J Math Model Algor (2013) 12:329–343
DOI 10.1007/s10852-012-9193-5

Reconstructing Convex Matrices by Integer Programming Approaches

Alain Billionnet · Fethi Jarray · Ghassen Tlig · Ezzedine Zagrouba

Received: 13 January 2012 / Accepted: 14 June 2012 / Published online: 12 July 2012

Abstract We consider the problem of reconstructing two-dimensional convex binary matrices from their row and column sums with adjacent ones. Instead of requiring the ones to occur consecutively in each row and column, we maximize the number of adjacent ones. We reformulate the problem by using integer programming and we develop approximate solutions based on linearization and convexification techniques.

Keywords Discrete Tomography (DT) · Integer programming · Convexification · Linearization

1 Introduction

Discrete tomography (DT) concerns the reconstruction (exactly or approximately) of discrete objects defined on discrete lattices from their projections. The projections are the sums along few angles of the pixel values. One of the main problems in DT is the reconstruction of binary matrices from two projections. DT was first studied

A. Billionnet · F. Jarray (✉) · G. Tlig
CEDRIC-CNAM, 292 rue St-Martin, 75003 Paris, France
e-mail: fethi_jarray@yahoo.fr

A. Billionnet
e-mail: alain.billionnet@ensiie.fr

G. Tlig
e-mail: ghassen.tlig@auditeur.cnam.fr

G. Tlig · E. Zagrouba
Faculté des Sciences de Tunis El Manar, 2092 El Manar II, Tunisie

E. Zagrouba
e-mail: ezzedine.zagrouba@fsm.rnu.tn

by Ryser [25] and Gale [13], and it has been investigated under various conditions [7, 10, 23, 27, 28].

DT is applicable in many interesting contexts such as nondestructive testing, image processing, electron microscopy, data security, industrial tomography and material sciences [17, 18, 22, 24, 26]. In general, the reconstruction of binary matrices from a small number of projections is undetermined and the number of solutions can be very large. Moreover, the projections data and the prior knowledge about the object to reconstruct are not sufficient to determine a unique solution. So DT is usually reduced to an optimization problem to select the best solution in a certain sense. For example, maximizing the number of adjacent ones in each row and column.

In this paper, we deal with the reconstruction of hv-convex matrices. In particular, research objectives are to derive three linearization methods, four convexification methods, and variable fixing methods and to compare them computationally.

A binary matrix is h-convex if the ones occur consecutively in a single block in each row. Similarly a binary matrix is v-convex if the ones occur contiguously in each column. A binary matrix is hv-convex if it is both h-convex and v-convex. Woeginger [28] shows that the existence problem for hv-convex matrices is NP-complete. In [8, 23], the authors present polynomially solvable classes of the hv-convex matrices. Kuba [20] proposes heuristics for the general class. Balázs [2] introduces the class of canonical hv-convex matrices (see Fig. 1) and proves that the reconstruction can be performed in polynomial time. Dahl and Flatberg [9] provide an algorithm to

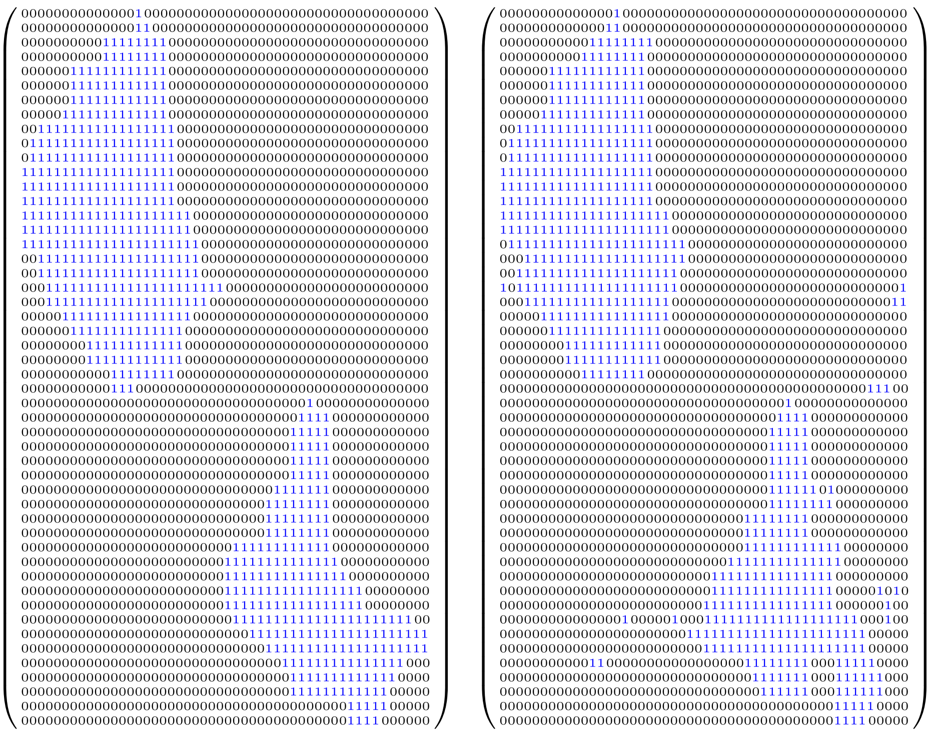

Fig. 1 Example of reconstructing a 50 × 50 matrix (solution obtained by $PLClassic_1$)

find hv-convex or nearly hv-convex matrices based on integer programming and variables splitting. They use a subgradient method to get an approximate solution. For an introduction to subgradient methods see [21]. Jarray et al. [19] provide an approximate solution based on a longest path and network flow algorithms.

The remainder of this paper is organized as follows. In Section 2, we introduce some definitions and notation. In Section 3, we propose linearization techniques. In Section 4, we propose convexification methods. In Section 5, we show how the different formulations can be improved by fixing some variables. In the last section, we present and discuss the numerical results.

2 Some Definitions and Notation

Let $H = (h_1, h_2, \ldots, h_m)$ and $V = (v_1, v_2, \ldots, v_n)$ be two positive integer vectors. We denote by $\mathcal{U}(\mathcal{H}, \mathcal{V})$ the class of $m \times n$ binary matrices A satisfying: $\sum_{j=1}^{n} a_{i,j} = h_i$, $i = 1, \ldots, m$ and $\sum_{i=1}^{m} a_{i,j} = v_j$, $j = 1, \ldots, n$. The vectors H and V is called respectively the horizontal and vertical projection of the matrix A. We suppose that the projections of rows and columns are consistent, i.e. $\mathcal{U}(\mathcal{H}, \mathcal{V})$ is not empty. This implies that $\sum_{j=1}^{n} v_j = \sum_{i=1}^{m} h_i = \tau$ is a necessary condition for non-emptiness of $\mathcal{U}(\mathcal{H}, \mathcal{V})$. Let $X = (x_{ij})_{i=1,\ldots,m;\, j=1,\ldots,n}$ be an $m \times n$ binary matrix. The number of adjacent ones is:

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

We model the reconstruction of an hv-convex matrix by the following integer program

$$P \quad \begin{cases} max\ f(X) & \\ s.t. & \\ \sum_{j=1}^{n} x_{ij} = h_i \quad i = 1, \ldots, m & (i) \\ \sum_{i=1}^{m} x_{ij} = v_j \quad j = 1, \ldots, n & (ii) \\ x_{ij} \in \{0, 1\} \quad i = 1, \ldots, m;\ j = 1, \ldots, n. & \end{cases}$$

The objective is to maximize the number of adjacent ones in the rows and columns over the class $\mathcal{U}(\mathcal{H}, \mathcal{V})$. Constraint (i) (resp. (ii)) ensures that the sum of ones in row i (resp. column j) is equal to the prescribed horizontal projection h_i (resp. vertical projection v_j). The program P is quadratic since the objective function contains some products of two variables. In general P is NP-hard. To handle P, we need to convert it into an easier program.

Proposition 1 *Let X be a binary matrix with orthogonal projections H and V. X is hv-convex if and only if:*

$$f(X) = 2\sum_{i=1}^{m} h_i - m - n = \sigma.$$

Proof Let X be a binary matrix respecting the projections H and V. X is hv-convex if and only if the ones in each row i and column j form a contiguous interval, i.e.

$$\sum_{j=1}^{n-1} x_{i,j}x_{i,j+1} = h_i - 1 \quad \text{and} \quad \sum_{i=1}^{m-1} x_{i,j}x_{i+1,j} = v_j - 1.$$

$$\text{So} \quad \sum_{i=1}^{m}\sum_{j=1}^{n-1} x_{i,j}x_{i,j+1} = \sum_{i=1}^{m} h_i - m \quad \text{and} \quad \sum_{j=1}^{n}\sum_{i=1}^{m-1} x_{i,j}x_{i+1,j} = \sum_{j=1}^{n} v_j - n.$$

$$\text{Thus} \quad f(X) = \sum_{i=1}^{m}\sum_{j=1}^{n-1} x_{i,j}x_{i,j+1} + \sum_{j=1}^{n}\sum_{i=1}^{m-1} x_{i,j}x_{i+1,j} = \sum_{j=1}^{n} v_j + \sum_{i=1}^{m} h_i - m - n$$

$$= 2\sum_{i=1}^{m} h_i - m - n \quad \text{since} \quad \sum_{j=1}^{n} v_j = \sum_{i=1}^{m} h_i.$$

□

3 Linearization

In this section we present an equivalent linear program to P by linearizing the objective function. The linearization method of quadratic programs replaces a quadratic term by a new variable.

3.1 Classical Linearization

The classical linearization [12] consists in replacing each product by a new variable and adding a set of linear constraints that force the equality between the new variable and the product. In program P, we replace the product $x_{i,j}x_{i,j+1}$ by $yh_{i,j}$ and $x_{i,j}x_{i+1,j}$ by $yv_{i,j}$. Since P is a maximization program, we add the following algebraic system of inequalities to ensure $yh_{i,j} = x_{i,j}x_{i,j+1}$

$$(I) \quad \begin{cases} yh_{i,j} \leq x_{i,j} \quad i = 1, \ldots, m;\ j = 1, \ldots, n-1 \quad (I.a) \\ yh_{i,j} \leq x_{i,j+1} \quad i = 1, \ldots, m;\ j = 1, \ldots, n-1\ (I.b). \end{cases}$$

We add also the following algebraic system of inequalities to ensure $yv_{i,j} = x_{i,j}x_{i+1,j}$

$$(II) \quad \begin{cases} yv_{i,j} \leq x_{i,j} \quad i = 1, \ldots, m-1;\ j = 1, \ldots, n \quad (II.a) \\ yv_{i,j} \leq x_{i+1,j} \quad i = 1, \ldots, m-1;\ j = 1, \ldots, n\ (II.b). \end{cases}$$

Finally, we obtain the following equivalent binary linear program to P.

$$PLClassic_1 \quad \begin{cases} max \sum_{i=1}^{m}\sum_{j=1}^{n-1} yh_{i,j} + \sum_{i=1}^{m-1}\sum_{j=1}^{n} yv_{i,j} & \\ s.t. & \\ \sum_{j=1}^{n} x_{i,j} = h_i \quad i = 1,\ldots,m & (i) \\ \sum_{i=1}^{m} x_{i,j} = v_j \quad j = 1,\ldots,n & (ii) \\ yh_{i,j} \le x_{i,j} \quad i = 1,\ldots,m;\ j = 1,\ldots,n-1 & (I.a) \\ yh_{i,j} \le x_{i,j+1} \quad i = 1,\ldots,m;\ j = 1,\ldots,n-1 & (I.b) \\ yv_{i,j} \le x_{i,j} \quad i = 1,\ldots,m-1;\ j = 1,\ldots,n & (II.a) \\ yv_{i,j} \le x_{i+1,j} \quad i = 1,\ldots,m-1;\ j = 1,\ldots,n & (II.b) \\ yh_{ij} \ge 0 \quad i = 1,\ldots,m;\ j = 1,\ldots,n-1 & \\ yv_{ij} \ge 0 \quad i = 1,\ldots,m-1;\ j = 1,\ldots,n & \\ x_{ij}, yh_{ij}, yv_{ij} \in \{0,1\} \quad i = 1,\ldots,m;\ j = 1,\ldots,n. & \end{cases}$$

Figure 1 shows an example of a 50×50 reconstructed matrix by $PLClassic_1$.

Another possible version of classical linearization aims to replace the algebraic system of inequalities (I) by $2yh_{i,j} \le x_{i,j} + x_{i,j+1}$ and (II) by $2yv_{i,j} \le x_{i,j} + x_{i+1,j}$ such that $yh_{ij}, yv_{ij} \in \{0,1\}$. We get the following binary program

$$PLClassic_2 \quad \begin{cases} max \sum_{i=1}^{m}\sum_{j=1}^{n-1} yh_{i,j} + \sum_{i=1}^{m-1}\sum_{j=1}^{n} yv_{i,j} & \\ s.t. & \\ \sum_{j=1}^{n} x_{i,j} = h_i \quad i = 1,\ldots,m & (i) \\ \sum_{i=1}^{m} x_{i,j} = v_j \quad j = 1,\ldots,n & (ii) \\ 2yh_{ij} \le x_{i,j} + x_{i,j+1} \quad i = 1,\ldots,m;\ j = 1,\ldots,n-1 & \\ 2yv_{ij} \le x_{i,j} + x_{i+1,j} \quad i = 1,\ldots,m-1;\ j = 1,\ldots,n & \\ x_{ij}, yh_{ij}, yv_{ij} \in \{0,1\} \quad i = 1,\ldots,m;\ j = 1,\ldots,n. & \end{cases}$$

3.2 Compact Linearization

The compact linearization [14] uses a different idea. It replaces in P the product of a binary variable and a linear expression by an integer variable. The objective function of P can be rewritten as

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

We replace each product $x_{i,j}(x_{i,j+1} + x_{i+1,j})$ in $f(X)$ by an integer variable $z_{i,j}$ and we introduce the following set of linear constraints

$$\begin{cases} z_{i,j} \le 2x_{i,j} & i = 1,\ldots,m-1;\ j = 1,\ldots,n-1 \\ z_{i,j} \le x_{i,j+1} + x_{i+1,j} & i = 1,\ldots,m-1;\ j = 1,\ldots,n-1. \end{cases}$$

For each product $x_{m,j}x_{m,j+1}$ (resp. $x_{i,n}x_{i+1,n}$) we introduce yh_j (resp. yv_i) such that $yh_j \leq x_{m,j}$ and $yh_j \leq x_{m,j+1}$ (resp. $yv_i \leq x_{i,n}$ and $yv_i \leq x_{i+1,n}$). Finally, we obtain the following linear compact integer program

$$PLCompact \quad \begin{cases} max \sum_{i=1}^{m} \sum_{j=1}^{n} z_{i,j} + \sum_{j=1}^{n-1} yh_j + \sum_{i=1}^{m-1} yv_i \\ s.t. \\ \sum_{j=1}^{n} x_{i,j} = h_i \quad i = 1, \dots, m & (i) \\ \sum_{i=1}^{m} x_{i,j} = v_j \quad j = 1, \dots, n & (ii) \\ z_{i,j} \leq 2x_{i,j} \quad i = 1, \dots, m-1;\ j = 1, \dots, n-1 \\ z_{i,j} \leq x_{i,j+1} + x_{i+1,j} \quad i = 1, \dots, m-1;\ j = 1, \dots, n-1 \\ yh_j \leq x_{m,j} \quad j = 1, \dots, n \\ yh_j \leq x_{m,j+1} \quad j = 1, \dots, n-1 \\ yv_i \leq x_{i,n} \quad i = 1, \dots, m \\ yv_i \leq x_{i+1,n} \quad i = 1, \dots, m-1 \\ x_{ij}, yh_{ij}, yv_{ij} \in \{0, 1\} \quad i = 1, \dots, m;\ j = 1, \dots, n. \\ z_{i,j} \geq 0 \quad i = 1, \dots, m-1;\ j = 1, \dots, n-1 \end{cases}$$

4 Convexification

A function of discrete variables is said to be convex if its extension to the convex hull is a convex function of continuous variables. A function f is called concave if (-f) is convex. Maximize a function f is equivalent to minimize (-f). Convex programming concerns the minimization (resp. maximization) of a convex (resp. concave) objective function under a convex constraint set. For example, we say that the quadratic function $f(R) = \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} c_{i,j} r_i r_j$, where the variables r_i are binary, is convex if and only if $\sum_{i=1}^{p-1} \sum_{j=i+1}^{p} c_{i,j} r_i r_j \geq 0$ for all $R \in \mathbb{R}^p$.

The convexification technique consists of finding an equivalent quadratic convex reformulation of P. In matrix notation, the objective function of P can be rewritten as $f(X) = \sum_{i=1}^{m-1} \sum_{j=1}^{n} x_{i,j}x_{i+1,j} + \sum_{i=1}^{m} \sum_{j=1}^{n-1} x_{i,j}x_{i,j+1} = X^T QX$, where X is an mn binary vector containing the variables of an $m \times n$ matrix and Q is an $mn \times mn$ matrix where the rows and columns are indexed by the elements of vector X. We will apply two convexification methods.

4.1 Classical Convexification

Hammer and Rubin [16] propose a simple convexification method to convert non convex 0-1 quadratic functions into convex functions by adding the term $-\lambda \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij}^2 - x_{ij})$, where λ is the minimum eigenvalue of Q. This addition modifies each diagonal element of Q to ensure that the matrix $Q - \lambda I$ is a positive semidefinite matrix. We call this program $PCClassic_1$.

The current solvers solve integer mathematical programs with quadratic convex objective function and linear constraints. Hence it is interesting to reformulate non convex problems into convex problem. Moreover, in certain cases, it can be even interesting to reformulate a convex quadratic program. In our case, the objective function $f(X)$ of program P is not concave. To transform it into a concave function, we add the term $-\alpha \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2$, where α is not less than the greatest eigenvalue of the quadratic form of P. We note that $\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = \sum_{i=1}^{m} h_i = \tau$, i.e. the sum of the ones in matrix X. We get the following concave function: $\hat{f}(X) = f(X) - \alpha \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2 = f(X) - \alpha\tau$ for $X \in \mathcal{U}(\mathcal{H}, \mathcal{V})$. We denote by $PCClassic_2$ the resulting convex program of P. To find the suitable value of α, we start by giving the definition of the spectral radius of a matrix:

Definition 1 The spectral radius of a square matrix A, noted $\rho(A)$, is defined by $\rho(A) = max_i(\lambda_i)$ where λ_i is an eigenvalue of A.

Lemma 1 *The spectral radius of a square matrix A is less than or equal to any norm of A.*

Proposition 2 $\rho(Q) \leq 4 \quad \forall m, n.$

Proof The infinity norm of an $n \times n$ matrix A is $\|A\|_\infty = \max_i \sum_{j=1}^{n} a_{ij}$. Thus $\|Q\|_\infty = 4$ and $\rho(Q) \leq 4$. □

We remind that the objective function of P is: $f(X) = \sum_{i=1}^{m-1} \sum_{j=1}^{n} x_{i,j}x_{i+1,j} + \sum_{i=1}^{m} \sum_{j=1}^{n-1} x_{i,j}x_{i,j+1} = X^T QX$. X is considered as a mn vector containing all the variables, i.e. $X = \begin{pmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1n} \\ x_{21} \\ \vdots \\ x_{mn} \end{pmatrix}$.

Q is the $mn \times mn$ matrix of the quadratic form where the rows and the columns are indexed by the elements of the vector X. We notice that each row of the matrix Q contains at most four ones because each variable x_{ij} is included in at most four products: $x_{ij}x_{i+1,j}$, $x_{ij}x_{i-1,j}$, $x_{ij}x_{i,j+1}$, $x_{ij}x_{i,j-1}$.

For an image of size 3×3, we get $X = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{pmatrix}$ and $Q = \begin{pmatrix} 0&1&0&1&0&0&0&0&0 \\ 1&0&1&0&1&0&0&0&0 \\ 0&1&0&0&0&1&0&0&0 \\ 1&0&0&0&1&0&1&0&0 \\ 0&1&0&1&0&1&0&1&0 \\ 0&0&1&0&1&0&0&0&1 \\ 0&0&0&1&0&0&0&1&0 \\ 0&0&0&0&1&0&1&0&1 \\ 0&0&0&0&0&1&0&1&0 \end{pmatrix}$.

In our simulation, we choose $\alpha = 4$ in $PCClassic_2$.

A third possible version of convexification consists of replacing the product xy by $-0.5((x-y)^2-x-y)$ which is equivalent for binary variables. We apply the method of convexification, obtaining the following program $PCClassic_3$

$$\begin{cases} max -0.5\left(\sum_{i=1}^{m-1}\sum_{j=1}^{n}\left(x_{i,j}-x_{i+1,j}\right)^2-x_{i,j}-x_{i+1,j}\right)-0.5\left(\sum_{i=1}^{m}\sum_{j=1}^{n-1}\left(x_{i,j}-x_{i,j+1}\right)^2-x_{i,j}-x_{i,j+1}\right) \\ s.t. \\ \sum_{j=1}^{n} x_{i,j}=h_i \quad i=1,\ldots,m \qquad (i) \\ \sum_{i=1}^{m} x_{i,j}=v_j \quad j=1,\ldots,n \qquad (ii) \\ x_{ij}\in\{0,1\} \quad i=1,\ldots,m;\ j=1,\ldots,n. \end{cases}$$

4.2 Quadratic Convex Reformulation (QCR) Method

Billionnet et al. [6] developed the QCR method to convert 0-1 quadratic programs into convex programs. They perturb all the terms of Q by adding new functions. The QCR method was extended to the case of Mixed Integer Quadratic Programs (MIQPs) [5]. We apply the QCR method by adding the following functions to $f(X)$

$$f_\alpha(X)=\alpha\sum_{i=1}^{m}\left(\sum_{j=1}^{n}x_{i,j}-h_i\right)^2,\ g_\alpha(X)=\alpha\sum_{j=1}^{n}\left(\sum_{i=1}^{m}x_{i,j}-v_j\right)^2,$$

$$f_\lambda(X)=\sum_{i=1}^{m}\sum_{j=1}^{n}\lambda_{i,j}\left(x_{i,j}^2-x_{i,j}\right).$$

The new objective function is $f_{c(\alpha,\lambda)}=f(X)+f_\alpha(X)+g_\alpha(X)+f_\lambda(X)$. It is easy to verify that $f_\alpha(X)=g_\alpha(X)=f_\lambda(X)=0$ for each feasible solution X. So we consider the following 0-1 parametric quadratic program depending on α and λ:

$$PCQCR \quad \begin{cases} max\ f_{c(\alpha,\lambda)}(X)=f(X)+f_\alpha(X)+g_\alpha(X)+f_\lambda(X) \\ s.t. \\ \sum_{j=1}^{n} x_{i,j}=h_i \quad i=1,\ldots,m \qquad (i) \\ \sum_{i=1}^{m} x_{i,j}=v_j \quad j=1,\ldots,n \qquad (ii) \\ x_{ij}\in\{0,1\} \quad i=1,\ldots,m;\ j=1,\ldots,n. \end{cases}$$

In [6], the authors proved that the optimal values α^* of α are equal to the values of the dual variables associated with constraints of the form (a) and the optimal values λ^* of λ are equal to the values of the dual variables associated with constraints (c) in the following semidefinite program SDP:

$$SDP\begin{cases} max\ \sum_{i=1}^{m}\sum_{j=1}^{n-1} y_{j+(i-1)m,j+1+(i-1)m}+\sum_{j=1}^{n}\sum_{i=1}^{m-1} y_{j+(i-1)m,im+j} \\ s.t. \\ \sum_{j=1}^{n} x_{i,j}=h_i \quad i=1,\ldots,m \qquad (i) \\ \sum_{i=1}^{m} x_{i,j}=v_j \quad j=1,\ldots,n \qquad (ii) \\ -h_ix_{p,q}+\sum_{t=1}^{n} y_{t+(i-1)n,q+(p-1)m}=0 \quad i=1,\ldots,m;\ p=1,\ldots,m;\ q=1,\ldots,n \ (a) \\ -v_jx_{p,q}+\sum_{t=1}^{m} y_{n(t-1)+1,q+(p-1)m}=0 \quad j=1,\ldots,n;\ p=1,\ldots,m;\ q=1,\ldots,n \ (b) \\ y_{m(i-1)+j,m(i-1)+j}=x_{i,j} \quad i=1,\ldots,m;\ j=1,\ldots,n \qquad (c) \\ x\in\mathbb{R}^{mn} \\ Y\in\mathcal{S}_{mn}. \end{cases}$$

where Y is a semidefinite real matrix, and $\mathcal{S}_{mn}$ represents the set of real symmetric matrices of order mn.

5 Algebraic System

We propose a new formulation (S) of the reconstruction of hv-convex matrices. A horizontal bar is a set of horizontally adjacent ones. A vertical bar is a set of vertically adjacent ones. Thus a matrix is hv-convex if and only if it contains a single horizontal bar per row and a single vertical bar per column.

We introduce two extra binary variables r_{ij} and c_{ij} such that $r_{ij} = 1$ if the horizontal bar of row i starts in cell (i, j) and $c_{ij} = 1$ if the vertical bar of column j starts in cell (i, j).

To ensure that $r_{i,j} = 1$ if and only if a horizontal bar of ones starts in cell (i, j), we add the constraints (*iii.a*) and (*iii.b*):

$$\begin{cases} \sum_{j=1}^{n-h_i+1} r_{i,j} = 1 \quad i = 1, \ldots, m & (iii.a) \\ \sum_{k=j}^{j+h_i-1} x_{i,k} \geq h_i r_{i,j} \quad i = 1, \ldots, m;\ j = 1, \ldots, n - h_i + 1 & (iii.b) \\ \sum_{k=1}^{j-1} x_{i,k} \leq h_i(1 - r_{i,j}) \quad i = 1, \ldots, m;\ j = 2, \ldots, n - h_i & (iii.c). \end{cases}$$

Similarly to guarantee that $c_{i,j} = 1$ if and only if a vertical bar of ones start in cell (i, j), we add the constraints (*iv.a*) and (*iv.b*):

$$\begin{cases} \sum_{i=1}^{m-v_j+1} c_{i,j} = 1 \quad j = 1, \ldots, n & (iv.a) \\ \sum_{k=i}^{i+v_j-1} x_{k,j} \geq v_j c_{i,j} \quad j = 1, \ldots, n;\ i = 1, \ldots, m - v_j + 1 & (iv.b) \\ \sum_{k=1}^{i-1} x_{i,k} \leq v_j(1 - c_{i,j}) \quad j = 1, \ldots, n;\ i = 2, \ldots, m - v_j & (iv.c). \end{cases}$$

We remark that constraints (*iii.c*) and (*iv.c*) are redundant. In fact constraint (*iii.c*) can be obtained by combining (*i*) and (*iii.b*).

Consider constraint (*i*):

$$\sum_{j=1}^{n} x_{i,j} = h_i \Rightarrow \sum_{k=1}^{j-1} x_{i,k} + \sum_{k=j}^{j+h_i-1} x_{i,k} + \sum_{k=j+h_i}^{n} x_{i,k} = h_i$$
$$\Rightarrow \sum_{k=1}^{j-1} x_{i,k} + \sum_{k=j}^{j+h_i-1} x_{i,k} \leq h_i.$$

Apply constraint (*iii.b*):

$$\sum_{k=j}^{j+h_i-1} x_{i,k} \geq h_i r_{i,j} \Rightarrow \sum_{k=1}^{j-1} x_{i,k} \leq h_i - \sum_{k=j}^{j+h_i-1} x_{i,k} \leq h_i - h_i r_{i,j}$$
$$\sum_{k=1}^{j-1} x_{i,k} \leq h_i(1 - r_{i,j}) \Rightarrow (iii.c).$$

Similarly, we prove that constraint (*iv.c*) is also redundant.

Finally the system S is

$$S \begin{cases} \sum_{j=1}^{n} x_{i,j} = h_i \quad i = 1,\dots,m & (i) \\ \sum_{i=1}^{m} x_{i,j} = v_j \quad j = 1,\dots,n & (ii) \\ \sum_{j=1}^{n-h_i+1} r_{i,j} = 1 \quad i = 1,\dots,m & (iii.a) \\ \sum_{k=j}^{j+h_i-1} x_{i,k} \geq h_i r_{i,j} \quad i = 1,\dots,m;\ j = 1,\dots,n-h_i+1 & (iii.b) \\ \sum_{k=1}^{j-1} x_{i,k} \leq h_i(1-r_{i,j}) \quad i = 1,\dots,m;\ j = 2,\dots,n-h_i & (iii.c) \\ \sum_{i=1}^{m-v_j+1} c_{i,j} = 1 \quad j = 1,\dots,n & (iv.a) \\ \sum_{k=i}^{i+v_j-1} x_{k,j} \geq v_j c_{i,j} \quad j = 1,\dots,n; i = 1,\dots,m-v_j+1 & (iv.b) \\ \sum_{k=1}^{i-1} x_{i,k} \leq v_j(1-c_{i,j}) \quad j = 1,\dots,n; i = 2,\dots,m-v_j & (iv.c) \\ x_{ij}, c_{i,j}, r_{i,j} \in \{0,1\} \quad i = 1,\dots,m;\ j = 1,\dots,n. \end{cases}$$

6 Variable Fixing

Many computational tests proved that the running time of a mathematical program strongly depends on the number of fixed variables. We try to fix certain variables in any solution of P. This preprocessing step reduces the size of the feasible region (sometimes all the variables can be fixed) before solving it by an exact method.

Definition 2 A cell (i, j) is 1-invariant if $x_{ij} = 1$ for any $x \in \mathcal{U}(\mathcal{H}, \mathcal{V})$. A cell (i, j) is 0-invariant if $x_{ij} = 0$ for any $x \in \mathcal{U}(\mathcal{H}, \mathcal{V})$. A cell is invariant if it is 1-invariant or 0-invariant.

Haber [15] gave necessary and sufficient conditions for a cell to be invariant.

Theorem 1 (Haber) *Suppose that H and V are monotone. If cell (i, j) is 1-invariant in $\mathcal{U}(H, V)$, then there exist two integers e and f with $i \leq e \leq m$ and $j \leq f \leq n$ such that every matrix $A \in \mathcal{U}(H, V)$ has the following form: $A = \begin{bmatrix} 1 & A1 \\ A2 & 0 \end{bmatrix}$, where 1 is an $e \times f$ matrix in which all its entries are one, 0 is an $(m-e) \times (m-f)$ matrix in which all its entries are zero, A1 is an $e \times (m-f)$ binary matrix, A2 is an $(m-e) \times f$ binary matrix.*

This theorem supposes that we have an initial matrix $A \in \mathcal{U}(H, V)$. This matrix can be found, for example, by using the algorithm of Ryser [25]:

```
1. $\alpha_i = h_i$, $i = 1,\dots,m$;
2. for $j = 1$ to $n$ do
  2.1 if there is no $v_j$ available row then stop;
  2.2 set $v_j$ '1' in each of the hightest priority $v_j$ available rows;
  2.3 if a '1' is placed in row $i$, then $\alpha_i \leftarrow \alpha_i - 1$;
   end do;
```

It is a greedy algorithm with complexity $O(mn + max(mlogm; nlogn))$. At each stage (column) j, v_j 1s are placed in column j in each of the hightest priority available rows. If the number of available rows is less than v_j, then the algorithm should stop

Table 1 CPU running time (s) for linearization methods

#matrix	σ	$PLClassic_1$			$PLClassic_2$			$PLCompact$			Direct use Cplex		
		Sol	$\widehat{Sol}$	Time	Sol	$\widehat{Sol}$	Time	Sol	$\widehat{Sol}$	Time	Sol	$\widehat{Sol}$	Time
10 × 10,1	80	80	80.88	**0.74**	80	80.52	**1.60**	80	80.89	**0.78**	80	80.90	**89.33**
10 × 10,2	36	36	38.15	**1.52**	36	37.45	**1.59**	36	37.44	**1.40**	36	37.10	**335.27**
10 × 10,3	18	18	20.09	**1.34**	18	18.67	**0.48**	18	19.18	**1.35**	18	19.02	**195.48**
10 × 10,4	20	20	22.07	**4.99**	20	21.06	**1.99**	20	21.20	**6.80**	20	21.22	**340.84**
20 × 20,1	390	390	390.00	**0.21**	390	390.00	**0.26**	390	390.00	**0.32**	390	390.00	**15.03**
20 × 20,2	154	154	155.07	**208.64**	154	155.15	**260.47**	154	155.00	**1783.56**	154	155.00	**2433.58**
20 × 20,3	90	90	91.00	**1925.23**	90	91.06	**967.48**	90	94.93	**3600.18**	90	92.00	**3433.66**
20 × 20,4	46	46	46.97	**213.40**	46	46.65	**203.23**	46	46.84	**3024.80**	46	46.80	**3433.59**
30 × 30,1	954	954	955.00	**34.08**	954	954.54	**78.38**	954	955.05	**106.08**	954	955.08	**333.08**
30 × 30,2	410	410	410.38	**165.15**	410	410.77	**986.73**	410	411.02	**3368.72**	410	410.38	**3429.19**
30 × 30,3	240	239	245.42	**3600.10**	234	245.30	**3600.18**	234	249.13	**3600.10**	232	250.13	**3600.20**
30 × 30,4	194	193	198.11	**3600.10**	194	197.09	**3600.13**	188	200.59	**3600.11**	188	200.58	**3600.38**
40 × 40,1	1390	1366	1398.14	**3600.24**	1353	1402.21	**3600.10**	1338	1403.15	**3600.10**	1360	1400.29	**3600.23**
40 × 40,2	762	757	766.64	**3600.11**	748	767.59	**3600.08**	732	770.02	**3600.17**	755	767.35	**3600.60**
40 × 40,3	512	492	520.37	**3600.11**	488	519.49	**3600.02**	480	522.22	**3600.16**	488	519.17	**3600.62**
40 × 40,4	274	253	288.00	**3600.53**	265	283.00	**3600.19**	258	290.82	**3600.13**	249	292.72	**3600.57**
50 × 50,1	2230	2230	2231.09	**424.11**	2228	2232.40	**3600.17**	2228	2231.54	**3600.13**	2224	2234.42	**3600.94**
50 × 50,2	1166	1133	1172.61	**3600.24**	1124	1174.17	**3600.16**	1103	1176.11	**3600.22**	1122	1174.55	**3600.01**
50 × 50,3	884	882	886.46	**3600.16**	838	888.76	**3600.22**	796	891.59	**3600.11**	853	888.32	**3600.38**
50 × 50,4	594	564	605.33	**3600.32**	570	606.26	**3600.11**	531	611.04	**3601.36**	557	600.02	**3600.52**

The bold items represent the running time

Table 2 CPU running time (s) for convexification methods

#matrix	σ	$PCClassic_1$			$PCClassic_2$			$PCClassic_3$			$PCQCR$		
		Sol	$\widehat{Sol}$	Time	Sol	$\widehat{Sol}$	Time	Sol	$\widehat{Sol}$	Time	Sol	$\widehat{Sol}$	Time
10 × 10,1	80	80	80.01	**99.38**	80	80.01	**103.42**	80	80.01	**61.24**	80	80.12	**43.22**
10 × 10,2	36	36	38.17	**3603.06**	36	38.31	**3603.29**	36	37.31	**3600.05**	36	36.99	**967.05**
10 × 10,3	18	18	18.00	**2661.71**	18	18.01	**2518.47**	18	18.00	**618.86**	18	18.01	**430.83**
10 × 10,4	20	20	24.28	**3603.30**	20	23.94	**3602.75**	20	22.00	**3600.07**	20	23.64	**3600.07**
20 × 20,1	390	390	391.00	**3.04**	390	391.02	**2.94**	390	391.00	**12.24**	390	391.00	**3600.04**
20 × 20,2	154	151	182.46	**3601.36**	148	182.39	**3601.94**	154	180.02	**3600.86**	154	180.02	**3600.00**
20 × 20,3	90	84	120.12	**3600.36**	84	120.18	**3601.38**	90	100.00	**3600.61**	90	100.01	**3600.10**
20 × 20,4	46	42	75.33	**3601.72**	42	75.37	**3601.27**	46	70.72	**3600.14**	45	75.33	**3600.21**
30 × 30,1	954	954	978.52	**3601.07**	954	978.45	**3601.00**	954	978.83	**3600.68**	954	978.40	**3600.45**
30 × 30,2	410	394	459.13	**3600.67**	394	459.08	**3600.77**	402	450.41	**3600.63**	400	450.21	**3600.17**
30 × 30,3	240	216	290.12	**3600.70**	217	290.08	**3600.69**	222	280.09	**3600.63**	222	280.18	**3600.28**
30 × 30,4	194	179	243.64	**3600.71**	172	243.57	**3600.64**	180	240.30	**3600.26**	188	200.99	**3600.41**
40 × 40,1	1390	1370	1458.08	**3601.62**	1359	1458.08	**3600.36**	1375	1450.05	**3600.01**	1385	1410.03	**3600.40**
40 × 40,2	762	747	832.29	**3600.34**	738	832.32	**3600.33**	750	800.22	**3600.07**	750	800.32	**3600.01**
40 × 40,3	512	473	582.08	**3600.39**	469	582.08	**3600.41**	480	555.07	**3600.94**	482	553.22	**3600.00**
40 × 40,4	274	238	346.04	**3600.58**	232	346.03	**3600.36**	240	346.42	**3600.21**	244	346.22	**3600.37**
50 × 50,1	2230	2208	2295.74	**3600.44**	2213	2295.80	**3600.42**	2222	2282.24	**3600.12**	2222	2282.48	**3600.05**
50 × 50,2	1166	1119	1255.46	**3600.19**	1118	1255.44	**3600.17**	1126	1222.07	**3600.84**	1120	1222.77	**3600.17**
50 × 50,3	884	840	973.81	**3600.16**	854	973.78	**3600.17**	855	956.00	**3600.06**	850	956.41	**3600.02**
50 × 50,4	594	533	685.33	**3600.27**	543	685.33	**3600.33**	547	660.45	**3600.92**	540	665.30	**3600.85**

The bold items represent the running time

and there is no solution. Row i is available at stage j if $\alpha_i > 0$. α_i is the number of 1s not yet placed at the stage j of the algorithm. The hightest priority rows are those having the greatest values α_i.

7 Computational Results

Our mathematical programs are solved using the AMPL modeling language and Ilog Cplex 11.0 solver. All our experiments were run on an AMD Athlon XP-M 1.7 GHz PC with 512 MB of memory.

We have used a set of square matrices of various sizes described and generated in [3, 4]. In [1], the authors presented a systematic approach to generate hv-convex binary matrices from uniform distributions each with a fixed number of components. A component is a maximal hv-convex connected set. The proposed generating algorithm is partially based on the decomposition studied in [11].

Results of the computational experiments evaluating the linearization methods and directly using Cplex are summarized in Table 1. Recent versions of CPLEX can treat non convex 0-1 quadratic problems. Table 2 displays computational experiments concerning the convexification methods. Table 3 displays computational experiments concerning $PLClassic_1$, $PLClassic_1$ with fixed variables and linear system S. In these tables, the first column contains the size of the matrix and the number of hv-convex components of the instance. For example $(10 \times 10, 3)$

Table 3 CPU running time in seconds for linear system (S) and for linearization $PLClassic_1$ with fixed variables; (–) means that the solution has not been found within one hour

#martix	σ	$PLClassic_1$			$PLClassic_1$ with fixed variables				Linear system S	
		Sol	$\widehat{Sol}$	Time	Sol	$\widehat{Sol}$	Time	% Fixed	Sol	Time
10×10,1	80	80	80.88	**0.74**	80	80.00	**0.74**	0	80	**0.04**
10×10,2	36	36	38.15	**1.52**	36	37.00	**1.02**	2	36	**0.09**
10×10,3	18	18	20.09	**1.34**	18	18.00	**1.03**	5	18	**0.07**
10×10,4	20	20	22.07	**4.99**	20	20.02	**3.09**	7	20	**0.28**
20×20,1	390	390	390.00	**0.21**	390	390.00	**0.21**	0	390	**0.06**
20×20,2	154	154	155.07	**208.64**	154	155.07	**208.64**	0	154	**36.92**
20×20,3	90	90	91.00	**1925.23**	90	91.00	**1925.23**	0	90	**64.74**
20×20,4	46	46	46.97	**213.40**	46	46.00	**150.40**	10	46	**24.72**
30×30,1	954	954	955.00	**34.08**	954	955.00	**3600.19**	0	954	**0.79**
30×30,2	410	410	410.38	**165.15**	410	410.00	**3600.17**	7	410	**9.32**
30×30,3	240	239	245.42	**3600.10**	240	240.13	**3600.22**	3	240	**33.29**
30×30,4	194	193	198.11	**3600.10**	194	195.51	**3600.00**	5	194	**584.13**
40×40,1	1390	1366	1398.14	**3600.24**	1390	1390.00	**3600.50**	0	1390	**3015.45**
40×40,2	762	757	766.64	**3600.11**	762	762.62	**3600.01**	9	762	**3600.07**
40×40,3	512	492	520.37	**3600.11**	510	514.33	**3600.30**	10	–	**3600.00**
40×40,4	274	253	288.00	**3600.53**	270	280.00	**3600.12**	15	–	**3600.00**
50×50,1	2230	2230	2231.09	**424.11**	2230	2230.03	**202.10**	20	2230	**133.42**
50×50,2	1166	1133	1172.61	**3600.24**	1155	1169.54	**3600.22**	13	–	**3600.00**
50×50,3	884	882	886.46	**3600.16**	884	885.76	**3600.31**	4	–	**3600.00**
50×50,4	594	564	605.33	**3600.32**	589	598.81	**3600.02**	31	–	**3600.00**

The bold items represent the running time

denotes a square 10×10 matrix with 3 components. The second column displays the maximal number of adjacent ones (see Proposition 1). The subcolumn labeled Sol (respectively $\widehat{Sol}$) contains the integer (resp. continuous) solution provided by each method. The subcolumn labeled "time" contains the CPU running time (in seconds) required by Cplex within a time limit fixed to 1 hr. The subcolumn labeled "% Fixed" displays the percentage of variables fixed in the preprocessing step.

We observe that the results given by the linearization methods are very similar. We note also that all the methods give the optimal solution for the set of images of size 10×10. From Tables 1 and 2, we conclude that the linearization methods are better than the convexification methods. From Table 1, we conclude that the method $PCClassic_1$ outperforms the other linearization methods. From Table 2, we note that $PCQCR$ cannot provide a solution within an hour. For instances of small size, we note that directly solving the system S is very fast. Finally, from Table 3, we remark that in all the cases the method $PCClassic_1$ is enhanced by fixing variables.

8 Conclusion

In this paper, we have considered the problem of reconstructing hv-convex binary matrices. We have formulated the problem as a quadratic integer program. We have proposed several techniques based on linearization and convexification to solve the nonlinear related programs. Linearization combined with fixing variables seems to be the best. Promising future research directions include generation of valid cutting planes to improve the quality of solutions obtained.

Acknowledgements The authors express their deep gratitude to the anonymous referees, whose comments have improved both the presentation and the readability of the paper.

References

1. Balázs, P.: A framework for generating some discrete sets with disjoint components by using uniform distributions. Discrete Appl. Math. **406**, 15–23 (2008)
2. Balázs, P.: Reconstruction of canonical hv-convex discrete sets from horizontal and vertical projections. In: 13th International Workshop on Combinatorial Image Analysis. Lecture Notes in Computer Science, vol. 5852, pp. 280–288. Playa del Carmen, Mexico (2009)
3. Balázs, P.: A benchmark set for the reconstructin of hv-convex discrete sets. Discrete Appl. Math. **157**, 3447–3456 (2009)
4. Balázs, P.: Benchmark collections for the reconstruction of hv-convex discret sets. http://www.inf.u-szeged.hu/pbalazs/benchmark/benchmark.html (2010). Accessed 11 Sept 2009
5. Billionnet, A., Elloumi, S., Lambert, A.: Extending the QCR method to general mixed-integer programs. Math. Program. **131**, 381–401 (2012)
6. Billionnet, A., Elloumi, S., Plateau, M.-C.: Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: the QCR method. Discrete Appl. Math. **25**, 1185–1197 (2007)
7. Chang, S.: The reconstruction of binary patterns from their projections. Commun. ACM **14**, 21–25 (1971)
8. Chrobak, M., Dürr, C.: Reconstructing hv-convex polyominoes from orthogonal projections. Inf. Process. Lett. **69**, 283–289 (1999)
9. Dahl, G., Flatberg, T.: Optimization and reconstruction of hv-convex (0, 1)-matrices. Discrete Appl. Math. **151**, 93–105 (2005)
10. Del Lungo, A.: Polyominoes defined by two vectors. Theor. Comp. Sci. **127**, 187–198 (1994)

11. Flajolet, P., Zimmermann, P., Van Custem, B.: A calculs for the random generation of labelled combinatorial structures. Theor. Comp. Sci. **132**, 1–35 (1994)
12. Fortet, R.: Applications de l'algèbre de boole en recherche opérationnelle. RAIRO Rech. Opér. **4**, 5–36 (1959)
13. Gale, D.: A theorem on flows in networks. Discrete Math. **187**, 1073–1082 (1957)
14. Glover, F.: Improved linear integer programming formulation of non linear integer problems. Manage. Sci. **22**, 445–460 (1975)
15. Haber, R.M.: Term Rank of 0,1 Matrices. Rendiconti del Seminario Matematico della Università di Padova, vol. 30, pp. 24–51 (1960)
16. Hammer, P.L., Rubin, A.A.: Some remarks on quadratic programming with 0-1 variables. RAIRO Rech. Opér. **3**, 67–79 (1970)
17. Herman, G.T., Kuba, A.: Discrete Tomography: Foundations, Algorithms and Applications. Birkhauser (1999)
18. Irving, R.W., Jerrum, M.R.: Three-dimensional statistical data security problems. SIAM J. Comput **23**, 170–184 (1994)
19. Jarray, F., Costa, M.-C., Picouleau, C.: Approximating hv-convex binary matrices and images from discrete projections. In: 14th Discrete Geometry for Computer Imagery. LNCS, vol. 4992, pp. 413–422. Lyon, France (2008)
20. Kuba, A.: The reconstruction of two-directionally connected binary patterns from their two orthogonal projections. Comput. Vis. Graph. Image Process. **27**(3), 249–265 (1984)
21. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. New York (1988)
22. Ourmazd, A., Kisielowski, C., Schwander, P., Baumann, F.H., Seibt, M., Kim, Y.: An approach to quantitative high-resolution transmission electron microscopy of crystalline materials. Ultramicroscopy **58**, 131–155 (1995)
23. Pinzani, R., Barcucci, E., Del Lungo, A., Nivat, M.: Reconstructing convex polyominoes from horizontal and vertical projections. Theor. Comp. Sci. **155**(2), 321–347 (1996)
24. Prause, G.P.M., Onnasch, D.G.W.: Binary reconstruction of the heart chambers from biplane angiographic image sequence. IEEE Trans. Med. Imag. **15**, 532–559 (1996)
25. Ryser, H.: Combinatorial properties of matrices of zeros and ones. Can. J. Math. **9**, 371–377 (1957)
26. Shliferstein, A.R., Chien, Y.T.: Switching components and the ambiguity problem in the reconstruction of pictures from their projections. Pattern Recogn. **10**, 327–340 (1978)
27. Wang, Y.: Characterization of binary patterns and their projections. IEEE Trans. Comput. **24**, 1032–1035 (1975)
28. Woeginger, G.: The reconstruction of polyominoes from their orthogonal projections. Inf. Process. Lett. **77**, 225–229 (2001)