Reconstructing hv-Convex Polyominoes from Orthogonal Projections

Marek Chrobak [a,1] Christoph Dürr [b]

[a] *Department of Computer Science, University of California, Riverside, CA 92521-0304.* marek@cs.ucr.edu

[b] *International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, CA 94704-1198.* cduerr@icsi.berkeley.edu

Key words: Combinatorial problems, discrete tomography, polyominoes.

1 Introduction

Tomography is the area of reconstructing objects from projections. In *discrete tomography*, an object T we wish to reconstruct is a set of cells of a multidimensional grid. We perform measurements of T, each one involving a projection that determines the number of cells in T on all lines parallel to the projection's direction. Given a finite number of such measurements, we wish to reconstruct T or, if unique reconstruction is not possible, to compute any object consistent with these projections. Gardner et al. [7] proved that deciding if there is an object consistent with given measurements is NP-complete, even for three non-parallel projections in the 2D grid.

In this paper we address the case of orthogonal (horizontal and vertical) projections of a 2D grid. A given object T, defined as a set of cells in a $m \times n$ grid, can be identified with an $m \times n$ 0-1 matrix, where the 1's determine the cells of T. We will use all three notations: set-theoretic, integer and boolean, whichever is most appropriate in a given context.

The *row* and *column sums* of an object T are defined by $rowsum_i(T) = \sum_j T_{i,j}$ for $i = 1, \ldots, m$ and $colsum_j(T) = \sum_i T_{i,j}$ for $j = 1, \ldots, n$. The vectors $rowsum(T)$ and $colsum(T)$ represent the horizontal and vertical projections of T. (See Figure 1.)

[1] Research supported by NSF grant CCR-9503498.

Given two integer vectors $\mathbf{r} = (r_1, \ldots, r_m)$ and $\mathbf{c} = (c_1, \ldots, c_n)$, a *realization* of $(\mathbf{r}, \mathbf{c})$ is an object T whose row and column sum vectors are $\mathbf{r}$ and $\mathbf{c}$, that is: $rowsum(T) = \mathbf{r}$ and $colsum(T) = \mathbf{c}$. In the *reconstruction problem*, given $(\mathbf{r}, \mathbf{c})$, we wish to find a realization T of $(\mathbf{r}, \mathbf{c})$, or to report failure if such T does not exist. The corresponding decision problem is called the *consistency problem*.

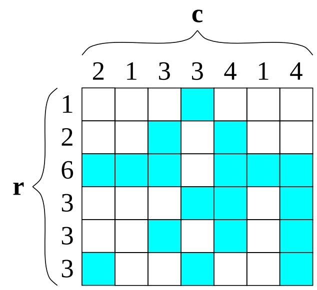

Fig. 1: An object with its row and column sums.

Properties of 0-1 matrices with given row and column sums have been studied in the literature since 1950's. We refer the reader to the work of Ryser [9], or to an excellent survey by Brualdi [4]. Ryser presents an $O(mn)$ algorithm for the reconstruction problem. In fact, the realizations constructed by his algorithm can be represented and computed in time $O(m + n)$.

For many objects, the orthogonal projections do not provide sufficient information for unique reconstruction. In this case, additional information about the object's structure could either lead to a unique realization, or at least substantially reduce the number of alternative solutions. Some research has been done on *polyominoes*, which are connected objects. More formally, we associate with an object T a graph, whose vertices are the cells of T, and edges join adjacent cells: $((i,j),(i',j'))$ is an edge iff $|i - i'| + |j - j'| \leq 1$. If this graph is connected, then T is called a polyomino. Woeginger [10] proved that the consistency problem for polyominoes is NP-complete.

Some geometric properties of polyominoes have been studied too. Call an object T *horizontally convex* (or *h-convex*) if the cells in each row i of T are consecutive, that is, if $(i,k),(i,l) \in T$ then $(i,j) \in T$ for all $k \leq j \leq l$. Vertically convex (v-convex) objects are defined analogously. The consistency problem for h-convex objects (whether we require that they are polyominoes or not) is also known to be NP-complete [2].

If T is both h-convex and v-convex, then we say that T is *hv-convex*. Kuba [8] initiated the study of hv-convex polyominoes and proposed a reconstruction algorithm with exponential worst-case time complexity. Quite surprisingly, as shown later by Barcucci et al [2], the reconstruction problem for hv-convex polyominoes can be solved in polynomial time. The algorithm given in [2] is, however, rather slow; its time complexity is $O(m^4n^4)$. In another paper, Barcucci et al. [3] showed that certain "median" cells of $(\mathbf{r}, \mathbf{c})$ must belong to any hv-convex polyomino realization, and, using this result, they developed an $O(m^2n^2)$-time heuristic algorithm. This new algorithm is not guaranteed to correctly reconstruct an object, although the experiments reported in [3] indicate that for randomly chosen inputs this method is very unlikely to fail.

The main contribution of this paper is an $O(mn\min(m^2,n^2))$-time algorithm for reconstructing hv-convex polyominoes. Our algorithm is several orders of magnitude faster than the best previously known algorithm from [2], and is also much simpler than the algorithms from [2,3,8]. In addition, we address a special case of *centered* hv-convex polyominoes, in which $r_i = n$ for some i. In other words, at least one row is completely filled with cells. For this case we show that the reconstruction problem can be solved in time $O(m+n)$.

2 hv-Convex Polyominoes

Throughout the paper, without loss of generality, we assume that $\mathbf{r}, \mathbf{c}$ denote strictly positive row and column sum vectors. We also assume that $\sum_i r_i = \sum_j c_j$, since otherwise $(\mathbf{r}, \mathbf{c})$ do not have a realization.

An object A is called an *upper-left corner region* if $(i+1,j) \in A$ or $(i,j+1) \in A$ implies $(i,j) \in A$. In an analogous fashion we can define other corner regions. By $\overline{T}$ we denote the complement of T. The definition of hv-convex polyominoes directly implies the following lemma.

Lemma 1 *T is an hv-convex polyomino if and only if $\overline{T} = A \cup B \cup C \cup D$, where A, B, C, D are disjoint corner regions (upper-left, upper-right, lower-left and lower-right, respectively) such that* (i) *$(i-1,j-1) \in A$ implies $(i,j) \notin D$, and* (ii) *$(i-1,j+1) \in B$ implies $(i,j) \notin C$.*

Given an hv-convex polyomino T and two row indices $1 \le k,l \le m$, we say that T is *anchored* at (k,l) if $(k,1),(l,n) \in T$. The idea of our algorithm is, given $(\mathbf{r}, \mathbf{c})$, to construct a 2SAT expression (a boolean expression in conjunctive normal form with at most two literals in each clause) $F_{k,l}(\mathbf{r}, \mathbf{c})$ with the property that $F_{k,l}(\mathbf{r}, \mathbf{c})$ is satisfiable iff there is an hv-convex polyomino realization T of $(\mathbf{r}, \mathbf{c})$ that is anchored at (k,l). $F_{k,l}(\mathbf{r}, \mathbf{c})$ consists of several sets of clauses, each set expressing a certain property: "Corners" (Cor), "Disjointness" (Dis), "Connectivity" (Con), "Anchors" (Anc), "Lower bound on column sums" (LBC) and "Upper bound on row sums" (UBR).

$$
\begin{aligned}
Cor &= \bigwedge_{i,j} \left\{ \begin{matrix} A_{i,j} \Rightarrow A_{i-1,j} & B_{i,j} \Rightarrow B_{i-1,j} & C_{i,j} \Rightarrow C_{i+1,j} & D_{i,j} \Rightarrow D_{i+1,j} \\ A_{i,j} \Rightarrow A_{i,j-1} & B_{i,j} \Rightarrow B_{i,j+1} & C_{i,j} \Rightarrow C_{i,j-1} & D_{i,j} \Rightarrow D_{i,j+1} \end{matrix} \right\} \\
Dis &= \bigwedge_{i,j} \left\{ X_{i,j} \Rightarrow \overline{Y}_{i,j} \quad : \text{for symbols } X,Y \in \{A,B,C,D\}, X \neq Y \right\} \\
Con &= \bigwedge_{i,j} \left\{ A_{i,j} \Rightarrow \overline{D}_{i+1,j+1} \quad B_{i,j} \Rightarrow \overline{C}_{i+1,j-1} \right\} \\
Anc &= \overline{A}_{k,1} \wedge \overline{B}_{k,1} \wedge \overline{C}_{k,1} \wedge \overline{D}_{k,1} \ \wedge \ \overline{A}_{l,n} \wedge \overline{B}_{l,n} \wedge \overline{C}_{l,n} \wedge \overline{D}_{l,n}
\end{aligned}
$$

$$LBC = \bigwedge_{i,j} \left\{ \begin{array}{ll} A_{i,j} \Rightarrow \overline{C}_{i+c_j,j} & A_{i,j} \Rightarrow \overline{D}_{i+c_j,j} \\ B_{i,j} \Rightarrow \overline{C}_{i+c_j,j} & B_{i,j} \Rightarrow \overline{D}_{i+c_j,j} \end{array} \right\} \quad \wedge \quad \bigwedge_j \left\{ \overline{C}_{c_j,j} \quad \overline{D}_{c_j,j} \right\}$$

$$UBR = \bigwedge_j \left\{ \begin{array}{ll} \bigwedge_{i \leq \min\{k,l\}} \overline{A}_{i,j} \Rightarrow B_{i,j+r_i} & \bigwedge_{k \leq i \leq l} \overline{C}_{i,j} \Rightarrow B_{i,j+r_i} \\ \bigwedge_{l \leq i \leq k} \overline{A}_{i,j} \Rightarrow D_{i,j+r_i} & \bigwedge_{\max\{k,l\} \leq i} \overline{C}_{i,j} \Rightarrow D_{i,j+r_i} \end{array} \right\}$$

Define $F_{k,l}(\mathbf{r},\mathbf{c}) = \mathit{Cor} \wedge \mathit{Dis} \wedge \mathit{Con} \wedge \mathit{Anc} \wedge \mathit{LBC} \wedge \mathit{UBR}$. All literals with indices outside the set $\{1,\ldots,m\} \times \{1,\ldots,n\}$ are assumed to have value 1.

Algorithm 1
Input: $\mathbf{r} \in \mathbb{N}^m$, $\mathbf{c} \in \mathbb{N}^n$
W.l.o.g assume: $\forall i : r_i \in [1,n]$, $\forall j : c_j \in [1,m]$, $\sum_i r_i = \sum_j c_j$ and $m \leq n$.
For $k,l = 1,\ldots,m$ **do begin**
  If $F_{k,l}(\mathbf{r},\mathbf{c})$ is satisfiable,
  then output $T = \overline{A \cup B \cup C \cup D}$ and **halt**.
end
output "failure".

Theorem 2 *$F_{k,l}(\mathbf{r},\mathbf{c})$ is satisfiable if and only if $(\mathbf{r},\mathbf{c})$ have a realization T that is an hv-convex polyomino anchored at (k,l).*

Proof ($\Leftarrow$) If T is an hv-convex polyomino realization of $(\mathbf{r},\mathbf{c})$ anchored at (k,l), then let A, B, C, D be the corner regions from Lemma 1. By Lemma 1, A, B, C and D satisfy conditions (Cor), (Dis) and (Con). Condition (Anc) is true because T is anchored at (k,l). (LBC) and (UBR) hold because T is a realization of $(\mathbf{r},\mathbf{c})$.

($\Rightarrow$) If $F_{k,l}(\mathbf{r},\mathbf{c})$ is satisfiable, take $T = \overline{A \cup B \cup C \cup D}$. Conditions (Cor), (Dis) and (Con) imply that the sets A, B, C, D satisfy Lemma 1, and thus T is an hv-convex polyomino. Also, by (Anc), T is anchored at (k,l).

It remains to show that T is a realization of $(\mathbf{r},\mathbf{c})$. Conditions (UBR) and (LBC) imply that $\mathit{rowsum}_i(T) \leq r_j$ for each row i, and $\mathit{colsum}_j(T) \geq c_j$ for each column j. Thus

$$\sum_i r_i \geq \sum_i \mathit{rowsum}_i(T) = \sum_j \mathit{colsum}_j(T) \geq \sum_j c_j.$$

Since $\sum_i r_i = \sum_j c_j$, T must be a realization of $(\mathbf{r},\mathbf{c})$, and the proof is complete. □

Each formula $F_{k,l}(\mathbf{r},\mathbf{c})$ has size $O(mn)$ and can be computed in time $O(mn)$. Since 2SAT can be solved in linear time [1,6], we obtain the following result.

Theorem 3 *Algorithm 1 solves the reconstruction problem for hv-convex polyominoes in time $O(mn\min(m^2,n^2))$.*

An implementation (see [5]) of Algorithm 1 can be made more efficient by reducing the number of choices for k, l. First, restrict k to multiples of c_1, and l to multiples of c_n. Second, let m_1 be the largest index for which $r_1 \leq \cdots \leq r_{m_1}$, and let m_2 be the smallest index for which $r_{m_2} \geq \cdots \geq r_m$. It is easy to see that we can assume that $\min\{k,l\} \leq m_1$ and $\max\{k,l\} \geq m_2$ (see [3].) Analogously we define column indices n_1 and n_2. With these restrictions on k, l, we run the 2SAT algorithm only $O(\min(m_1 m_2/c_1 c_n, n_1 n_2/r_1 r_m))$ times.

One can also remove redundant clauses from formulas $F_{k,l}(\mathbf{r}, \mathbf{c})$. For example, in the clauses (Dis), we only need to state that regions $A \cap B = \emptyset$ and $C \cap D = \emptyset$. The other disjointness relations follow from condition (LBC). Finally, it is not necessary to reconstruct each $F_{k,l}(\mathbf{r}, \mathbf{c})$ from scratch, since only clauses (UBR) and (Anc) depend on k and l.

3 Centered hv-Convex Polyominoes

Structure of realizations. A *rectangle* R is the intersection of a set of rows and a set of columns, say $R = I \times J$, for $I \subseteq \{1, \ldots, m\}$ and $J \subseteq \{1, \ldots, n\}$. The rectangle *orthogonal* to R is $R^\perp = \overline{I} \times \overline{J}$, where $\overline{I}$ and $\overline{J}$ are the complements of I, J.

The lemma below follows from the work of Ryser [9] (see also [4]). We include a simple proof for completeness.

Lemma 4 *Let $R = I \times J$ be a rectangle such that $\sum_{i \in I} r_i = \sum_{j \notin J} c_j + |I| \cdot |J|$. Then each realization T of $(\mathbf{r}, \mathbf{c})$ satisfies $R \subseteq T$ and $R^\perp \cap T = \emptyset$.*

Proof If T is a realization of $(\mathbf{r}, \mathbf{c})$ then

$$
\begin{aligned}
|R \cap T| - \left|R^\perp \cap T\right| &= \sum_{i \in I} \sum_{j \in J} T_{i,j} - \sum_{i \notin I} \sum_{j \notin J} T_{i,j} \;=\; \sum_{i \in I} \sum_{j} T_{i,j} - \sum_{i} \sum_{j \notin J} T_{i,j} \\
&= \sum_{i \in I} r_i - \sum_{j \notin J} c_j \;=\; |I| \cdot |J|,
\end{aligned}
$$

and therefore $R \subseteq T$ and $R^\perp \cap T = \emptyset$. □

Representation of hv-convex realizations. Since we are interested in *centered hv-convex* realizations, we assume now that $r_k = n$, for some row k. We could also assume that the row sums satisfy $r_1 \leq \cdots \leq r_k = n \geq \cdots \geq r_m$, although our algorithm does not use this assumption.

We represent an hv-convex realization as a vector $T = (t_1, \ldots, t_m)$, where each $t_i \in [1, m - r_i + 1]$ is the smallest j for which $(i,j) \in T$. Thus $(i,j) \in$

T is equivalent to $t_i \leq j < t_i + r_i$. By definition, such realizations satisfy the horizontal convexity and the row-sum conditions. The vertical convexity condition can be written as

$$t_{i+\delta} \leq t_i \leq t_{i+\delta} + r_{i+\delta} - r_i \tag{VC}$$

where $\delta = 1$ for $i < k$ and -1 for $i > k$. The column sums can be expressed by the t_i as $colsum_j(T) = |\{i : t_i \leq j < t_i + r_i\}|$.

Partial realizations. Let $X = (t_p, \ldots, t_q)$, where $(p, q) \neq (1, m)$ and $k \in [p, q]$, be a vector in which $t_i \in [1, m - r_i + 1]$, for each i. We call column j *unsaturated* if $colsum_j(X) < c_j$, where $colsum_j(X) = |\{i : t_i \leq j < t_i + r_i\}|$. Let α_X be the first unsaturated column and β_X the last. (We will omit the subscript X if X is understood from context.) X is called a *$[p, q]$-realization* (or a *partial realization*) if it satisfies condition (VC) for $i = p, \ldots, q$, and

$$\begin{aligned} \min\{t_p, t_q\} \leq \alpha \leq \beta < \max\{t_p + r_p, t_q + r_q\}, \\ colsum_j(X) = c_j \quad \text{for } j \notin [\alpha, \beta]. \end{aligned}$$

If $\max\{t_p, t_q\} \leq \alpha$ and $\beta < \min\{t_p + r_p, t_q + r_q\}$, then X is called *balanced.* Note that in the definition of partial realizations we do not insist that

$$colsum_j(X) \leq c_j \quad \text{for } j = \alpha + 1, \ldots, \beta - 1. \tag{1}$$

We call X *valid* if it satisfies (1). Clearly, an invalid partial realization cannot be extended to a realization, but to facilitate a linear-time implementation we will verify (1) only for balanced partial realizations.

Our algorithm will attempt to construct a realization row by row in the order of decreasing sums. A $[p, q]$-realization X will be extended to row $p - 1$ or $q + 1$, whichever has larger sum. Each X has at most two extensions, and it may have two extensions only if it is balanced. The naive approach would be to explore recursively all extensions, but, if the number of balanced realizations is large, it could result in exponential running time. To reduce the number of partial realizations to explore we show, using Lemma 4, that if $(\mathbf{r}, \mathbf{c})$ has any hv-convex realization at all, then each valid balanced $[p, q]$-realization X can be extended to a realization of $(\mathbf{r}, \mathbf{c})$. Therefore when we encounter a valid balanced $[p, q]$-realization, we can discard all other $[p, q]$-realizations. Consequently, we keep at most two partial realizations at each step.

Suppose that either $q = m$ or $r_{p-1} \geq r_{q+1}$, and let $lext_{p-1}(X)$, $rext_{p-1}(X)$ be the extensions of X with $t_{p-1} = \alpha$ and $t_{p-1} = \beta - r_{p-1} + 1$, respectively. If X is non-balanced, or balanced and valid, $Ext_{p-1,q}(X)$ is the set containing these vectors in $\{lext_{p-1}(X), rext_{p-1}(X)\}$ which are $[p-1, q]$-realizations. If X is balanced but not valid then $Ext_{p-1,q}(X) = \emptyset$. Analogously we define the set $Ext_{p,q+1}(X)$ for the case when $p = 1$ or $r_{q+1} \geq r_{p-1}$.

Algorithm 2 (See Fig. 2)
Input: $\mathbf{r} \in \mathbb{N}^m$, $\mathbf{c} \in \mathbb{N}^n$
W.l.o.g. assume: $\forall i : r_i \in [1,n]$, $\forall j : c_j \in [1,m]$, $\sum_i r_i = \sum_j c_j$ and $\exists k : r_k = n$.

$t_k \leftarrow 1$, $X \leftarrow (t_k)$, $\mathbf{P} \leftarrow \{X\}$, $(p,q) \leftarrow (k,k)$,
While $(p,q) \neq (1,m)$ **do begin**
  If $q = m$ or $r_{p-1} \geq r_{q+1}$ **then** $p \leftarrow p-1$ **else** $q \leftarrow q+1$
  If $\exists X \in \mathbf{P}$ such that X is valid and balanced
  then $\mathbf{P} \leftarrow \mathit{Ext}_{p,q}(X)$
  else $\mathbf{P} \leftarrow \bigcup_{X \in \mathbf{P}} \mathit{Ext}_{p,q}(X)$
end
If P contains a realization T **then** output T **else** output "failure".

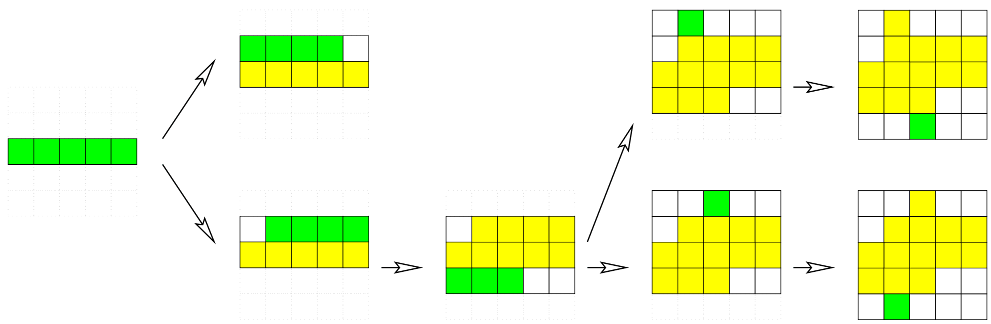

Fig. 2. Execution of Algorithm 2 on $\mathbf{r} = (1,4,5,3,1)$ and $\mathbf{c} = (2,4,4,2,2)$.

The correctness of Algorithm 2 follows from the following lemma.

Lemma 5 *Suppose that* $(\mathbf{r},\mathbf{c})$ *has an hv-convex realization. Then, after each step of Algorithm 2,* $|\mathbf{P}| \leq 2$ *and there is* $X \in \mathbf{P}$ *that can be extended to an hv-convex realization of* $(\mathbf{r},\mathbf{c})$.

Proof The proof is by induction on $q-p+1$. Denote by $\mathbf{P}_{p,q}$ the set $\mathbf{P}$ at the beginning of the **while** loop. The lemma holds trivially for $p = q = k$. Assume that it holds for some $\mathbf{P}_{p,q}$. Without loss of generality, assume that either $q = m$ or $r_{p-1} \geq r_{q+1}$.

Suppose first that $\mathbf{P}_{p,q}$ contains a valid balanced $[p,q]$-realization X. Let $I = [p,q]$, $J = [\alpha_X, \beta_X]$ and $R = I \times J$. Then $\sum_{i\in I} r_i = \sum_{j \notin J} c_j + |I| \cdot |J|$. By Lemma 4, we get that for each realization $T = (z_1,\ldots,z_m)$, $R \subseteq T$ and $R^\perp \cap T = \emptyset$. If we replace $z_p,\ldots,z_q$ by $t_p,\ldots,t_q$, we get another realization T' that is an extension of X. Since $q = m$ or $r_{p-1} \geq r_{q+1}$, z_{p-1} must be either α_X or $\beta_X - r_{p-1} + 1$. Therefore T' is an extension of one of $\mathit{rext}_{p-1}(X)$, $\mathit{lext}_{p-1}(X)$. Since $\mathbf{P}_{p-1,q} = \mathit{Ext}_{p-1,q}(X)$, the lemma holds for $\mathbf{P}_{p-1,q}$.

If $\mathbf{P}_{p,q}$ does not contain a valid balanced $[p,q]$-realization then, for $X \in \mathbf{P}_{p,q}$, either $\alpha_X < \max\{t_p, t_q\}$ or $\beta_X \geq \min\{t_p + r_p, t_q + r_q\}$. Thus each $X \in \mathbf{P}_{p,q}$ gives rise to at most one extension in $\mathbf{P}_{p-1,q}$, so we have $|\mathbf{P}_{p-1,q}| \leq 2$. Pick $X \in \mathbf{P}_{p,q}$ that can be extended to a realization $T = (t_1, \ldots, t_m)$. We can assume $\alpha_X < \max\{t_p, t_q\}$, since the other case is symmetric. If $t_p \leq \alpha_X < t_q$, then $t_{p-1} = \alpha_X$, and T is an extension of $\mathit{lext}_{p-1}(X) \in \mathbf{P}_{p-1,q}$. If $t_q \leq \alpha_X < t_p$, then $t_{q+1} = \alpha_X$, and thus, by $t_{p-1} \geq t_{q+1}$, we have $t_{p-1} = \beta_X - r_{p-1} + 1$. Therefore T is an extension of $\mathit{rext}_{p-1}(X) \in \mathbf{P}_{p-1,q}$. □

Linear-time implementation. A naive implementation of Algorithm 2 takes time $O(mn)$. We show how to reduce the running time to $O(m+n)$.

The computation can be divided into phases, where each phase starts when a valid balanced partial realization Y is encountered. During a phase, for each $X \in \mathbf{P}$, we store only rows which are not in Y, and when the phase ends the new rows are copied into Y. We also maintain the indices $\alpha = \alpha_X$ and $\beta = \beta_X$. Call a column j *active* if $j \in [\alpha, \beta]$. To keep track of the sums of the active columns, we use two arrays hisum_j and losum_j which represent, respectively, the column sums of rows $[p, k-1]$ and $[k+1, q]$ in X. Only the values of hisum_j for $j \in [\alpha_X, t_p - 1] \cup [t_p + r_p, \beta_X]$ are stored explicitly. For $j = [t_p, t_p + r_p - 1]$, we know that $\mathit{hisum}_j = k - p - 1$. When t_p increases or $t_p + r_p$ decreases, we simply fill the new entries with the correct values. The array losum_j is maintained analogously. In this way, we can maintain these arrays in total time $O(m+n)$ and we can query colsum_j in time $O(1)$, since $\mathit{colsum}_j = \mathit{hisum}_j + 1 + \mathit{losum}_j$.

It remains to show that condition (1) can be tested in total time $O(m+n)$. We keep the list A of the active columns j in the order of increasing sums. Let ℓ be the first column in A, that is $c_\ell = \min_{j \in [\alpha,\beta]} c_j$. Then for a balanced realization X, (1) holds iff $q - p + 1 \leq c_\ell$, and thus each verification of (1) takes time $O(1)$. Initially, A can be constructed in time $O(m+n)$ by bucket-sort. Each active column has a pointer to its record in A. When we increase α or decrease β, we simply remove from A the columns that become non-active, at cost $O(1)$ per deletion.

In summary, we obtain the following result.

Theorem 6 *Algorithm 2 solves the reconstruction problem for centered hv-convex polyominoes, and it can be implemented in time $O(m+n)$.*

4 Final Comments

To speed up Algorithm 1 further, one can explore the fact that the consecutive expressions $F_{k,l}(\mathbf{r}, \mathbf{c})$ differ only very slightly. Thus, an efficient *dynamic* algorithm for 2SAT could be used to improve the time complexity. The 2SAT problem is closely related to strong connectivity in directed graphs, for which no efficient dynamic algorithms are known. Nevertheless, it may be possible to take advantage of the special structure of the 2SAT instances arising in our algorithm, as well as the fact that the sequence of clause insertions/deletions in $F_{k,l}(\mathbf{r},\mathbf{c})$ is known in advance.

It would be interesting to investigate polyominoes which are digital images of convex shapes. Call a polyomino T *convex* if its convex hull does not contain any whole cells outside T. How fast can we reconstruct convex polyominoes?

References

[1] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[2] E. Barcucci, A. Del Lungo, M. Nivat, and R. Pinzani. Reconstructing convex polyominoes from horizontal and vertical projections. *Theoretical Computer Science*, 155(2):321–347, 1996.

[3] E. Barcucci, A. Del Lungo, M. Nivat, and R. Pinzani. Medians of polyominoes: a property for the reconstruction. *International Journal of Imaging Systems and Technology*, 1998. to appear.

[4] R.A. Brualdi. Matrices of zeros and ones with fixed row and column sum vectors. *Linear Algebra and Applications*, 33:159–231, 1980.

[5] P. Couperous and C. Dürr. *http://www.icsi.berkeley.edu/˜cduerr /Xray.*

[6] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, 1976.

[7] R.J. Gardner, P. Gritzmann, and D. Prangenberg. On the computational complexity of reconstructing lattice sets from their X-rays. Technical Report 970.05012, Techn. Univ. München, Fak. f. Math., 1997.

[8] A. Kuba. The reconstruction of two-directional connected binary patterns from their two orthogonal projections. *Computer Vision, Graphics, and Image Processing*, 27:249–265, 1984.

[9] H.J. Ryser. *Combinatorial Mathematics.* Mathematical Association of America and Quinn & Boden, Rahway, New Jersey, 1963.

[10] G.J. Woeginger. The reconstruction of polyominoes from their orthogonal projections. Technical Report SFB-65, TU Graz, Graz, Austria, April 1996.