ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Discrete Applied Mathematics 151 (2005) 93–105

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Optimization and reconstruction of hv-convex (0, 1)-matrices

Geir Dahl, Truls Flatberg*

Centre of Mathematics for Applications, University of Oslo, P.O. Box 1053 Blindern, 0316 Oslo, Norway

Received 1 October 2003; received in revised form 30 April 2004; accepted 10 February 2005
Available online 6 July 2005

Abstract

We consider a variant of the NP-hard problem of reconstructing hv-convex (0, 1)-matrices from known row and column sums. Instead of requiring the ones to occur consecutively in each row and column, we maximize the number of neighboring ones. This is reformulated as an integer programming problem. A solution method based on variable splitting is proposed and tested with good results on moderately sized test problems.

Keywords: Discrete tomography; Integer programming

1. Introduction

Consider a matrix $A = [a_{i,j}]$ of size $m \times n$ with elements 0 or 1. Let $R = (r_1, \ldots, r_m)$ and $S = (s_1, \ldots, s_n)$ denote the row and column sums of A, respectively, that is, $r_i = \sum_{j=1}^{n} a_{i,j}$ and $s_j = \sum_{i=1}^{m} a_{i,j}$ for all $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$. Let $\mathfrak{A}(R, S)$ be the set of all (0, 1)-matrices with row sums R and column sums S. Clearly, for such matrices to exist we must have that $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} s_j$. We denote this sum (value) by τ. We assume throughout the paper that $1 \leqslant r_i \leqslant m$ and $1 \leqslant s_j \leqslant n$.

* Corresponding author.
E-mail address: trulsf@ifi.uio.no (T. Flatberg).

0166-218X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2005.02.023

In 1957 it was shown independently by Gale [5] and Ryser [12] that a necessary and sufficient condition for the existence of a $(0, 1)$-matrix with row sums R and column sums S is that $S \prec R^*$ where R^* is the conjugate sequence of R and $\prec$ denotes the majorization ordering. See Marshall and Olkin [10] for a thorough discussion of majorization theory and applications.

The results of Gale and Ryser have found a recent revival in the field of discrete tomography [7]. In discrete tomography one of the problems is to reconstruct a discrete-valued function f from knowledge of weighted sums of function values over subsets of the domain. Applications can be found in crystallography [13] and medical imaging [3] amongst others. A much studied special case is $m \times n$ $(0, 1)$-matrices with known row and column sums, precisely matrices in the class $\mathfrak{A}(R, S)$.

As the number of matrices in this class may be high [14], it is of interest to study the reconstruction problem where we impose additional constraints on the $(0, 1)$-matrices. The most common restrictions are of a geometrical nature. A $(0, 1)$-matrix is h-convex if the ones in a row form a contiguous interval, similarly a $(0, 1)$-matrix is v-convex if the ones occurs contiguously in each column. A $(0, 1)$-matrix is hv-convex if it is both h- and v-convex. If the pattern of ones is four-connected, it is called a polyomino. It was shown by Barcucci et al. [2] that the existence problem for both h- and v-convex matrices having given row and column sums is NP-complete while it is polynomially solvable for hv-convex polyominoes. Similarly, Woeginger [15] showed that the existence problem for hv-convex matrices is NP-complete.

In this paper we examine a new approach to finding hv-convex or nearly hv-convex matrices. Optimizing over the class $\mathfrak{A}(R, S)$, we try to find a matrix with a maximum number of neighboring ones in rows and columns. Let X be a $(0, 1)$-matrix and let $x_{i,j}$ denote the entry of the matrix in position (i, j). We introduce a function $f(X)$ that counts the number of neighboring ones

$$f(X) = \sum_{i=1}^{m-1} \sum_{j=1}^{n} \min\{x_{i,j}, x_{i+1,j}\} + \sum_{j=1}^{n-1} \sum_{i=1}^{m} \min\{x_{i,j}, x_{i,j+1}\}. \tag{1}$$

Using this function as our objective function, we consider the following optimization problem:

$$\max\{f(X) : X \in \mathfrak{A}(R, S)\}. \tag{2}$$

We denote this problem by *MAX_NB*.

This problem may be formulated as an integer programming problem. Let $G = (V, E)$ be the graph with a vertex for each entry in X and edges between neighboring entries in both row and column directions. Introduce a binary variable y_e for each edge in this graph. We denote by I the following maximization problem

$$v^I = \max \sum_{e \in E} y_e$$

$$\begin{aligned}\text{subject to} \quad & \text{(i)} \ \sum_{j=1}^{n} x_{i,j} = r_i \quad \text{for } i = 1, \ldots, m,\\ & \text{(ii)} \ \sum_{i=1}^{m} x_{i,j} = s_j \quad \text{for } j = 1, \ldots, n,\\ & \text{(iii)} \ y_e \leqslant x_{i,j} \quad \text{for all } e \in E, (i,j) \in e,\\ & \text{(iv)} \ x_{i,j}, y_e \in \{0,1\}. \end{aligned} \tag{3}$$

Constraints (i) and (ii) ensure that the matrix $X = [x_{i,j}]$ is a member of $\mathfrak{A}(R, S)$. Constraint (iii) ensures that y_e can be set to one only if both neighboring entries are set to one. Note that constraint (iii) represents two constraints for each edge.

We observe that for each optimal solution (x, y) of I, it holds that

$$y_e = \min\{x_{i,j}, x_{i',j'}\} \quad \text{for } e = [(i,j),(i',j')], \tag{4}$$

and therefore, $v^I = \max f(X)$. Thus, the problem *MAX_NB* corresponds to solving problem I.

We mention that it may be of interest to study the problem with a more general objective function $\sum_e y_e + \sum_{i,j} d_{i,j} x_{i,j}$, allowing us to model situations where we have a priori knowledge about some preferred positions for entries 1 in the matrix. In this article we will only focus on the simpler version.

The paper is organized as follows. Section 2 gives complexity results and some properties of the linear programming relaxation of I. Section 3 introduces a solution method based on variable splitting. Section 4 interprets the method in terms of linear programming. We implemented and tested the methods. The results of these tests are reported in Section 5.

2. Bounds

If a $(0, 1)$-matrix is hv-convex, both rows and columns have ones occurring contiguously, then the number of neighboring ones is

$$\hat{v}_1 = \sum_{i=1}^{m} (r_i - 1) + \sum_{j=1}^{n} (s_j - 1) = 2\tau - m - n.$$

Clearly this is an upper bound for $f(X)$. Moreover, a $(0, 1)$-matrix X is hv-convex if and only if $f(X) = \hat{v}_1$.

If we can find an optimal solution (x, y) to problem I, the corresponding matrix $X = [x_{i,j}]$ will be hv-convex if $f(X) = v^I = \hat{v}_1$. If v^I is strictly less than $\hat{v}_1$, no hv-convex matrix with the given row and column sums exists. Since the existence problem for a hv-convex matrix is NP-complete [15], we have proved the following.

Theorem 2.1. *MAX_NB is NP-hard.*

An alternative upper bound for $v^I = \max f(X)$ can be derived by considering two neighboring rows (or columns). The maximum number of possible neighbors between two rows

(or columns) is equal to the minimum number of ones in each of the two rows. Summing this over all neighboring rows and columns, we get the bound

$$f(X) \leqslant \hat{v}_2 = \sum_{i=1}^{m-1} \min\{r_i, r_{i+1}\} + \sum_{j=1}^{n-1} \min\{s_j, s_{j+1}\}.$$

An easy observation is that $\hat{v}_1 \leqslant \hat{v}_2$ is a necessary condition for the existence of a hv-convex $(0, 1)$-matrix in $\mathfrak{A}(R, S)$. If both R and S are unimodal vectors, i.e. $r_1 \leqslant r_2 \leqslant \cdots \leqslant r_k > r_{k+1} \geqslant \cdots \geqslant r_m$ and $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_l > s_{l+1} \geqslant \cdots \geqslant s_n$, we have that $\hat{v}_2 = 2\tau - r_k - s_l$. Since $r_k + s_l \leqslant m + n$, it follows that $\hat{v}_1 \leqslant \hat{v}_2$ holds in general for this class of problems.

Let RI be the linear programming relaxation of I

$$\begin{aligned} v^{RI} = \ & \max \sum_e y_e \\ \text{subject to} \quad & \text{(i)} \ \sum_{j=1}^{n} x_{i,j} = r_i \quad \text{for } i = 1, \ldots, m, \\ & \text{(ii)} \ \sum_{i=1}^{m} x_{i,j} = s_j \quad \text{for } j = 1, \ldots, n, \\ & \text{(iii)} \ 0 \leqslant y_e \leqslant x_{i,j} \quad \text{for all } e \in E, (i, j) \in e, \\ & \text{(iv)} \ 0 \leqslant x_{i,j} \leqslant 1 \quad i \leqslant m, \quad j \leqslant n. \end{aligned}$$

Define the corresponding matrix $X = [x_{i,j}]$ and vector $y = [y_e]$. Then the following result holds.

Proposition 2.2. $v^{RI} \leqslant \hat{v}_2$.

Proof. Let (X, y) be a feasible solution to RI. Consider two neighboring rows i and $i + 1$ and let $y_{i,i+1,j}$ denote the variable corresponding to the edge between rows i and $i + 1$ in column j. Then

$$\begin{aligned} \sum_{j=1}^{n} y_{i,i+1,j} &\leqslant \sum_{j=1}^{n} x_{i,j} = r_i, \\ \sum_{j=1}^{n} y_{i,i+1,j} &\leqslant \sum_{j=1}^{n} x_{i+1,j} = r_{i+1}. \end{aligned}$$

Combining these two inequalities leads to the inequality

$$\sum_j y_{i,i+1,j} \leqslant \min\{r_i, r_{i+1}\}.$$

Since this inequality holds in general for all $1 \leqslant i \leqslant m$, and due to a similar inequality for columns, we get the desired result. □

Remark. Assume that $\max_{i,j} r_i s_j \leqslant \tau$, then equality holds in Proposition 2.2. In fact an optimal solution to *RI* is given by

$$x_{i,j} = \frac{r_i s_j}{\tau} \tag{5}$$

with the y variables set according to (4)

$$y_e = \min\left\{\frac{r_i s_j}{\tau}, \frac{r_{i'} s_{j'}}{\tau}\right\} \quad \text{for } e = [(i, j), (i', j')].$$

Clearly this is a feasible solution (as $x_{i,j} \leqslant 1$) and the corresponding objective, v^{RI} is

$$\begin{aligned} v^{RI} &= \sum_{i=1}^{m-1}\sum_{j=1}^{n} \min\left\{\frac{r_i s_j}{\tau}, \frac{r_{i+1} s_j}{\tau}\right\} + \sum_{j=1}^{n-1}\sum_{i=1}^{m} \min\left\{\frac{r_i s_j}{\tau}, \frac{r_i s_{j+1}}{\tau}\right\} \\ &= \sum_{i=1}^{m-1} \min\{r_i, r_{i+1}\} + \sum_{j=1}^{n-1} \min\{s_j, s_{j+1}\} = \hat{v}_2. \end{aligned}$$

The matrix with elements defined by (5) is one example of a larger class of matrices for which equality holds. We call a matrix row-graded if $r_i \geqslant r_{i+1}$ implies $x_{i,j} \geqslant x_{i+1,j}$ for $j = 1, \ldots, n$ and $r_i \leqslant r_{i+1}$ implies $x_{i,j} \leqslant x_{i+1,j}$ for $j = 1, \ldots, n$. Column-graded matrices are defined similarly. The matrix defined by (5) is clearly both row- and column-graded. We have the following result.

Lemma 2.3. *If X is both row- and column-graded, then $v^{RI} = \hat{v}_2$.*

Proof. We consider the edge variables connecting two neighboring rows (or columns) with row sums r_i and r_{i+1}. Each edge variable will be set to the minimum of the two adjacent x variables. Since X is row-graded the minimum values will all occur in the same row. Thus the sum of all edge variables is equal to $\min(r_i, r_{i+1})$. Repeating this for all neighboring pairs of rows and columns gives the result. □

Empirical testing on a large number of random generated test cases indicates that the optimal solutions of *RI* are both row- and column-graded also for the case $\max_{i,j} r_i s_j > \tau$, and thus the bound is obtained.

3. Variable splitting

We propose to solve problem *I* using Lagrangian relaxation techniques on the integer programming formulation (3). More specifically, we will use a technique known as variable splitting, see [6].

We duplicate the set of $x_{i,j}$ variables, obtaining two sets of variables, $x^h_{i,j}$ and $x^v_{i,j}$. To ensure that two corresponding variables $x^h_{i,j}$ and $x^v_{i,j}$ share the same value in the feasible solutions, we add a constraint stating that the corresponding variables should be equal.

The following is a reformulation of I:

$$
\begin{aligned}
v^I = \quad & \max \sum_e y_e \\
\text{subject to} \quad & \text{(i)} \ \sum_{j=1}^{n} x^h_{i,j} = r_i \quad \text{for } i = 1, \ldots, m, \\
& \text{(ii)} \ \sum_{i=1}^{m} x^v_{i,j} = s_j \quad \text{for } j = 1, \ldots, n, \\
& \text{(iiia)} \ y_e \leqslant x^h_{i,j} \quad \text{for all horizontal } e, (i,j) \in e, \\
& \text{(iiib)} \ y_e \leqslant x^v_{i,j} \quad \text{for all vertical } e, (i,j) \in e, \\
& \text{(iv)} \ x^h_{i,j} = x^v_{i,j} \quad \text{for all } i, j, \\
& \text{(v)} \ x^h_{i,j}, x^v_{i,j}, y_e \in \{0,1\}.
\end{aligned} \tag{6}
$$

We relax constraint (iv) using Lagrangian multipliers $\lambda_{i,j}$ and get the problem

$$
\begin{aligned}
\max \quad & \sum_e y_e + \sum_{i,j} \lambda_{i,j}(x^h_{i,j} - x^v_{i,j}) \\
\text{subject to} \quad & \text{(i)} \ \sum_{j=1}^{n} x^h_{i,j} = r_i \quad \text{for } i = 1, \ldots, m, \\
& \text{(ii)} \ \sum_{i=1}^{m} x^v_{i,j} = s_j \quad \text{for } j = 1, \ldots, n, \\
& \text{(iiia)} \ y_e \leqslant x^h_{i,j} \quad \text{for all horizontal } e, (i,j) \in e, \\
& \text{(iiib)} \ y_e \leqslant x^v_{i,j} \quad \text{for all vertical } e, (i,j) \in e, \\
& \text{(v)} \ x^h_{i,j}, x^v_{i,j}, y_e \in \{0,1\}.
\end{aligned} \tag{7}
$$

We denote this problem $VSI(\lambda)$ and its optimal value $v^{VSI}(\lambda)$. Because of the variable splitting the problem naturally divides into separate horizontal and vertical problems which can be further separated into subproblems for each row and column.

Since the above problem is a relaxation of the original problem, $v^{VSI}(\lambda)$ is an upper bound on the value of I. We want to find the best possible bound, that is, solve the Lagrangian dual problem

$$
v^{VSD} = \min_{\lambda} v^{VSI}(\lambda). \tag{8}
$$

This is a convex, non-differentiable optimization problem which can be solved by a subgradient method. For an introduction to subgradient methods see [11].

During the subgradient procedure we have to evaluate $VSI(\bar{\lambda})$ for a given $\bar{\lambda}$. As mentioned this reduces to solving a separate subproblem for each row and column. Each of these

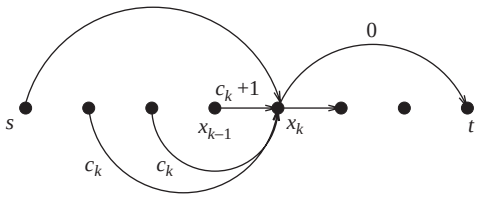


Fig. 1. Arcs going to and from a vertex x_k.

subproblems will be of the form

$$\begin{aligned} \max \quad & \sum_{k=1}^{N-1} y_k + \sum_{k=1}^{N} c_k x_k \\ \text{subject to} \quad & \text{(i)} \ \sum_k x_k = b, \\ & \text{(ii)} \ y_k \leqslant x_k, \\ & \text{(iii)} \ y_k \leqslant x_{k+1}, \\ & \text{(iv)} \ y_k, x_k \in \{0,1\}, \end{aligned} \tag{9}$$

where N is n (m) for row (column) subproblems, b is the suitable r_i or s_j and c_k is the suitable Lagrange multiplier (or its negative if we consider a column). This problem may be solved as the longest (simple) path problem with exactly $b+1$ edges in a directed graph. Let the graph $D=(V, A)$ have a vertex set consisting of one vertex for each x_k variable together with a source vertex s and a target vertex t. The arc set A consists of the following arcs:

- (s, x_k) with cost c_k,
- (x_{k-1}, x_k) with cost c_k+1,
- (x_l, x_k) for $l<k-1$ with cost c_k,
- (x_k, t) with cost 0.

The construction is illustrated in Fig. 1. The longest path from s to t in D with exactly $b+1$ edges corresponds to a solution of problem (9) by setting x_k equal to one if vertex x_k is used in the longest path and zero otherwise. Similarly, we set y_k to one if the longest path visits both x_{k-1} and x_k and zero otherwise.

The longest path problem with a prescribed number of arcs in an acyclic graph can be solved efficiently by dynamic programming. Let $d(i, j)$ denote the value of the longest path from vertex s to vertex i using exactly j edges. The values of $d(i, j)$ can be calculated from

$$\begin{aligned} & d(i,1) = c_i \\ & d(i,j) = \max\{d(u, j-1) + c_j \text{ for } u < i-1,\ d(i-1, j-1) + c_i + 1\} \end{aligned}$$

The optimal value is then $d(t, b+1) = \max_i d(i, b)$ and the corresponding solution can be found by retracing the path in the graph. The algorithm has running time $O(bn^2)$. Thus the problem $VSI(\lambda)$ can be solved efficiently. Still, since b typically is $O(n)$ and the problem

has to be solved for each row and column, the total complexity is $O(n^4)$. Thus, the algorithm may have scaling problems when n becomes large.

4. A reformulation of the Lagrangian dual

In this section we again consider the Lagrangian dual (8) and its optimal value v^{VSD}. We explain how to construct a linear programming problem whose optimal value equals v^{VSD}. Note that this reformulation is in terms of continuous variables (in contrast with problem (7)). Moreover, this LP is a *compact formulation*, meaning that the number of variables and constraints grow polynomially as a function of m and n.

Let $z^h = [x^h_{i,j}, y_e]$ be the vector of the "horizontal" variables $x^h_{i,j}$ $(i \leqslant m,\ j \leqslant n)$ and the variables y_e. Similarly, z^v is the vector of the "vertical" variables $x^v_{i,j}$ $(i \leqslant m,\ j \leqslant n)$ and the variables y_e. As mentioned the Lagrangian subproblem (6) decomposes into a horizontal problem involving z^h and a vertical problem involving z^v. Let $A^h z^h \leqslant b^h$ $(A^v z^v \leqslant b^v)$ denote the constraints in the horizontal (vertical) problem. Then we obtain from the general theory of Lagrangian relaxation and variable splitting (confer [11, Corollary 6.12]) that

$$v^{VSD} = \max \left\{ \sum_e y_e : z = [x_{i,j}, y_e] \in P^h \cap P^v \right\}, \tag{10}$$

where $P^h = \mathrm{conv}(\{z : A^h z \leqslant b^h\})$ and $P^v = \mathrm{conv}(\{z : A^v z \leqslant b^v\})$. Now, the maximization problem in (10) is an LP, but, unfortunately, a complete linear description of P^h (or P^v) is not known. However, we get around this problem by finding an extended formulation for P^h by introducing additional variables. To this end it suffices to find an extended formulation of the feasible set of problem (9).

This can be done by constructing a certain acyclic directed graph $G = (V, A)$. The construction is such that there is a correspondence between feasible solutions in (9) and certain directed paths in the graph.

The graph is organized into $2k-1$ layers with $N_k = N - k + 1$ vertices in each layer, together with a source vertex s and a sink vertex t. Thus the total number of vertices is $(2k-1)N_k + 2$. For the odd numbered layers, $l = 1, 3, \ldots, 2k-1$, label the vertices in layer l as $v^l_{i+(l-1)/2}$, $i = 1, \ldots, N_k$. The vertices in the even-numbered layers are labeled $w_{i+l/2-1}$. The arc set A consists of the following arcs:

- (s, v^1_i) for $i = 1, \ldots, N_k$,
- (v^l_i, v^{l+2}_j) for all l and $i < j$,
- (v^l_i, w^{l+1}_i) for all i, l,
- (w^l_i, v^{l+1}_{i+1}) for all i, l,
- (v^{2k-1}_i, t) for $i = k, \ldots, N$.

The total number of arcs is $\frac{1}{2}N_k(N_k-1)(k-1) + 2kN_k$. The graph is illustrated for a small case in Fig. 2.

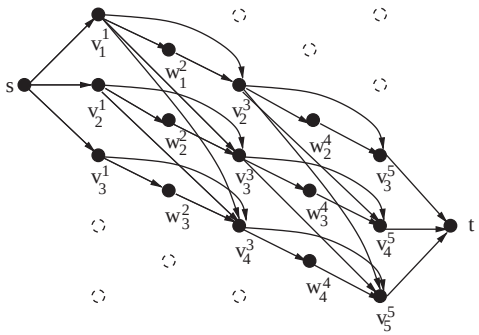


Fig. 2. The graph G for the case $N = 5$ and $k = 3$.

We first show that there is a one-to-one correspondence between directed *st*-paths in the graph G and feasible solutions in (9). Assume we have a directed *st*-path, then we create a feasible solution to (9) by the following procedure. If the path uses a vertex v_i^l we set x_i to one, and similarly if it uses w_j^l we set y_j to one. Since there are k odd numbered layers and there are no possible paths from a vertex v_i^l to any vertex v_j^k for $k > l$ and $j < i$, this will lead to k different x_i set to one. By construction any path that uses w_j^l will also have to use v_j^{l-1} and v_{j+1}^{l+1}, thus we have that y_j is one only if both x_j and x_{j+1} is one. In a similar way, we may construct an *st*-path from a feasible solution to (9).

We now return to the maximization problem in (10). We introduce flow variables f for all arcs in the graph. Let b be a column vector whose components correspond to the vertices in G, and where $b_s = -1$, $b_t = 1$ and $b_i = 0$ otherwise. Let A be the incidence matrix of the graph G. Then the *st*-paths in G correspond to the vertices of the network flow polyhedron

$$Af = b, \quad 0 \leqslant f \leqslant 1, \tag{11}$$

due to the fact that A is totally unimodular. The values of the variables in the original subproblem (9) are obtained from the flow variables following the procedure described in the previous section. We write this as

$$\begin{aligned} x &= Bf, \\ y &= Cf, \end{aligned} \tag{12}$$

where B and C are suitable $(0, 1)$-matrices. Let f_r be the vector of flow variables in (11) for the problem associated with row r, let $z_r = (x_1, \ldots, x_N, y_1, \ldots, y_N)$ and define the polytope

$$EP_r = \{(z_r, f_r) : z_r \text{ and } f_r \text{ are feasible solutions to (11) and (12)}\}.$$

Since the problems in each row are independent we may define the extended polytope for the complete horizontal problem as

$$EP^h = EP_1 \times EP_2 \times \cdots \times EP_m.$$

From the previous discussion it is clear that $z \in P^h$ if and only if $(z, f^h) \in EP^h$ for some vector $f^h = (f_1, \ldots, f_m)$ of extended variables and that the number of variables and inequalities is polynomially bounded by m and n. A similar construction may also be done for the vertical problem leading to a compact description of the polytope EP^v.

Combining the discussion above with the result in (10), we obtain the following theorem.

Theorem 4.1. *The value of the Lagrangian dual* v^{VSD} *in problem* (8) *is equal to the optimal value of the compact linear programming problem*

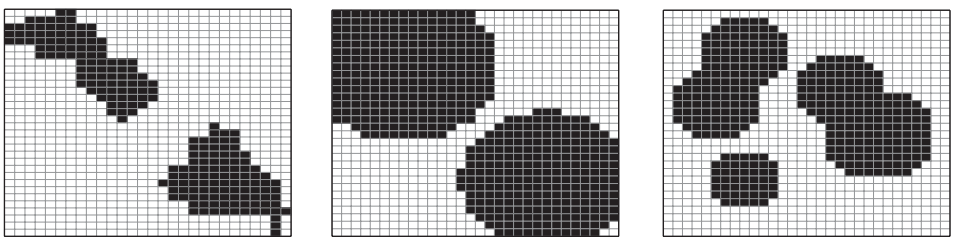
$$\max \left\{ \sum y_e : (z, f^h) \in EP^h, (z, f^v) \in EP^v \right\}. \tag{13}$$

As a consequence one may solve this compact LP in order to find the bound v^{VSD}. However, in practice, this LP becomes so large that the alternative approach of solving the Lagrangian dual via the subgradient method is preferable.

5. Computational results

Test data: Test data were generated for three different classes of problems. Since the algorithm is designed to find hv-convex or nearly hv-convex (0, 1)-matrices, we constructed binary test images ((0, 1)-matrices) having this property. The classes are illustrated in Fig. 3. hv-convex data were obtained by creating two hv-convex polyominoes, see [8], and then combining them. The *two circles* data consist of two disjoint circles with overlap in horizontal and vertical placements. One circle is centered in the upper left corner while the other is centered in the lower left corner. The radius of both circles were approximately a third of the image size. The result is a matrix that is not hv-convex, but with a maximum of two intervals of ones in each row and column. The images in the third class, *random circles*, were generated by placing a specified number of circles (10 for the three smallest cases and 20 for the two largest) with random radius and center in the image. The maximal value for the radius was a quarter of the image size. Based on the images constructed, test data were generated by calculating the number of black pixels (ones) in each row and column.

The subgradient procedure: The subgradient procedure is an iterative search procedure that may be used to solve (8). In each iteration the Lagrange multipliers are updated



(a) (b) (c)

Fig. 3. Classes of test data: (a) hv-convex; (b) two circles; (c) random circles.

according to

$$\lambda_{i,j}^{s+1} = \lambda_{i,j}^{s} + t^s d_{i,j}^{s},$$

where t^s is the steplength and $d_{i,j}^{s}$ is the search direction. For search direction we used the subgradient $d_{i,j} = x_{i,j}^{h} - x_{i,j}^{v}$ and the steplength was $t^s = \rho_0 \rho^k$, where k is increased after a specified number of iterations. We performed tests with more sophisticated schemes, but gaining no better performance.

To check the accuracy of the subgradient procedure we compared it to the results obtained by solving the variable splitting problem directly using ILOG CPLEX 7.5 [9] on the LP formulation described in Section 4. If we allowed up to 1000 subgradient iterations, the maximal deviation was 0.1%. The LP problems quickly became too large, so we tested only for problem sizes up to 20 times 20.

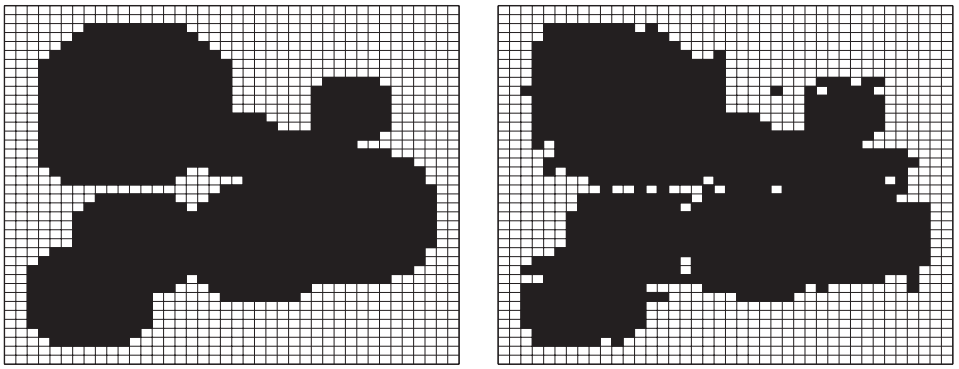
Primal heuristic: To obtain primal feasible solutions for calculating lower bounds on v^I, we use the following heuristic. Define a cost matrix $C = [c_{i,j}]$ by $c_{i,j} = x_{i,j}^{h} + x_{i,j}^{v}$, i.e. a cost function that prefers entries where both solutions have a value of one. Using this as input, we solve the following network flow problem:

$$\begin{aligned} \max \quad & \sum_{i,j} c_{i,j} x_{i,j} \\ \text{subject to} \quad & \sum_{j} x_{i,j} = r_i, \quad i \leqslant m \\ & \sum_{i} x_{i,j} = s_j, \quad j \leqslant n \\ & 0 \leqslant x_{i,j} \leqslant 1. \end{aligned}$$

This may be solved efficiently using e.g. the network simplex method, see [1]. For integer data, the vertices of the corresponding polytope are integral. Any method that finds an optimal vertex solution will give an integer solution and the matrix $X = [x_{ij}]$ will be a member of $\mathfrak{U}(R, S)$.

Example: Fig. 4 shows the result of solving a reconstruction problem using a subgradient procedure to solve the Lagrangian dual (8). Feasible solutions were obtained with the heuristic described above.

Results: Table 1 gives the results of five test runs of various size for each class of test data. The tests were performed on a Dell Latitude PC with a 1.2 GHz processor and 512 MBs of RAM. The reported value of v^{VSD} was obtained with a subgradient procedure with a maximum of 1000 iterations, the steplength was reduced with a factor of 0.7 every 50th iteration. The final value was rounded down to the nearest integer. v^{LB} is the value of the best feasible solution obtained with the primal heuristic run at intervals of 100 subgradient iterations. g is the gap between v^{VSD} and v^{LB}. The reported time T_{SG} involves only the subgradient optimization and excludes the time used for the lower bound calculations.



(a) (b)

Fig. 4. Reconstruction example, $m = n = 40$: (a) Original figure; (b) best feasible solution.

Table 1
Computational results

Type	m	n	$\hat{v}_1$	$\hat{v}_2$	v^{VSD}	v^{LB}	g (%)	T_{SG} (sec.)
hv-convex	14	16	98	102	98	98	0	0.77
	34	36	432	444	432	432	0	3.67
	56	54	1140	1163	1140	1140	0	15.1
	73	77	1748	1784	1748	1748	0	38.2
	100	90	3094	3124	3094	3094	0	103
Two circles	20	20	432	438	426	424	0.47	2.59
	40	40	2148	2148	2144	2144	0	11.6
	60	60	4196	4198	4176	4158	0.43	72.7
	80	80	8200	8192	8180	8180	0	190
	100	100	11812	11818	11781	11657	1.05	500
Random circles	20	20	268	267	265	265	0	1.08
	40	40	1682	1674	1674	1674	0	7.84
	60	60	3101	3082	3072	2914	5.14	59.7
	80	80	8499	8489	8483	8408	0.884	253
	100	100	9506	9426	9423	8347	11.4	456

6. Conclusions

In this paper we have studied a variant to the problem of reconstructing hv-convex $(0, 1)$-matrices from horizontal and vertical projections. Instead of requiring the ones to occur contiguously, we maximize the number of neighboring ones. This leads to an NP-hard combinatorial optimization problem. The problem is formulated as an integer problem and several upper bounds are derived. To solve the problem, we suggest an algorithm based

on a Lagrangian relaxation technique called variable splitting. By duplicating some of the variables, the problem decomposes into small problems that can be solved efficiently. The Lagrangian dual problem is solved by a subgradient procedure and a heuristic based on a network flow formulation is used to obtain feasible $(0, 1)$-matrices.

Even though the number of test cases are limited, the results seem to indicate that the algorithm is more successful with the cases having a large degree of hv-convexity in the original image. For the hv-convex data, optimal solutions were produced for all cases. There is a considerable increase in running time as the size of the problems increases. Thus, the algorithm in its current form will have problems handling large problems.

The algorithm can be generalized to handle problems with more than two projection directions and general neighborhood relations. A forthcoming paper [4] treats these extensions in detail.

References

[1] R.K. Ahuja, T.L. Magnati, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, Englewood Cliffs, NJ, 1993.
[2] E. Barcucci, A. Del Lungo, M. Nivat, R. Pinzani, Reconstructing convex polyominoes from horizontal and vertical projections, Theoret. Comput. Sci. 155 (1996) 321–347.
[3] S. Chang, C.K. Chow, The reconstruction of three-dimensional objects from two orthogonal projections and its application to cardiac cineangiography, IEEE Trans. Computers c-22 (1973) 18–28.
[4] G. Dahl, T. Flatberg, Reconstructing $(0, 1)$-matrices from projections using integer programming, submitted for publication.
[5] D. Gale, A theorem on flows in networks, Pacific J. Math. 7 (1957) 1073–1082.
[6] M. Guignard, S. Kim, Lagrangean decomposition: a model yielding stronger lagrangean bounds, Math. Prog. 39 (1987) 215–228.
[7] G.T. Herman, A. Kuba (Eds.), Discrete Tomography: Foundations, Algorithms and Applications, Birkhäuser, 1999.
[8] W. Hochstättler, M. Loebl, C. Moll, Generating convex polyominoes at random, Discrete Math. 153 (1996) 165–176.
[9] Ilog cplex, http://www.ilog.com/products/cplex.
[10] A.W. Marshall, I. Olkin, Inequalities: Theory of Majorization and its Applications, Academic Press, New York, 1979.
[11] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley, New York, 1988.
[12] H.J. Ryser, Combinatorial properties of matrices of zeros and ones, Canad. J. Math. 9 (1957) 371–377.
[13] P. Schwander, C. Kisielowski, M. Seibt, F.H. Baumann, Y. Kim, A. Ourmazd, Mapping projected potential, interfacial roughness, and composition in general crystalline solids by quantitative transmission electron microscopy, Phys. Rev. Lett. 71 (25) (1993) 4150–4153.
[14] B. Wang, F. Zhang, On the precise number of $(0, 1)$-matrices in $\mathfrak{A}(r, s)$, Discrete Math. 187 (1998) 211–220.
[15] G.J. Woeginger, The reconstruction of polyominoes from their orthogonal projections, Inform. Process. Lett. 77 (2001) 225–229.