathcal{H}(F) = H$ and $\mathcal{V}(F) = V$ (see again Fig. 4.1c for an example).

As (1.11) of Section 1.4 states, at each point $p \in \mathcal{S}(F,Y)$ there is a unique skeletal label κ_p value with $p \in \mathcal{S}_{\kappa_p}(F,Y)$. Thus, the image F can be uniquely represented by a vector $K(\mathcal{S}(F,Y)) = (\kappa_{p_1}, \kappa_{p_2}, \ldots, \kappa_{p_{|\mathcal{S}(F,Y)|}}) \in \mathbb{Z}^{|\mathcal{S}(F,Y)|}$. Using the notions of (1.4) of Section 1.1 and given a set of points S, our goal is to find a $K^*(S) = (\kappa^*_{p_1}, \kappa^*_{p_2}, \ldots, \kappa^*_{p_{|S|}})$ which corresponds to the image F^* generated by (1.11), such that $f(\mathbf{x}^*) = ||\mathbf{A}\mathbf{x}^* - \mathbf{b}||_2^2$ is minimal. Here, $\mathbf{x}^*$ is the column vector representing F^*. Figure 4.12 shows an example. Note that even if there is no F such that $S = \mathcal{S}(F,Y)$ and the function value of f is zero (e.g. in case of noisy projection data), it can be still possible to give a solution, whose projections are

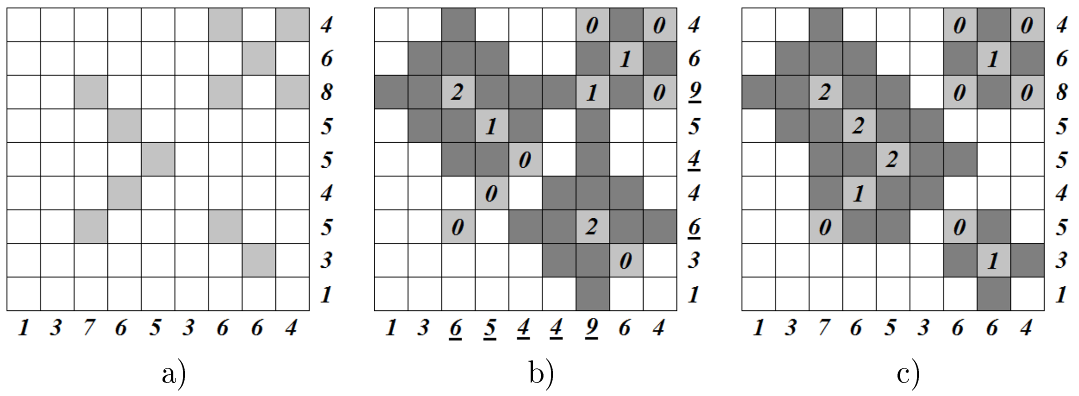


Figure 4.12: An example of the studied reconstruction problem: (a) the skeleton S and the projections H and V; (b) a reconstruction attempt F with $\mathcal{H}(F)$ and $\mathcal{V}(F)$ given by some $K(S)$; (c) the optimal solution F^* with $\mathcal{H}(F^*) = H$ and $\mathcal{V}(F^*) = V$ given by $K^*(S)$ (c). Projection elements that differ from the required ones are shown underlined.

close to the required ones.

The following lemma gives an upper bound for each element of $K(\mathcal{S}(F,Y))$ of an arbitrary binary image F.

Lemma 4.4.1 *Let F be a binary image of size $n \times n$ and $K(\mathcal{S}(F,Y)) = (\kappa_{p_1}, \kappa_{p_2}, \ldots, \kappa_{p_{|\mathcal{S}(F,Y)|}}) \in \mathbb{Z}^{|\mathcal{S}(F,Y)|}$. Then $\kappa_{p_i} < \lfloor \frac{n+1}{2} \rfloor$ for each $i = 1, \ldots, |\mathcal{S}(F,Y)|$, where $\lfloor . \rfloor$ stands for the floor function.*

Proof

From (1.10) of Section 1.4 we know that the maximum value of $K(\mathcal{S}(F,Y))$ is $\max\{k \mid F \ominus_k Y \neq \emptyset\}$. Since the size of the structuring element Y is 3×3, it follows that the size of $F \ominus_{t+1} Y$ is smaller by two in each dimension than the size of $F \ominus_t Y$ for some non-negative integer t (see the definition of the morphological erosion in (1.6) of Section 1.4). As a consequence, $F \ominus_{\lfloor \frac{n+1}{2} \rfloor} Y = \emptyset$. Thus, the possible maximum value in $K(\mathcal{S}(F,Y))$ is less than $\lfloor \frac{n+1}{2} \rfloor$. □

Since the size of the image is known, the searching space is bounded by Lemma 4.4.1. The following lemma provides a sharper upper bound.

Lemma 4.4.2 *For any skeletal set S of points and any $(i,j) \in S$ it holds that*

$$\kappa_{(i,j)} \leq \min\left\{i-1,\ j-1,\ n-i,\ n-j,\ \left\lfloor \frac{h_i - 1}{2} \right\rfloor,\ \left\lfloor \frac{v_j - 1}{2} \right\rfloor \right\},$$

where $\kappa_{(i,j)} \in K(S)$ is the corresponding skeletal label in K, h_i and v_j is the corresponding horizontal and vertical projection value, respectively.

Lemma 4.4.2 is trivial due to the size of the image and the fact that $F = \bigcup_{p \in \mathcal{S}(F,Y)} (p \oplus_{\kappa_p} Y)$. The lemma simply states that the skeletal labels must be small enough, otherwise the corresponding projection values would be bigger. Furthermore, together with Lemma 4.4.1, they define a unique maximum value for each $\kappa_p \in K(\mathcal{S}(F, Y))$.

Note that in some variations of the parametric SA we will integrate Lemma 1.4.1 into the energy function as a restriction on the skeletal labels.

4.5 Solving the Reconstruction Problem with Parametric SA

We focus on the reconstruction of general binary images from at most two projections and the morphological skeleton. Since the decision problem is NP-complete, a straightforward consequence is that there is no polynomial-time algorithm for finding an exact solution to the problem described in the first paragraph in Section 4.4 with a 0 valued minimum of (1.4) of Section 1.1 (if the morphological skeleton is involved), unless P=NP.

Nevertheless, we still have the chance to solve (1.4) approximately with SA. An adjusted version of SA is described in Algorithm 14.

Algorithm 14 Simulated Annealing on the Introduced Problem

```
Require: Projections $H$ and $V$, set of skeletal points $S$, and starting position
  $K_0(S)$
Ensure: $K(S)$
  $K(S) \leftarrow K_0(S)$
  $t \leftarrow 0$
  repeat
    $K'(S) \leftarrow$ MODIFY$(K(S))$
    Calculate $\mathbf{x}'$ and $\mathbf{x}$ from $K'(S)$ and $K(S)$, respectively
    if $f(\mathbf{x}') < f(\mathbf{x})$ or RAND $< \exp\left(\frac{f(\mathbf{x})-f(\mathbf{x}')}{T(t)}\right)$ then
      $K(S) \leftarrow K'(S)$
    end if
    $t \leftarrow t + 1$
  until the termination criterion is satisfied
  return $K(S)$ and the corresponding image
```

The basic energy function f we use is simply $f(\mathbf{x}) = ||\mathbf{Ax} - \mathbf{b}||_2^2$, where $\mathbf{x}$ is defined by the actual solution F. The goal is to find $K^*(S)$ which describes an image $\mathbf{x}^*$ where $f(\mathbf{x}^*)$ is minimal, i.e., it has the lowest energy. We know that if $f(\mathbf{x}_1) < f(\mathbf{x}_2)$, then the image F_1 is better than F_2 in the sense that its projections are closer to the required ones, therefore function $f(\mathbf{x})$ is a proper energy function. $T(t)$ is the temperature function or the cooling schedule, such that $T(0)$ is positive, and $T(t) \to 0$ as $t \to \infty$.

We choose the following exponential function

$$T(t) = T_0 \cdot \left(\frac{T_s}{T_0}\right)^{t/M},$$

where t denotes time, so the temperature will decrease over time, M is the maximal number of allowed iterations, T_0 is the chosen value for the starting temperature and T_s is a parameter controlling the shape of the cooling schedule. We empirically established the starting temperature $T_0 = 10$ and the parameter $T_s = 0.001$. In each iteration the time t is increased by 1. The process terminates when the iteration number t reaches M, or the energy of the current solution is 0.

RAND is a floating point number taken in each iteration from a uniform random distribution ($0 \leq \text{RAND} \leq 1$). With the function MODIFY we alter a state to another one simply by choosing a $\kappa_p \in K(S)$ randomly, and updating its value between the corresponding bounds defined by Lemmas 4.4.1 and 4.4.2. For the initial solution we choose the κ_p-s such that the initial image satisfies Lemma 1.4.1 (see Section 1.4) for every 8-adjacent skeletal points – i.e., their Manhattan-distance is not greater than 2 – and its projections are close to the required ones. We choose to consider only 8-adjacency, because its validity is easy to compute locally. We developed three different strategies for the reconstruction:

1. *No Skeletal Constraint* (NSC): In the SA modification step, we choose a κ_p randomly, and change it randomly between its bounds, omitting Lemma 1.4.1.

2. *Dynamic Skeletal Constraint* (DSC$_C$): We apply Lemma 1.4.1 in the following way: in each step, we modify a randomly chosen κ_p by defining its new value such that $|\kappa_p - \kappa_q| \leq C$ holds for each q 8-adjacent to p. If $C = 1$, we allow only those differences that mentioned in Lemma 1.4.1 for 8-adjacent skeletal points. Because it also means slow convergence during iterations, we allow higher C values in the beginning of the reconstruction, and decrease C through time. For that we use a function $C(t)$, which is similar to the cooling schedule:

$$C(t) = \left\lceil C_0 \cdot \left(\frac{C_s}{C_0}\right)^{t/M} \right\rceil,$$

   where $\lceil . \rceil$ denotes the ceil function, C_0 is the starting parameter, so $C(0) = C_0$, C_s is a parameter established to 0.15 explicitly. Note that $C(t) \to 1$ as $t \to M$, so we force SA to search a solution that satisfies Lemma 1.4.1 (for 8-adjacency) as much as possible.

3. *Combined Energy Function* (CEF$_\alpha$): We incorporate the constraints of

Lemma 1.4.1 by using an extended energy function:

$$f(\mathbf{x}) = \alpha||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + (1-\alpha)g(\mathbf{x}),$$

where α is a weighting parameter ($0 \leq \alpha \leq 1$),

$$g(\mathbf{x}) = \sum_{0<d_1(p,q)\leq 2} h(\kappa_p, \kappa_q) \quad \big(\ p, q \in S,\ \ \kappa_p, \kappa_q \in K(S)\ \big),$$

and

$$h(\kappa_p, \kappa_q) = \begin{cases} 0 & \text{if } |\kappa_p - \kappa_q| \leq 1 \\ |\kappa_p - \kappa_q|/2 & \text{otherwise.} \end{cases}$$

Note that if a solution F satisfies Lemma 1.4.1 for 8-adjacent skeletal points, then $g(\mathbf{x}) = 0$. In case of $\alpha = 1$, this method is equivalent to the No Skeletal Constraint method (i.e. $\text{CEF}_1 = \text{NSC}$).

4.6 Numerical Results

4.6.1 Implementation Details

For testing our proposed algorithm we developed a general reconstruction framework. For initialization, one has to specify the initial temperature T_0, the parameter T_s, the maximal number of allowed iterations and the initialization strategy. Some of the variants of the presented SA method have also additional parameters, such as α or C_0. Certain parameters were fixed, such as C_s or the structuring element Y. We also fixed the cooling schedule. These parameters were chosen empirically, since we found that the reconstruction is robust for those settings. The framework was implemented in C++ language using Dev-C++ environment, and the test was performed under Windows 7 on one core of an Intel Core 2 Duo T2520 of 1.5 GHz PC with 2GB of RAM.

4.6.2 Experimental Results for Two Projections

We tested our algorithm on 50 artificial images. Here we show 8 samples of them, which we found the best representatives of our results. Six of our test samples have one point thin morphological skeleton consisting of few 8-connected components. However, we also show two other images which have more complex skeletons. All of the test images have size of 256×256.

Since SA is a randomized algorithm, we performed each test 10 times and measured the mean CPU time and errors of the reconstruction. For the numerical

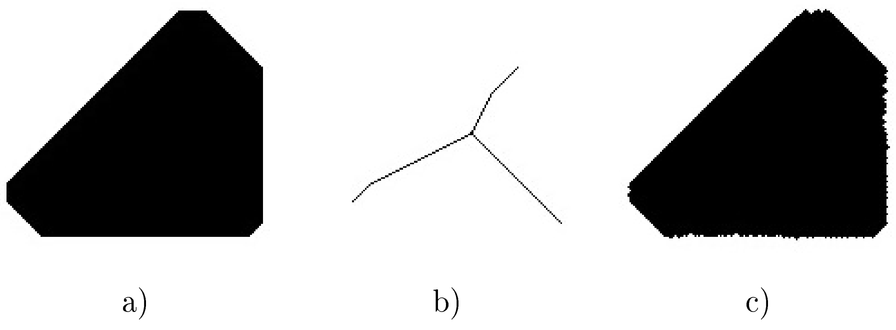


Figure 4.13: A test image and a reconstruction attempt: (a) the original image; (b) its morphological skeleton; (c) one of the reconstructed images with $\mathrm{CEF}_{0.5}$.

evaluation of the quality of the reconstructed images, we calculated

$$E = ||\mathbf{b} - \mathbf{b}'||_2 ,$$

where $\mathbf{b}$ and $\mathbf{b}'$ is the projection vector of the original and the reconstructed image, respectively. We also calculated the relative mean error *(RME)* to measure the distance between the original and the reconstructed image,

$$RME = \frac{\sum_{i=1}^{n^2} |p_i - p_i'|}{\sum_{i=1}^{n^2} p_i} \cdot 100,$$

where p_i and p_i' is the i-th pixel value of the original and the reconstructed image, respectively. Note that E and RME do not necessarily correlate (although $RME = 0$ implies $E = 0$). For all tests, we set $T_0 = 10$, $T_s = 0.001$, and $M = 50\,000$.

First, we tested the images containing just one convex object (see the upper two images of Table 4.3). An example of the reconstruction is shown in Fig. 4.13. We found that 50 000 iterations were more than enough to converge to such a reconstructed image in most cases. All three variants of the SA method provided results with low projection error and *RME*, and DSC turned out to be the best choice in term of *RME*. In one case, setting the parameter $C = 1$ of DSC we could even perfectly reconstruct the original image in all 10 runs, using only 21 220 iterations on average.

In the second turn, we studied images of convex objects arranged in a 2×2 and a 3×3 array (bottom two images of Table 4.3). We observed that the initial state misleaded the DSC algorithm in some cases. The main reason was that the initial image was very dissimilar to the original one, and DSC converged very slowly, meaning that it would have needed much more than 50 000 iterations.

The third group of test data contained images consisting of convex objects forming random groups (upper two images of Table 4.4). For the first image,

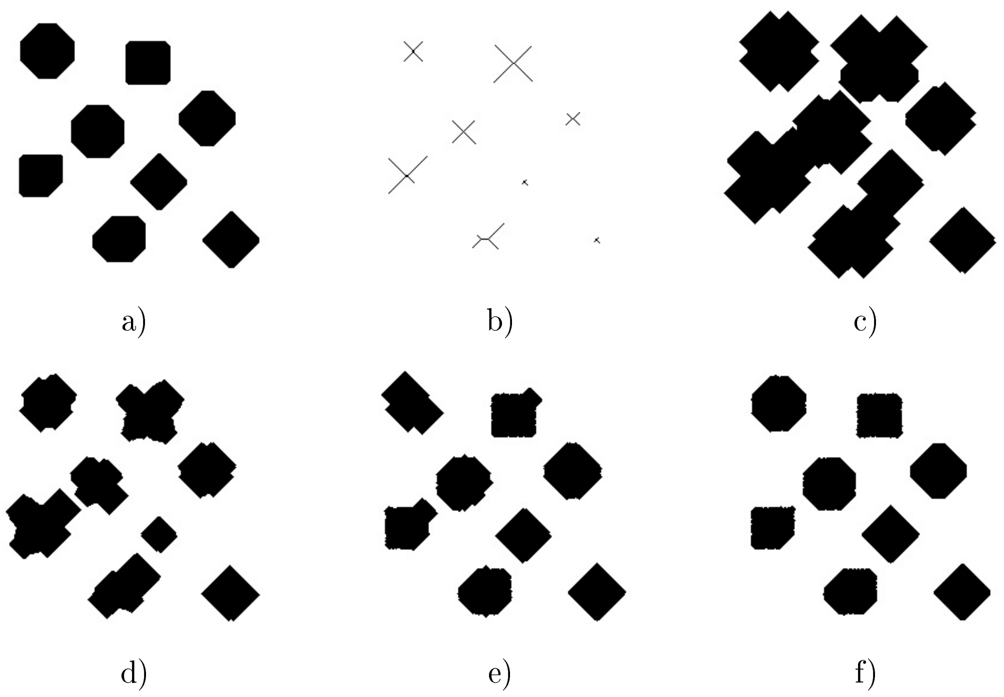


Figure 4.14: An example of the convergence of DSC_{10}: (a) original image; (b) the morphological skeleton; (c) the initial reconstruction; (d) result after 5 000 iterations; (e) after 25 000 iterations; (f) final result after 50 000 iterations with $E = 325$ and $RME = 2.08$.

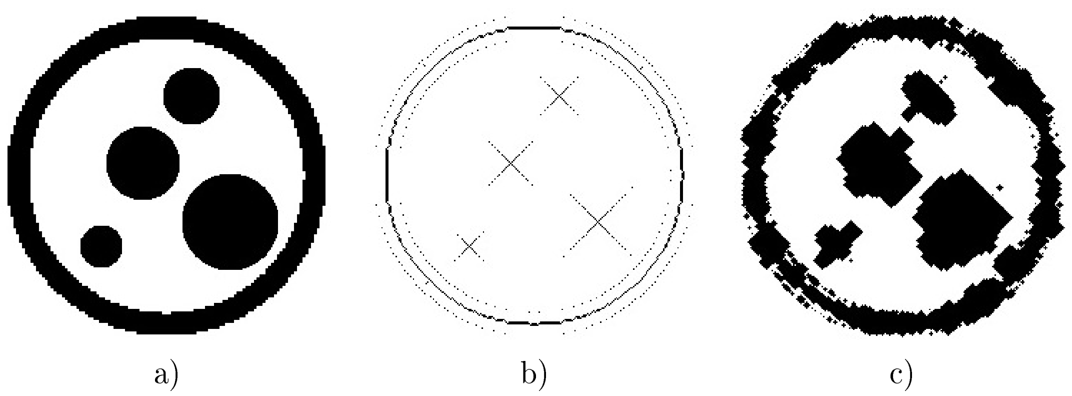
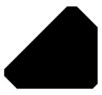
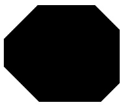
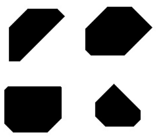
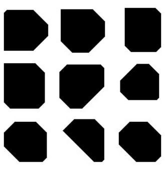


Figure 4.15: A test image with coarse boundaries: (a) the test image; (b) its morphological skeleton that contains numerous isolated pixels; (c) one of the reconstructed images with NSC.

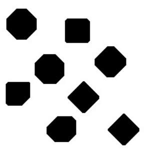
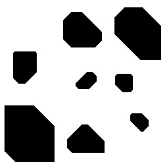
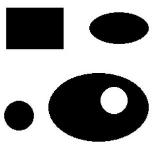
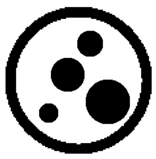
Table 4.3: Reconstruction results from two projections. CPU values are in milliseconds and E values are rounded to integers. Results with lowest errors are typeset in boldface.

Image	Method	CPU	E	RME
	NSC	3842	1060	0.64
	DSC_{10}	4030	98	0.16
	DSC_5	4116	97	0.15
	$\mathbf{DSC}_1$	**4563**	**18**	**0.04**
	$\mathrm{CEF}_{0.3}$	4358	2468	1.18
	$\mathrm{CEF}_{0.5}$	4415	1675	0.86
	$\mathrm{CEF}_{0.7}$	4435	1305	0.72
	NSC	7276	1285	0.90
	DSC_{10}	7900	174	0.28
	DSC_5	8127	146	0.13
	$\mathbf{DSC}_1$	**4473**	**0**	**0**
	$\mathrm{CEF}_{0.3}$	7626	2578	1.09
	$\mathrm{CEF}_{0.5}$	7665	1849	0.82
	$\mathrm{CEF}_{0.7}$	7691	1505	0.79
	NSC	3784	3405	7.27
	$\mathbf{DSC}_{10}$	**3038**	**1291**	**3.09**
	DSC_5	3164	4288	4.26
	DSC_1	3566	5307	5.74
	$\mathrm{CEF}_{0.3}$	5412	5665	4.20
	$\mathrm{CEF}_{0.5}$	5387	4829	3.62
	$\mathrm{CEF}_{0.7}$	5328	3212	3.62
	**NSC**	**4346**	**6136**	**4.43**
	DSC_{10}	4733	1066145	48.48
	DSC_5	4609	1722350	52.87
	DSC_1	4926	3302481	59.77
	$\mathrm{CEF}_{0.3}$	7308	14371	5.94
	$\mathrm{CEF}_{0.5}$	7243	8896	5.30
	$\mathrm{CEF}_{0.7}$	7222	7402	4.67

the results are similar to the first group's results (see the upper two images of Table 4.3), even if there are more skeletal points now yielding a bigger searching space. Figure 4.14 shows an example of the convergence of the DSC_{10} method. However, for the second image NSC produced the best results.

Finally, we examined images that have many skeletal points with few connections (bottom two images of Table 4.4). An example reconstruction result can be seen in Fig. 4.15. One of the reasons of the poor results could be the skeleton, which contains many isolated pixels. It makes the method slow and ambiguous due to the large searching space. Here, NSC proved to be the best choice, since it did not use the constraints of Lemma 1.4.1, yielding the most robust approach of all. Although even this method could reach just a rough approximation of the original object, the RME of the results is suprisingly low regarding that just two

Table 4.4: Reconstruction results of more complex images from two projections. CPU values are in milliseconds and E values are rounded to integers. Results with lowest errors are typeset in boldface.

Image	Method	CPU	E	RME
	NSC	1666	1341	4.34
	DSC$_{10}$	**1215**	**292**	**2.01**
	DSC$_5$	1234	314	2.06
	DSC$_1$	1302	294	2.03
	CEF$_{0.3}$	2904	2534	6.56
	CEF$_{0.5}$	2827	1950	5.59
	CEF$_{0.7}$	2851	1732	5.61
	**NSC**	**2165**	**2709**	**3.14**
	DSC$_{10}$	1713	6042	5.00
	DSC$_5$	1724	7962	13.06
	DSC$_1$	1910	6360	6.28
	CEF$_{0.3}$	4123	5688	4.55
	CEF$_{0.5}$	4131	4178	3.69
	CEF$_{0.7}$	4114	3346	3.24
	**NSC**	**3537**	**2530**	**7.49**
	DSC$_{10}$	2852	9154	15.73
	DSC$_5$	2981	13138	18.79
	DSC$_1$	3226	67493	27.37
	CEF$_{0.3}$	6380	5183	8.90
	CEF$_{0.5}$	6367	4102	8.69
	CEF$_{0.7}$	6343	3029	8.65
	**NSC**	**2757**	**4034**	**21.75**
	DSC$_{10}$	2304	4523	27.98
	DSC$_5$	2467	7472	28.91
	DSC$_1$	2430	13096	38.14
	CEF$_{0.3}$	8884	6663	25.24
	CEF$_{0.5}$	8856	5012	24.13
	CEF$_{0.7}$	8959	4407	25.16

projections were used.

Beside our previous database we tested the NVC algorithm – which resulted generally low RME – on hv-convex 4-connected images, a well-defined class of images where uniform random generation is possible [50]. We selected 50 images from the benchmark of [3] with the size of 150×150, and ran the algorithm 10 times with the same parameters. Some of the images can be seen in Fig. 4.16, while the reconstruction results for an image are shown in Fig. 4.17. The mean of the energies for the reconstructed images were $E = 723$ with standard deviation of $\sigma = 184.86$, the mean of the RMEs were 8.40 with $\sigma = 3.76$, while the mean of the CPU times were 1715 miliseconds with $\sigma = 629.83$.

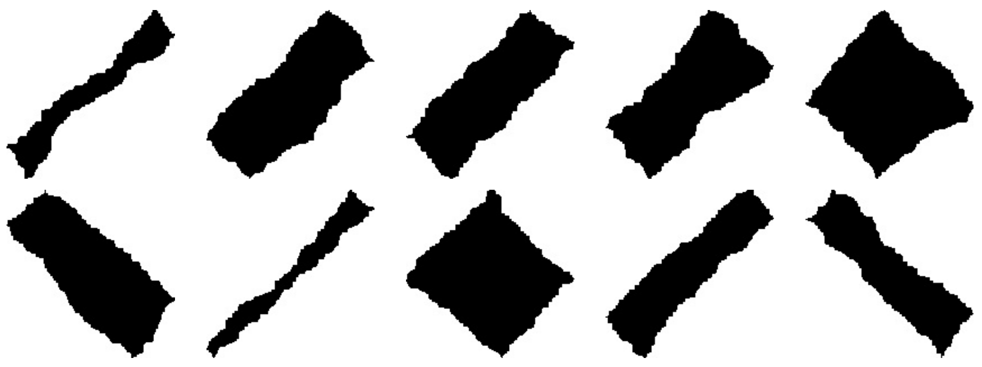


Figure 4.16: Samples from the *hv*-convex 4-connected set.

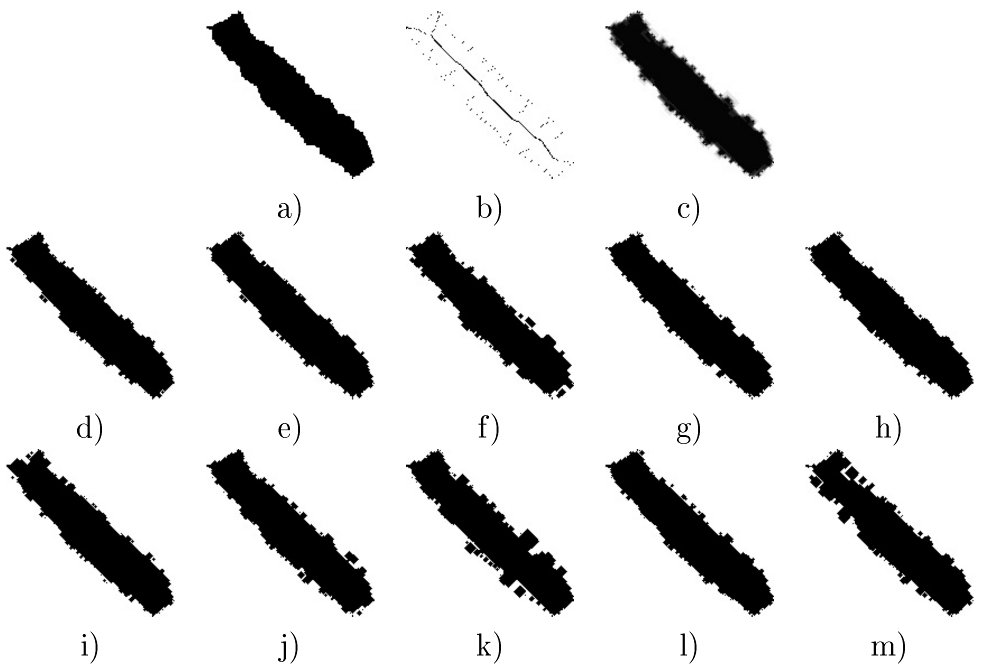
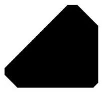
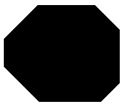
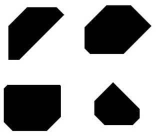
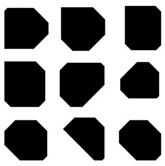


Figure 4.17: Reconstruction results for an *hv*-convex 4-connected image: (a) original image; (b) morphological skeleton; (c) the combined gray-value image that we get by superimposing the reconstruction results (d-m) on each other, and calculating the average value on each pixel. Numbers indicate the *RME* values.

Table 4.5: Reconstruction results from the horizontal projection. CPU values are in milliseconds and E values are rounded to integers. Results with lowest errors are typeset in boldface.

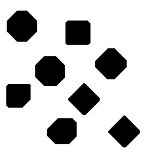
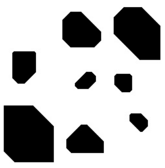
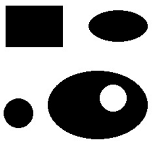
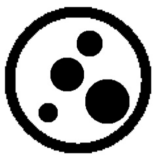
Image	Method	CPU	E	RME
	NSC	5524	995	1.52
	DSC$_{10}$	**6020**	**106**	**0.29**
	DSC$_5$	6012	422	1.01
	DSC$_1$	6675	318	1.11
	CEF$_{0.3}$	5618	2332	2.80
	CEF$_{0.5}$	5689	1499	2.01
	CEF$_{0.7}$	5737	1177	1.89
	NSC	10401	1353	0.94
	DSC$_{10}$	11871	124	0.18
	DSC$_5$	12327	153	0.19
	DSC$_1$	**14006**	**8**	**~0**
	CEF$_{0.3}$	10302	2600	1.40
	CEF$_{0.5}$	10470	1901	1.05
	CEF$_{0.7}$	10441	1546	0.93
	NSC	4707	2381	11.48
	DSC$_{10}$	**4135**	**1310**	**10.41**
	DSC$_5$	4265	1511	10.54
	DSC$_1$	4913	5189	12.82
	CEF$_{0.3}$	5796	4501	11.25
	CEF$_{0.5}$	5784	3482	11.42
	CEF$_{0.7}$	5713	2892	10.74
	**NSC**	**5527**	**4118**	**8.00**
	DSC$_{10}$	6940	600270	60.82
	DSC$_5$	8131	967214	57.92
	DSC$_1$	10943	1823806	63.09
	CEF$_{0.3}$	7470	7037	9.83
	CEF$_{0.5}$	7503	5072	8.39
	CEF$_{0.7}$	7326	4436	8.44

4.6.3 Experimental Results for One Projection

Since the results show that the morphological skeleton carries so much information about the shape of the original image, and can greatly improve the quality of the reconstruction, we decided to run the tests considering only the horizontal projection of the image besides its morphological skeleton. The parameters were identical to the ones before, except the lack of the second projection. Table 4.5 and Table 4.6 show the results.

The results are fairly similar to the reconstruction results of two projections, however, the RME values are significantly higher, due to the lesser amount of information. Since the algorithms try to minimize E, which holds now only the horizontal projection, there is also a significant difference between the correspond-

Table 4.6: Reconstruction results of more complex images from the horizontal projection. CPU values are in milliseconds and E values are rounded to integers. Results with lowest errors are typeset in boldface.

Image	Method	CPU	E	RME
	NSC	1999	1583	13.14
	DSC_{10}	1445	1445	17.48
	DSC_{5}	1560	1363	12.66
	DSC_{1}	1706	2328	14.86
	$\text{CEF}_{0.3}$	2592	2427	17.95
	$\text{CEF}_{0.5}$	2590	2216	14.51
	$\mathbf{CEF}_{0.7}$	**2598**	**1360**	**9.23**
	NSC	2475	3927	29.88
	DSC_{10}	2202	4086	32.17
	DSC_{5}	2303	3970	34.16
	DSC_{1}	2558	5004	35.67
	$\text{CEF}_{0.3}$	3623	5472	28.44
	$\text{CEF}_{0.5}$	3642	4229	28.42
	$\mathbf{CEF}_{0.7}$	**3641**	**3375**	**21.96**
	NSC	4984	2563	15.63
	DSC_{10}	4407	3789	20.63
	DSC_{5}	4602	6744	25.50
	DSC_{1}	5070	34355	30.81
	$\mathbf{CEF}_{0.3}$	**6738**	**2395**	**13.37**
	$\text{CEF}_{0.5}$	6736	3043	16.31
	$\text{CEF}_{0.7}$	6712	3152	15.40
	**NSC**	**3342**	**1896**	**43.85**
	DSC_{10}	3025	1638	46.89
	DSC_{5}	3108	2152	43.37
	$\mathbf{DSC}_{1}$	**3155**	**3524**	**39.52**
	$\text{CEF}_{0.3}$	7949	3474	45.32
	$\text{CEF}_{0.5}$	7897	2920	44.73
	$\text{CEF}_{0.7}$	7903	2431	44.32

ing E and RME errors. A good example is the last image of Table 4.6, where NSC managed to minimize E better, but DSC_1 achieved better RME, thus a more similar result to the original one. However, the reconstruction results of those images are quite poor, similarly to Table 4.4. On the other hand, some of the reconstruction results are surprisingly good. For example, for the second image of Table 4.5, the reconstructed image differed from the original one in just a few pixels, yielding a nearly 0 error in both E and RME. This indicates that the morphological skeleton carries really much information about the original image.

Fig. 4.18 shows some of the reconstruction results for an hv-convex 4-connected image. The mean of the energies for the reconstructed images were $E = 266$ with standard deviation of $\sigma = 27.85$, the mean of the RMEs were 11.86 with $\sigma = 2.93$,

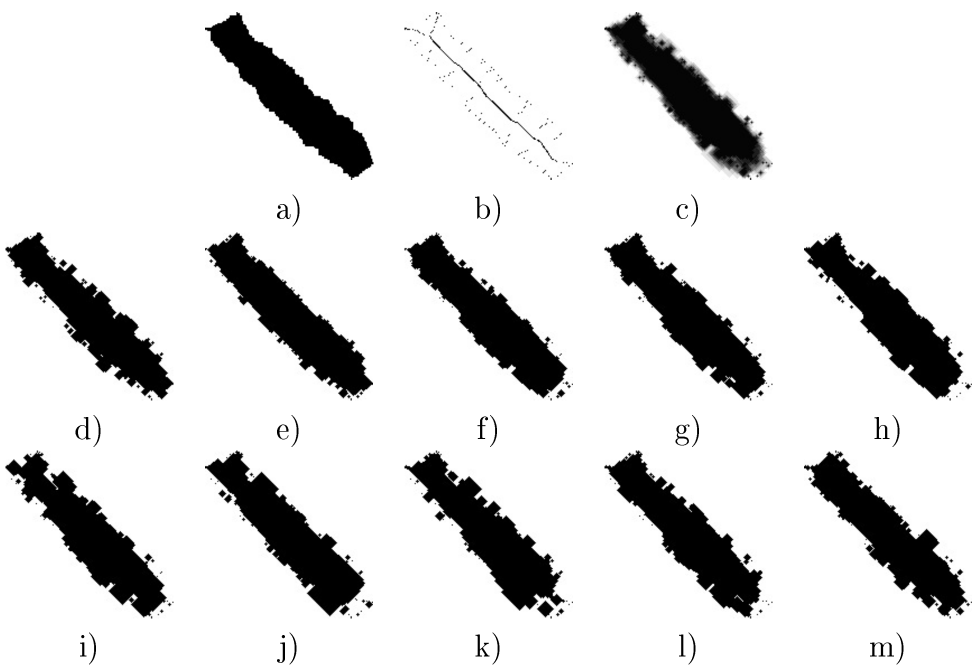


Figure 4.18: Reconstruction results from the horizontal projection for an *hv*-convex 4-connected image: (a) original image; (b) morphological skeleton; (c) the combined gray-value image that we get by superimposing the reconstruction results (d-m) on each other, and calculating the average value on each pixel. Numbers indicate the *RME* values.

while the mean of the CPU times were 1697 miliseconds with $\sigma = 640.77$.

4.7 A Uniqueness Result for Reconstructing *hv*-Convex Polyominoes

As mentioned before, uniqueness of certain type of binary images is a related issue to the reconstruction. In [30], the authors determined an upper and lower bound to the maximum number of *hv*-convex polyominoes having the same orthogonal projections, and proved that under some conditions, the ambiguity can be exponential. In [55], the authors gave a formula for enumerating certain type of parallelogram polyominoes according to their symmetry types and their perimeter or area. In this section, we study the uniqueness of the reconstruction of certain type of 4-connected *hv*-convex images, using two projections and the morphological skeleton. We show that the uniqueness of a certain parametric subclass of *hv*-convex binary images is strongly connected to its parameters.

First, we introduce some definitions. Given a finite set $F \subset \mathbb{Z}^2$ and a point $p \in \mathbb{Z}^2$, we say that $q \in F$ is a *closest point* of F to p, if there is no $r \in F$ such that $d_1(p, r) < d_1(p, q)$.

Let $\overline{pq}$ denote the sequence $p = (i_p, j_p), (i_p + 1, j_p + 1), \ldots, (i_p + n, j_p + n) = q$. We use the same notion for the line segment between p and q if $q \in \{\ (i_p + n, j_p - n),\ (i_p - n, j_p + n),\ (i_p - n, j_p - n)\ \}$.

Let $N_4(p)$ denote the set of 4-adjacent points to p. A point $p \in F$ is called a *border point* if p has a 4-adjacent background point.

For the definition of 4-connectedness and polyominoes, see the first paragraph of Section 4.2. Furthermore, a binary image is *diagonally convex* (*antidiagonally convex*), if the object points are consecutive in each diagonal (antidiagonal).

We can define now a special class of polyominoes, described by two parameters. Later we will show that the uniqueness of the reconstruction of these images depends only on their parameters.

For given $k, l \in \mathbb{N}_0$, let $G_{k,l}$ be the binary image

$$G_{k,l} = (p \oplus_k Y) \cup (q \oplus_l Y) \cup (r \oplus_l Y) \cup (s \oplus_k Y), \tag{4.20}$$

where $p = (i, j)$, $q = (i,\ j{+}k{+}l{+}1)$, $r = (i{+}k{+}l{+}1,\ j)$, $s = (i{+}k{+}l{+}1,\ j{+}k{+}l{+}1)$, and Y is the structuring element given by (1.12) of Section 1.4. Furthermore, let $H_{k,l} \subset \mathbb{Z}^2$ be constructed from $G = G_{k,l}$ by

$$\begin{aligned} H_{k,l} \ &= \ \big\{(u,v) \mid \exists (u_1,u_2)\ \ u_1{<}u{<}u_2, \\ &\qquad (u_1,v) \in G, (u_2,v) \in G, (u,v) \notin G\big\} \\ &\bigcup \ \big\{(u,v) \mid \exists (v_1,v_2)\ \ v_1{<}v{<}v_2, \\ &\qquad (u,v_1) \in G, (u,v_2) \in G, (u,v) \notin G\big\}. \end{aligned}$$

Finally, let $B_{k,l} = G_{k,l} \ \dot{\cup}\ H_{k,l}$, where $\dot{\cup}$ denotes the disjoint union. Figure 4.19 shows examples with $k = 4$, $l = 2$, and $k = 5$, $l = 1$.

In the sequel, let p, q, r, and s denote the points in (4.20) for an arbitrary $B_{k,l}$ with fixed k and l. The following attributes are easy to verify for $B_{k,l}$:

- the size of the image is $n \times n$, where $n = 3 \cdot \max\{k, l\} + \min\{k, l\} + 2$,
- $B_{k,l}$ is 4-connected, hv-convex, diagonally and antidiagonally convex,
- $B_{k,l}$ has rotational symmetry of order 2,
- $B_{k,l} = T_x(B_{l,k}) = T_y(B_{l,k})$, where T_x and T_y denotes the reflection transformation across the horizontal and vertical axis, respectively,
- $\mathcal{H}(B_{k,l}) = \mathcal{H}(T_x(B_{k,l}))$ and $\mathcal{V}(B_{k,l}) = \mathcal{V}(T_y(B_{k,l}))$.

The images with $k = l$ are trivial cases; in the reconstruction from two projections can be performed uniquely in polynomial time [11, 26, 65], and – in a final step

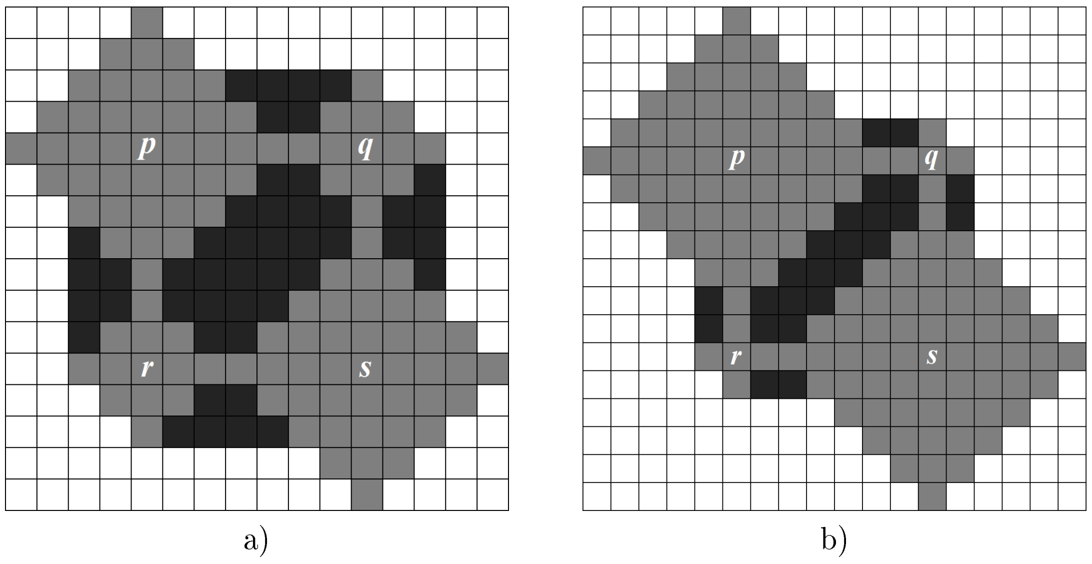


Figure 4.19: Example images of $B_{k,l}$: (a) the $B_{4,2}$ image; (b) the $B_{5,1}$ image. Grey and black pixels indicate the corresponding subset G and H, respectively. From the horizontal and vertical projections and the morphological skeleton the reconstruction of $B_{4,2}$ is non-unique, $B_{5,1}$ is uniquely reconstructable.

– we only need to check whether the morphological skeleton of the reconstructed image is the set S. Furthermore, since $B_{k,l} = T_x(B_{l,k})$, it is sufficient to focus on the case $k > l$. We will prove that there is a connection between the uniqueness of the reconstruction and the value of k and l. Namely, the reconstruction of $B_{k,l}$ $(k > l)$ is non-unique if and only if

$$\left\lfloor \frac{k+l}{2} \right\rfloor \leq 2l. \tag{4.21}$$

4.7.1 Some Properties of the Morphological Skeleton and the $B_{k,l}$ Images

First, we show some general properties of the morphological skeleton with the structuring element Y in (1.12) of Section 1.4. The following lemma gives another definition of the morphological skeleton with Y.

Lemma 4.7.1 *Let $p \in F$ be an object point of a binary image F, and let Y be the structuring element in (1.12) of Section 1.4. Then*

$$p \in \mathcal{S}_k(F, Y) \iff \begin{cases} (p \oplus_k Y) \subset F, \\ (p \oplus_{k+1} Y) \not\subset F, \\ \forall q \in N_4(p) \quad (q \oplus_{k+1} Y) \not\subset F. \end{cases} \tag{4.22}$$

Proof
For the proof we use the equivalency

$$p \in (F \ominus_k Y) \iff (p \oplus_k Y) \subset F. \tag{4.23}$$

Assume $p \in \mathcal{S}_k(F,Y)$. Then $p \in (F \ominus_k Y) \setminus [(F \ominus_{k+1} Y) \oplus Y] \subset (F \ominus_k Y) \setminus (F \ominus_{k+1} Y)$, according to the definition of the skeletal subset in (1.9). Therefore, $(p \oplus_k Y) \subset F$ and $(p \oplus_{k+1} Y) \not\subset F$, using (4.23). Assume to the contrary that there exists a $q \in N_4(p)$ such that $(q \oplus_{k+1} Y) \subset F$. According to (4.23), $q \in (F \ominus_{k+1} Y)$. But then $p \in [(F \ominus_{k+1} Y) \oplus Y]$, since $d_1(p,q) = 1$, which contradicts to the definition of the skeletal subset.

For the other direction, let $p \in F$ a particular point. From $(p \oplus_k Y) \subset F$, it follows that $p \in (F \ominus_k Y)$, and from $(p \oplus_{k+1} Y) \not\subset F$ we get $p \notin (F \ominus_{k+1} Y)$, using (4.23). Since there is no point $q \in N_4(p)$ with $q \in (F \ominus_{k+1} Y)$, we get $p \notin [(F \ominus_{k+1} Y) \oplus Y]$. Therefore, according to (1.9), $p \in \mathcal{S}_k(F,Y)$. □

We describe a property of the points in $B_{k,l}$ with its border points.

Lemma 4.7.2 *Let $B = B_{k,l}$ for fixed k and l. Moreover, let $u = (i_u, j_u) \in B$ an arbitrary object point of B and $t \in \mathbb{N}_0$ such a value that $(u \oplus_t Y) \subset B$ and $(u \oplus_{t+1} Y) \not\subset B$. Then, there exists a border point $v = (i_v, j_v)$ such that $d_1(u,v) = t$ and either $i_u = i_v$ or $j_u = j_v$.*

Proof
Since $(u \oplus_0 Y) = u \in B$ and B is finite, the unique existence of t is clear. Assume to the contrary that none of the $v_1 = (i_u - t, j_u)$, $v_2 = (i_u, j_u + t)$, $v_3 = (i_u + t, j_u)$ and $v_4 = (i_u, j_u - t)$ points are border points (as Fig. 4.20 shows). Thus, every point 4-adjacent to them is in B, including the o_i points shown in Fig. 4.20 ($i = 1, \ldots, 4$). Since B is both diagonally and antidiagonally convex, every point in $\overline{o_i o_{i+1}}$ is in B ($i = 1, \ldots, 4$, $o_5 = O_1$), too. Therefore, $(u \oplus_{t+1} Y) \subset B$, which is a contradiction.□

Now, we give three lemmas to decide whether a certain point of $B_{k,l}$ is an element of the morphological skeleton of the image. First, we identify which object points cannot be skeletal points.

Lemma 4.7.3 *Let $B = B_{k,l}$ for arbitrary fixed k and l, and let $u = (i_u, v_u) \in B$ an arbitrary object point of B. If $u \notin (\overline{ps} \cup \overline{rq})$ then $u \notin \mathcal{S}(B,Y)$.*

Proof
Assume that $u \notin (\overline{ps} \cup \overline{rq})$ and let $t \in \mathbb{N}_0$ be a value such that $(u \oplus_t Y) \subset B$ and $(u \oplus_{t+1} Y) \not\subset B$. Without loss of generality, let us assume that a closest point to u from the set $\{p, q, r, s\}$ is p (the proof of the other cases are similar). Let c be a closest border point to u in a way that either $c = (i_u - t, j_u)$ or $c = (i_u, j_u - t)$. We know from Lemma 4.7.2 that at least one of the cases holds.

The rest of the proof can be followed in Fig. 4.21. If $c = (i_u - t, j_u)$, then let $w = (i_w, j_w)$ be the closest point to u such that $w \in \overline{ps}$ and $j_w = j_u$ (in the other

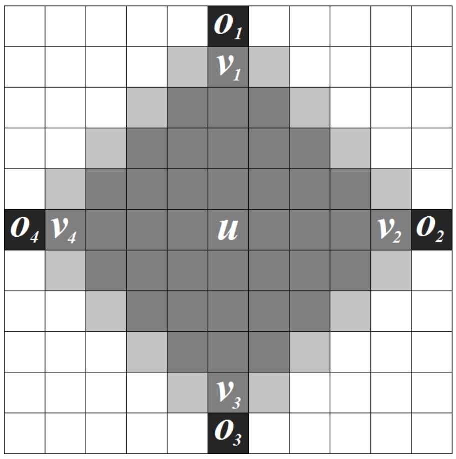


Figure 4.20: Assumption for Lemma 4.7.2, where $o_i \in B$ $(i = 1, \ldots, 4)$ are denoted by black pixels. Dark grey pixels indicate the set $(u \oplus_t Y)$. Since B is both diagonally and antidiagonally convex, every element indicated by light grey pixels are in B. Therefore, $(u \oplus_{t+1} Y) \subset B$, which is a contradiction.

case, if $c = (i_u, j_u - t)$, then $i_w = i_u$). Since c is also the closest border point to w and $d_1(c, w) = d_1(u, w) + t$, we get $(w \oplus_{d_1(u,w)+t} Y) \subset B$ (in a similar way as in Lemma 4.7.2). As a consequence, $(u \oplus_t Y) \subsetneq (w \oplus_{d_1(u,w)+t} Y)$ (note that $u \neq w$), therefore there is a v point such that $V \in N_4(u)$ and $(v \oplus_{t+1} Y) \subset B$. According to Lemma 4.7.1, $u \notin \mathcal{S}_t(F, Y)$ for any $t \in \mathbb{N}_0$, therefore, $u \notin \mathcal{S}(F, Y)$. □

The next lemma describes which object points must be skeletal points.

Lemma 4.7.4 *Let $B = B_{k,l}$ for arbitrary fixed $k \geq l$. Then $\overline{ps} \subset \mathcal{S}(B, Y)$.*

Proof
Let $u \in \overline{ps}$ and let $t \in \mathbb{N}_0$ be a value such that $(u \oplus_t Y) \subset B$ and $(u \oplus_{t+1} Y) \not\subset B$. According to Lemma 4.7.1, we only have to prove that $(v \oplus_{t+1} Y) \not\subset B$ for any $v \in N_4(u)$ point. Without loss of generality, we only have to check two of the 4-adjacent points, since B has rotational symmetry of order 2. Let $u = (i, j)$, $v_1 = (i - 1, j)$, and $v_2 = (i, j + 1)$ (see Fig. 4.22). Through the symmetries, either $c_1 = (i - t, j)$ or $c_2 = (i, j + t)$ is a closest border point to u.

Assume the first case holds. Then, $(i - t - 1, j) \notin B$. Since B is *hv*-convex and $k \geq l$, $w = (i - t - 1, j + 1) \notin B$. Note that $d_1(w, v_1) = d_1(w, v_2) = t + 1$. Therefore $(v_1 \oplus_{t+1} Y) \not\subset B$ and $(v_2 \oplus_{t+1} Y) \not\subset B$, which yield that $u \in \mathcal{S}(B, Y)$, indeed. The proof is similar if c_2 is the closest border point to u. □

Finally, we give a necessary and sufficient condition for the remaining points of $B_{k,l}$.

Lemma 4.7.5 *Let $B = B_{k,l}$ for arbitrary fixed $k \geq l$. Then $\overline{rq} \subset \mathcal{S}(B, Y)$ if and only if*

$$\left\lfloor \frac{k + l}{2} \right\rfloor \leq 2l. \qquad (4.24)$$

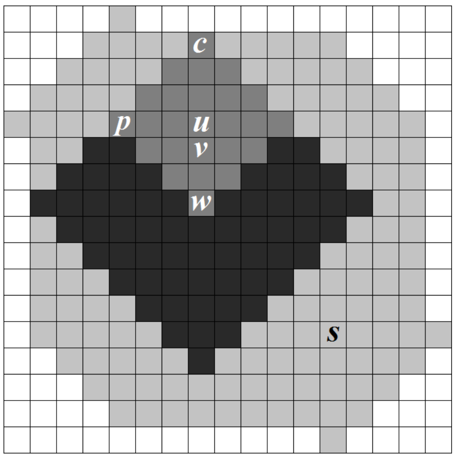


Figure 4.21: Illustration for the proof of Lemma 4.7.3. Grey pixels indicate $B_{4,3}$. Here, $t = 3$, $d_1(u, w) = 3$. Note that, since $(u \oplus_3 Y) \subsetneq (w \oplus_6 Y) \subset B$ (dark grey and black pixels), it holds that $(v \oplus_4 Y) \subset B$, therefore u cannot be a skeletal point.

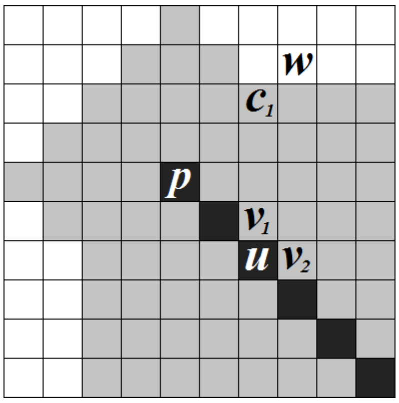


Figure 4.22: Illustration for the proof of Lemma 4.7.4, assuming c_1 is a closest border point to u. Note that $d_1(u, c_1) = t$, $d_1(v_1, w) = t + 1$ and $d_1(v_2, w) = t + 1$. Since $w \notin B$ (due to the hv-convexity of B) and $k > l$, it follows that $(v_1 \oplus_{t+1} Y) \not\subset B$ and $(v_2 \oplus_{t+1} Y) \not\subset B$. Therefore, u is a skeletal point.

Proof

Let us denote the points of $\overline{rq}$ with $r = r_0 = (i, j)$, $r_1 = (i - 1, j + 1)$, $r_2 = (i - 2, j + 2)$, ..., r_n, where $n = \lfloor (k + l + 1)/2 \rfloor$. Note that $d_1(r, r_t) \leq d_1(q, r_t)$ for $0 \leq t \leq n$, while the remaining points of $\overline{rq}$ are always closer to q than to r. Furthermore, let $c_0 = (i + l, j)$, $c_1 = (i + l, j + 1)$, $c_2 = (i + l, j + 2)$, ..., $c_{2l+1} = (i + l, j + 2l + 1)$, which are all border points of B below the r_t points (see Fig. 4.23). Note, that a similar set can also be defined to the left of the r_t points.

If $n \leq 2l$, then c_t is a closest border point to r_t, having $d_1(c_t, r_t) = l + t$. Therefore $(r_t \oplus_{l+t} Y) \subset B$ and $(r_t \oplus_{l+t+1} Y) \not\subset B$. Since c_t and c_{t+1} are both border points, for all the 4-adjacent $v \in N_4(r_t)$ points it is true that $(v \oplus_{l+t+1} Y) \not\subset B$. According to Lemma 4.7.1, $r_t \in \mathcal{S}(B, Y)$. As a consequence, $\overline{rq} \subset \mathcal{S}(B, Y)$.

If $n > 2l$ and $k + l$ is odd, then $r_n \in \overline{ps}$, therefore $r_n \in \mathcal{S}(B, Y)$ according to Lemma 4.7.4. If $n = 2l + 1$, then $\overline{rq} \subset \mathcal{S}(B, Y)$ still holds. However, if $n >$

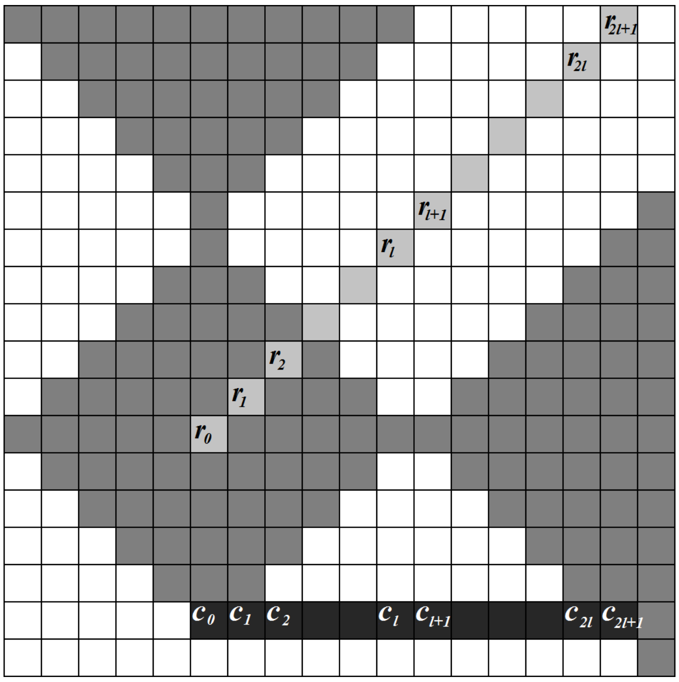


Figure 4.23: Illustration for the proof of Lemma 4.7.5. Dark grey pixels indicate the $G_{k,l}$ sub-image in $B_{k,l}$. Light grey pixels indicate the r_t elements ($t = 0, \dots, n$), where $r = r_0 = (i, j)$. Here, only the first $2l + 1$ of them are shown. Black pixels indicate c_0, c_1, …, c_{2l+1}. Note that one of the closest border points to r_a is c_a.

$2l + 1$, then $r_t \notin \mathcal{S}(B, Y)$ for $2l + 1 \leq t < n$, since there exist a $v \in N_4(r_t)$ that $(v \oplus_{l+t+1} Y) \subset B$. Therefore, $\overline{rq} \subset \mathcal{S}(B, Y)$ if and only if $n = \left\lfloor \frac{k+l+1}{2} \right\rfloor \leq 2l + 1$, or equivalently, $\left\lfloor \frac{k+l}{2} \right\rfloor \leq 2l$.

Finally, if $n > 2l$ and $k+l$ is even, then $r_n \notin \overline{ps}$. If $n \geq 2l+1$, then $r_t \notin \mathcal{S}(B, Y)$ for every $2l + 1 \leq t \leq n$, similarly to the previous case. Therefore, $\overline{rq} \subset \mathcal{S}(B, Y)$ if and only if $n = \left\lfloor \frac{k+l+1}{2} \right\rfloor \leq 2l$, i.e., $\left\lfloor \frac{k+l}{2} \right\rfloor \leq 2l$. □

4.7.2 Proof of the Uniqueness Result

Now we establish the relation between k, l and the uniqueness of the reconstruction of the image $B_{k,l}$.

Theorem 4.3 *Let $B = B_{k,l}$ for arbitrary fixed $k \geq l$. Then the reconstruction of B is non-unique if and only if*

$$\left\lfloor \frac{k+l}{2} \right\rfloor \leq 2l. \tag{4.25}$$

Proof
Note that if $k = l$, the reconstruction is unique (see the last paragraph of Section 4.7). From now we assume that $k > l$.

First, let assume that (4.25) holds for $B = B_{k,l}$ with arbitrary fixed $k > l$. According to Lemma 4.7.4, $\overline{ps} \subset \mathcal{S}(B, Y)$. As Lemma 4.7.5 states, $\overline{rq} \subset \mathcal{S}(B, Y)$.

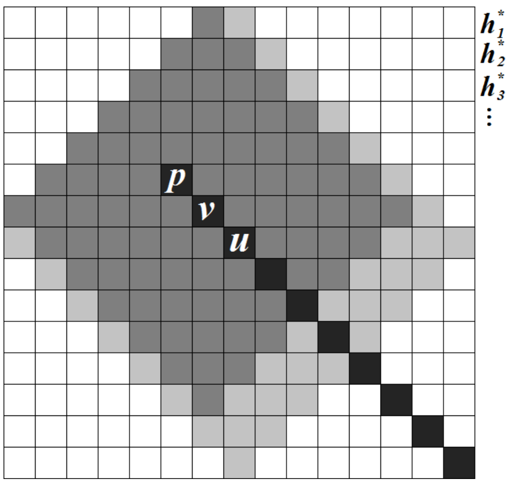


Figure 4.24: Illustration for the proof of Theorem 4.3: The assumption that there is a v skeletal point closer to p than u. If $(u \oplus_{\kappa_u} Y)$ reaches the first row (light grey pixels), then $(v \oplus_{\kappa_v} Y)$ must reach the first row (dark grey pixels), according to Lemma 1.4.1, which is a contradiction to $h_1^* = 1$. Black pixels denote $\overline{ps}$.

There is no other skeletal point, according to Lemma 4.7.3, therefore $\mathcal{S}(B, Y) = \overline{ps} \cup \overline{rq}$ (note that the skeleton has a reflection symmetry to both horizontal and vertical axis). As a consequence, $\mathcal{S}(B_{k,l}, Y) = \mathcal{S}(B_{l,k}, Y)$. Since $\mathcal{H}(B_{k,l}) = \mathcal{H}(B_{l,k})$ and $\mathcal{V}(B_{k,l}) = \mathcal{V}(B_{l,k})$ (although $B_{k,l} \neq B_{l,k}$), the reconstruction is non-unique.

For the other direction of the proof, we assume to the contrary that there is an $F \neq B$ image (which is not necessarily an hv-convex polyomino) with $H^* = \mathcal{H}(B) = \mathcal{H}(F)$, $V^* = \mathcal{V}(B) = \mathcal{V}(F)$ and $S^* = \mathcal{S}(B, Y) = \mathcal{S}(F, Y)$, but $\lfloor \frac{k+l}{2} \rfloor > 2l$. We use the notation of p, q, r and s in S^* as in definition (4.20). Let $n = 3k + l + 2$ denote the width (and height) of the image B. Since $k > l$, there is only 1 object point in the first and last row of B, and in the first and last column of B. Therefore $h_1^* = 1$, $h_n^* = 1$, $v_1^* = 1$, $v_n^* = 1$. According to (1.11), we have to find the skeletal labels of S^* in order to reconstruct F. Let $u \in S^*$ be the skeletal point such that the projection value h_1^* came from $(u \oplus_{\kappa_u} Y)$. Then u and p (or q) must coincide with $\kappa_u = k$, otherwise, there exists a v skeletal point such that $d_1(u, v) = 2$ and v is closer to p (or q). Since $\kappa_v \geq \kappa_u - 1$, according to Lemma 1.4.1 (i.e., $2 = d_1(u, v) > |\kappa_u - \kappa_v|$), $(v \oplus_{\kappa_v} Y)$ reaches the first row. Therefore, $h_1^* \geq 2$, which is a contradiction (see Fig. 4.24). In a similar way, we get either $\kappa_p = \kappa_s = k$ or $\kappa_q = \kappa_r = k$.

If $\kappa_p = \kappa_s = k$, then let $F' = S^* \cup (p \oplus_k Y) \cup (s \oplus_k Y) \subset F$. Let $H' = \mathcal{H}(F')$. We observe that $h_1^* = h_1', \ldots, h_{k-l}^* = h_{k-l}'$ (see Fig. 4.25), and also the difference $h_{k-l+1}^* - h_{k-l+1}' = 2l + 1$.

Let q_i denote the i-th skeletal point in $\overline{qr}$ in a way that $q_0 = q$ and $q_i = q_{i-1} + (1, -1)$ $(i = 0, \ldots, 2l)$. It follows that $\kappa_{q_i} \leq l + i$, because h_{k-l}^* is already fulfilled, and $\kappa_{q_i} \geq l + i$. Otherwise h_{k-l+1}^* cannot be fulfilled, since none of the q_i points are in the same column. Therefore, $\kappa_{q_i} = l + i$. The same holds for the r_i series defined similarly. Note that $\kappa_q = \kappa_r = l$.

Now, let $F'' = F' \cup (q_i \oplus_{l+i} Y) \cup (r_i \oplus_{l+i} Y)$ with $i = 0, \ldots, 2l$. If F'' fulfils the

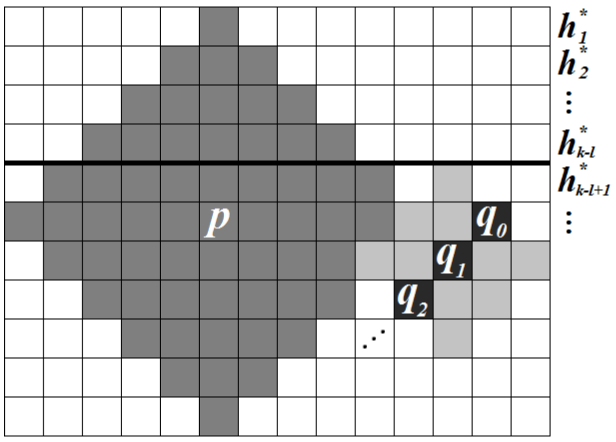


Figure 4.25: An example of $\kappa_p = \kappa_s = k$ in the proof of Theorem 4.3. Dark grey pixels indicate $(p \oplus_k Y)$. Note that every horizontal projection value $h'_1, \ldots, h'_{k-l}$ fulfils the corresponding horizontal projection values in H^*, due to $(p \oplus_k Y)$. Therefore, for every i, $(q_i \oplus_{\kappa_{q_i}} Y)$ must reach the thick line in order to fulfil the value h^*_{k-l+1}. Here, $\kappa_{q_1} = 2$ (light grey pixels).

projections, then $F = F''$. Otherwise, we have to add additional points to F'' in order to fulfil the projections. We have to fill the $pqrs$ rectangle in order to get F (since the rest of the projection values are already fulfilled). In this way we get $F = B$, which is a contradiction to our assumption $F \neq B$.

The reconstruction is similar if $\kappa_q = \kappa_r = k$. In this case, $B = T_x(F)$. However, $\overline{ps} \subset S^*$, but $\overline{rq} \not\subset S^*$ according to Lemmas 4.7.4 and 4.7.5. Therefore, $\mathcal{S}(F, Y) \neq T_x(\mathcal{S}(F, Y)) = \mathcal{S}(T_x(F), Y) = \mathcal{S}(B, Y) = S^*$. F cannot be a solution to the reconstruction problem (see Fig. 4.26).

Since there cannot be any other image $\tilde{F}$ with $\mathcal{H}(\tilde{F}) = H^*$, $\mathcal{V}(\tilde{F}) = V^*$ and $\mathcal{S}(\tilde{F}, Y) = S^*$, our assumption that $\left\lfloor \frac{k+l}{2} \right\rfloor > 2l$ was false. $\square$

4.8 Summary

Determining the computational complexity of the reconstruction problems is important to analyze the efficiency of the reconstrucion algorithms. We showed that even though additional information, such as the morphological skeleton – with or without considering 4-connectedness – of the image with a particular structuring element may reduce the ambiguity of the reconstruction, the problem still remains NP-complete. Thus, we redefined the problem as an optimization task, and proposed three variants of a method based on Simulated Annealing to solve the task. Without assuming 8-connected morphological skeletons, a rough reconstruction is always possible in a short time and a small number of iterations. With additional restrictions the result will be smoother, although the convergency of the method becomes slower. The No Skeletal Constraint variant provides overall satisfactory

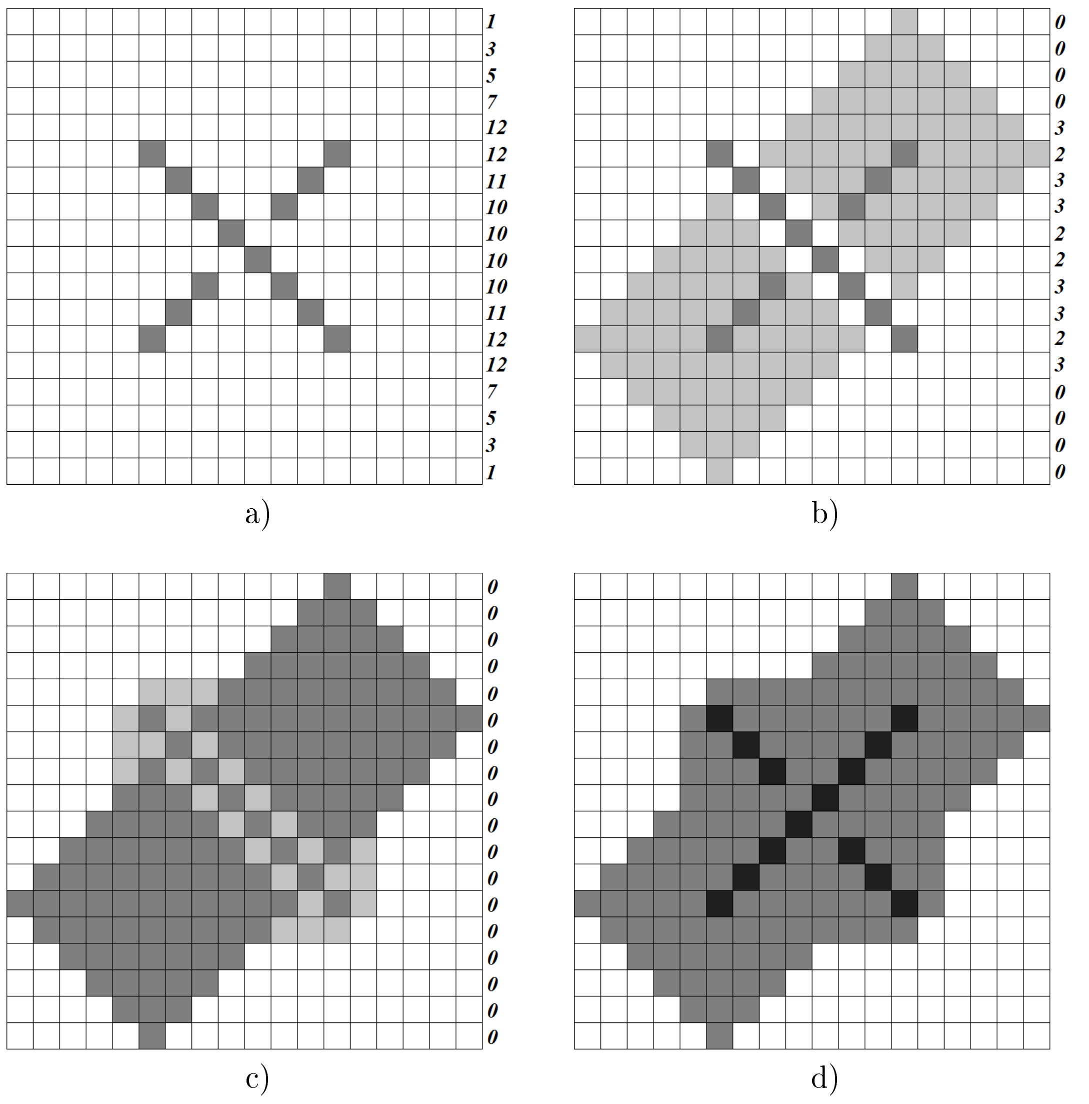


Figure 4.26: A reconstruction attempt, where $k = 5$ and $l = 1$. Here, the assumption is $\kappa_q = \kappa_r = k$. The steps are (a) the skeleton S^*, (b) - (c) intermediate steps (light grey pixels indicate the newly added object points), (d) the final image F and its skeleton (black pixels). Numbers show the difference of the actual and the required horizontal projections. Note that $\mathcal{S}(F, Y) \neq S^*$, hence F cannot be a solution: our assumption was false.

results. The Dynamic Skeletal Constraint creates smoother results in most cases, but needs more iterations to converge. The Combined Energy Function variant is just slightly worse than the NSC, but much slower. Beside that, in all the three considered variants we found that the result is much more dependent on the number of the skeletal points, rather than on the size of the image.

Furthermore, the number of *hv*-convex polyominoes satisfying the given horizontal and vertical projections can be exponential, although there exists a polynomial time algorithm for finding one of those solutions. We showed that the reconstruction of *hv*-convex polyominoes is non-unique even if the morphological skeleton of the image is additionally given. However, for a certain parametric type of those images the question of the uniqueness can be answered by the parameter values.

The findings of this research have been published in two conference proceedings [40, 46], and two journal papers [47, 45]. Up to date, there have been four independent citations [16, 19, 20, 72] to the results of this chapter.

Chapter 5

Conclusion

This dissertation gives a summary of the Author's research in the field of binary matrices and binary tomography.

First, we investigated switching components that play an essential role in binary reconstruction and data analysis. Searching for switching components in a binary matrix is a relevant task in discrete image reconstruction, as well as in biogeography and ecology. Although finding the minimal number of 0-1 flips in order to make a binary image switching component free is generally NP-hard, we managed to give two heuristics that outperform the previous methods in the number of 0-1 flips to make a binary image switching component free. For that aim we have shown how to reduce the size of the search space radically while keeping the optimal solutions in the search space. Moreover, we explained how to use those heuristics for binary image compression using Chang's binary reconstruction algorithm. The results could lead to design more efficient lossless and lossy image compression methods based on storing projections, 0-1 and 1-0 flips; not just for binary images, but for grayscale and color images as well.

Furthermore, we proposed a method to reconstruct *hv*-convex polyominoes from a given horizontal projection with minimal number of columns in linear time. The method can be extended in numerous ways, including searching for solutions with arbitrary number of columns, or reconstructing 8-connected binary images with given number of 4-connected components. We also provided formulas for counting the possible solutions, and gave a method to generate uniform random *hv*-convex binary images satisfying the given horizontal projection in polyominal time. We believe that an efficient heuristic could be to solve two instances of the one-projection reconstruction problem (one for the horizontal and one for the vertical projection), and then to combine the results of both of them. By combining the results of the one-projection reconstructions we could also develop a novel method to reconstruct *hv*-convex polyominoes from two projections, by which we could gain a deeper understanding of the problem.

Finally, we introduced a novel problem where the task is to reconstruct certain type of binary images from the given projections and the morhological skeleton. We proved that for polyominoes and general binary images the problem is generally NP-complete. Nevertheless, we showed that a rough reconstruction is always possible in a short time and a small number of iterations using optimization methods. We also investigated the uniqueness of certain sub-classes of polyominoes.

Reconstruction from the projections and morphological skeleton brings many open questions. What is the complexity of reconstructing hv-convex polyominoes with given morphological skeleton from two projections? Do three or more projections make the reconstruction easier? Does the NP-completeness result of this dissertation hold for other structuring elements, too? What is the number of the solutions if the reconstruction is not unique? Moreover, what algorithms are efficient to give an acceptable solution in a reasonable time for an NP-hard reconstruction? Since SA is rather sensitive to the initial state, in a further work, one could try to apply further strategies for choosing a starting image, e.g., by using Ryser's algorithm to obtain an initial solution. Apart from SA, tabu search and genetic algorithms [2] can also be used as function minimizers, altough the latter could have significantly higher running time and require more memory to handle each instance in a generation. These open questions are worth to investigate in the future.

Appendix A

Summary in English

Analysis of patterns in binary matrices plays a vital role in numerous applications of computer science. One such important application is binary tomography, where the task is to reconstruct binary images representing two-dimensional cross-sections of three-dimensional homogeneous objects from their projections. Very often, just few projections of the object can be measured, since the acquisition of the projection data can be expensive or damage the object. The presence of certain binary patterns in the image can violate the uniqueness of the reconstruction of the image, especially from small number of projections. Moreover, the physical limitations of the imaging devices make it sometimes impossible to take projections from numerous angles. Owing to the small number of projections the binary reconstruction can be extremely ambiguous. A common way to reduce the number of solutions of the reconstruction task is to assume that certain geometrical properties are satisfied. Such property can be the horizontal or vertical convexity, 4- or 8-connectedness, etc. We can also reduce the number of solutions if the morphological skeleton of the image is given.

Another fundamental question is the complexity of the reconstruction. Although many variants of the original reconstruction problem can be NP-hard, the prior knowledge can be often incorporated into an energy function, thus the reconstruction task is equivalent to a function minimization problem. A further question is whether a lower or upper bound, or even an exact formula can be given for the number of solutions.

This thesis is a summary of the Author's research in the field of binary tomography. The main focus of this work was to examine additional prior information for the reconstruction task from at most two projections, examine the theoretical background for complexity, give formulas for the number of solutions for certain classes of binary images, and develop new algorithms for binary tomography.

Key points of the thesis

The findings of the research can be divided into three thesis groups. Table A.1 gives the connection between the results and the publications of the Author.

In the first thesis group, I examined the methods for eliminating switching components in binary matrices with possibly low number of 0-1 flips. The results were published in a conference proceeding [42] and accepted for publication in a journal [41].

I/1. I provided a proof to reduce the search space drastically while the optimal solutions still can be found in the reduced search space. I managed to give two heuristics that outperform the previous methods in the number of 0-1 flips to make a binary image switching component free.

In the second thesis group I examined the binary reconstruction of *hv*-convex polyominoes and *hv*-convex canonical images where only the horizontal projection is given. The results were published in two conference proceedings [39, 44], and one journal paper [43].

II/1. I managed to give an algorithm to reconstruct *hv*-convex polyominoes with running time linear in the size of the horizontal projection. I proved that the algorithm always gives a result with minimal number of columns. Moreover, the algorithm can be easily modified to provide an image with a given size. Furthermore, I provided a formula for the exact number of solutions with arbitrary number of columns according to a given horizontal projection, and a recursive formula with fixed number of columns.

II/2. I provided an algorithm for the uniform random generation of *hv*-convex polyominoes, according to a given horizontal projection. The worst case running time of the algorithm is $O(m^2)$, where m is the size of the projection. The algorithm can be modified to generate polyominoes with fixed number of columns.

II/3. I showed how to reconstruct *hv*-convex canonical images from one projection. I provided an algorithm which always gives an 8-connected result minimal in size.

In the third thesis group, I examined the reconstruction problem of binary images if the morphological skeleton with a certain structuring element is also provided. The results were published in two conference proceedings [40, 46], and two journal papers [47, 45].

III/1. I proved that the reconstruction of polyominoes from two projections and the morphological skeleton (considering a certain structuring element) is NP-complete. Furthermore, without the restriction of the 4-connectedness the problem is still NP-complete. If only the horizontal projection is given with the morphological skeleton, finding a solution is, again, NP-complete.

III/2. I redefined the problem as an energy minimization problem, and used Simulated Annealing to solve the reconstruction of general binary images. I studied three variants of a parametric SA, and showed that a rough reconstruction is usually possible in a short time and a small number of iterations.

III/3. I defined a certain parametric subclass of hv-convex polyominoes, and showed that the uniqueness of the reconstruction from two projections and the morphological skeleton is determined by the parameters.

	[39]	[40]	[41]	[42]	[43]	[44]	[45]	[46]	[47]
I/1.			●	●					
II/1.						●			
II/2.					●				
II/3.	●								
III/1.							●		●
III/2.								●	●
III/3.		●							

Table A.1: The connection between the thesis points and the Author's publications.

Appendix B

Summary in Hungarian

A bináris mátrixok elemzése fontos szerepet játszik a számítástudomány több területén is. Egy ilyen terület a bináris tomográfia, ahol a cél egy háromdimenziós homogén objektum kétdimenziós szeleteit ábrázoló képeinek előállítása a vetületek ismeretében. Mivel a vetületképzés költséges, illetve roncsolhatja a vizsgált objektumot, legtöbbször csak kevés vetület áll rendelkezésre. Ennek következményeként az előállítandó kép a rendelkezésre álló adatok alapján nem egyértelműen meghatározott, bizonytalan lehet. Éppen ezért fontos, hogy bináris tomográfia esetén a bináris mátrixok tulajdonságait alaposan megvizsgáljuk. A lehetséges megoldások számának csökkentése érdekében gyakran feltételeznek a rekonstruálandó képről bizonyos geometriai tulajdonságokat. Ilyen tulajdonságok lehetnek a horizontális illetve vertikális konvexitás, 4- illetve 8-összefüggőség, és így tovább. Szintén segíthet a megoldások számának csökkentésében, ha ismertnek tekintjük a kép morfológiai vázát.

A rekonstrukció egy másik alapvető kérdése a probléma bonyolultsága. Habár az eredeti rekonstrukciós probléma sok változata ismerten NP-nehéz, a rekonstruálandó képre tett megszorítások gyakran megfogalmazhatók energiafüggvényként. Ily módon a probléma ekvivalens egy függvény minimumának megkeresésével. Az efféle optimalizációs probléma közelítő megoldására sokféle matematikai módszer ismert. A rekonstrukció egy másik lényeges kérdése, hogy milyen módon adható alsó illetve felső becslés egy adott rekonstrukciós feladat megoldásainak számára, esetenként megadható-e akár pontosan ez az érték.

Jelen értekezés a Szerző bináris tomográfiában, illetve az azzal kapcsolatos bináris mátrixok elemzésében végzett kutatásait foglalja össze. A kutatás fő célja a két vetületből történő bináris rekonstrukcióra tett megszorítások, a problémához kapcsolodó bonyolultságelmélet, valamint a lehetséges megoldások számának vizsgálata.

Az eredmények tézisszerű összefoglalása

A kutatás eredményei három csoportba oszthatók. Az eredmények és a hozzájuk kapcsolódó publikációk viszonyát a B.1 táblázat tartalmazza.

Az első téziscsoportban azt vizsgáltam, hogyan lehet a lehető legkevesebb 0-1 váltással kapcsoló komponens mentessé tenni bináris mátrixokat. Az eredmények egy konferencia kiadványban [42] jelentek meg, illetve egy folyóiratban [41] kerültek elfogadásra.

I/1. Megmutattam, hogy hogyan lehetet a keresési teret drasztikusan csökkenteni úgy, hogy közben optimális megoldást ne veszítsünk el. Ez alapján megadtam két olyan heurisztikát, amik az eddig publikált módszereknél a legtöbb esetben jobbnak bizonyultak a 0-1 váltások számának minimalizálásában.

A második tézispontban *hv*-konvex poliominók és *hv*-konvex kanonikus mátrixok rekonstrukcióját vizsgáltam egy vetületből. Az eredmények két konferenciakiadványban [39, 44], és egy folyóiratcikkben [43] kerültek publikálásra.

II/1. Megadtam egy olyan algoritmust, amely a horizontális vetület méretével lineáris időben megad egy, a vetületet kielégítő *hv*-konvex poliominót. Bebizonyítottam, hogy az eljárás mindig minimális méretű mátrixot állít elő, továbbá megmutattam, hogy az eljárás könnyen módosítható tetszőleges szélességű mátrix előállításához. Ezen felül megadtam egy zárt képletet a megoldások számának pontos meghatározásához tetszőleges mátrixméret esetén, illetve egy rekurzív formulát rögzített mátrixméret esetén.

II/2. Megadtam egy eljárást olyan *hv*-konvex poliominók véletlen generálásához, amelyek egy adott horizontális vetületet kielégítenek. Az eljárás legrosszabb futásideje $O(m^2)$, ahol m a vetület méretét jelöli. Az eljárás módosítható rögzített méretű mátrixok véletlen generálásához.

II/3. Megmutattam, hogyan lehet *hv*-konvex kanonikus mátrixokat előállítani egy vetületből. Az eljárás bizonyítottan mindig minimális szélességű, 8-összefüggő képpel tér vissza.

A harmadik téziscsoportban a rekonstrukció azon változatát vizsgáltam, ahol két vetület mellett a rögzített szerkesztőelemmel előállított morfológiai váz is ismert. A téziscsoport eredményei két konferenciakiadványban [40, 46] és két folyóiratcikkben [47, 45] jelentek meg.

III/1. Bebizonyítottam, hogy poliominók rekonstrukciója két vetület illetve a morfológiai váz ismeretében NP-teljes. Ezenfelül, ha eltekintünk a 4-összefüggőségtől, a probléma továbbra is NP-teljes. Akkor is NP-teljes a probléma, ha a morfológiai váz mellett csak egy vetületnek kell megfelelnie a rekonstrukciós eredménynek.

III/2. Megadtam egy módszert a rekonstrukció függvényminimalizációs problémaként történő leírására. A rekonstrukcióhoz a szimulált hűtés három változatát használtam, és megmutattam, hogy egy elfogadható minőségű rekonstrukció már rövid időn belül, kevés iterációval is előállítható, a probléma nehézsége ellenére.

III/3. Megadtam a *hv*-konvex poliominók egy speciális, parametrikus alosztályát, és bebizonyítottam, hogy a két vetületből és morfológiai vázból történő rekonstrukció egyértelműsége csak a paraméterektől függ.

	[39]	[40]	[41]	[42]	[43]	[44]	[45]	[46]	[47]
I/1.			●	●					
II/1.						●			
II/2.					●				
II/3.	●								
III/1.							●		●
III/2.								●	●
III/3.		●							

B.1. táblázat. A tézispontok és a Szerző publikációinak kapcsolata.

Bibliography

[1] S.V. Aert, K.J. Batenburg, M.D. Rossell, R. Erni, G.V. Tendeloo, *Three-dimensional atomic imaging of crystalline nanoparticles*, Nature 470, 374–377 (2011).

[2] T. Back, *Evolutionary Algorithms in Theory and Practice – Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, New York, 1996.

[3] P. Balázs, *A benchmark set for the reconstruction of hv-convex discrete sets*, Discrete Applied Mathematics, 157, 3447–3456 (2009).

[4] P. Balázs, *Generation and empirical investigation of hv-convex discrete sets*, Lecture Notes in Computer Science, 4522, 344–353 (2007).

[5] P. Balázs, *On the number of hv-convex discrete sets*, Lecture Notes in Computer Science, 4958, 112–123 (2008).

[6] P. Balázs, *Reconstruction of canonical hv-convex discrete sets from horizontal and vertical projections*, Lecture Notes in Computer Science, 5852, 280–288 (2009).

[7] P. Balázs, E. Balogh, A. Kuba, *Reconstruction of 8-connected but not 4-connected hv-convex discrete sets*, Discrete Applied Mathematics, 147, 149–168 (2005).

[8] E. Balogh, A. Kuba, Cs. Dévényi, A. Del Lungo, *Comparison of algorithms for reconstructing hv-convex discrete sets*, Linear Algebra and Its Applications, 339(1–3), 23–35 (2001).

[9] E. Barcucci, S. Brlek, S. Brocchi, *PCIF: An algorithm for lossless true color image compression*, Lecture Notes in Computer Science, 5852, 224–237 (2009).

[10] E. Barcucci, A. Del Lungo, M. Nivat, R. Pinzani, *Medians of polyominoes: A property for the reconstruction*, International Journal of Imaging Systems and Technology, 9, 69–77 (1998).

[11] E. Barcucci, A. Del Lungo, M. Nivat, R. Pinzani, *Reconstructing convex polyominoes from horizontal and vertical projections*, Theoretical Computer Science, 155, 321–347 (1996).

[12] K.J. Batenburg, S. Bals, J. Sijbers, C. Kuebel, P.A. Midgley, J.C. Hernandez, U. Kaiser, E.R. Encina, E.A. Coronado, G.V. Tendeloo, *3D imaging of nanomaterials by discrete tomography*, Ultramicroscopy, 109(6), 730–740 (2009).

[13] J. Baumann, Z. Kiss, S. Krimmel, A. Kuba, A. Nagy, L. Rodek, B. Schillinger, J. Stephan, *Discrete tomography methods for nondestructive testing*, Chapter 14 of [48].

[14] P. van Beek, R. Dechter, *On the minimality and global consistency of row-convex networks*, Journal of the ACM, 42(3), 543–561 (1995).

[15] M.R. Berthold, C. Borgelt, F. Höppner, F. Klawonn, *Guide to Intelligent Data Analysis*, Springer, 2010.

[16] S. Bilotta, S. Brocchi, *Discrete tomography reconstruction algorithms for images with a blocking component*, Lecture Notes in Computer Science, 8668, 250–261 (2014).

[17] H. Blum, *Biological shape and visual science (Part I)*, Journal of Theoretical Biology, 38, 205–287 (1973).

[18] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest, R.E. Tarjan, *Time bounds for selection*, Journal of Computer and System Sciences, 7(4), 448–461 (1973).

[19] S. Brocchi, *A new approach for the reconstruction of object-based images in discrete tomography*, Fundamenta Informaticae, 135(1–2), 43–57 (2014).

[20] S. Brocchi, *An object-based tomographic recostruction algorithm exploiting interest points in image projections*, Proceedings of the 8th International Symposium on Image and Signal Processing and Analysis (ISPA), IEEE, 606–611 (2013).

[21] S. Brocchi, E. Barcucci, *BCIF: Another algorithm for lossless true color image compression*, Lecture Notes in Computer Science, 6636, 372–384 (2011).

[22] S. Brunetti, A. Daurat, *Random generation of Q-convex sets*, Theoretical Computer Science, 347(1–2), 393–414 (2005).

[23] S. Brunetti, A. Del Lungo, F. Del Ristoro, A. Kuba, M. Nivat, *Reconstruction of 4- and 8-connected convex discrete sets from row and column projections*, Linear Algebra and Its Applications, 339(1–3), 37–57 (2001).

[24] V. Černý, *Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm*, Journal of Optimization Theory and Applications, 45(1), 41–51 (1985).

[25] S.K. Chang, *The reconstruction of binary patterns from their projections*, Communications of the ACM, 14, 21–25 (1971).

[26] M. Chrobak, C. Dürr, *Reconstructing hv-convex polyominoes from orthogonal projections*, Information Processing Letters, 69(6), 283–289 (1999).

[27] A. Del Lungo, *Polyominoes defined by two vectors*, Theoretical Computer Science, 127, 187–198 (1994).

[28] A. Del Lungo, E. Duchi, A. Frosini, S. Rinaldi, *Enumeration of convex polyominoes using the ECO method*, Discrete Mathematics and Theoretical Computer Science, Proceedings, 103–116 (2003).

[29] A. Del Lungo, E. Duchi, A. Frosini, S. Rinaldi, *On the generation and enumeration of some classes of convex polyominoes*, The Electronic Journal of Combinatorics, 11, R60 (2004).

[30] A. Del Lungo, M. Nivat, R. Pinzani, *The number of convex polyominoes reconstructible from their orthogonal projections*, Discrete Mathematics, 157(1–3), 65–78 (1996).

[31] Y. Deville, Y., O. Barette, P. Van Hentenryck, *Constraint satisfaction over connected row convex constraints*, Artificial Intelligence 109(1–2), 243–271 (1999).

[32] R.J. Gardner, P. Gritzmann, D. Prangenberg, *On the computational complexity of reconstructing lattice sets from their X-rays*, Discrete Mathematics, 202(1–3), 45–71 (1999).

[33] M.R. Garey, D.S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[34] I. Gessel, *On the number of convex polyominoes*, Annales des Sciences Mathématiques du Québec, 24, 63–66 (2000).

[35] V.D. Gesù, G.L. Bosco, F. Millonzi, C. Valenti, *A memetic algorithm for binary image reconstruction*, Lecture Notes in Computer Science, 4958, 384–395 (2008).

[36] P. Giblin, B.B. Kimia, *A formal classification of 3D medial axis points and their local geometry*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(2), 238–251 (2004).

[37] S.W. Golomb, *Polyominoes*, Charles Scriber's Sons, New York, 1965.

[38] R.C. Gonzalez, R.E. Woods, *Digital Image Processing (3rd Edition)*, Prentice Hall, 2008.

[39] N. Hantos, P. Balázs, *A fast algorithm for reconstructing hv-convex binary images from their horizontal projection*, Lecture Notes in Computer Science, 8888, 789–798 (2014).

[40] N. Hantos, P. Balázs, *A uniqueness result for reconstructing hv-convex polyominoes from horizontal and vertical projections and morphological skeleton*, Proceedings of the 8th International Symposium on Image and Signal Processing and Analysis (ISPA), IEEE Signal Processing Society, 788–793 (2013).

[41] N. Hantos, P. Balázs, *Eliminating switching components in binary matrices by 0-1 flips and columns permutations*, accepted for publication in Fundamenta Informaticae (2015).

[42] N. Hantos, P. Balázs, *Fast heuristics for eliminating switching components in binary matrices by 0-1 flips*, Lecture Notes in Computer Science, 8827, 62–69 (2014).

[43] N. Hantos, P. Balázs, *Random generation of hv-convex polyominoes with given horizontal projection*, Fundamenta Informaticae, 135(1–2), 103–115 (2014).

[44] N. Hantos, P. Balázs, *Reconstruction and enumeration of hv-convex polyominoes with given horizontal projection*, Lecture Notes in Computer Science, 8258, 100–107 (2013).

[45] N. Hantos, P. Balázs, *The reconstruction of polyominoes from horizontal and vertical projections and morphological skeleton is NP-complete*, Fundamenta Informaticae, 125(3–4), 343–359 (2013).

[46] N. Hantos, P. Balázs, K. Palágyi, *Binary image reconstruction from two projections and skeletal information*, Lecture Notes in Computer Science, 7655, 263–273 (2012).

[47] N. Hantos, S. Iván, P. Balázs, K. Palágyi, *Binary image reconstruction from a small number of projections and the morphological skeleton*, Annals of Mathematics and Artificial Intelligence, in press (2015).

[48] G.T. Herman, A. Kuba (Eds.), *Advances in Discrete Tomography and Its Applications*, Birkhäuser, Boston, 2007.

[49] G.T. Herman, A. Kuba, (Eds.), *Discrete Tomography: Foundations, Algorithms and Applications*, Birkhäuser, Boston, 1999.

[50] W. Hochstättler, M. Loebl, C. Moll, *Generating convex polyominoes at random*, Discrete Mathematics, 153(1–3), 165–176 (1996).

[51] R.W. Irving, M.R. Jerrum, *Three-dimensional data security problems*, SIAM Journal on Computing, 23, 170–184 (1994).

[52] K.J. Kearfott, S.E. Hill, *Simulated annealing image reconstruction method for a pinhole aperture single photon emission computed tomograph (SPECT)*, IEEE Transactions on Medical Imaging, 9(2), 128–143 (1990).

[53] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, *Optimization by simulated annealing*, Science 220, 671–680 (1983).

[54] A. Kuba, *Reconstruction in different classes of 2D discrete sets*, Lecture Notes in Computer Science, 1568, 153–163 (1999).

[55] P. Leroux, E. Rassart, *Enumeration of symmetry classes of parallelogram polyominoes*, Annales des sciences mathématiques du Québec, 25(1), 71–90 (2001).

[56] X. Li, L. Ma, *Minimizing binary functions with simulated annealing algorithm with applications to binary tomography*, Computer Physics Communications, 183(2), 309-315 (2012).

[57] T. Lukić, B. Nagy, *Energy-minimization based discrete tomography reconstruction method for images on triangular grid*, Lecture Notes in Computer Science, 7655, 274–284 (2012).

[58] H. Mannila, E. Terzi, *Nestedness and segmented nestedness*, KDD '07 Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, 480–489 (2007).

[59] P. Maragos, R.W. Schafer, *Morphological skeleton representation and coding of binary images*, IEEE Transactions on Acoustics, Speech and Signal Processing, 34(5), 1228–1244 (1986).

[60] T.C. Martins, E.D.L.B. Camargo, L. Gonzales, R.G. Lima, M.B.P. Amato, M.S.G. Tsuzuki, *Image reconstruction using interval simulated annealing in electrical impedance tomography*, IEEE Transactions on Biomedical Engineering, 59(7), 1861–1870 (2012).

[61] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, *Equation of state calculations by fast computing machines*, Journal of Chemical Physics, 21, 1087–1092 (1953).

[62] T.M. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[63] A. Nagy, A. Kuba, *Parameter settings for reconstructing binary matrices from fan-beam projections*, Journal of Computing and Information Technology, 14(2), 100–110 (2006).

[64] H.J. Ryser, *Combinatorial mathematics*, Mathematical Association of America, New York, 1963.

[65] H.J. Ryser, *Combinatorial properties of matrices of zeros and ones*, Canadian Journal of Mathematics, 9, 371–377 (1957).

[66] T. Schüle, C. Schnörr, S. Weber, J. Hornegger, *Discrete tomography by convex-concave regularization and D.C. programming*, Discrete Applied Mathematics, 151, 229–243 (2005).

[67] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, New York, 1982.

[68] L. Shapiro, G. Stockman, *Computer Vision*, Prentice Hall, 2002.

[69] K. Siddiqi, S. Pizer (Eds.), *Medial Representations – Mathematics, Algorithms and Applications*, Computational Imaging and Vision, 37, Springer, 2008.

[70] G.W. Woeginger, *The reconstruction of polyominoes from their orthogonal projections*, Information Processing Letters, 77(5–6), 225–229 (2001).

[71] X. Wu, V. Kumar (Eds.), *The Top Ten Algorithms in Data Mining*, Chapman & Hall/CRC, 2007.

[72] A. Zamuda, J. Brest, *Vectorized procedural models for animated trees reconstruction using differential evolution*, Information Sciences, 278, 1–21 (2014).

Index