Efficient Approach for Reconstruction of Convex Binary Images Branch and Bound Method

Shiv Kumar Verma, Tanuja Shrivastava and Divyesh Patel

Abstract In this paper reconstruction algorithm of convex binary image in discrete tomography made efficient by implementing branch and bound method. We focus on diagonal and anti-diagonal (dad) projections and comparison done with the conventional horizontal and vertical (hv) projections. It was shown that proposed strategy is computationally strong and gives fast reconstruction.

Keywords Discrete tomography · Convexity · Branch and bound · Anti-diagonal and diagonal projections

1 Introduction

Reconstruction of binary matrices from few projections is the field of study in discrete tomography. We can provide information about any unknown binary images from the projections in few directions and some prior information about the object such as convexity and connectivity may give close approximate reconstruction result of unknown objects [1–3]. Reconstruction of binary matrix were based on horizontal and vertical projections and using branch and bound method it was first implemented by Miklós and Csongor in [4] and discussed that constructing of tree exponential grow if the size of matrix increase but work well for matrix of order less than 10. In present approach considering different view of binary images,

S. K. Verma (✉) · T. Shrivastava · D. Patel
Department of Mathematics, Indian Institute of Technology-Roorkee, Roorkee, India
e-mail: shivkdma@gmail.com

T. Shrivastava
e-mail: tanujfma@iitr.ernet.in

D. Patel
e-mail: divyshpatel@gmail.com

M. Pant et al. (eds.), *Proceedings of the Third International Conference on Soft Computing for Problem Solving*, Advances in Intelligent Systems and Computing 259,
DOI: 10.1007/978-81-322-1768-8_17,

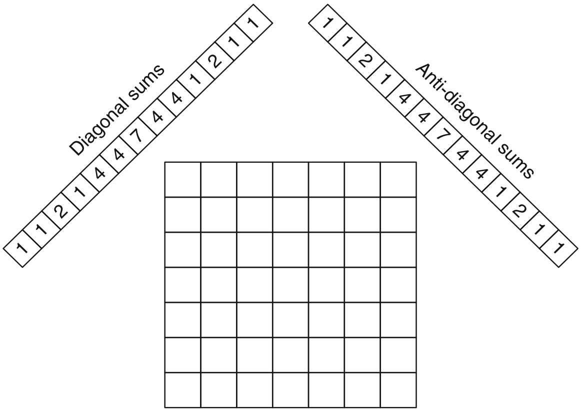

Fig. 1 Binary matrix and projection set in anti-diagonal and diagonal directions

restricting to only two orthogonal projections anti-diagonally and diagonally (Detail description given for anti-diagonal and diagonal characteristics can be find in [5]) shown in Fig. 1. Let $\mathcal{P}(D, A)$ is the given projection set such that,

$$A = \{1 \quad 1 \quad 2 \quad 1 \quad 4 \quad 4 \quad 7 \quad 4 \quad 4 \quad 1 \quad 2 \quad 1 \quad 1\}$$
$$D = \{1 \quad 1 \quad 2 \quad 1 \quad 4 \quad 4 \quad 7 \quad 4 \quad 4 \quad 1 \quad 2 \quad 1 \quad 1\}$$

where A represents the anti-diagonal vector sum, and D represents the diagonal vector sum. If solution exists, then we can reconstruct the binary matrix from the given two projection vectors shown in Fig. 1. The condition for existence of the solution given in [5] for an $m \times n$ binary matrix, where a_k and d_k are number of 1's in anti-diagonal and diagonal projection are as follows,

- $\sum_{k=1}^{m+n-1} a_k = \sum_{k=1}^{m+n-1} d_k$ and
- $\sum_{k=(2l-1)}^{m+n-1} a_k = \sum_{k=(2l-1)}^{m+n-1} d_k$ and $\sum_{k=2l}^{m+n-1} a_k = \sum_{k=2k}^{m+n-1} d_k$:

  for $l = 1, \ldots m + n - 1$.

If suppose there exist exactly single 1 in every row (column), then the possibility to place single one in any row (column) can be perform by $n!$ number of ways and for simplicity if the matrix is square then there will be $n! \times n!$ ways to place a one in binary matrix. In our approach, if the matrix is viewed diagonally and anti-diagonally then placing of 1's accordingly in anti-diagonal (diagonals) sums, since diagonal and anti-diagonal are not intersect each other as in case of horizontal and vertically, thus we can separate those intersecting diagonals with anti-diagonal and observed that odd and even indexing of anti-diagonals (diagonal), then only odd index diagonals (anti-diagonal) will have a single one while the even diagonals (anti-diagonal) will have no 1's. Thus, the total possibility to place single 1's in every row (Column) will be $n!$ which is nothing but the main anti-diagonal (diagonal) and can be considered as best case for anti-diagonal (diagonal). Hence, the total number possibility having exactly single one's in every anti-diagonal (diagonal) will have $n!((n-1)!)$.

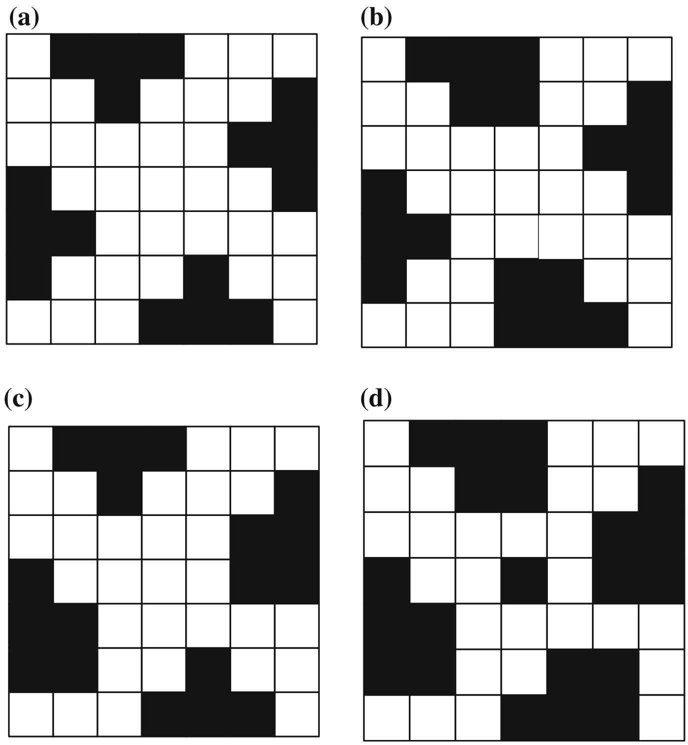

Fig. 2 **a** Dad-convex image, **b** a-convex image, **c** d-convex image, and **d** non-convex image

1.1 Dad-*Convex Matrix*

A binary matrix with respect to anti-diagonal and diagonal projection will called as *ad*-convex if there are no 0's between the sequence of 1's in all the anti-diagonal sums as shown in Fig. 2b, similarly *d*-convex if there are no 0's between the sequence 1's in every diagonals as shown in Fig. 2c. A binary matrix will be dad-convex if it is a-convex as well as *d*-convex as shown in Fig. 2a otherwise non-convex shown in Fig. 2d.

1.2 Branch and Bound Method

The *Branch and Bound* is a convex optimization method for global optimization, and is not always fast (indeed, are often slow). The method is based on the observation that the enumeration of integer solutions has a tree structure. The main idea in branch and bound is to restricting unbound growth of tree by calculating

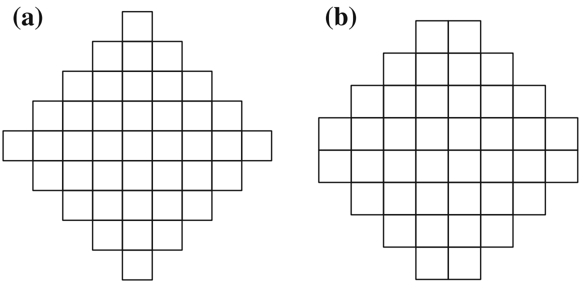

Fig. 3 Building of two tree for anti-diagonal and diagonal projection **a** odd–odd indexing, and **b** even–even indexing

bound at every branching point and at intermediate node point the current version of the branch and bound tree will be available and consists of nodes labeled with their bounding values.

The *node selection policy* governs how to choose the next node for expansion. There are three popular and basic policies for node selection:

- *Best-first* or *global-best node selection*: in this process best bounding value of node is selected for maximizing while for minimizing select the lowest bounding value of nodes.
- *Depth-first*: selection from the current set of nodes i.e. first anti-diagonal sums and go to one step deeper into the branch and bound tree after each iteration by checking the corresponding diagonal sums. Hence, it reaches the last nodes quickly. If it cannot proceed to any deeper into the tree, back track one level and choose another child node from that level.
- *Breadth-first*: expanding nodes in the same order in which they were created.

2 Method of Building Tree

Building tree is done by two separate odd index Fig. 3a binary matrix and even index Fig. 3b binary matrix. For simplicity, square odd order matrix considered.

The development of tree is started with anti-diagonal line sum and the corresponding intersecting diagonal line sums, complying with the *dad*-convexity. If any of the corresponding diagonal sum and dad-convexity properties are not conflicting, then that sub tree will be expanded further, up to leaves of the tree or till diagonal sums and dad-convexity matches else the sub tree will be cut out narrowing the tree. All possible solutions will be obtained upon reaching to the leaves of the tree. The leaves node of the tree will represent the number of solutions.

3 Reconstruction Strategy

The basic idea for constructing the tree will be started with the very first contents of the anti-diagonal sums of a binary matrix. If at any point tree construction is not possible that some of the diagonal sums or the dad-convexity condition may not satisfy, hence these sub trees will be cut out, narrowing the solution. Whenever we reached to the leaf node we get a solution.

Consider a 7×7 binary matrix with anti-diagonal sums, red and black color indicating the intersecting diagonal with the corresponding anti-diagonal. Let the anti-diagonal sums is given by

$$A = [\, 1 \quad 1 \quad 2 \quad 1 \quad 4 \quad 4 \quad 7 \quad 4 \quad 4 \quad 1 \quad 2 \quad 1 \quad 1 \,]$$

and diagonal sums

$$D = [\, 1 \quad 1 \quad 2 \quad 1 \quad 4 \quad 4 \quad 7 \quad 4 \quad 4 \quad 1 \quad 2 \quad 1 \quad 1 \,]$$

Separate all anti-diagonal and diagonal sums into even and odd indexing as follows

$$A_{odd} = [\, 1 \quad 2 \quad 4 \quad 7 \quad 4 \quad 2 \quad 1 \,], \quad A_{even} = [\, 1 \quad 1 \quad 4 \quad 4 \quad 1 \quad 1 \,],$$

Similarly,

$$D_{odd} = [\, 1 \quad 2 \quad 4 \quad 7 \quad 4 \quad 2 \quad 1 \,], \quad D_{even} = [\, 1 \quad 1 \quad 4 \quad 4 \quad 1 \quad 1 \,],$$

Now check for consistency or existence of the solutions

$$\sum A = 33 = \sum D \text{ also } \sum A_{odd} = 21 = \sum D_{odd} \text{ and } \sum A_{even} = 12 = \sum D_{even}$$

Starting with the first anti-diagonal sums $A_{odd}(1) = 1$ and placed in top most block of odd–odd index of Fig. 3a, next for $A_{odd}(2) = 2$ and the available positions are 3 so we can place two 1's in three places by $(P_n - k + 1)$ way for the convex sums, while for the non convex there will be $C_k^{P_n}$ possibility to place 1's where $(P_n = \{1, 2, \ldots n-1, n, n-1, \ldots 2, 1\})$ is a set containing the number of pixel in diagonally or anti-diagonally. The tree will be expanded for 110 and 011. The corresponding diagonal sums matches up to $A_{odd}(4) = 7$, when $A_{odd}(5) = 4$, the diagonal sums became 1345311 which is out of bound to the diagonal sums 1247421, hence tree will not expanded from this node and this node will be pruned. In the last $A_{odd}(7) = 1$, we get four possible matches with the corresponding sums, hence the four different solution as shown in Fig. 4. In the similar manner we will build the tree for the even anti-diagonal and intersecting diagonal sums showing in Fig. 5a and 5b.

In all Figs. 4, 5a and 5b, red box indicating not matching with the diagonal sum, while green box matches with the diagonal sum and represent the possible solution. Thus, we get six possible solution of the given projection set.

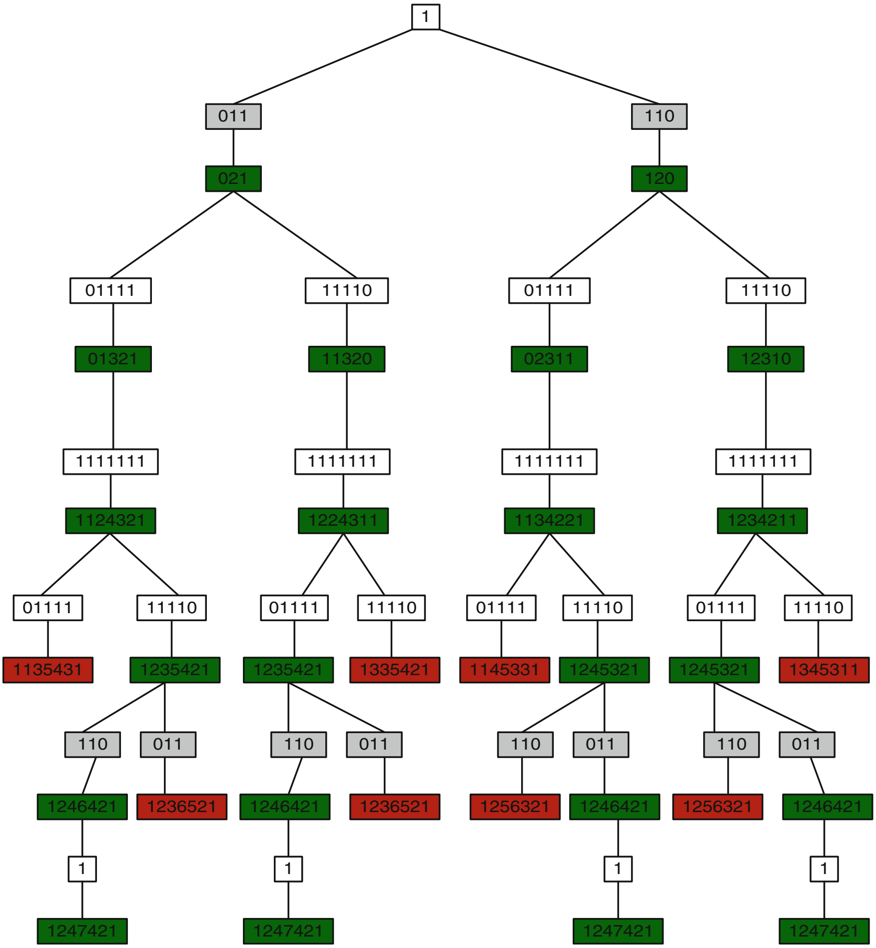

Fig. 4 First solution odd–odd indexed of anti-diagonal and diagonal sums

4 Computational Results

4.1 Unique Solution

If the anti-diagonal and diagonal vector sums are consistent then it is possible to reconstruct unique binary image [6] and from the experiments it was verified that, branch and bound method give fast solution. If we consider the following anti-diagonal and diagonal vector sum

$$A = [1 \quad 2 \quad 1 \quad 2 \quad 1 \quad 2 \quad 3 \quad 2 \quad 5 \quad 1 \quad 0 \quad 1 \\ 1 \quad 1 \quad 0 \quad 5 \quad 2 \quad 3 \quad 2 \quad 1 \quad 2 \quad 1 \quad 2 \quad 1]$$

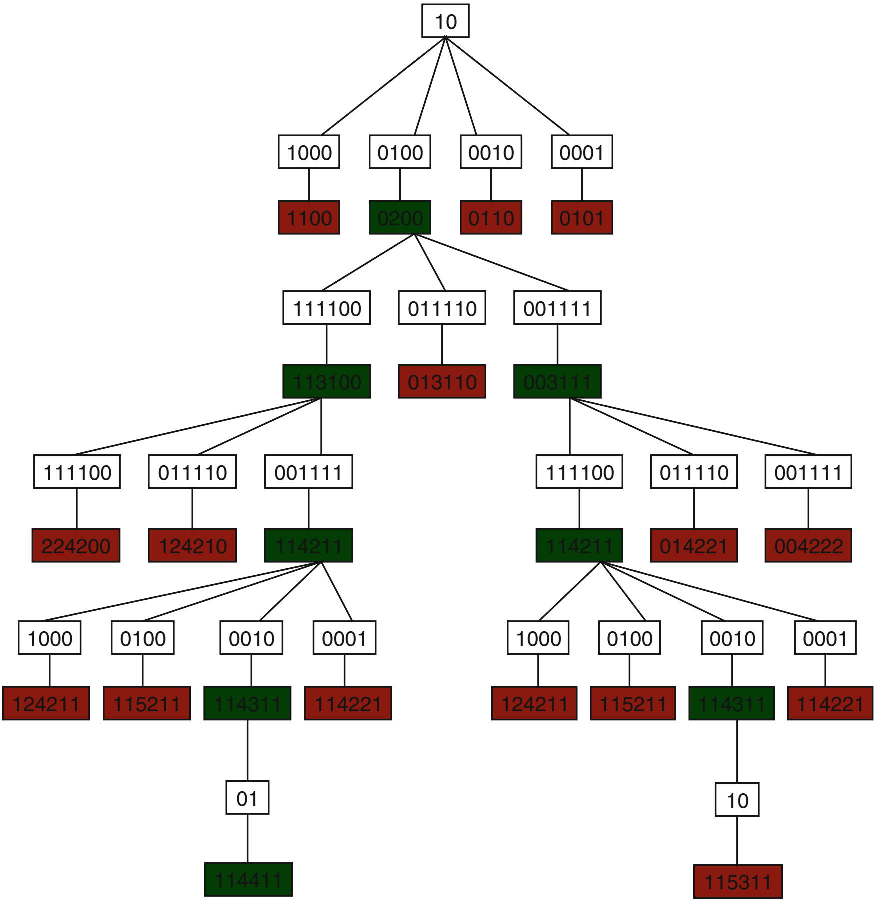

Fig. 5a Second solution for even index of anti-diagonal and diagonal

$$D = [1 \quad 2 \quad 1 \quad 2 \quad 1 \quad 2 \quad 3 \quad 2 \quad 5 \quad 1 \quad 0 \quad 1$$
$$1 \quad 1 \quad 0 \quad 5 \quad 2 \quad 3 \quad 2 \quad 1 \quad 2 \quad 1 \quad 2 \quad 1]$$

We get the following unique solution shown in Fig. 6.

4.2 Other Solutions

There are possibilities to get many solutions of same given projections. It is due to presence of switching elements presents and we follow the dad-convexity condition then we get many valid solution of the same anti-diagonal and diagonal vector

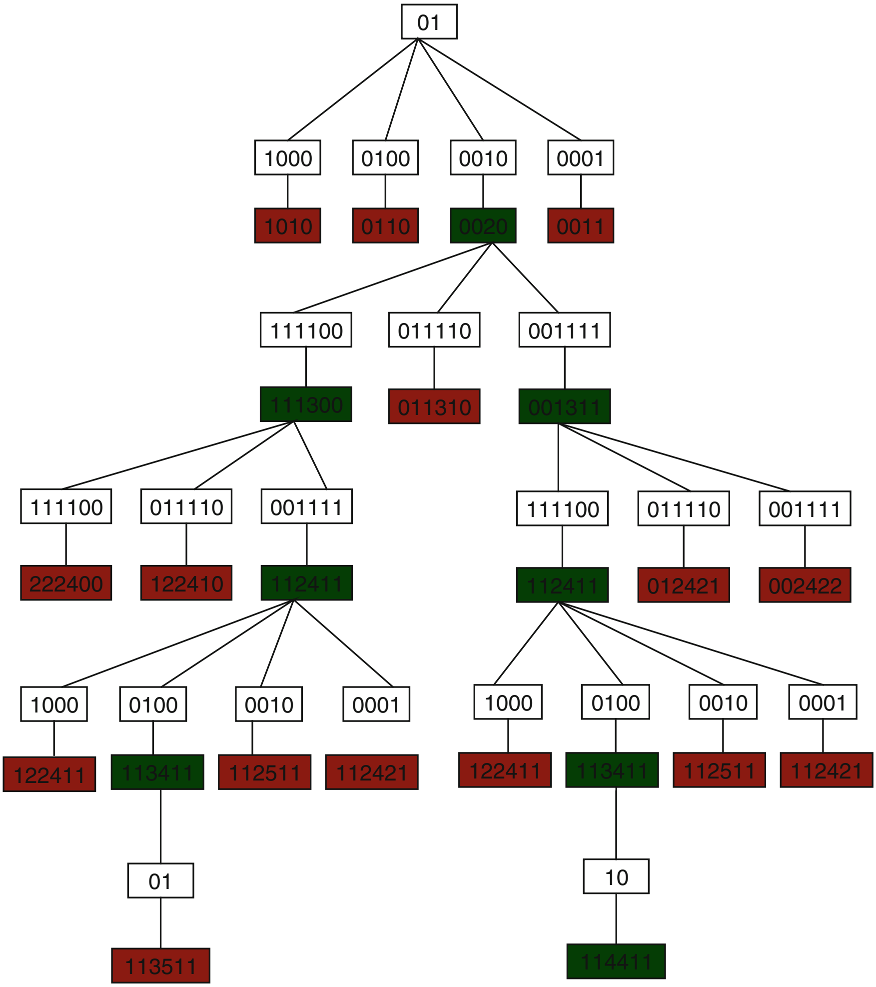

Fig. 5b Third solution for even indexing of diagonals and anti-diagonals

sums as shown in Fig. 7a–d, if we are not following the convexity condition then we get more solution from Sect. 3, shown in Fig. 7e–i.

4.3 Comparison with Horizontal and Vertical Projections

Experiment done on various synthetic binary images and compared our computational result with the *hv*-convex images [4]. It was observed that execution time and number of nodes increases exponentially in the tree in case of *hv*-convex images, while in case of dad-convex, execution time is drastically decreases, although the height of tree is more as compare to *hv*-convex images.

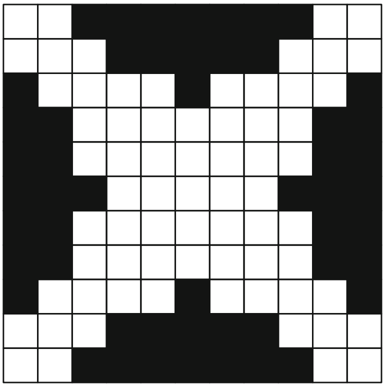

Fig. 6 Unique reconstruction from the given anti-diagonal and diagonal sums

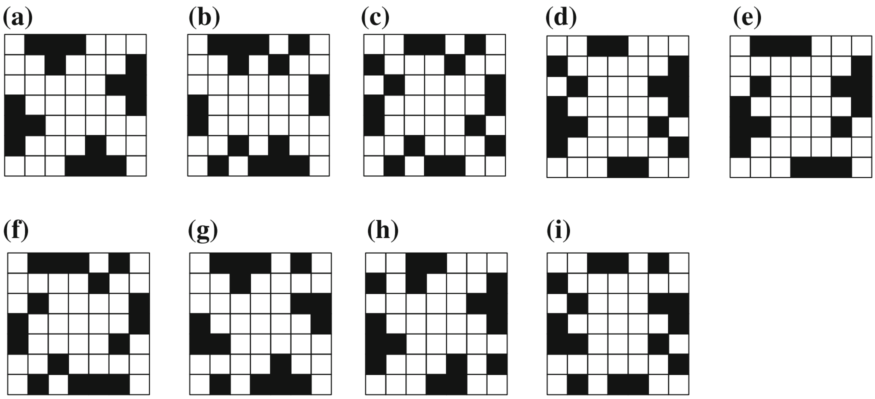

Fig. 7 Other possible solution of **a** due to switching, **b**–**d** with dad-convex and from **e**–**i** without dad-convexity

Consider an example, the row and column sum of a binary image as $R = [2 \quad 4 \quad 4 \quad 4 \quad 2]$, $C = [2 \quad 4 \quad 4 \quad 4 \quad 2]$, than we get four different solution with an average of 38 nodes of all in Fig. 8.

Since the above binary images are *hv*-convex as well as *dad*-convex, if we reconstruct using our approach, unique reconstruction is possible for all images as shown in Fig. 8, an in the tree 8 nodes for Fig. 8a, 7 nodes for Fig. 8b, 6 nodes for Fig. 8c and 7 nodes for Fig. 8d respectively are appeared.

For comparison purpose the proposed strategy and method given in [4]. We have created a database of more than 200 images of various sizes of 5×5 to 30×30, having different ratio of 1's against 0's in every images and varies from 30, 60 and 80 %. Programs were developed in MATLAB 2009, and executed on AMD (Phenom) II Quad Core with 4 GB RAM.

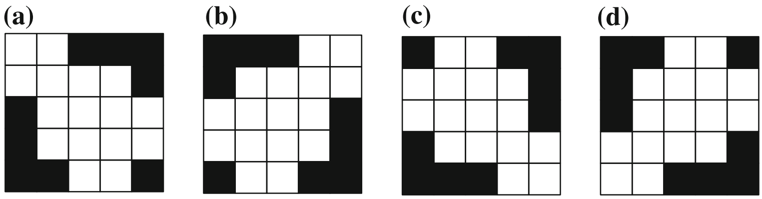

Fig. 8 Four solutions for the given $R = [2 \quad 4 \quad 4 \quad 4 \quad 2]$ and $C = [2 \quad 4 \quad 4 \quad 4 \quad 2]$ sums

Table 1 Comparison of 7 × 7 binary images

1's (%)	Avg. time (s)		Avg. solution		Avg. nodes	
	hv	*dad*	*hv*	*dad*	*hv*	*dad*
30	1.0205	0.4644	1.5	2	84.5	13.84
60	1.0917	0.5023	1.3	1.3	109	11.33
80	0.5416	0.5108	1.3	1.7	61	10

Table 2 Comparison of 11 × 11 binary images

1's (%)	Avg. time (s)		Solution		Nodes	
	hv	*dad*	*hv*	*dad*	*hv*	*dad*
30	24.1593	1.6065	1.6	3	755.4	32
60	22.36	1.3198	1.4	1.2	1,377	58
80	2.3462	0.7129	1.2	1.8	284	30

Table 3 Comparison of 15 × 15 binary images

1's (%)	Avg. time (s)		Solution		Nodes	
	hv	*dad*	*hv*	*dad*	*hv*	*dad*
30	146.32	12.788	2	10	3,360	148.5
60	665.473	8.0052	1	1	18,561	176
80	124.6752	6.0142	4	1	15,632	128

Table 4 Comparison of 21 × 21 binary images

1's (%)	Avg. time (s)		Solution		Nodes	
	hv	*dad*	*hv*	*dad*	*hv*	*dad*
30	245.32	42.365	4	1	42,532	253
60	186.423	14.436	1	1	36,452	187
80	145.471	5.146	1	1	25,732	145

Tables 1, 2, 3, 4, and 5, represents comparison, on the basis of the execution time of *hv*-convex and *dad*-convex, number of node exist in the tree and the number of possible solution exists for the binary image consistent data.

5 Conclusions

Result obtained and displayed in Tables 1, 2, 3, and 4, are clearly indicating that the proposed method is better in execution time, number of nodes presented in tree as compare to horizontal and vertical approach given in [4].

We have implemented our approach for the reconstruction of binary images through branch and bound method using only two orthogonal projections in different directions. Although the branch and bound method is comparatively slow converges rate comparing to other optimization methods. For large size of the binary images it is not feasible for *hv*-convex method of solutions, while it works well for *dad*-convex up to 50×50 order of matrices.

This reconstruction method can be applied for in crystalline tomography, data hiding and security by using the coding and decoding of the transmission data for which further research is in progress.

Acknowledgments This research was supported by the MHRD (Ministry of Human Resource Development) INDIA under the QIP (Quality Improvement Program) sponsored by AICTE (All Indian Council for Technical Education) and Subhash Institute of Software Technology Kanpur.

References

1. Chrobak, M., Durr, C.: Reconstructing hv-convex polyominoes from orthogonal projection. Inf. Process. Lett. **69**(6) (1999)
2. Del Lungo, A.: Reconstructing permutation matrices from diagonal sums. Theor. Comput. Sci. 281(1–2), pp. 235–249 (2002)
3. Kuba, A.: Reconstruction of two directionally connected binary patterns from their orthogonal projections. Comput. Vis. Graph. Image Process. **27**, 249–265 (1984)
4. Miklós, P., Csongor, G.: Discrete tomographic reconstruction of binary matrices using branch and bound method. In: 7th International Symposium on Intelligent Systems and Informatics (SISY 2009), Subotica, Serbia, 25–26 Sept 2009
5. Srivastava, T., Verma, S.K., Patel, D.: Reconstruction of binary images from two orthogonal projections. IJTS **21**(2), 105–114 (2012)
6. Srivastava, T., Verma, S.K.: Uniqueness algorithm with diagonal and anti-diagonal projections. IJTS **23**(2), 22–31 (2013)