Reconstructing hv-convex polyominoes with multiple colours

by

Adam Bains

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2009

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis examines the problem of reconstructing multiple discrete 2D objects, represented by a set of cells arranged in an $m \times n$ grid, from their projections. The objects being constructed are disjoint, hv-convex polyominoes, each of which has a separate colour. The main results presented here are two algorithms for unordered C-colour reconstruction that have time complexities of $O(C^2 n^{2C+1} m^{2C+1})$ and $O(C^2 \min(n^{2C}, m^{2C}) nm)$, an ordered C-colour reconstruction algorithm that is $O(C \min(n^{2C}, m^{2C}) nm)$, and an NP-completeness proof when the number of colours is unbounded.

Acknowledgements

I would like to thank all of my family and friends for their help and support, as well as everyone who helped me edit and polish this thesis into what it is today.

Dedication

This is dedicated to my friends and family, as well as all of the people that supported me throughout the process of writing this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Discrete tomography, for the purposes of this thesis, deals with the problem of reconstructing a 2D object, which is represented as a discrete set of cells, from projections of the object's thickness. In fact, the discrete tomography problem as considered here is similar to a kind of logic puzzle typically referred to as nonograms, although they are known by a number of other names (e.g. paint-by-numbers, pic-a-pix, Edel) [20]. Such puzzles give the reader information about the number of filled-in cells that should be in each row and column of a matrix of cells. A puzzle is solved when a cell colouring that matches these criteria is found.

The field of discrete tomography has recently experienced a large surge of interest. Many papers have been published in the last decade. Some examine the basic problems of the field from new perspectives, while others apply algorithms and techniques from the field to a number of different areas, such as medical imaging [19], electron microscopy [9, 8], data security [8], and image processing [23]. It should be noted that this is by no means an exhaustive list of papers or applications; the collections on the subject by Gabor and Kuba [15, 14] provide a wealth of papers discussing both the algorithmic aspects of the field and possible applications.

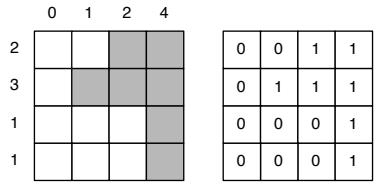


Figure 1.1.1: (a) An instance of a single-colour reconstruction, (b) a $\{0, 1\}$ matrix representation of that object.

This paper examines the subarea of discrete tomography dealing with discrete

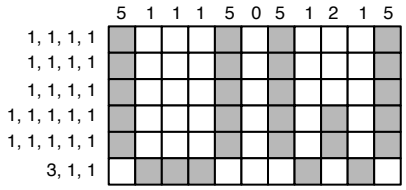


Figure 1.1.2: A simple single-colour nonogram puzzle.

sets, which are represented here as sets of cells within a larger matrix of cells. Traditionally, the cells can be either black or empty. In this thesis, we deal with multiple colours, so the cells can either be empty or have one of many colours. Note, however, that each cell can only ever contain one colour. The problem then becomes to reconstruct these discrete sets from sets of sums (one sum for each colour) over the rows and columns of the matrix of cells. A formal definition of the problem will be given in Section 1.2.

Our problem has two motivations: the reconstruction of polyatomic structures, and multiple colour nonogram puzzles. For the reconstruction of polyatomic structures, applications consider different types of atoms instead of different colours, but the two concepts are identical from an algorithmic viewpoint. A two-dimensional projection of a structure composed of multiple types of atoms can easily be thought of as a picture containing multiple colours, one for each type of atom. There is a technique called QUANTITEM [16, 22] that uses high-resolution transmission electron microscopes to determine the number of atoms in the atom columns of a crystal in certain directions, which are essentially projections. As there are no degrees of thickness—a position within the crystal either contains an atom or it is empty—the problem of reconstructing the crystal from these atom counts is well-suited to discrete tomography, as shown by the consideration other authors have given to both the 3D and the 2D versions of the problem [8].

The multiple-colour nonogram puzzle is another motivation for our work. As noted previously, the single-colour version gives you the number of cells in any given row and column, but with one significant difference: gaps of empty cells between consecutive blocks of filled-in cells are shown in the row and column information for the problem. The exact length of the gap is not provided; all that is known is that a gap of some length exists between blocks of filled-in cells whose lengths are known. Figure 1.1.2 provides an example demonstrating how the row and column information is conveyed. The multiple-colour problem is the same as the single-colour problem, except that the blocks of consecutive cells can be of different colours; for example, you might be given a puzzle where you know that a row must contain five black cells, four red cells, and three green cells. Also, there are not necessarily gaps between colours, whereas in the single-colour version there is always a gap between two groups of filled-in cells. If the picture being considered is known to make use of only hv-convex blocks of colour, the multiple-colour nonogram problem then becomes a variant of the hv-convex reconstruction problem for multiple colours.

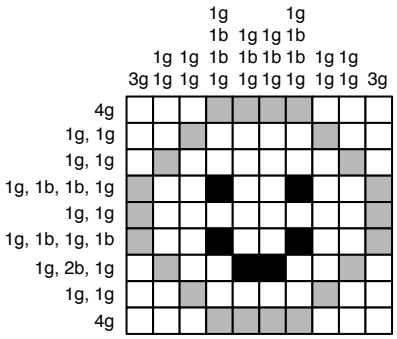


Figure 1.1.3: A simple two-colour nonogram puzzle.

1.2 Definitions

We now give formal definitions. A *projection* is a measurement of the density of an object (or set of objects) taken from a single direction. For our purposes, we will only need to consider reconstruction problems from two projections, one along the x-axis and one along the y-axis of the integer lattice. An *object* is a set of cells of the form i, j, where $1 \leq i \leq m$ and $1 \leq j \leq n$. Such an object is called a 01-matrix $(x_{i,j})$, where $x_{i,j} = 1$ means that cell (i, j) belongs to the object. A *set of objects* is described by a set of colours $\{1, \ldots, C\}$ where each colour is associated with an object $\{x^c_{i,j} \in \{0, 1\} : 1 \leq i \leq m, 1 \leq j \leq n\}$. We will later restrict the objects to be disjoint, so that for any given i and j, the sum over all c of $x^c_{i,j} \leq 1$. Note that since the cells associated with each object can only take on binary values, the entries in it can also be thought of as boolean variables.

A projection for such an object that is of colour c is defined as:

$$h^c_i = \sum_{j=1}^{n} x^c_{i,j}, i = 1, \ldots, m \tag{1.1a}$$

$$v^c_j = \sum_{i=1}^{m} x^c_{i,j}, j = 1, \ldots, n \tag{1.1b}$$

where h^c_i is the density measurement for the i^{th} row of the object, also referred to as the i^{th} *row sum*, and v^c_j is the density measurement for the j^{th} column, also referred to as the j^{th} *column sum*. The sets of all row and column sums for the object of colour c are denoted as $H^c = (h^c_1, \ldots, h^c_m)$ and $V^c = (v^c_1, \ldots, v^c_n)$. In the case of a single-colour reconstruction problem, the c's are omitted from these definitions, as there is only one colour available (black). If $x^c_{i,j} = 0$ for all c, then the cell is said to be white, or empty. Note that white is not considered a colour in this sense, but an absence of colour. It has no row or coloumn sums, nor variables to indicate whether a cell is white or not.

The object of colour c is referred to as P^c. If a cell is known to belong to P^c (i.e. if $x^c_{i,j} = 1$), then $(i, j) \in P^c$. The set of all cells, which includes white cells

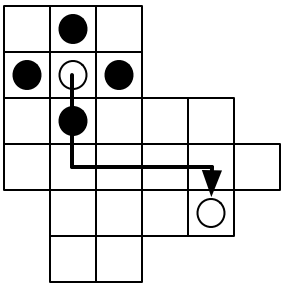


Figure 1.2.4: Example of 4-connectivity. The black circles are 4-neighbours of the empty circle, and the empty circles are shown to be 4-connected by the indicated path

as well as cells belonging to each of the objects, is referred to as the *matrix*. The outermost rows and columns within the matrix are defined as the *boundary rows* and *boundary columns*, and the cells within them are the *boundary cells*.

The reconstruction problem (to be defined formally below) consists of reconstructing a set of disjoint objects from a given set of row/column-sum vectors, possibly under additional constraints. A *solution to the reconstruction problem* is an assignment of colour values to the cells such that the number of coloured cells in each row and column in the matrix is exactly equal to the given row and column sums for each colour. Note that in order to be valid, a solution may have to meet a number of other requirements, depending on the reconstruction problem.

We consider a restricted version of the reconstruction problem, where the objects must be connected and convex in some sense. For each cell (i, j) in an $m \times n$ matrix, the set $N((i, j)) = \{(i + 1, j), (i - 1, j), (i, j + 1), (i, j - 1)\}$ is referred to as the *4-neighbours* of (i, j). Note that cells in the outer rows and columns of the matrix may not have all four of these 4-neighbour cells available. Two cells are said to be *4-connected* if there exists a path between the two cells such that each cell in the path is a 4-neighbour of the next cell in the path. More precisely, $P = (i_1, j_1), \ldots, (i_n, j_n)$ is 4-connected if for all $1 \leq k \leq n$, $(i_k, j_k) \in N((i_{k+1}, j_{k+1}))$. A set is said to be a *polyomino* if all cells in the set are 4-connected. A row (column) is said to be *h-convex* (*v-convex*) if it consists of a contiguous block of cells, i.e. it is 4-connected. A polyomino is said to be *h-convex* (*v-convex*) if all of its rows (columns) are h-convex (v-convex), and *hv-convex* if it is both h-convex and v-convex [3]. [1] Note that if a row (column) is h-convex (v-convex), then it is 4-connected, as it is easy to draw a valid path between any two cells in the row (column). This thesis deals largely with the reconstruction of *hv-convex polyominoes*. The problem definition for the hv-convex reconstruction problem is as follows:

Problem: hv-Convex Polyomino Reconstruction from Orthogonal Projections

[1] Being *hv-convex* is the same as being *orthogonally convex*, which is a term that is sometimes used in Computational Geometry, but not traditionally in Discrete Tomography.

Given: Row and column sum vectors $H = (h_1, \ldots, h_m), V = (v_1, \ldots, v_n)$.

Goal: Construct an hv-convex polyomino contained in an $m \times n$ matrix that satisfies the given row and column sums, i.e., row i has exactly h_i black cells and column j has exactly v_i black cells.

In the process of developing a reconstruction, we will also develop a *boolean assignment* using 2-SAT that assigns boolean values to variables representing the cells. As noted earlier, the cells of the various objects being constructed are analogous to boolean variables, so the boolean assignment and the actual reconstruction are largely interchangeable. In other words, the boolean assignment from the 2-SAT can be thought of as a possible solution for the instance of the reconstruction problem it is based on.

We will also have cause to refer to the unrestricted reconstruction problem, which is formally defined here:

Problem: Unrestricted Reconstruction from Orthogonal Projections

Given: Row and column sums $H = (h_1, \ldots, h_m), V = (v_1, \ldots, v_n)$.

Goal: Fill the cells of an $m \times n$ matrix in a manner that satisfies the given row and column sums, i.e., all rows i have exactly h_i black cells and all columns have exactly v_i black cells.

Lastly, our results make use of the k-satisfiability problem in a number of ways. The basic problem definition for k-SAT is as follows:

Problem: k-Satisfiability (k-SAT)

Given: A set X of variables, and a set $\mathbb{C}$ of m clauses such that each clause $c_i \in \mathbb{C}$ is a disjunction of k literals, e.g., $c_i = (x_1 \vee \overline{x_2} \vee \ldots x_k)$

Goal: To assign boolean values to all $x_i \in X$ such that every clause $c_j \in C$ is satisfied, i.e., each clause contains a true literal.

Note that when considering the number of variables and clauses, we refer to them as $|X|$ and $|C|$. This is because m and n, which are typically used to refer to these values, are already used to represent the number of rows and columns in the matrices we consider. Two different versions of the satisfiability problem are made use of in this thesis: 3-SAT and 2-SAT. The 2-SAT problem is solvable in polynomial time, whereas the 3-SAT problem is known to be NP-complete.

1.3 Related Results

Discrete tomography problems were being discussed as far back as 1957, when Ryser published a paper that provided the first reconstruction algorithm for the unrestricted single-colour reconstruction problem. [21] (see also the first chapter of Herman and Kuba [15].) With those preliminary results in place, other authors began to examine variants of the single-colour reconstruction problem. Woeginger proved that two problems were NP-complete: reconstructing a 4-connected object, and reconstructing an hv-convex object [24]. Barcucci et al. presented NP-completeness proofs for h-convex objects, v-convex objects, h-convex polyominoes, and v-convex polyominoes [3]. Note that an h-convex object is an object that is not 4-connect where all rows of the object are h-convex, an h-convex polyomino is a 4-connected object where all rows of the polyomino are h-convex, and so forth.

Surprisingly, if an object is required to be both hv-convex and 4-connected, the reconstruction problem becomes polynomial. Barcucci et al. developed an algorithm for the reconstruction of hv-convex polyominoes that has a total time complexity of $O(n^3m^3(n+m))$ [3]. This time was subsequently improved upon by Chrobak and Dürr, who published an $O(mn\min(m^2,n^2))$ time algorithm [7]. Both of these algorithms will be discussed in greater detail in Section 1.4, as their results are the groundwork for the material presented in this thesis. A special case of the reconstruction problem for hv-convex polyominos was also presented by Chrobak and Dürr, where the polyomino has at least one row (column) that is n (m) cells long, i.e., it spans the length (width) of the matrix; they presented a modified algorithm that could solve this special case in $O(m+n)$ time [7].

Some researchers have been examining further restricted variants of the polyomino reconstruction problem. Balázs developed an algorithm for the reconstruction of objects that are 8-connected but not 4-connected, and arrives at a total time complexity of $O(mn\min(m,n))$ [2]. 8-connectivity is similar to 4-connectivity, except it includes cells $(i-1,j-1),(i-1,j+1),(i+1,j+1)$, and $(i+1,j-1)$ as 8-neighbours of (i,j). Castiglione and Restivo examined so-called L-convex polyominoes and found that such polyominoes can be uniquely determined by their row and column sums, as well as providing an efficient algorithm for their reconstruction [5], [6]. Exact time complexity was not given in their papers. Similarly, Duchi et al. examined the class of so-called Z-convex polyominoes, although they provided a generating function for the class of polyominoes and not an algorithm to reconstruct them more efficiently [10]. These convexity classes further restrict the shape of a valid solution; further details are not relevant to this thesis, but can be found in the relevant papers.

Thus far, all results mentioned have dealt with single-colour reconstruction. For more colours, all NP-hardness results carry over, and few algorithmic results are known. A number of NP-hardness results specific to higher numbers of colours are known, however. Gardner et al. showed that the unrestricted reconstruction problem was NP-complete for 6 colours, and Chrobak and Dürr presented an NP-

hardness proof for 3 colours [13, 8]). As this result can be extended to 4 and 5 colours, only 2 colours remains an open problem for the unrestricted case.

Further, other researchers have examined reconstruction problems for objects in more than two dimensions, as well as reconstructions from more than two projections. Balázs examined the class of decomposable discrete sets and provided a single-colour reconstruction algorithm for two dimensions and four projections [1]; Batenburg developed an exact algorithm for the single-colour reconstruction problem in 3D with two projections, and tested an algorithm for reconstruction with three or more projections. Batenburg's results showed that, despite the NP-hardness of the problem, it was still feasible to reconstruct relatively smooth objects from a small number of projections [4]. Gardner et al. showed that the single-colour unrestricted reconstruction problem was NP-hard where the object considered is 3-dimensional or higher and there are three or more non-parallel projections used; in the same paper, they also showed that the same problem is polynomial for all other values of dimensions and projections [12].

1.4 Review of existing algorithms

1.4.1 Overview

As mentioned previously, there are two papers that are particularly important to understand when considering the work presented in this thesis. The first, "Reconstructing convex polyominoes from horizontal and vertical projections" by Barcucci et al., presents an early approach to reconstruction of hv-convex polyominoes [3]. Note, however, that we follow the review of this algorithm by Del Lungo and Nivat for the sake of clarity [17]. Although the paper by Barcucci et al. has since been improved upon, their work makes use of an idea that is used by the algorithms presented here. The second paper is "Reconstructing hv-convex polyominoes from orthogonal projections" by Chrobak and Dürr [7]. This paper provides a much faster solution than the one by Barcucci et al., but is not well-suited to a multiple-colour reconstruction problem. Nevertheless, the method they use to develop their reconstruction, which is to make use of the empty space around the polyomino to reconstruct it, can be partially adapted to a multiple-colour reconstruction problem, as we will show later in the algorithms we present.

1.4.2 Barcucci et al. – Reconstructing hv-convex polyominoes via spine expansion

The first method of reconstructing hv-convex polyominoes that we will examine is the one by Barcucci et al. [17]. Roughly, the algorithm consists of the following steps: first, they find a set of cells that must belong to the polyomino being reconstructed. Next, they expand this set of cells as much as possible, using the

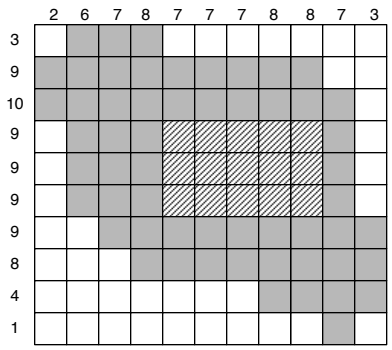


Figure 1.4.5: An example of a polyomino's feet. The gray cells are those that belong to the feet, and the dashed cells are those that belong to the rest of the polyomino.

basic constraints of the problem to determine more cells that must belong to the polyomino. Finally, they phrase the resulting reconstruction as a 2-SAT problem to develop colour assignments for "ambiguous" cells, which could be either black or white.

Foot configuration and spine

The main idea of the algorithm involves the *feet* of a polyomino. There are four feet in a polyomino, each of which corresponds to one of the boundary rows and columns of the matrix. Note that we can assume all boundary rows and columns have non-zero sums, as such a column could be safely ignored when developing any solution. Now, consider the north foot, i.e. the foot associated with the top row of the matrix. Each of the h_1 black cells in this row are considered to be part of the foot. As these cells are at the top of the matrix, we know that each one of these cells is the first black cell in its column. By v-convexity, each column j associated with a cell from the foot has a block of black cells with length v_j extending from the top of the matrix. Since these cells must then be in the polyomino as well, we will also include them in our definition of the foot. See Figure 1.4.5 for a visual example. Note that we assume that the row and column sums we are given are all non-zero; this is because any boundary rows or columns that cannot contain cells of the polyomino may simply be ignored.

The paper refers to the top, bottom, left, and right feet as the North, South, West, and East feet respectively, with the columns occupied by the north foot referred to as $n_1, \ldots, n_2$, the columns associated with the south foot referred to as $s_1, \ldots, s_2$, and so forth. A *foot configuration* is a partial solution where positions for each of the feet have been selected. The north foot has $n - h_1 + 1$ possible positions, the west foot has $m - v_1 + 1$, the south has $n - h_m + 1$, and the east has $m - v_n + 1$. This results in a total of $O(n^2m^2)$ possible foot configurations.

Throughout this thesis, unless stated otherwise, we are dealing with a descending polyomino, a type of polyomino discussed by Duchi et al. [10]. In order to

categorize a polyomino as either ascending or descending, the row index of the southernmost cell in the west foot (w_2) is compared to the position of the east foot. If the east foot begins beneath w_2, then the polyomino is classified as descending; if it ends above w_2, the polyomino is ascending. More precisely, if $w_1 \leq w_2 < e_1 \leq e_2$ the polyomino is descending, and if $e_1 \leq e_2 < w_1 \leq w_2$, it is ascending.

We will ignore the cases $e_1 \leq w_1 \leq e_2 \leq w_2$ and $w_1 \leq e_1 \leq w_2 \leq e_2$, as those necessarily have a *spanning row* – a row i such that $h_i = m$ – and as such can be solved easily, as shown by Chrobak and Dürr [7]. We similarly assume that there is no overlap between the north and south feet, again because it would mean that the polyomino would have a spanning column, and could be detected and solved quickly. We will also require that $n_1 \leq s_2$ for the sake of simplicity; Barucci et al. handle the case where $s_2 < n_1$ as well [17], but since it is not used by the faster of the two multiple-colour reconstructions we will present, the details have been omitted here.

Thus, we assume from now on that in the chosen foot configuration we have $w_1 < e_2$; the other case is handled in a manner similar to the one we will describe. The next step is to define a *spine*, which requires the following technical definition for sets S_{WE} and S_{NS}:

Definition. *Let $H_i = \sum_{a=1}^{i} h_a$ be the i^{th} cumulative row sum, $V_j = \sum_{b=1}^{j} v_b$ be the j^{th} cumulative column sum. We define S_{WE} and S_{NS} as sets of cells satisfying the following properties:*

$$S_{WE} = \{(i,j) | w_1 \leq i \leq e_2, V_j \geq H_{i-1}, H_i \geq V_{j-1}\} \tag{1.2}$$

$$S_{NS} = \{(i,j) | n_1 \leq j \leq s_2, V_j \geq H_{i-1}, H_i \geq V_{j-1}\} \tag{1.3}$$

These sets consist of cells that must be in any valid solution to the reconstruction problem given the current foot configuration, with S_{NS} containing all such cells in columns between the north and south feet, and S_{WE} containing those between the west and east feet. We also define $S = S_{NS} \cup S_{WE}$ as the set of all cells satisfying either set of properties. The *NS-spine* is then defined as the union of S_{NS} with the north and south feet, the *WE-spine* is defined as the union of S_{WE} with the west and east feet, and the *spine* is defined as the union of the NS-spine and the WE-spine. Our goal is to show that the NS-spine contains a cell in every row of the matrix and is 4-connected. Also, we will demonstrate that for any foot configuration, the cells in S_{NS} must belong in any solution with that foot configuration; the same holds for S_{WE} and S. Most of this section will be spent presenting properties and lemmas to work our way toward these goals.

Our lemmas are slightly different that those presented by Barcucci et al., as we will need to refer to the results for the NS- and WE-spines later, and they phrased their proofs in terms of the spine. For this reason, we present full proofs here.

For each column $n_1 \leq j \leq s_2$, we define two row indices d_j and u_j. Index d_j is the highest row index where (1.3) holds, i.e., $(i,j) \notin S_{NS}$ for $i > d_j$, and u_j is the

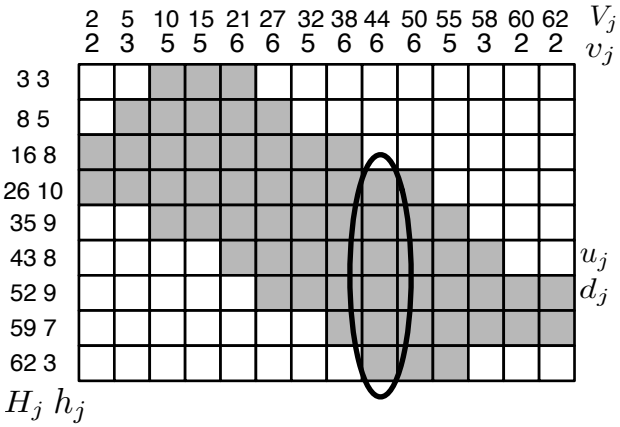


Figure 1.4.6: Example illustrating d_j and u_j for a descending polyomino where $n_1 \leq j \leq s_2$. The row indices given for d_j and u_j are associated with the circled column.

lowest row index where (1.3) holds, i.e., $(i, j) \notin S_{NS}$ for $i < u_j$. See Figure 1.4.6 for an example. More formally, for d_j and u_j, we know the following:

$$V_j \geq H_{i-1} \text{ for each } i \leq d_j, \text{ and } V_j < H_{d_j} \tag{1.4a}$$

$$H_i \geq V_{j-1} \text{ for each } i \geq u_j, \text{ and } H_{u_j-1} < V_{j-1} \tag{1.4b}$$

Note that it is possible for $d_j = u_j$, but it can never be the case that $d_j < u_j$, as this would require that $V_j < H_{d_j} \leq H_{u_j-1} < V_{j-1}$. Also, if $n_1 = 1$ and/or $s_2 = n$, d_j and u_j will not exist for n_1 and s_2. These cases will not pose a problem, however, as the spines we discuss include the foot cells in addition to those in S_{NS} and S_{WE}; even if S_{NS} does not have a cell in column n_1, the NS-spine will. Lastly, similar indices (e.g. r_i, l_i) can be defined for rows $w_1 \leq i \leq e_2$, and the following proofs regarding d_j, u_j, and S_{NS} hold for these indices as well. However, due to the similarity to the proofs presented for d_j and u_j, the details for l_i and r_i will be omitted.

Lemma 1.4.1. *The indices d_j and u_j must exist for every column $j \neq 1, n$ where $n_1 \leq j \leq s_2$. Furthermore, $(i, j) \in S_{NS}$ if and only if $n_1 \leq j \leq s_2$ and $u_j \leq i \leq d_j$.*

Proof. We know that $H_m \geq V_{j-1}$ for $n_1 \leq j \leq s_2$, as H_m includes all black cells in the matrix. Similarly, we know that $H_0 = 0 < V_{j-1}$. Therefore, $1 \leq u_j \leq m$ must exist for $n_1 \leq j \leq s_2$. As for d_j, $V_j \geq H_0$ and $V_j < H_m$ for all $n_1 \leq j \leq s_2$, so d_j must exist as well.

If $u_j \leq i \leq d_j$, we know that $V_j \geq H_{i-1}$ and $H_i \geq V_{j-1}$ by the definition of d_j and u_j. These are exactly the properties that ensure that $(i, j) \in S_{NS}$ if $n_1 \leq j \leq s_2$, allowing us to conclude that $(i, j) \in S_{NS}$. Similarly, if $(i, j) \in S_{NS}$, then $V_j \geq H_{i-1}$ implies $i \leq d_j$ and $H_i \geq V_{j-1}$ implies $i \geq u_j$ □

Lemma 1.4.2. *The NS-spine is 4-connected.*

Proof. Assume $j \neq 1, n$. The j^{th} column of S_{NS} consists of all cells (i, j) with $u_j \leq i \leq d_j$, so it must be 4-connected. Now, consider column index $j+1$, where $j+1 \leq s_2$. From the definition of d_j, u_j, d_{j+1}, and u_{j+1}, we can conclude that:

$$H_{u_j-1} < V_{j-1} < V_j \leq H_{u_{j+1}} \tag{1.5a}$$
$$H_{u_{j+1}-1} < V_j < H_{d_j} \tag{1.5b}$$
$$H_{d_j-1} \leq V_j < V_{j+1} < H_{d_{j+1}} \tag{1.5c}$$

Expression 1.5a follows from the requirement that $H_{u_j-1} < V_{j-1}$ for column j, and $H_i \geq V_{j-1}$ for column j. However, the second requirement is expressed in terms of $j+1$ in expression 1.5a, so it is instead written as $V_j \leq H_{u_{j+1}}$. From this, we know that $u_j \leq u_{j+1}$. Expression 1.5b applies the requirement $H_{u_j-1} < V_{j-1}$ to row $j+1$, giving us $H_{u_{j+1}-1} < V_j$, and $V_j < H_{d_j}$ applies the requirement that $V_j < H_{d_j}$ to column j, thereby demonstrating that $u_{j+1} \leq d_j$. Together with our previous statement, this allows us to conclude that $u_j \leq u_{j+1} \leq d_j$. Lastly, 1.5c's leftmost inequality, $H_{d_j-1} \leq V_j$, follows directly from the definition of d_j, the middle inequality follows from the definition of the cumulative column sum, as all columns must have positive non-zero column sums, and the rightmost inequality also follows from the definition of d_{j+1}, allowing us to conclude that $d_j \leq d_{j+1}$. In combination with our previous statements, this implies that $u_j \leq u_{j+1} \leq d_j \leq d_{j+1}$.

This implies that set S_{NS} is 4-connected. Since $u_j \leq u_{j+1} \leq d_j$, we know that column j is 4-connected to column $j+1$, and we can extend this reasoning to cover all columns k where $n_1 \leq j < k \leq s_2$, implying overall 4-connectivity. Since this covers columns that are included in the north and south feet, which are themselves 4-connected, the union of the north and south feet with S_{NS} is 4-connected as well. Therefore, the NS-spine must be a polyomino.

If $j = 1$ or n, i.e., $n_1 = 1$ or $s_2 = n$, the proof is very similar. As mentioned earlier, d_j and u_j will not exist for columns $j = n_1 = 1$ and $j = s_2 = n$. Since a cell (i, j) is in S_{NS} if and only if $u_j \leq i \leq d_j$, these columns of S_{NS} will be empty; thus, S_{NS} is 4-connected by the previous case. There are then a few possibilities to consider: the case where $n_1 = n_2$ and/or $s_1 = s_2$, and the case where $n_1 \neq n_2$ and/or $s_1 \neq s_2$. The second situation is trivial, as S_{NS} will contain cells in the spine and as such the NS-spine must be 4-connected. In the first situation, let (α_j, j) be the uppermost cell of the south foot. We know that $V_{n-1} = V_n - v_n$, which is to say it is missing exactly the cells of the south foot. As such, it must be the case that $d_j > \alpha_j$ in order for $V_j < H_{d_j}$ to hold, which implies that the NS-spine is 4-connected. A similar proof holds for the case where $j = n_1 = 1$. □

Lemma 1.4.3. *The NS-spine has at least one black cell in every row of the matrix.*

Proof. Recall that a cell is referred to as being black if it is in P, and white otherwise. The north foot contains cells in row 1, and the south foot contains cells in row m. As the NS-spine is 4-connected, it must be possible to draw a 4-connected path between row 1 and row m using only the cells in the NS-spine, and such a

path would need to pass through every row of the matrix. As such, the NS-spine must contain at least one black cell in every row of the matrix. □

A similar process can be used to prove that the WE-spine is 4-connected and contains a black cell in every column. As such, the union of $S_{NS} \cup S_{WE}$ with the feet results in a polyomino with cells in every row and column.

Next, we will show that all cells in S_{NS} must belong to any valid solution to the reconstruction problem using the current foot configuration. Before proving this, a few preliminary lemmas and definitions must be presented. Let $NW(i,j) = \{(h,k) : 1 \leq h \leq i, 1 \leq k \leq j\}$, $NE(i,j) = \{(h,k) : 1 \leq h \leq i, j \leq k \leq n\}$, $SW(i,j) = \{(h,k) : i \leq h \leq m, 1 \leq k \leq j\}$, and $SE(i,j) = \{(h,k) : i \leq h \leq m, j \leq k \leq n\}$. These are four quadrants of the matrix relative to (i,j).

Lemma 1.4.4. *Let (i,j) be a cell in the matrix, and let P be a valid solution.*

1. *If $V_j \geq H_{i-1}$, then $SW(i,j) \cap P \neq \varnothing$*
2. *If $H_i \geq V_{j-1}$, then $NE(i,j) \cap P \neq \varnothing$*
3. *If $H_i \geq A - V_j$, then $NW(i,j) \cap P \neq \varnothing$*
4. *If $A - V_{j-1} \geq H_{i-1}$, then $SE(i,j) \cap P \neq \varnothing$*

Proof. Assume that $SW(i,j) \cap P = \varnothing$. Let $W(g)^P$ be the set of cells belonging to P in all columns to the left of and including column g (i.e. $W(g)^P = \{(i,j) : (i,j) \in P, j \leq g\}$), and let $N(i)^P$ be the set of all cells belonging to P in rows above and including i. If $SW(i,j) \cap P = \varnothing$, then $W(j)^P$ must be a subset of $N(i)^P$. This is because $W(j)^P - N(i-1)^P \subseteq SW(i,j)$ and $SW(i,j) \cap P = \varnothing$. Also, if $SW(i,j) \cap P = \varnothing$, then no path connecting a black cell in $NW(i,j)$ to a black cell in $SE(i,j)$ can pass through $SW(i,j)$; instead, all such paths must pass through $NE(i,j)$. As only $N(i-1)^P$ will contain cells (h,k) such that $1 \leq h \leq i$ and $j < k \leq n$, $W(j)^P$ must be a proper subset of $N(i-1)^P$.

Since $W(j)^P$ contains exactly V_j black cells and $N(i-1)^P$ contains H_{i-1} black cells, we then know that $V_j < H_{i-1}$. As such, if $V_j \geq H_{i-1}$ holds, then $SW(i,j) \cap P \neq \varnothing$. Similar arguments apply for the other three properties. □

Lemma 1.4.5. *Let P be a valid solution to the reconstruction problem. Cell (i,j) is in P if and only if $SW(i,j) \cap P \neq \varnothing, NW(i,j) \cap P \neq \varnothing, NE(i,j) \cap P \neq \varnothing$, and $SE(i,j) \cap P \neq \varnothing$, i.e. all four quadrants contain at least one black cell.*

Proof. Assume all four quadrants contain at least one black cell, i.e. they have a non-empty intersection with P. Next, assume (i,j) is white. Then, by h-convexity, one of the sets $\{(i',j) : i' \geq i\}$ or $\{(i',j) : i' \leq i\}$ must be white; assume the former. Similarly, by v-convexity one of the sets $\{(i,j') : j' \geq j\}$ or $\{(i,j') : j' \geq j\}$ must be white; again, assume the former is white. Therefore, all cells in $L = \{(i',j) : i' \geq i\} \cup \{(i,j') : j' \geq j\}$ are white. The cells in L correspond exactly to the cells along

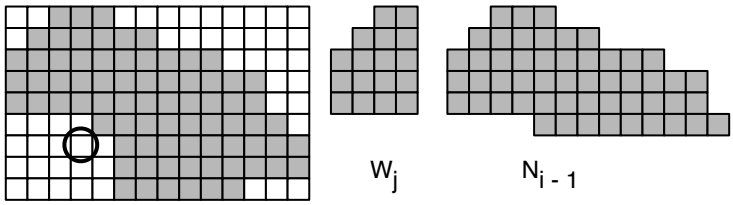
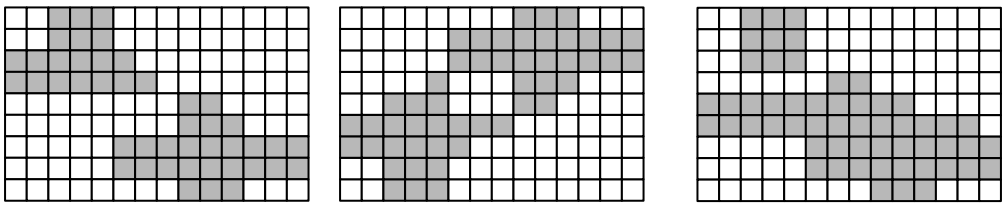


Figure 1.4.7: An example demonstrating the relationship between $SW(i,j), W_i$, and N_{i-1}.



(a) (b)

Figure 1.4.8: (a) Two examples of consecutive pairs of feet with non-empty intersections. (b) A violation of vertical convexity as a result of disconnected feet.

the border of $NE(i,j)$. Therefore, there is no path of black cells that could connect a cell in $NE(i,j)$ to a cell in any other region of the matrix, thereby contradicting the 4-connectivity requirements.

Next, assume $(i,j) \in P$; as $(i,j) \in NE(i,j), NW(i,j), SW(i,j), SE(i,j)$, all four quadrants have a non-empty intersection with P. Therefore, we have proven that $(i,j) \in P$ if and only if all four quadrants contain at least one black cell. □

Lemma 1.4.6. *Let P be a valid solution to the reconstruction problem. Then $S_{NS} \subseteq P$.*

Proof. Let $(i,j) \in S_{NS}$, and recall that $n_1 \leq i \leq s_2$. By the definition of S_{NS}, this implies that $V_j \geq H_{i-1}$ and $H_i \geq V_{j-1}$. Then $NW(i,j) \cap P \neq \varnothing$ and $SE(i,j) \cap P \neq \varnothing$, as they contain the north and south feet, respectively. By $V_j \geq H_{i-1}$, $SW(i,j) \cap P \neq \varnothing$, and by $H_i \geq V_{j-1}$, $NE(i,j) \cap P \neq \varnothing$ (see Lemma 1.4.4). Therefore, (i,j) must be in P by Lemma 1.4.5. □

Similarly, if considering a cell (i,j) where $w_1 \leq i \leq e_2$, we know $NE(i,j), SW(i,j) \cap P \neq \varnothing$ from the properties of cells in S_{WE}, $NW(i,j)$ contains the west foot, and $SE(i,j)$ contains the east foot. As such, if $(i,j) \in S_{WE}$, it must be in P. This allows us to conclude that if $(i,j) \in S$, $(i,j) \in P$.

Based on all of the material presented in this section, we can conclude that the NS-spine is 4-connected, contains a cell in every row, and that every cell in it must

be black in any solution to the reconstruction problem. The same holds for the WE-spine with similar arguments.

With regards to the algorithm presented by Barcucci et al., we now consider how to generate the spine. In order to develop the spine, valid positions for the feet need to be found, and the algorithm tests each of the $O(n^2m^2)$ possible foot configurations to determine whether or not it is valid. The authors say that a foot position is valid if the two consecutive pairs of feet have a non-empty intersection. A valid foot position is required in order to allow for hv-convexity; if e.g. the north foot had an empty intersection with both the west and east feet, then there would necessarily be a violation of vertical convexity. See Figure 1.4.8 for an example. Validity of any given foot position can be checked in $O(m+n)$ time by examining each row and column associated with each of the feet to determine whether or not there is an intersection with an adjacent foot. The spine for a given foot configuration can be generated in $O(nm)$ time: the values H_i and V_j for all i and j can be precomputed in $O(m+n)$ time, and each of the $n \times m$ cells can be checked in constant time to see whether it satisfies the properties of S_{NS} or S_{WE} (Definition on page 9) .

Polyomino expansion

Once a spine is found, Barcucci et al.'s algorithm proceeds to attempt to generate the solution (P) by constructing three subsets of the remaining cells: a set of cells definitely belonging to P called the *kernel*, a set of ambiguous cells called the *shell*, and the remaining cells that definitely do not belong to P. At the beginning of the algorithm's execution, the kernel $\mathbb{K}$ is defined as the union of the NS-spine and the WE-spine, and the shell $\mathbb{S}$ is the entire matrix. In other words, the kernel is initialized as the set of all cells we know must be in P, while the shell initially contains *all* cells in the matrix. Note that, although the shell is defined as containing ambiguous cells, it also contains the cells of the kernel; as such, it can also be thought of as the kernel, with the addition of some cells that may or may not be black in a valid solution.

After the initial values of the shell and the kernel have been found, their algorithm proceeds to attempt to expand the kernel and contract the shell through the use of filling operations. These operations slowly expand the kernel and remove cells of the matrix that cannot be in P from the shell, causing it to contract. The operations themselves take a constant amount of time, which means that every time the rows and columns of the matrix are processed, it takes $O(n+m)$ time. Since the paper guarantees that at least one cell will be added to the kernel or removed from the shell with every pass, at most $O(nm)$ passes can be performed, giving this portion of the algorithm time complexity $O(nm(n+m))$. These concepts were originally used to provide a base for the 2-SAT reconstruction technique that Barcucci et al. used to develop a solution. Our algorithm makes use of a different 2-SAT reconstruction technique, and as such does not make use of any of these operations

or concepts. As such, we will not review the polyomino expansion operations in detail.

Reconstruction via SAT

The algorithm then goes on to reconstruct the polyomino by applying a 2-satisfiability (2-SAT) construction to the expanded spine. When applying 2-SAT to the spine, Barcucci et al. identify "cycles" of ambiguous cells where if one cell is set to black, another cell must be set to white, which requires a further cell to be black, and so on. Our algorithm skips the expansion step entirely, and uses the 2-SAT to express the basic constraints of the problem—4-connectivity and hv-convexity—thereby removing the need to expand the spine or search for cycles of ambiguous cells. As such, the details of the 2-SAT used by Barcucci et al. have been omitted; an examination of the 2-SAT that we make use of will follow in Section 1.4.3.

Time complexity

Since the filling operations have a worst-case time complexity of $O(nm(n+m))$, and can be performed on any or all of the $O(n^2m^2)$ possible foot configurations, the algorithm has a total time complexity of $O(n^3m^3(n+m))$.

1.4.3 1-colour 2-SAT

The process of expanding the spine and searching for cycles of ambiguous cells that is used by Barcucci et al. is useful in a single-colour setting, but becomes cumbersome when considering a multiple-colour problem. Instead of directly adapting their algorithm and the idea of expanding the spine, we make use of a different 2-SAT reconstruction technique. This section contains the details for a simplified, 1-colour version of this 2-SAT technique, which will be used in Section 2.5.

2-SAT is known to be solvable in $O(|X| + |\mathbb{C}|)$ time. It can be applied to the 1-colour reconstruction problem by considering each individual cell (i, j) as a boolean variable, and the colours black and white as $x_{i,j}$ and $\overline{x_{i,j}}$. For each row i, let α_i be a column index chosen such that cell (i, α_i) belongs to the spine. We choose cells belonging to the spine because they are known to exist in every row and column of the matrix, and it has been shown that they must belong to any solution using the foot configuration that they are generated from in Lemmas 1.4.3 and 1.4.6. To ensure that a solution derived from the 2-SAT instance is 4-connected

and hv-convex, we place the following restrictions for each row i:

$$\overline{x_{i,j}}, \forall j \leq \alpha_i - h_i \tag{1.6a}$$
$$\overline{x_{i,j}}, \forall j \geq \alpha_i + h_i \tag{1.6b}$$
$$x_{i,j} \iff \overline{x_{i,j+h_i}}, \forall j \in \{\alpha_i - h_i + 1, \ldots, \alpha_i - 1\} \tag{1.6c}$$
$$x_{i,j} \Rightarrow x_{i,j+1}, \forall j \in \{\alpha_i - h_i + 1, \ldots, \alpha_i - 2\} \tag{1.6d}$$
$$x_{i,j} \Rightarrow x_{i,j-1}, \forall j \in \{\alpha_i + 2, \ldots, \alpha_i + h_i - 1\} \tag{1.6e}$$
$$x_{i,\alpha_i} \tag{1.6f}$$

Lemma 1.4.7. *Clauses 1.6a - 1.6f ensure that row i is h-convex and will contain exactly h_i black cells.*

Proof. Let j_1 (j_2) be the smallest (largest) column index such that $x_{i,j} = 1$. Then $j_1 \leq \alpha_i \leq j_2$ by 1.6f, and by 1.6d and 1.6e, $x_{i,j} = 1$ for all $j_1 \leq j \leq j_2$, guaranteeing h-convexity. Each black cell to the left of column α_i forces the cell h_i columns to its right to be white (by Clause 1.6c), so the row must contain at most h_i black cells, since it is h-convex. Clause 1.6c simultaneously guarantees that there must be at least h_i black cells in the column, because for every white cell to the right of α_i, the cell h_i columns to its left must be black. □

We also know that any valid solution to the reconstruction problem must satisfy the clauses, but as the proof is trivial, it has been omitted. These clauses only express restrictions for the rows. In order to ensure hv-convexity, we also need to add similar restrictions for each column j. Let (α_j, j) represent an arbitrary cell in column j of the spine:

$$\overline{x_{i,j}}, \forall i \leq \alpha_j - v_j \tag{1.7a}$$
$$\overline{x_{i,j}}, \forall i \geq \alpha_j + v_j \tag{1.7b}$$
$$x_{i,j} \iff \overline{x_{i+v_j,j}}, \forall i \in \{\alpha_j - v_j + 1, \ldots, \alpha_j - 1\} \tag{1.7c}$$
$$x^r_{i,j} \Rightarrow x_{i+1,j}, \forall i \in \{\alpha_j - v_j + 1, \ldots, \alpha_j - 2\} \tag{1.7d}$$
$$x_{i,j} \Rightarrow x_{i-1,j}, \forall i \in \{\alpha_j + 2, \ldots, \alpha_i + h_i - 1\} \tag{1.7e}$$
$$x_{\alpha_j,j} \tag{1.7f}$$

The purpose of each clause here mirrors that of the corresponding clause in Equations 1.6a–1.6f, and as such they guarantee that each column j is v-convex and has exactly v_j black cells. Overall 4-connectivity follows trivially, as the rows and columns being built are extensions of the spine, which is already known to be hv-convex and 4-connected.

1.4.4 Chrobak and Dürr – Reconstructing hv-convex polyominoes directly from 2-SAT

In 1999, Chrobak and Dürr presented an improved algorithm for the reconstruction of hv-convex polyominoes that has a total time complexity of $O(mn \min(m^2, n^2))$, a significant improvement over the previous algorithm [7].

The algorithm presented by Chrobak and Dürr avoids the process of developing a spine for the polyomino being reconstructed, and instead moves directly to a 2-SAT problem instance. Before generating the 2-SAT problem, however, they make use of an idea that is similar to the feet of the polyomino: the anchor cell. An hv-convex polyomino P is *anchored* at rows k and l if the cells $(k, 1)$ and (l, n) are elements of P; k and l are referred to as the *anchor rows*. This is similar to the idea of the feet of the polyomino in that it looks for cells on the boundary of the matrix, but only two rows are used: one from what would be the East foot, and one from the West. So, where Barcucci et al. examined $O(m^2n^2)$ possible configurations, Chrobak and Dürr only have to check $O(\min(m^2, n^2))$ possible positions for the anchor rows.[2]

In addition to the anchor rows, the algorithm also defines four areas in the matrix: the upper left, upper right, lower left, and lower right corner regions, which are denoted as A, B, C, and D. These regions represent the empty space that may or may not exist in the corners of the matrix, with A representing the upper-left corner, B the upper-right, C the lower-left, and D the lower-right. These regions do not have to exist; for example, a polyomino where $H_m = V_n = mn$ would have no white cells, and as such would have empty corner regions. If cell (i, j) is an element of A, then $(i-1, j) \in A$ and $(i, j-1) \in A$, which implies that $NW(i, j) \in A$, in the terminology of the Barcucci et al. algorithm. Similar rules apply to the other three corner regions. The hv-convex polyomino P is defined as being the negation of the union of all four regions, i.e., $P = \overline{A \cup B \cup C \cup D}$. Also note that A, B, C, and D are disjoint and satisfy the following properties:

$$(i-1, j-1) \in A \text{ implies } (i, j) \notin D \tag{1.8}$$

$$(i-1, j+1) \in B \text{ implies } (i, j) \notin C \tag{1.9}$$

These require that the boundaries of A and D, as well as B and C, don't touch; therefore, P is 4-connected. The reconstruction algorithm then moves immediately

[2] $\min(m^2, n^2)$ is used because if $n < m$, the solution can be rotated 90°, so that the north and south feet become the east and west feet.

into an instance of 2-SAT using the following clauses:

$$
\begin{aligned}
Corners &= \bigwedge_{i,j} \left\{ \begin{array}{ll} A_{i,j} \Rightarrow A_{i-1,j} & B_{i,j} \Rightarrow B_{i-1,j} \\ C_{i,j} \Rightarrow C_{i+1,j} & D_{i,j} \Rightarrow D_{i+1,j} \\ A_{i,j} \Rightarrow A_{i,j-1} & B_{i,j} \Rightarrow B_{i,j+1} \\ C_{i,j} \Rightarrow C_{i,j-1} & D_{i,j} \Rightarrow D_{i,j+1} \end{array} \right\} \\
Disjoint &= \bigwedge_{i,j} \left\{ X_{i,j} \Rightarrow \overline{Y}_{i,j} : \text{for symbols } X, Y \in \{A, B, C, D\}, X \neq Y \right\} \\
Connected &= \bigwedge_{i,j} \left\{ \begin{array}{ll} A_{i,j} \Rightarrow \overline{D}_{i+1,j+1} & B_{i,j} \Rightarrow \overline{C}_{i+1,j-1} \end{array} \right\} \\
Anchors &= \overline{A}_{k,1} \wedge \overline{B}_{k,1} \wedge \overline{C}_{k,1} \wedge \overline{D}_{k,1} \wedge \overline{A}_{l,n} \wedge \overline{B}_{l,n} \wedge \overline{C}_{l,n} \wedge \overline{D}_{l,n} \\
LBC &= \bigwedge_{i,j} \left\{ \begin{array}{ll} A_{i,j} \Rightarrow \overline{C}_{i+v_j,j} & A_{i,j} \Rightarrow \overline{D}_{i+v_j,j} \\ B_{i,j} \Rightarrow \overline{C}_{i+v_j,j} & B_{i,j} \Rightarrow \overline{D}_{i+v_j,j} \end{array} \right\} \wedge \bigwedge_{j} \left\{ \begin{array}{ll} \overline{C}_{v_j,j}, & \overline{D}_{v_j,j} \end{array} \right\} \\
UBR &= \bigwedge_{j} \left\{ \begin{array}{ll} \bigwedge_{i \leq \min\{k,l\}} \overline{A}_{i,j} \Rightarrow B_{i,j+h_i} & \bigwedge_{k \leq i \leq l} \overline{C}_{i,j} \Rightarrow B_{i,j+h_i} \\ \bigwedge_{l \leq i \leq k} \overline{A}_{i,j} \Rightarrow D_{i,j+h_i} & \bigwedge_{\max\{k,l\} \leq i} \overline{C}_{i,j} \Rightarrow D_{i,j+h_i} \end{array} \right\}
\end{aligned}
$$

The first clause defines the corner regions, and the second and third ensure that the corner regions are disjoint and that any solution is connected. This ensures 4-connectivity of the solution because it ensures that the borders of A and D (B and C) cannot touch; if they did, then a 4-neighbour path along the border of A would be blocked by cells of D. The corner clause restates the requirement that (i, j) is an element of A whenever $(i + 1, j) \in A$ or $(i, j + 1) \in A$. Disjointness is guaranteed by explicitly stating that if a cell (i, j) is in one corner region, it cannot be in any of the other corner regions. The connectivity clause is similarly straightforward, as it just restates 1.8 and 1.9.

The last three clauses represent the anchor rows, a lower bound on column sums, and an upper bound on row sums, respectively. The anchors clause requires that the anchor rows are in $P = \overline{A \cup B \cup C \cup D}$, and it does so through a simple conjunction of literals. The LBC clause, which gives a lower bound on the column sums, is written in two parts: the second half of the clause (for each j, $\overline{C}_{v_j,j}, \overline{D}_{v_j,j}$) is a statement ensuring that there are at least v_j cells in column j that are not occupied by the lower corners, and the first half ensures that there are always at least v_j cells beneath the bottom-most cell of A or B in column j and the upper-most cells of C and D. The final set of clauses is similar, except it exerts an upper bound on the row sums by guaranteeing that for any cell that could be in T, all cells more than $h_i - 1$ cells to the right of it cannot be in T.

This is where the anchor rows become important. The set of clauses establishing an upper bound on the row sums uses the anchor rows to determine the relationship between the corner regions for each row index. If the current row is above both anchor rows, then A and B are used, as the fact that both anchor rows are beneath the current row tells us that we must be dealing exclusively with upper regions. On the other hand, if $k \leq i \leq l$, then we know that the region on the left could not be

A, as the anchor row k—which is in P, and as such not in A, B, C, or D—divides A from C. Therefore, the corner region on the left must be C.

Once these individual clauses have been converted from full propositional logic to CNF, the entire system can be described by the formula $F_{k,l}(H, V) = Cor \wedge Dis \wedge Con \wedge Anc \wedge LBC \wedge UBR$. The algorithm then tests $F_{k,l}(H, V)$ using a 2-SAT algorithm and determines whether or not the formula is satisfiable. If it is, then $T = \overline{A \cup B \cup C \cup D}$ is an hv-convex polyomino satisfying the row and column sums (H and V), and the algorithm can terminate. Otherwise, it moves on to the next pair of anchor rows and repeats.

Time Complexity

As mentioned previously, there are $O(\min(m^2, n^2))$ possible pairs of anchor rows to be worked through. As we already know, a 2-SAT problem instance can be solved in $O(|X| + |\mathbb{C}|)$ time. This algorithm has exactly four variables per cell $(A_{i,j}, B_{i,j}, C_{i,j}, D_{i,j})$ and a constant number of clauses per cell, and so the solution to the 2-SAT can be found in $O(nm)$ time. Hence, the final time complexity for the algorithm is $O(nm \min(m^2, n^2))$.

Chapter 2

Multiple-colour Reconstruction

2.1 Problem Overview

For this thesis, instead of dealing with a single hv-convex polyomino, we consider the reconstruction problem when we know that there are multiple non-overlapping hv-convex polyominoes within the problem space. This is typically described as the C-colour reconstruction problem for hv-convex polyominoes, where C is some constant number. Each colour corresponds to one of the polyominoes to be reconstructed. A more precise definition follows.

Note that there are two ways in which colour information can be presented: ordered and unordered. Basically, if colours are unordered, then we are presented with sets of sums for every row and column, whereas ordered colours present us with tuples of sums. These tuples provide a total ordering on the colours, in that if colour a occurs before colour b in a row tuple, all cells of colour a must be to the left of cells with colour b in the reconstruction. Unless explicitly stated otherwise, we are considering the unordered version of the reconstruction problem. Also, note that we are not putting any restrictions on the shape (e.g., 4-connectivity, convexity) of the union of the C colours. We now define this problem formally.

> **Problem:** Multiple-Colour Reconstruction of hv-convex Polyominoes from Orthogonal Projections
>
> **Given:** A finite set of colours $\{1, \ldots, C\}$, row sum vectors $\{H^1, \ldots, H^c\}$ where $H^i = (h^i_1, \ldots, h^i_m)$, and column sum vectors $\{V^1, \ldots, V^c\}$ where $V^i = (v^i_1, \ldots, v^i_n)$.
>
> **Goal:** Construct C hv-convex, non-overlapping, coloured polyominoes that satisfy the row and column sums for their respective colours, i.e row i has exactly h^c_i cells of colour c, and column j has exactly v^c_j cells of colour c.

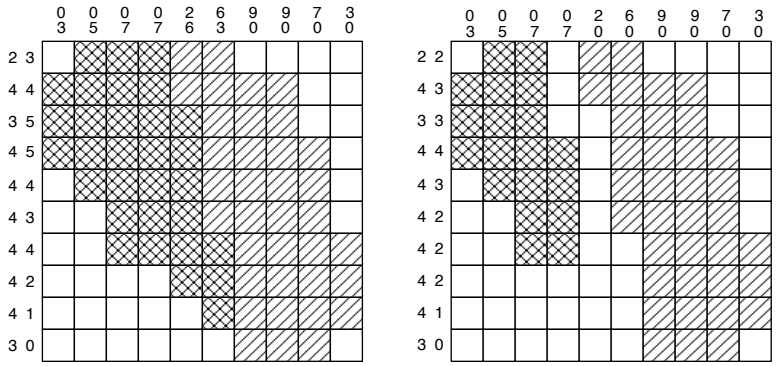


Figure 2.1.1: Example solutions to the 2-colour reconstruction problem on *hv*-convex polyominoes. The upper/leftmost set of numbers corresponds to the polyomino whose cells are filled in with dashes, and the other numbers correspond to the polyomino shaded using cross- hatching

Two polyominoes, P^r and P^b, are *non-overlapping* if for every cell $(i, j) \in P^r$, $(i, j) \notin P^b$ and vice versa. An example of a solution to a 2-colour reconstruction problem is shown in Figure 2.1.1.

2.2 Algorithm

To solve this version of the reconstruction problem, we make use of a reconstruction algorithm that has three phases. First, we generate a foot configuration for each of the C polyominoes that are to be reconstructed. Next, this set of foot configurations is used to create the spine for each polyomino. Finally, a 2-SAT formula based on the one presented in Section 1.4.3 is used to find a colour assignment for the cells that satisfies the row and column sums, if such an assignment exists. If a valid assignment exists, that assignment is returned as a solution; if not, a new foot configuration is generated and the process repeats. This continues until a valid solution is found or until all possible foot positions are exhausted.

2.2.1 Generating foot configurations and spines

When considering the multiple-colour reconstruction problem, each colour will only have row (column) sums for some of the rows (columns) of the matrix; other rows will simply be listed as containing 0 cells of that colour. Since there is nothing to reconstruct in these rows and columns, we can ignore them.

As mentioned in Section 1.4.2, a single polyomino can have as many as $O(n^2m^2)$ possible foot configurations. Since we are constructing C polyominoes, there will be a total of $O(n^{2C}m^{2C})$ possible foot configurations in total. The details of generating the foot configurations and spines are otherwise identical to those presented earlier. From here, we move to the 2-SAT portion of the algorithm.

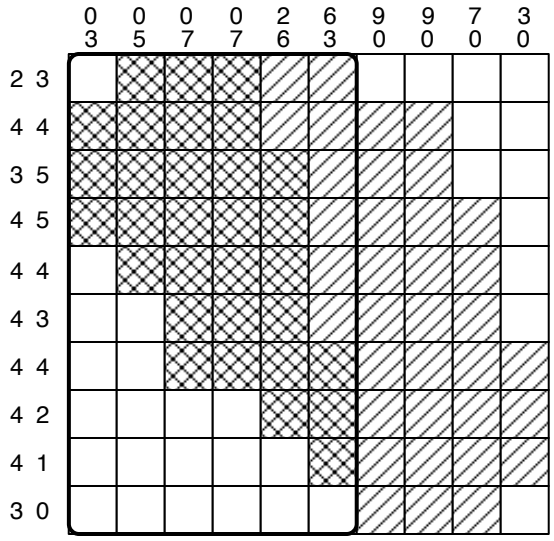


Figure 2.2.2: Diagram illustrating the idea of a polyomino's matrix. The section of the matrix associated with the darker polyomino (shaded using cross-hatching) has been circled. The uppermost/leftmost sets of numbers provide row/column information for the gray polyomino, which is shaded using dashes.

2.2.2 C-colour 2-SAT

For the C-colour 2-SAT representation, each cell (i, j) will be represented by C variables: $x^c_{i,j} \in \{0, 1\}$ for each colour c. The clauses being used are the same as those presented in Section 1.4.3, but as we have C different polyominoes to reconstruct, we will need to have C copies of the clauses, one for each colour. By Lemma 1.4.7, we know that if these clauses are satisfied, the resulting solutions will be 4-connected and hv-convex. In order to fulfill the final requirement, which states that the various polyominoes must not overlap, we need to introduce one more set of clauses. For all r and b in the set of colours such that $r \neq b$, and all cells (i, j), we have a clause of the form:

$$\overline{(x^r_{i,j} \wedge x^b_{i,j})}.$$

In a C-colour problem, each of the $m \times n$ cells will require $\frac{C(C-1)}{2}$ clauses, leading to a total of $O(C^2nm)$ clauses needed for the 2-SAT. As there are exactly C variables per cell, one for each colour, the 2-SAT will involve $O(Cnm)$ variables, and will take $O(Cnm + C^2nm)$, or $O(C^2nm)$ time to solve.

2.2.3 Time Complexity

Theorem 2.2.1. *The C-colour reconstruction problem for hv-convex polyominoes from orthogonal projections can be solved in $O(C^2n^{2C+1}m^{2C+1})$ time.*

Proof. As mentioned previously, there are a total for $O(n^{2C}m^{2C})$ possible foot configurations. For each configuration, we must generate a spine and solve the relevant 2-SAT problem; these steps take $O(Cnm)$ and $O(C^2nm)$ time respectively. Any solution of the 2-SAT can then be converted into a colour assignment in $O(Cnm)$ time, giving a final time complexity of $O(n^{2C}m^{2C}(Cnm + C^2nm))$, or $O(C^2n^{2C+1}m^{2C+1})$. □

2.3 Alternate Approach

2.3.1 Preliminaries

The previous algorithm is based on the algorithm presented by Barcucci et al. [15] as opposed to the much faster algorithm proposed by Chrobak and Dürr [7] because the algorithm by Barcucci et al. is better suited to a multiple-colour problem. Chrobak and Dürr's approach never explicitly mentions the black cells of the polyomino that is being reconstructed; instead, it reconstructs the polyomino by computing the empty space around it. This would prevent us from ever explicitly stating that the various polyominoes must be disjoint, or that if a cell belongs to P^r it must be in a corner region of P^b. This could cause severe difficulties when considering tightly-packed polyominoes: for example, if there were two valid configurations for a 2-colour reconstruction problem given the current foot configuration, one polyomino might be reconstructed in a manner that is incompatible with the other polyomino's position.

To avoid this problem, we present an alternate approach to the C-colour reconstruction problem that combines the idea of using "empty" space with the spine and portions of the previous C-colour reconstruction algorithm. It still refers to the coloured cells directly, allowing us to express our non-overlapping property, but by placing restrictions on the space not belonging to a given polyomino, it allows us to relax the requirements for the spines we generate. Note that all of the material here is written as though we were dealing with a single colour; this is done for the sake of readability. The clauses are the same for multiple colours, simply replacing the relevant variables with the versions that use the multiple-colour notation, e.g. replacing $x_{i,j}$ with $x^c_{i,j}$.

This algorithm will have a final time complexity of $O(C^2nm\min(n^{2C}, m^{2C}))$. Note that in the single-colour case, this simplifies to $O(nm\min(n^2, m^2))$, which is exactly that of Chrobak and Dürr's single-colour reconstruction algorithm [7].

2.3.2 Spine Generation

Instead of dealing with a full spine, as in the algorithm that has just been presented, this approach deals with the WE-spine, which consists of the union of S_{WE} and the west and east feet, as per the definition in Section 1.4.2. Instead of developing

a foot configuration where the positions of all four feet are fixed, we now only need to fix the positions of the west and east feet. We then find S_{WE}, which consists of the cells that satisfy Equation 1.2, which is shown on page 9.

Lemma 1.4.2 proved that the NS-spine was 4-connected and contained a cell in every row; later, we noted that a similar proof could be used to show that the WE-spine was also 4-connected and contained a cell in every column. These are exactly the properties that we require for the reconstruction process, so spine generation is finished once S_{WE} has been determined.

Note that either S_{WE} or S_{NS} could be used for this algorithm, as it doesn't really matter whether we have at least one cell in all of the columns or all of the rows. The one important difference is that the number of foot positions for the north and south feet is $O(n^2)$, while the number of positions for the west and east feet is $O(m^2)$. We use either the NS- or WE-spine, depending on whether n or m is smaller, giving us a total of $O(\min(n^2, m^2))$ foot configurations, or $O(\min(n^{2C}, m^{2C}))$ total foot configurations in the C-colour case. We assume without loss of generality that $m < n$ when explaining the algorithm, and as such we will use S_{WE} in the coming examples.

2.3.3 Reconstruction

As we have already shown that the WE-spine has at least one cell in every column of the polyomino, we can apply the 2-SAT clauses from the first algorithm to the columns. We can do this because the clauses from the first algorithm require a cell in every column j that is known to be in P to act as the α_j index for that column. As such, for every $1 \leq j \leq n$, we add the following clauses:

$$\overline{x_{i,j}}, \forall i \leq \alpha_j - v_j \tag{2.1a}$$
$$\overline{x_{i,j}}, \forall i \geq \alpha_j + v_j \tag{2.1b}$$
$$x_{i,j} \iff \overline{x_{i+v_j,j}}, \forall i \in \{\alpha_j - v_j + 1, \ldots, \alpha_j - 1\} \tag{2.1c}$$
$$x_{i,j} \Rightarrow x_{i+1,j}, \forall i \in \{\alpha_j - v_j + 1, \ldots, \alpha_j - 2\} \tag{2.1d}$$
$$x_{i,j} \Rightarrow x_{i-1,j}, \forall i \in \{\alpha_j + 2, \ldots, \alpha_i + h_i - 1\} \tag{2.1e}$$
$$x_{\alpha_j,j} \tag{2.1f}$$

Because S_{WE} is not guaranteed to have at least one cell in every row, however, the rows cannot be reconstructed using the same clauses. Instead, we borrow the idea of restricting coloured cells by defining the surrounding white space from Chrobak and Dürr [7]. Their paper, previously covered in Section 1.4.4, revolves around the idea of a corner region: an area of white cells surrounding cells belonging to the solution. We use a similar idea: for each colour and cell, we define a variable $R_{i,j}$ that is true if (i, j) is to the right of all black cells in row i; in the multiple-colour case, then, $R^c_{i,j}$ means that cell (i, j) is to the right of all cells of colour c. We add the following clauses to the 2-SAT to define this region:

$$R_{i,j} \Rightarrow R_{i,j+1}, \forall i, \forall j \tag{2.2a}$$
$$R_{i,j} \Rightarrow \overline{x_{i,j}}, \forall i, \forall j \tag{2.2b}$$
$$x_{i,j} \Rightarrow R_{i,j+h_i}, \forall i, \forall j \tag{2.2c}$$

Clause 2.2a ensures that if a cell (i, j) is to the right of all black cells in row i (i.e. $R_{i,j}$ holds), $R_{i,k}$ must hold for $k > j$ as well. Clause 2.2b requires that if $R_{i,j}$, cell (i, j) cannot be black in the solution, enforcing the region's definition as an area of non-black cells. Clause 2.2c ensures that there can be no more than h_i black cells in row i. Combining these clauses with the previous 2-SAT clauses, we can prove the following lemmas:

Lemma 2.3.1. *A reconstruction satisfying both 2.1 and 2.2 guarantees that the sum of black cells in each row i is equal to the row sum h_i.*

Proof. We prove this via contradiction. Lemma 1.4.7 shows us that all column sums must be satisfied by the column reconstruction. This gives us an exact bound on the number of black cells in the reconstruction, in that it equals $V_n = \sum_{j=1}^{n} v_j$, and therefore $H_m = \sum_{i=1}^{m} h_i$ as well.

Clause 2.2c provides an upper bound for the rows, as it guarantees that no more than h_i black cells can be to the right of any given black cell. This, in combination with the exact bounds on the column cells, provides us with our guarantee that the number of black cells in the rows is exactly equal to h_i, in the same manner as Chrobak and Dürr's reconstruction approach [7]. The column bounds guarantee that there will be exactly H_m black cells, and Clause 2.2c requires that there will be no more than h_i black cells in any row i. If there were fewer than h_i black cells in the row, there would have to be fewer than H_m black cells in total, or some other row would have more black cells than possible, both of which are contradictions. □

Lemma 2.3.2. *A reconstruction satisfying both 2.1 and 2.2 must be hv-convex and 4-connected.*

Proof. By Lemma 1.4.7, we know that the columns are v-convex. Next, consider the rows. By Lemma 2.3.1, we know that there are exactly h_i black cells in any row. If (i, j) is black, then by Clause 2.2c, cell $(i, j + h_1)$, must be in R (i.e., $R_{i,j+h_i}$ is true) and must be white, so there is no room for any white cells in-between the black cells in the row. As such, the rows must be h-convex, in which case the solution must be hv-convex.

Lemma 1.4.2 can be used to show that the WE-spine is 4-connected. Since the columns are v-convex extensions of the spine, in that they use a key cell from the spine and build outwards, the union of the 2-SAT-generated columns and the spine is still 4-connected. Similarly, as the rows are h-convex and the clauses used to build them just impose convexity requirements on the column cells, the union of the row cells generated by the 2-SAT and the spine are 4-connected as well. As such, 4-connectivity in the final reconstruction must be maintained. □

As before, this single-colour algorithm can be generalized to a multiple-colour approach by adding the relevant notation, i.e. $R_{i,j}$ becomes $R^c_{i,j}$ for each colour c, $x_{i,j}$ becomes $x^c_{i,j}$, and so forth. Note that since we are dealing with a problem involving multiple colours, $R^c_{i,j}$ only implies that a cell is not of colour c; it might be white or any other colour in the reconstruction. We will also need to include the $O(C^2nm)$ non-overlapping clauses (e.g., $\overline{x^{c_1}_{i,j} \wedge x^{c_2}_{i,j}}$).

2.3.4 Time Complexity

Theorem 2.3.3. *The multiple-colour discrete reconstruction problem for hv-convex polyominoes from orthogonal projections can be solved in $O(C^2 \min(n,m)^{2C} nm)$ time.*

Proof. As previously mentioned, we assume that $m < n$ and as such use the WE-spine, thereby ignoring the north and south feet. By removing the north and south feet from the spine generation process, we remove $O(n^2)$ possible foot configurations from consideration, lowering the time complexity for that step from $O(n^{2C}m^{2C})$ to $O(m^{2C})$. By adding the new variable $R_{i,j}$ for each colour being considered, we add $O(Cnm)$ new variables and $O(Cnm)$ new clauses to the 2-SAT. These changes do not affect the final runtime of $O(C^2nm)$ for the 2-SAT, and so the final run time is $O(C^2 \min(n,m)^{2C} nm)$. □

Note that the techniques used in this algorithm remove a factor of n^{2C} from the reconstruction process, which is a significant decrease in time complexity from the first algorithm developed in this thesis. As previously mentioned, in the single-colour case, the run-time simplifies to $O(nm \min(n^2, m^2))$, which is exactly that of Chrobak and Dürr's single-colour reconstruction algorithm [7].

2.4 Reconstruction for Ordered Colours

Up until now, we have been considering the unordered c-colour reconstruction problem. We now show that using a modified version of the alternate reconstruction approach, we can also develop a solution to the ordered c-colour reconstruction problem. The key idea behind this approach is that the R region defined in the previous section gives us an easy way to determine if a cell is to the right of all cells of a given colour.

As a result of Clause 2.2a's restriction that if $R_{i,j}$ holds, $R_{i,k}$ must hold for $k \geq j+1$, we can easily develop clauses to ensure that a strict colour order is maintained. For example, consider the following colour ordering for row i in a k-colour reconstruction problem: $(c_1, c_2, \ldots, c_k)$. In this case, colour order could be enforced via the following clause:

$$x^{c_l}_{i,j} \Rightarrow R^{c_{l-1}}_{i,j} \text{ for } 2 \leq l \leq k \tag{2.3}$$

These clauses guarantee that all cells of colour c_l will be to the right of all cells of colour c_{l-1}, for $2 \leq l \leq k$, and by transitivity they will be to the right of all cells of colour c_j as well, for all $1 \leq j < l-1$. Note that the $R_{i,j}$ clauses only offer a horizontal colour ordering; in order to ensure that our solutions will satisfy an overall colour ordering, we need to apply a similar set of clauses to the columns of the reconstruction using a new region. We introduce the variable $B_{i,j}$ to denote that cell (i,j) is beneath all cells of colour c in column j. The clauses used to define $B_{i,j}$ are as follows:

$$B_{i,j} \Rightarrow B_{i+1,j}, \forall i, j \tag{2.4a}$$
$$B_{i,j} \Rightarrow \overline{x_{i,j}}, \forall i, \forall j \tag{2.4b}$$
$$x_{i,j} \Rightarrow B_{i+v_j,j}, \forall i, \forall j \tag{2.4c}$$

As $R_{i,j}$ and $B_{i,j}$ are simply another way of defining cells that are not of colour c, $B_{i,j}$ will not affect v-convexity or v_j requirements which have already been established by the previous column clauses (Clauses 2.1a to 2.1f).

2.4.1 Time Complexity

By imposing a strict colour order on the solution to the reconstruction problem, we actually decrease time complexity significantly. This is because a strict colour ordering gives us non-overlapping polyominoes "for free".

Lemma 2.4.1. *If a solution to the reconstruction problem satisfies a total colour order, each cell will have been assigned at most one colour.*

Proof. Assume some cell is assigned colour c in addition to some other colour d. If d is earlier in the total order, then the ordering has been violated, as all cells of colour c are not to the right of all cells of colour d. A similar order violation occurs if d is later in the total order, as all cells of colour d are not to the right of all cells of colour c. □

Theorem 2.4.2. *The ordered, multiple-colour reconstruction problem for hv-convex polyominoes from orthogonal projections can be solved in $O(C \min(n,m)^{2C} nm)$ time.*

Proof. All aspects of the algorithm other than the 2-SAT remain unchanged from what is described in Theorem 2.3.3. Lemma 2.4.1 allows us to eliminate all $O(C^2nm)$ non-overlapping clauses from the 2-SAT problem. This decreases the 2-SAT time complexity to $O(Cnm)$ from $O(C^2nm)$, thereby resulting in an overall time complexity of $O(C \min(n,m)^{2C} nm)$. □

Note that it is possible to use the techniques presented above to solve an ordered colour reconstruction problem where we are only given a partial order on the set of colours. In this case, Clause 2.3 would only be applied to colours involved in the

given order. However, since order requirements would no longer exist for the other colours, we would need to continue using the non-overlapping clause $(\overline{x_{i,j}^{c_1}} \vee \overline{x_{i,j}^{c_2}})$ for all such colours. This would result in a return to the previous time complexity of $O(C^2 \min\{n, m\}^{2C} nm)$.

2.5 Final Notes

Throughout this section, specifics have not been mentioned regarding a 2-SAT solver as the actual process of solving the 2-SAT can be regarded as a black box solving process. However, the choice of 2-SAT solver does impact the final algorithm in that it determines whether the final algorithm returns a single solution to the reconstruction problem or all possible solutions to the reconstruction problem. Up until the 2-SAT is encountered, the algorithm is largely generic; the spine that is constructed for a given foot configuration is the same for all possible reconstructions based on that foot position. As such, if the goal were to generate all possible *hv*-convex reconstructions of a given set of data, it could easily be accomplished by iterating through all possible foot configurations and using a 2-SAT solver that returns all possible solutions to a 2-SAT instance, such as the algorithm by Feder [11]. This could pose a problem with respect to time complexity, however, because it was shown that in the worst case, there can be an exponential number of *hv*-convex polyominoes that satisfy the same row and column sums [18].

Chapter 3

NP-Completeness Results

For the second part of the thesis, we will be providing NP-completeness results for the *hv*-convex reconstruction problem for k colours, where k is unbounded. The proof will involve a reduction from 3-SAT, which was defined in Section 1.4.3, using the standard "gadget" template. The proof given is for the NP-completeness of the unordered colour version of the problem; the ordered version will be discussed briefly in Section 4.2.

3.1 Overview

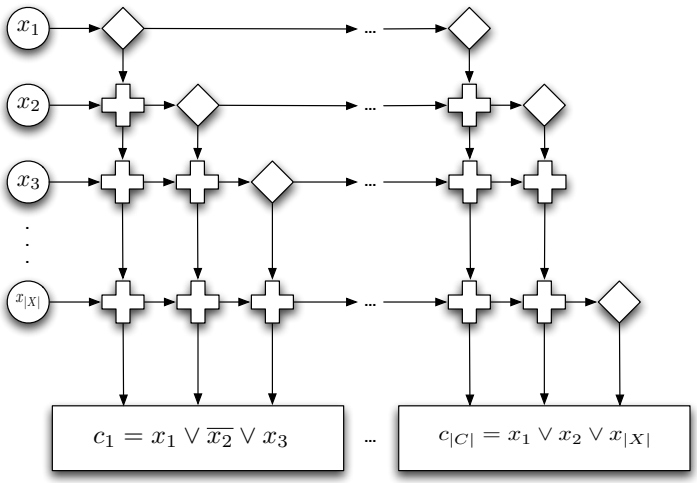


Figure 3.1.1: The basic layout of a reduction from 3-SAT to the reconstruction problem. Circles represent variable gadgets, diamonds are splitter gadgets, crosses are crosser gadgets, and rectangles are clause gadgets. Transmitter gadgets are not shown for the sake of simplicity.

In our reduction, we make use of the idea of *gadgets*: representations of 3-SAT variables and clauses that can communicate boolean values while still adhering to

the restrictions of the reconstruction problem. In order to handle the reduction, we will need to make use of five different types of gadgets: boolean variable gadgets, splitters, crossers, transmitters, and clause gadgets. These will ultimately be laid out in a manner similar to that shown in Figure 3.1.1, with each gadget using different colours.

The clause gadgets are laid out at the bottom of the matrix, while the boolean variables are aligned against the left border of the matrix. Boolean values are then transmitted horizontally across the matrix by transmitter gadgets, with values being sent downwards to clause gadgets at the appropriate horizontal position with the aid of a splitter gadget. When a vertical boolean-value transmission crosses a horizontal boolean-value transmission, the crosser gadget is used to transmit values without modifying them.

3.2 Gadgets

The simplest component of the reduction is the *boolean variable gadget*; this is what represents each boolean variable $x_i \in X$, and acts as the source of the value for that variable. Its row and column sum values are: $H = (3, 1, 3)$, $V = (1, 1, 3, 1, 1)$. Due to the *hv*-convexity and 4-connectivity restrictions, this input has two possible reconstructions, which are shown in Figure 3.2.2. These configurations represent the literals x_i and $\overline{x_i}$. If the upper-left corner of the polyomino's matrix is empty, then the variable represents x_i; similarly, if it is occupied, it represents $\overline{x_i}$.

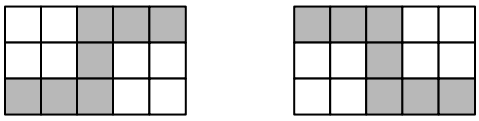


Figure 3.2.2: Two representations of a boolean variable in the form of an *hv*-convex polyomino. The left polyomino represents x_i, while the right one represents $\overline{x_i}$.

The second gadget is the *transmitter gadget*, which transmits the boolean values of our variable gadgets horizontally across the matrix. It has the following row and column sums: $H = \{1, 1, 1, 1, 3, 1, 1, 1, 1\}, V = \{5, 1, 5\}$. It is essentially a boolean variable gadget with longer top and bottom rows, and as such has the same two possible configurations as the variable gadget.

A transmitter gadget is shown in Figure 3.2.3, where it is being used to transmit a boolean value. Note that it is not possible to switch the transmitter to its "false" configuration, as that would introduce an overlap between it and the variable gadget; the transmitter must mirror the variable's truth value in any valid reconstruction. This basic idea is used whenever we need to transmit truth values between adjacent gadgets.

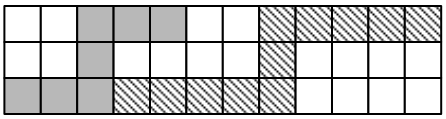


Figure 3.2.3: A transmitter gadget being used to project a boolean assignment. A variable gadget is on the left, while the transmitter is the striped polyomino on the right.

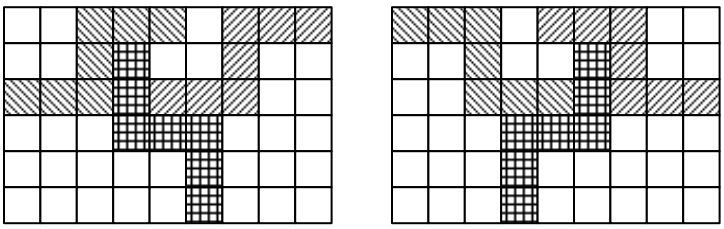


Figure 3.2.4: A splitter gadget being used to split a truth-value assignment, transmitting it both horizontally and vertically. The left figure depicts a splitter gadget in the "true" position, while the right figure shows the "false" position.

Next is the *splitter gadget*, which is composed of multiple variable gadgets. The true and false versions of the gadget can be seen in Figure 3.2.4. Note that the vertical component is a variable gadget that has been rotated 90° clockwise, and the true and false positions follow from this: if the upper-left corner is occupied, it is in the true position. The row and column sums for each component should be obvious, and have been omitted. We can also derive the following lemma:

Lemma 3.2.1. *All components of the splitter gadget must have the same boolean value.*

Proof. Each of the shapes that compose the splitter gadget have only two possible configurations, as they are variable gadgets. There are no conflicts when all components are set to the same boolean value, as can be seen in Figure 3.2.4. The two horizontal variable gadgets must have the same value, otherwise they would overlap. As the vertical gadget is similarly constrained to two possible reconstructions, each of the two possible configurations for the horizontal gadgets leaves only one possible configuration for the vertical gadget. We have defined the positions of the vertical gadget so that the value for each configuration mirrors the value transmitted by the horizontal gadgets. □

The *crosser gadget* is used to transmit values horizontally and vertically without allowing them to interfere, and so the horizontal and vertical parts of the gadget are able to switch between true and false configurations without conflicting

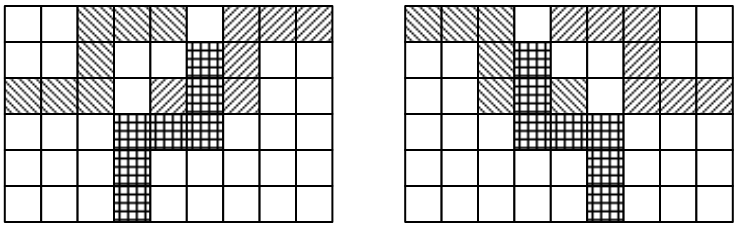


Figure 3.2.5: Examples of conflicts caused by changing the configuration of the vertical component in the splitter gadget.

with the other set. Its row and column sums are $H = \{1, 1, 7, 5, 7, 5, 1, 1\}, V = \{1, 1, 4, 6, 4, 6, 4, 1, 1\}$.

Lemma 3.2.2. *The crosser gadget's row and column sums permit exactly four valid hv-convex reconstructions.*

Proof. First, exactly one row (column) needs to extend to the border in each of the four cardinal directions in order to satisfy the row (column) sums of 1 – if two rows were extended in the same direction, the row sum would be exceeded, and if no rows were extended, the sums would not be satisfied. Additionally, only the rows (columns) with sums of 7 (6) can extend to the border of the bounding box; otherwise, at least two rows or columns would be one cell short of their respective sums. This covers all ways of modifying the reconstruction while satisfying *hv*-convexity. □

The *clause gadget* is used to represent 3-SAT clauses. Each clause gadget has three inputs, and restricts the boolean value of one of these inputs to either true or false in any given configuration in a similar manner as the other gadgets. The values that each clause gadget restricts are exactly those that would not cause the 3-SAT clause it represents to evaluate as true. For example, the clause gadget associated with the 3-SAT clause $(x_i \vee \overline{x_j} \vee x_k)$ would have three inputs, one each from x_i, x_j, and x_k, and would be able to restrict x_i to be true, x_j to be false, or x_k to be true. The row and columns sums for the clause gadget corresponding to the 3-SAT clause $(x_1 \vee x_2 \vee x_3)$ are $H = \{1, 1, 14, 14, 14, 14, 14, 1, 1\}$ and $V = \{4, 5, 5, 5, 5, 5, 5, 7, 5, 5, 5, 5, 5, 5, 3\}$. Each clause gadget has four possible configurations, which are shown in Figure 3.2.7. Two of these configurations are functionally equivalent insofar as they both restrict the same variable, and as such one of them can be safely ignored.

Lemma 3.2.3. *The clause gadget has exactly four possible configurations.*

Proof. All columns j such that $v_j = 5$ are fixed in place for any possible solution to the reconstruction problem. This is because rows 3 through 7 have row sums of 14, and the gadget has 16 columns. The 13 cells from column 2 to 15 must then

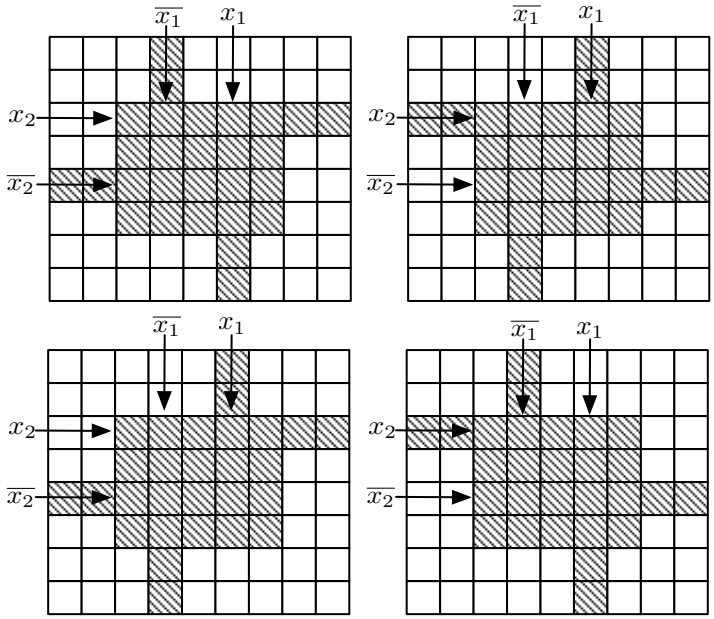


Figure 3.2.6: The four possible configurations of the crosser gadget. Arrows indicate which boolean-values are associated with each row/column; filled-in rows/columns are blocked by the gadget, as no other gadget could occupy those cells without causing an overlap.

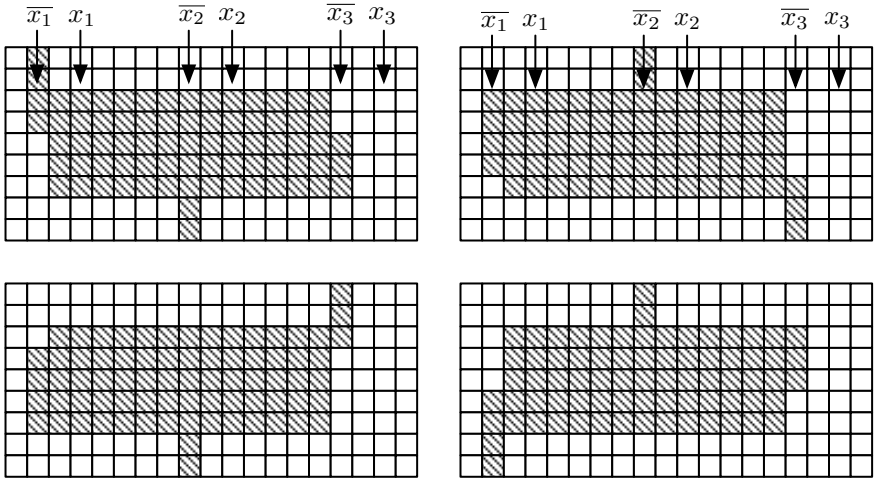


Figure 3.2.7: Four configurations of the clause gadget representing $(x_1 \vee x_2 \vee x_3)$. The upper left restricts the first variable in the clause to true by blocking a false input, the upper-right restricts the second variable, the bottom-right restricts the third variable, and the bottom-right restricts the second variable.

be black in each of these rows, completely satisfying the column sum requirements when $v_j = 5$.

With these columns removed from consideration, we only need examine the columns with lengths 3, 4, and 7: the rightmost, leftmost, and (approximately) middle columns. As the matrix containing a clause gadget has only 9 rows, there are exactly two possible positions for the middle column that satisfy its column sum: these positions span either rows 1 to 7, or rows 3 to 9. The position spanning 2 to 8 does not work, as it requires that the cell satisfying the uppermost row sum ($h_i = 1$) be in a different column, which would result in a violation of v-convexity. This leaves only the side columns. Due to row sum requirements, these columns cannot occupy any of the same rows, as this would result in rows with more than 14 black cells. Row sum requirements also ensure that they must occupy neighbouring rows; if the left column spanned rows 1-4 and the right spanned 6-8, row 5 would have 13 black cells, violating row sum requirements. With this in mind, the union of the two can be considered as roughly analogous to the middle column, placing similar restrictions on its possible positions.

This would indicate two possible configurations of the clause gadget. However, as either the rightmost or leftmost column can be extended into the rows where $h_i = 1$, there are two possible configurations when the union of these two columns is pointing up, and two when it is pointing down, resulting in four possible configurations. □

There are eight different clause gadgets, one for each of the other possible forms a 3-SAT clause can take, e.g., $(x_1 \vee \overline{x_2} \vee x_3)$. The one shown in Figure 3.2.7 represents a clause of the form $(x_i \vee x_j \vee x_k)$, and the remaining clause gadgets are shown in Figure 3.2.8.

3.3 Layout

We now explain how all gadgets are put together; a full example is given in Figure 3.3.9 on page 38. Row and column positions for each individual gadget are dependent on both the index of the variable they represent and the position of that variable within the clause it is being inserted into. There are $|\mathbb{C}|$ clauses and each clause has three indices associated with it that correspond to the three literals in the clause. As such, literal positions range from $1 \ldots 3|\mathbb{C}|$ with 1 representing the first literal in the first clause and $3|\mathbb{C}|$ the third literal in the $|C|^{\text{th}}$ clause. Gadget positions will be described in terms the rows and columns that the gadgets occupy; Table 3.1 lists these positions for each gadget. Positions are described in terms of i and j, where i is the current variable's index (i.e., the i in x_i) and j is the current literal's position; e.g., the third literal in the fifth clause would be in the 15^{th} position. Note that the rows and columns range from $1 \ldots m$ and $1 \ldots n$ respectively, so $(1, 1)$ is considered the upper-left cell.

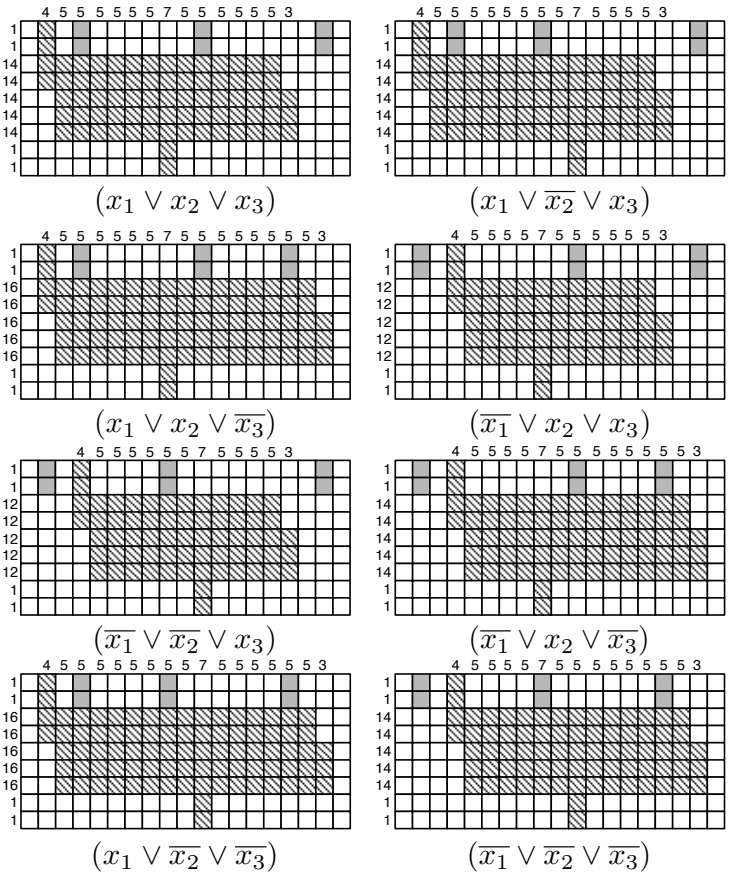


Figure 3.2.8: All variations of the clause gadget, with the corresponding clause types listed below each gadget. Solid gray cells represent cells from other splitter/crosser gadgets transmitting boolean values

We list only one of the clause gadgets in Table 3.1; all other clause gadgets are variations on this, with the gadget's left border moved to $21j - 12$ if the first literal is negated, and the right border moved to $21j + 2$ if the third literal is negated.

Gadget	Rows	Columns
Variable	$6i - 5$ to $6i - 3$	1 to 5
Transmitter	$6i - 5$ to $6i - 3$	$7j - 3$ to $7j + 5$
Splitter	$6i - 5$ to $6i - 1$	$7j - 3$ to $7j + 5$
Crosser	$6(i - 1) - 1$ to $6i - 1$	$7j - 3$ to $7j + 5$
Clause $(x_1 \vee x_2 \vee x_3)$	$6\|X\| - 2$ to $6\|X\| + 6$	$21t - 14$ to $21t$

Table 3.1: Row and column information for the position of each gadget within the layout, where i is the index of the boolean variable associated with the gadget, and j is the position of the literal being considered. The index t indicates the clause being considered, and ranges from 1 to $|\mathbb{C}|$.

The area of the matrix associated with variable x_i and the j^{th} literal's position is used by either a transmitter gadget, a crosser gadget, or a splitter gadget as follows. If the literal at the j^{th} position is either x_i or $\overline{x_i}$, the gadget must be a splitter, as the variable's truth value needs to be transmitted down to the clause gadget. If the j^{th} literal is x_h or $\overline{x_h}$ for $h < i$, then a crosser gadget is used. Otherwise, a transmitter gadget is used.

From this layout, we can derive an upper bound of on the number of colours needed. Each row can have at most $3|C|$ gadgets, plus the truth variable gadget. Furthermore, there are $|C|$ clause gadgets, giving us a total of $|X|(3|C| + 1) + |C|$ gadgets. However, $3|\mathbb{C}|$ of these gadgets are splitter gadgets, which use three colours instead of just one. This means we will need $6|\mathbb{C}|$ colours in addition to the previous total, giving us a final total of $|X|(3|\mathbb{C}| + 1) + 7|\mathbb{C}| \in O(|X||\mathbb{C}|)$ colours.

3.4 NP-Completeness Proof

Theorem 3.4.1. *The multiple-colour discrete reconstruction problem for hv-convex polyominoes from orthogonal projections is NP-complete if the number of colours is part of the input.*

Proof. We first show that a solution can be verified in polynomial time. Given a reconstruction, we need to verify three properties: that the row and column sums are satisfied, that each polyomino is *hv*-convex, and that each polyomino is 4-connected. Row and column sums can be verified trivially by summing up the cells in each row/column. Convexity can be determined by a simple scan through each row/column to ensure that for each colour c, if a non-c cell is found after a cell of colour c has already been observed, no other c cells occur later in the row/column. Finally, 4-connectivity can be verified in polynomial time by constructing a graph

for each colour c where the nodes are cells of colour c and edges are drawn between cells that are 4-neighbours of each other. Connectivity can then be checked in the usual manner for a graph.

We now show that the problem is NP-hard via reduction from 3-SAT. Given an instance of 3-SAT, develop an instance to the reconstruction problem as described in the previous section. Given a truth-value assignment satisfying the 3-SAT formula, select one literal from each clause that evaluates as true. Next, assume all clause gadgets within the matrix are realized in the configurations that restrict their respective literal's value. For example, if some clause $c_1 = (x_1 \vee x_2 \vee \overline{x_3})$ evaluates as true because x_3 is false in the 3-SAT assignment, that clause's gadget would be reconstructed such that it accepts a true input as the third literal, but disallows a false input.

If this is done, then a valid reconstruction can be found in polynomial time by setting each variable gadget to the value associated with it in the 3-SAT assignment. If each of the gadgets in the layout described above are set to configurations corresponding with the truth values of the relevant variables, then each gadget will not overlap with any of its neighbouring gadgets. Additionally, since we have already set the clause gadget to allow values associated with key literals, we know that there will be no conflicts between the clause gadgets and any adjacent gadgets. As we already know exactly which objects need to be constructed and where they will be located, reconstruction in polynomial time is trivial.

Given a valid reconstruction, consider a clause of the form $c_j = (x_1 \vee x_2 \vee x_3)$. From Lemma 3.2.3 and Figure 3.2.7, we know that the first two rows of c_j's clause gadget have row sum 1, and these cells will be located in one of three columns, which have lengths 4, 7, and 3. The clause gadget is arranged such that these columns are exactly those corresponding to the boolean values $\overline{x_1}, \overline{x_2}$, and $\overline{x_1}$ respectively. As such, the literals $\overline{x_1}, \overline{x_1}, \overline{x_1}$ cannot all be true simultaneously, so clause c_j must be satisfied. From the layout and Lemmas 3.2.1 and 3.2.2, all variable gadgets, transmitter gadgets, splitter gadgets, and crosser gadgets involving variable x_i must communicate the same boolean value as x_i, so we can trace back from the clause gadget inputs to each variable's gadget in order to find a solution to the 3-SAT.

As discussed earlier, the number of gadgets, colours, rows, and columns is polynomial in the number of variables and clauses, which proves NP-hardness. Since there is also a polynomial-time verifier, the problem is therefore NP-complete.

□

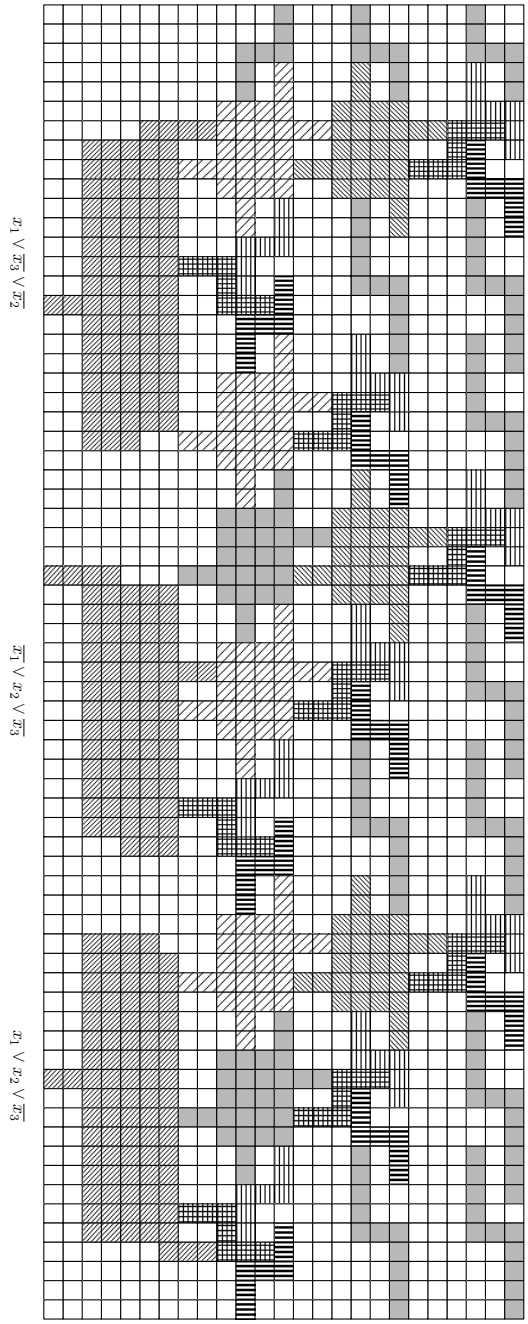


Figure 3.3.9: A simple example of the layout for the NP-completeness proof with three boolean variables and three clauses. Note that some shades are re-used throughout the layout; this is for the sake of readability. Each gadget should be thought of as having its own, unique colour.

Chapter 4

Conclusion

4.1 Results

In this thesis, two main algorithms were presented: an $O(C^2 n^{2C+1} m^{2C+1})$ time hv-convex unordered C-colour reconstruction algorithm based on the results presented by Barcucci et al. [3], and an $O(C^2 \min(n^{2C}, m^{2C})nm)$ time algorithm incorporating aspects of the algorithm by Chrobak and Dürr [7]. Additionally, a method for adapting the latter algorithm to the ordered version of the reconstruction problem was presented. Finally, an NP-completeness proof was presented for the multiple-colour hv-convex reconstruction problem if the number of colours is not a constant.

4.2 Open Problems

We will conclude by discussing a few open problems. One possible avenue for exploration is to examine cases where the polyominoes are further restricted to be certain types of hv-convex polyominoes, such as L-convex or Z-convex polyominoes, which are discussed by Castiglione et al [5] and Duchi et al. [10] respectively. All of these cases can be considered as either ordered or unordered colour reconstruction problems. We could also think about applying other notions of connectivity—such as 8-connectivity, which is discussed by Balázs et al. [2]—to the polyominoes being reconstructed. Our notions of connectivity could also be modified: we could examine problems where each colour is guaranteed to form a constant number (e.g., two) of hv-convex polyominoes which are not connected to each other. All of these problems remain open.

The most fascinating open problem that remains for us is to generalize the NP-completeness proof to the ordered colour version of the reconstruction problem. This would require gadgets that have multiple configurations, each representing a different boolean value being transmitted, that all maintain the same colour ordering. We have already been able to convert the transmitter gadget into a new

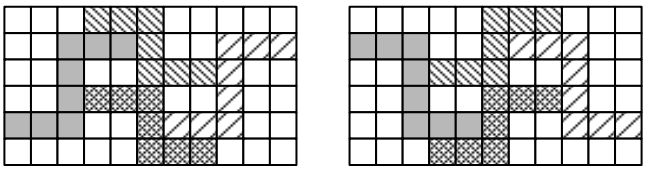


Figure 4.2.1: A version of the transmitter gadget designed specifically for the ordered k-colour reconstruction problem. Both versions obey the same total colour ordering, but transmit different truth values.

representation that respects these requirements, but the remaining gadgets appear to be more difficult to convert.

The transmitter gadget that we have developed works differently than those previously presented; instead of using the non-overlapping property to restrict the number of configurations, the colour ordering is used. Figure 4.2.1 shows the new, ordered gadget. Note that any of its components can be switched without overlapping with one of the others, but doing so violates the colour ordering. This idea also is used to transmit values between gadgets. Instead of placing gadgets within the matrix such that inconsistent boolean values would cause overlaps, they would be positioned such that an inconsistent pair of configurations would result in a colour ordering violation.

The splitter, crosser and clause gadgets would have to be reworked, and are a non-trivial problem. In particular, the clause gadget is difficult to rework into a format that respects colour ordering; it is easy to arrange it such that the colour ordering is respected for two of the three boolean inputs, but the third is difficult to restrict without violating either the ordering or the hv-convexity requirement. Finally, these problems that we have mentioned can be made more interesting by introducing additional layers of complexity. For example, all such problems become worth studying if we consider higher dimensions. Alternatively, we could consider instances without solutions, or with noisy data. Are there algorithms that can handle such situations by giving a solution that is "close" to the correct solution, i.e., that are provably-good approximation algorithms for some measure of error?

References

[1] Péter Balázs. A decomposition technique for reconstructing discrete sets from four projections. *Image and Vision Computing*, 25(10):1609–1619, October 2007. 7

[2] Péter Balázs, Emese Balogh, and Attila Kuba. Reconstruction of 8-connected but not 4-connected hv-convex discrete sets. *Discrete Applied Mathematics*, 147(2-3):149–168, April 2005. 6, 39

[3] Elena Barcucci, Alberto Del Lungo, Maurice Nivat, and Renzo Pinzani. Reconstructing convex polyominoes from horizontal and vertical projections. *Theor. Comput. Sci.*, 155(2):321–347, 1996. 4, 6, 7, 39

[4] K.J. Batenburg. A new algorithm for 3D binary tomography. *Electronic Notes in Discrete Mathematics*, 20:247–261, July 2005. 7

[5] Giusi Castiglione and Antonio Restivo. Reconstruction of L-convex polyominoes. *Electronic Notes in Discrete Mathematics*, 12:290 – 301, 2003. 9th International Workshop on Combinatorial Image Analysis. 6, 39

[6] Giusi Castiglione, Antonio Restivo, and Roberto Vaglica. A reconstruction algorithm for L-convex polyominoes. *Theoretical Computer Science*, 356(1-2):58 – 72, 2006. In honour of Professor Christian Choffrut on the occasion of his 60th birthday. 6

[7] Marek Chrobak and Christoph Dürr. Reconstructing hv-convex polyominoes from orthogonal projections. *Inf. Process. Lett.*, 69(6):283–289, 1999. 6, 7, 9, 17, 23, 24, 25, 26, 39

[8] Marek Chrobak and Christoph Dürr. Reconstructing polyatomic structures from discrete X-rays: NP-completeness proof for three atoms. *Theoretical Computer Science*, 259(1-2):81–98, May 2001. 1, 2, 7

[9] Albert V. Crewe and David A. Crewe. Inexact reconstruction: Some improvements. *Ultramicroscopy*, 16(1):33 – 40, 1985. 1

[10] Enrica Duchi, Simone Rinaldi, and Gilles Schaeffer. The number of Z-convex polyominoes. *Advances in Applied Mathematics*, 40:54–72, 2008. 6, 8, 39

[11] Toms Feder. Network flow and 2-satisfiability. *Algorithmica*, 11(3):291–319, March 1994. 28

[12] R. J. Gardner, P. Gritzmann, and D. Prangenberg. On the computational complexity of reconstructing lattice sets from their X-rays. *Discrete Math.*, 202(1-3):45–71, 1999. 7

[13] R. J. Gardner, P. Gritzmann, and P. Prangenberg. On the computational complexity of determining polyatomic structures by X-rays. *Theoretical Computer Science*, 233(1-2):91–106, February 2000. 7

[14] Gabor T. Herman and Attila Kuba. *Advances in Discrete Tomography and Its Applications*. Birkhäuser, 2007. 1

[15] G.T. Herman and A. Kuba, editors. *Discrete Tomography: Foundations, Algorithms, and Applications*. Birkhäuser, 1999. 1, 6, 23

[16] C. Kisielowski, P. Schwander, F.H. Baumann, M. Seibt, Y. Kim, and A. Ourmazd. An approach to quantitative high-resolution transmission electron microscopy of crystalline materials. *Ultramicroscopy*, 58(2):131 – 155, 1995. 2

[17] Alberto Del Lungo and Maurice Nivat. Reconstruction of connected sets from two projections. In Gabor T. Herman and Attila Kuba, editors, *Discrete Tomography*, article 7, pages 163–188. Birkhäuser, 1999. 7, 9

[18] Alberto Del Lungo, Maurice Nivat, and Renzo Pinzani. The number of convex polyominoes reconstructible from their orthogonal projections. *Discrete Math.*, 157(1-3):65–78, 1996. 28

[19] G.P.M. Prause and D.G.W. Onnasch. Binary reconstruction of the heart chambers from biplane angiographic image sequences. *Medical Imaging, IEEE Transactions on*, 15(4):532–546, Aug 1996. 1

[20] Conceptis Puzzles. Color pic-a-pix, Last accessed in June 2009. `http://www.conceptispuzzles.com/`. 1

[21] H. J. Ryser. Combinatorial properties of matrices of zeroes and ones. *Canad. J. Math.*, 9:371–377, 1957. 6

[22] P. Schwander, C. Kisielowski, M. Seibt, F. H. Baumann, Y. Kim, and A. Ourmazd. Mapping projected potential, interfacial roughness, and composition in general crystalline solids by quantitative transmission electron microscopy. *Phys. Rev. Lett.*, 71(25):4150–4153, Dec 1993. 2

[23] Abe Shliferstein and Y. T. Chien. Switching components and the ambiguity problem in the reconstruction of pictures from their projections. *Pattern Recognition*, 10(5-6):327 – 340, 1978. 1

[24] Gerhard J. Woeginger. The reconstruction of polyominoes from their orthogonal projections. *Information Processing Letters*, 77(5-6):225–229, March 2001. 6