Journal of Cellular Automata, Vol. 0, pp. 1–30
Reprints available directly from the publisher
Photocopying permitted by license only

Solutions on 1-D and 2-D Density Classification Problem Using Programmable Cellular Automata

Sudhakar Sahoo[1], Pabitra Pal Choudhury[2], Amita Pal[2] and Birendra Kumar Nayak[3]

[1]*Institute of Mathematics and Applications, Andharua, Bhubaneswar, 751003, India*
Email: sudhakar.sahoo@gmail.com
[2]*Indian Statistical Institute, Kolkata, 700108, India*
Email: pabitrapalchoudhury@gmail.com, pamita@isical.ac.in
[3]*P.G. Department of Mathematics, Utkal University, Bhubaneswar-751004,India*
Email: bknatuu@yahoo.co.uk

Received: October 7, 2013. Accepted: November 29, 2013

This paper deals with the solutions to the problem of Density Classification Task (DCT) using a variant of non-uniform Cellular Automata rules defined and named Programmable Cellular Automata (PCA). The translation properties as well as the density preserving properties of fundamental CA rules are explored. These two properties are utilized to develop different programmable CA to obtain the solution of Density Classification Task (DCT) both for 1-D and 2-D under periodic and fixed boundary conditions. The PCA based DCT solution rely on non-uniform CAs, with fixed rules for fixed boundaries, and rules changing along time for periodic boundary and can be generalisable to the other kinds of boundaries. Various solutions are explicitly given for the 1-D case; for 2-D, a single solution is discussed, and others could be derived, following the same principles of the solution given. Finally, the conventional problem of 2-D DCT is modified to arbitrary shapes and sizes, its applicability, and its solutions are discussed.

Keywords: Cellular Automata Rules, non-uniform Cellular Automata, Density Classification Task

1. INTRODUCTION

Cellular Automata (CA), introduced by John von Neumann [30] have attracted many researchers of various disciplines. The computing process of CA is undertaken by simple interconnected components in a finite number of possible

states, which may be updated in parallel by the same updating rule. The updating rule is only dependent on the states of the component and its neighbors.

Density Clasification Task (DCT) is a simple counting problem having special importance because it leads to finding solutions of the problems related to image processing, analysis of balanced Boolean functions, k-class pattern classification problem [19], testing the density of a matrix, recognizing acid-base concentration in a solution depending on its PH values [27] etc.. Although, elementary Cellular Automata have the capability to solve many complicated problems in various other disciplenes but, it is incapable and facing difficulty when trying to solve the simple DCT problem. This is due to the absence of memory in the CA to keep cumulative count. Therefore, it is always a chalange for CA researchers to solve the fundamental DCT problem on using CA [16], because it involves the potential aspect of the CA theory.

In literature, the methods so far used to solve the DCT problem are (i) determination of rules' performance by using evolutionary algorithms [7, 9, 11, 17, 20-22, 34], (ii) reverse engineering approach using the basins of attraction [3], and (iii) analytic formulation of some specific CA rules [18, 19, 29]. All these methods lead to approximate solutions but not giving exact solutions.

In this paper, following an algorithm, deterministic non-uniform cellular automata rules (named as PCA) are used to obtain the exact solutions of both 1-D and 2-D DCT problems. The initial definition of DCT due to Gacs, Kurdyumov and Levin [16] is slightly modified to its equivalent variation, and the solutions are obtained, following a two-step procedure. In the first step, the initial CA configuration, on application of non-uniform CA rules evolves into an intermediate CA configuration. In the second step, based on the patterns of 0's and 1's in the intermediate configuration, a specific CA rule is used that leads to the DCT solution. The paper is organised as follows:

Section 2 provides the basic concepts of CA. Survey of earlier works that elucidate DCT solutions are given in section 3. The fundamental CA rules, which are used in the proposed algorithms, are given in section 4. Section 5 and 6 describes the proposed algorithms (PCA) to solve 1-D and 2-D DCT problems. Further, in section 6, variations of 2-D DCT problem, its practical applications, and solutions using the concept of 2-D CA are discussed. Section 7, is the conclusion section of the article.

2. ELEMENTARY CELLULAR AUTOMATA (CA) RULES IN 1-D AND 2-D

A 1-D elementary CA is a pair, consisting of a row of "cells" and a set of "rules". The ***row of cells*** at any time-instant t is represented by a vector $(x_1^t, x_2^t, ..., x_n^t)$. The bit in the i^{th} cell x_i^t, at the "next" time-instant $(t+1)$, is changed to x_i^{t+1} by a ***local mapping*** denoted by f_i^t, and all other cells too change their states at the same time. The change of states by a local mapping takes place

$$x_i^{t+1} = f_i^t(X = x_{i-1}^t, Y = x_i^t, Z = x_{i+1}^t),\ i = 2,3,...,n-1$$

FIGURE 1
Shows an 1-D neighborhood of radius 3.

in the following manner: each cell looks around and gathers information on its neighbor's states. Based on its own state Y (ref. fig-1), its left neighbor's state X, its right neighbor's state Z, and the rule of the CA, the cell decides what its new state should be. Thus a local rule at time $(t + 1)$ is of the form, x_i^{t+1}as shown in fig-1.

For two extreme cells x_1 and x_n, the changes occur at the time instant $(t + 1)$, according to the boundary conditions imposed. Those boundary conditions could be Null, Periodic, Fixed, Adiabatic, Reflexive, and Intermediate. In case of **Fixed boundary** conditions the extreme cells are connected by a pre-assigned fixed logic states either 0/1. If the extreme cells are connected by logic 0 states then it is called **Null boundary** CA. In **Periodic boundary** conditions the extreme cells are connected to each other and form a cycle. In **Adiabatic boundary** conditions the missing neighbor states are same as the corresponding boundary cell values. In **Reflexive** boundary conditions the value of left and right neighbors are same with respect to the boundary cell. In **Intermediate** boundary conditions the value of the left (right) boundary will be the same as the cell value present at it's next to next (previous to previous) cell. Different boundary conditions in 1-D, 3- neighbourhoods CA of length n are represented as follows:

Fixed boundary condition: $0(x_1, x_2, x_3, ..., x_{n-1}, x_n)0$ or
$0(x_1, x_2, x_3, ..., x_{n-1}, x_n)1$ or
$1(x_1, x_2, x_3, ..., x_{n-1}, x_n)0$ or $1(x_1, x_2, x_3, ..., x_{n-1}, x_n)1$
Null boundary condition: $0(x_1, x_2, x_3, ..., x_{n-1}, x_n)0$
Periodic boundary condition: $x_n(x_1, x_2, x_3, ..., x_{n-1}, x_n)x_1$
Adiabatic boundary condition: $x_1(x_1, x_2, x_3, ..., x_{n-1}, x_n)x_n$
Reflexive boundary condition: $x_2(x_1, x_2, x_3, ..., x_{n-1}, x_n)x_{n-1}$
Intermediate boundary condition: $x_3(x_1, x_2, x_3, ..., x_{n-2}, x_{n-1}, x_n)x_{n-2}$

If the same CA rule is applied to each cell of a CA configuration, the CA will be called a **Uniform Cellular Automaton**; otherwise it will be called a **non-Uniform / Hybrid Cellular Automaton**. In case of 1-D, 3-neighborhood, 2-state Cellular Automaton, the number of all possible uniform CA rules is 256. That rules can be enumerated using Wolfram's naming convention [32, 33] from rule number 0 to rule number 255 and can be represented by a 3-variable Boolean function. The algebraic expression [31] for any 3-variable Boolean function contains terms including X, Y, Z in general. Under this perspective, that three rules, which will be used in latter sections, are considered fundamental and are shown in table 1.

XYZ	111	110	101	100	011	010	001	000
Rule X or Rule 240	1	1	1	1	0	0	0	0
Rule Y or Rule 204	1	1	0	0	1	1	0	0
Rule Z or Rule 170	1	0	1	0	1	0	1	0

TABLE 1
Shows 3-fundamental 1-D CA rules with 3-neighborhood structure.

$Initial:$	0	0	0	0	0	0	0	1	1	1	1	1
$time = 1:$	0	0	0	0	0	0	1	1	1	1	1	0
$time = 2:$	0	0	0	0	0	1	1	1	1	1	0	0
$time = 3:$	0	0	0	0	1	1	1	1	1	0	0	0
$time = 4:$	0	0	0	1	1	1	1	1	0	0	0	0
$time = 5:$	0	0	1	1	1	1	1	0	0	0	0	0
$time = 6:$	0	1	1	1	1	1	0	0	0	0	0	0
$time = 7:$	1	1	1	1	1	0	0	0	0	0	0	0

FIGURE 2
Shows the Space-time diagram of Rule Z (or Rule 170) in Null boundary condition.

64	**128**	**256**
32	**1**	**2**
16	**8**	**4**

FIGURE 3
Shows a naming scheme for 2-D nine- neighborhood fundamental CA rules.

The CA evolution (also known as Space-time diagram) of Rule Z (equivalently Rule 170 in Wolfram's convention) starting from an initial configuration is shown in fig-2. It may be noted that applying Rule Z clustering of 1's in right side have been shifted to left side.

Similar to 1-D, in 2-D nine-neighborhood standard CA, the next state of a particular cell is influenced by its own current state and the states of other eight cells in its nearest neighbourhood. This nine-neighborhood is also referred as Moore neighborhood and shown in fig-3.

Such neighborhood dependencies are described by various 2-D CA rules or called transition functions. For the sake of simplicity, in this section we take into consideration only nine fundamental rules. A specific rule convention that is adopted in [10] is shown in fig-3 where, the central box represents the current cell (i.e. the cell being considered) and all other boxes represent the eight nearest neighbors of that cell. The number within each box represents the rule number characterizing the dependency of the current cell on that particular neighbor

only. Rule 1 characterizes dependency of the central cell on its own present state where as such dependency only on its top neighbor is characterized by Rule 128, and so on. Similar to 1-D case, these nine rules 1, 2, 4, 8, 16, 32, 64, 128, and 256 are called fundamental rules. The application of these rules can be realized on a problem matrix which is also called an information matrix where every matrix entry is either 0 or 1. It may be mentioned that instead of applying the same rule to each entry of the problem matrix, it is admissible to apply different fundamental rules to different entries at the same time. While the former characterizes the **Uniform CA,** the latter characterizes **non-Uniform CA** in 2-D. Different boundary conditions in 2-D CA such as Null, Periodic, Fixed, Adiabatic, and Reflexive etc. can also be defined similar to 1-D CA.

Illustration (For Uniform CA)
In this example, Rule 128 is applied uniformly to each cell of a 2-D problem matrix of order (3 x 4) with periodic boundary condition. It may be noted that on applying Rule 128 the rows of the problem matrix being shifted from top to bottom in a periodic way as shown in fig-4.

Illustration (For non-Uniform CA)
Here is an example of a non-uniform (or hybrid) CA in null boundary condition (see fig-5), where 3 rules (Rule 2, Rule 8, and Rule 16) are applied in 3 different rows (1st, 2nd and 3rd rows respectively) in the problem matrix and the corresponding output matrix is obtained.

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}_{(3\times4)} \xrightarrow{Rule\ 128\,(Periodic\ Boundary)} \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}_{(3\times4)}$$

FIGURE 4
Shows the CA Rule 128 is applied to all cells. That is each cell will change its state by the states of its top neighboring cells using the definition of periodic boundary condition for boundary cells.

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}_{(3\times4)} \xrightarrow{Rule\ 2, Rule 8, Rule 16\ (Null\ Boundary)} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}_{(3\times4)}$$

FIGURE 5
Shows 3 CA rules (Rule 2, Rule 8, and Rule16) are applied in 3 different rows (1st, 2nd and 3rd rows respectively) in the problem matrix using the definition of Null boundary conditions for extreme cells.

3. SURVEY ON DENSITY CLASSIFICATION TASK (DCT)

Density Classification Task, a remarkable problem in the field of Cellular Automata (CA) is a simple computing problem in which the CA starts with an initial configuration and forms a configuration with either all 1's or all 0's depending on which had the higher initial density. For example, if the CA's initial state contains 60% 1's and 40% 0's then there are more 1's than 0's hence the CA can attain a configuration with 100% 1's and remain in that state. Like wise, if the initial state contains 30% 1's and 70% 0's then the CA can attain a configuration where every cell has 0.

The most investigated rule space related to DCT is radius-3, 1-D uniform CA. A brief chronological resume about the results in this rule space and the related references are given below.

1. The first rule proposed to solve DCT problem was by Gacs, Kurdyumov and Levin (1978) and known as GKL rule [16].
2. Packard in 1988 proposed, for the first time, the rules for DCT by using evolutionary method [26].
3. The first published evolved rule for DCT, but with efficacy below GKL was due to Mitchel, Hraber and Crutchfield [20].
4. Andre et al. gave the first evolved rule better than GKL [1].
5. The rule that remained as the best-known rule for about 10 years was given by Juille and Pollack [17].
6. Currently the best known rule with efficacy close to 90% is given by D. Wolz and Pedro P. B. de Oliveira [34].
7. The CA rules for DCT, by using Genetic Algorithm (GA), are available in the studies by Mitchell et al., [21, 22], Crutchfield and Mitchell [7], Jullie and Pollack [17] and Bossomaier et al. [2], Oliveira et al. [25].

Land and Belew [18] proved that no 1-D two-state CA rule can be constructed, which classifies binary strings according to their densities of 1's and 0's. On building upon the proof of Land and Belew; Chau, Siu and Yan [8] proved the impossibility of finding the n-ary version of DCT. Further, it has been demonstrated by Capcarrere et al., [5] that a solution to the density classification problem does exist, with a different output compared to that of Gacs, Kurdyumov and Levin [16], and Packard [26]. Capcarrere and Sipper [6] demonstrated that a rule, which resolves the density classification problem, for 1-D standard CA, has to satisfy two conditions:

1. The density of the initial configuration must be preserved over time.
2. The rules table must exhibit density of 0.5 implying that it should be a balanced rule.

The multi-state 1-D CA rules obtained by the application of GA are used for the first time to solve DCT by Gabriele [15], where the result shows that there is a substantial homogeneity between elementary CA and multi state CA.

It has been shown by Fuks [14], the "traffic rule" 184 and the "majority rule" 232 taken together, perform the density classification task perfectly. The main result of that paper is the following proposition, where the density ρ is defined as the ratio between the numbers of 1's with respect to the total number of cells in the initial configuration.

Proposition 1 Let s be a configuration of length L and density ρ, and let $n = \lfloor (L-2)/2 \rfloor$, $m = \lfloor (L-1)/2 \rfloor$. Then $F_{232}^{m}(F_{184}^{n}(s))$ consists of only 0's if $\rho < 1/2$ and of only 1's if $\rho > 1/2$. If $\rho = 1/2$ then $F_{232}^{m}(F_{184}^{n}(s))$ is an alternating sequence of 0 and 1 that is of the form 0101...0101.

Although the solution proposed by Fuks [14] performs the task in L time steps, it is straightforward to construct a faster algorithm, provided that rules of larger radius are allowed. If F is a radius-1 rule, then F^n, the rule iterated n times, is itself a CA rule of radius n. Therefore, the pair (g, h), where $g = F_{184}^{n}$ and $h = F_{232}^{n}$, performs the classification task in L/n time steps, assuming that both g and h are iterated for L/2n time steps.

On utilizing the characteristic of CA rather than any statistical approach involving GA, Maiti et al. have given an analytical framework for 1-D DCT solution [19]. In their work Best Rule Vector (BRV) has been derived from the analysis of Rule Vector Graph (RVG) generated from the Rule Vector (RV) of a CA. They have also analyzed the error, obtained in the DCT solution by introducing two types of error namely primary and secondary, followed by the attempt to reduce the error by extending the neighborhood to k (k = 3, 5, 7, 9) in 1-D CA.

In the literature [23, 27, 24], attempts are made to find solutions of 2-D DCT, though Reynaga and Amthauer [27] have shown that, no perfect rule for 2-D DCT exists. It has been shown by Busic, Fates, Mairesse and Marcovici [4] that the so-called Toom's rule perfectly solves infinite length 2-D lattices and according to Fates [13], one might not rely upon the Tooms's rule for solving DCT of finite lattices.

Till date, what is known and what is not, both for standard DCT and for various alternative formulations of DCT are very accurately reviewed by Pedro P. B. deOliveira [12].

But in literature, one does not find any attempt being made to design a general algorithm to solve 1-D and 2-D density classification problems for an arbitrary critical density. In this paper, this attempt is made; a general DCT problem is designed and an algorithm is developed to find its solution with the help of non-Uniform CA named as PCA.

4. REDEFINING DCT IN THE CONTEXT OF FUNDAMENTAL CA RULES

The solutions presented in this paper both for 1-D and 2-D cases are obtained, by applying a two-phase procedure albeit slightly modifying the classical definition of DCT originally used by Gacs, Kurdyumov and Levin [16], and Packard [26]. For our purpose, an equivalent density classification problem is defined as follows:

Definition: The initial CA configuration with the application of proper CA rules results in an intermediate CA configuration of 0's and 1's and then the PCA picks up the classification decision based on the pre-produced densities.

This way of defining density classification problem is equivalent to the original DCT problem defined by Gacs et al. [16]. It is because of the existence of two trivial CA rules; Rule 0 and Rule $(2^{2^k}-1)$ for a neighborhood of radius k. For example when $k = 3$, the Rule number $(2^{2^k}-1)$ is equal to 255 in 1-D and in 2-D for $k = 9$, the rule number is $(2^{512}-1)$. An interesting property of these trivial rules is that, starting from any random configuration, Rule 0 and Rule $(2^{2^k}-1)$ cause this random configuration to evolve, to a quiescent configuration of cell values all 0's and all 1's respectively.

4.1 Translation of images and density preservation by 1-D fundamental rules

In table 2, two fundamental 1-D CA rules, Rule X in Algebraic Normal Form (ANF) [31] equivalently Rule 240 in Wolfram convention [32], and Rule Z (equivalently Rule 170) has the ability to translate 1's from left to right and from right to left respectively. Again the identity rule, Rule Y (ANF) equivalently Rule 204 (Wolfram convention) can preserve the number of 1's and 0's constant throughout the evolution. Thus the combined effect of these three rules can serve the dual purpose of translation and density preservation as shown in table 2.

4.2 Translation of images and density preservation by 2-D fundamental rules

It is reported in [10] that, the fundamental rules of fig-3 namely rule 2, 4, 8, 16 and their transpose rules 32, 64, 128, 256 respectively, causes translation of the image in the directions mentioned against each rule in table 3. The fundamental rule, Rule 1 being an identity rule, satisfies density preservation property (that is number of 1's and 0's are constant throughout the evolution).

For applying 2-D fundamental rules for translation of images, a binary matrix of size (100×100) may be taken. Each element of the matrix may be mapped to a unique pixel on the screen and may be colored a pixel White for 0 and Black for 1 of the matrix elements. 2-D CA fundamental rule is when

Direction of the translation of the image	Rules to be applied
0s move towards left 1s move towards right	Rule X (ANF) or Rule 240 (Wolfram)
No translation but densities of 0's and 1's are preserved	Rule Y (ANF) or Rule 204 (Wolfram)
0s move towards right 1s move towards left	Rule Z (ANF) or Rule 170 (Wolfram)

TABLE 2
Directions of translation of an image on applying 3-fundamental 1-D CA rule.

Direction of the translation of the image	Rules to be applied
Top (↑)	Rule 8
Bottom (↓)	Rule 128
Left (←)	Rule 2
Right (→)	Rule 32
No translation but densities of 0's and 1's are preserved	Rule 1
Top-Left (↖)	Rule 4
Top-Right (↗)	Rule 16
Bottom-Left (↙)	Rule 256
Bottom-Right (↘)	Rule 64

TABLE 3
Directions of translation of an image on applying 9-fundamental 2-D CA rule.

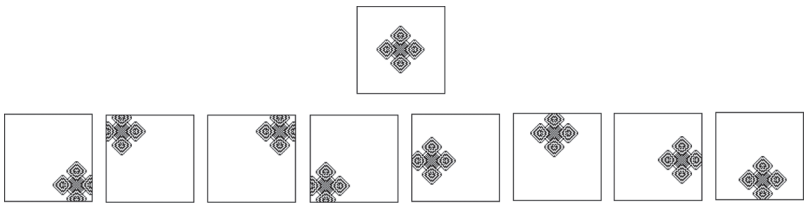

FIGURE 6
Shows the translation of an image after application of CA rules 64, 4, 16, 256, 2, 8, 32 and128 to a given image.

applied uniformly on that (100×100) matrix; the matrix changes such that a new image is redrawn. Initial image is taken by making some elements of the (100×100) matrix as 1, in a suitable fashion. We call this initial image as 'seed image'. The extent of shift in a particular direction is dependent on the number of repetitions of the application of the fundamental rules as illustrated in the fig-6 followed by table 3.

5. SOLUTION OF 1-D DCT USING PROGRAMMABLE CELLULAR AUTOMATA

5.1 Algorithm to solve 1-D Density Classification Problem

The algorithm discussed here to solve 1-D DCT problem can be divided into two-phases, a pre-processing phase and a decision phase. In the pre-processing phase the initial CA configuration of length n is caused to evolve by CA rules to reach a particular type of configuration, wherefrom the phase to make a decision (decision phase) begins regarding the densities. Similar to Sieve of Eratosthenes technique to filter prime numbers and composite numbers, the pre-processing phase in the proposed algorithm separates 1's and 0's to form separate clusters. One such intermediate configuration to be used in the pre-processing phase of the proposed 1-D DCT algorithm is shown in fig-7.

It has been shown in fig-8 that starting from an initial 1-D configuration $(x_1, x_2, x_3, ..., x_{n-1}, x_n)$ of length n, we have to findout some CA rules, which translates all 1's towards left, and all 0's towards right with a condition that

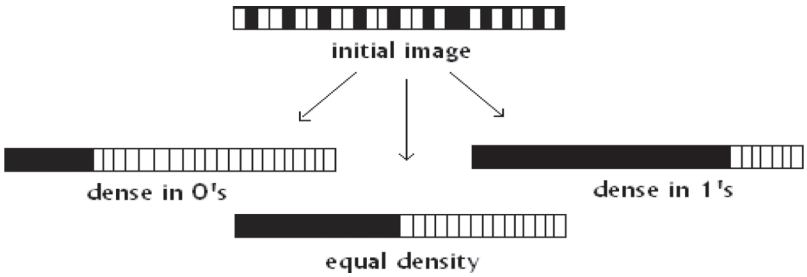

FIGURE 7
Possible intermediate outputs for taking decision in 1-D DCT algorithm..

Input: 1 0 1 1 0 1 0 1

Pre-processing phase:

for i = 1: 1 1 0 1 1 0 1 0
i = 2: 1 1 1 0 1 1 0 0
i = 3: 1 1 1 1 0 1 0 0
i = 4: 1 1 1 1 1 0 0 0
i = 5: 1 1 1 1 1 0 0 0
i = 6: 1 1 1 1 1 0 0 0
i = 7: 1 1 1 1 1 0 0 0

Decision phase:

As (middle) and (middle+1) cell is 11, So, *Apply Rule 255* to the configuration gives:

i = 8: 1 1 1 1 1 1 1 1

Output: input image is dense in 1's

FIGURE 8
Shows the pre-processing phase and the decision phase of 1-D DCT algorithm for a random input string of length 8.

number of 0's and 1's should not be changed throughout the evolution process. To achieve translation, two fundamental CA rules Rule X and Rule Z and for density preservation Rule Y of table 2, can be used because they satisfy these requirements. Also some special care should be taken for different boundary conditions. For example, under periodic boundary condition, the 1's that translate to wards left should not again return back from the right end. Further, it may be mentioned here that the effect of an arbitrary rule vector can be captured by the combined effect of three fundamental rules X, Y or Z or their complements. This has been exemplified in table 4 stating that at different neighborhood situation, rules X, Y or Z or their complements are invoked and as a result, rule 226 is obtained. Again selection of these rules X, Y or Z or their complements for a particular neighborhood situation may not be unique. For example in table 4, the "XYZ" values "110" $\rightarrow$ "1" can be obtained by applying any one of the rule say either Rule X or Y or Z^c (complement of Z).

Table 5 shows the list of 1-D CA rules that can be used in the pre-processing phase proposed in the 1-D DCT algorithm. Rules, which achieve both translation and density preservation, as discussed above, are obtained as follows.

For all the cells from x_2 to x_{n-1}, except the boundary, the output requirements for all possible 3-neighborhood situations are given in table 4. As an example, for the neighborhood situation 111, 110, 100, and 000 where, 1s, if any, are present in the left side and 0s, if any, are in the right side so there is no need to change the state for this type of neighborhoods. But, for other neighborhood situation 101, 011, 010, and 001 the 1s are on the right of 0s, so there is a need for translation of 1s from right to left. The complete change of all these eight situations is shown in table 4 from which the desired rule vector can be constructed.

From table 4, collecting the middle cell values from the requirement row the 1-D CA rule that achieves both translation as well as density preservation is the rule vector "11100010", which is rule 226 (by Wolfram's naming convention).

But, for the boundary cells, the 1-D CA rules that achieve translation and density preservation are obtained as follows. If the value of left neighbor of the first cell x_1 is 0 then out of eight possible 3-neighborhood situations only

XYZ	111	110	101	100	011	010	001	000
Requirement	111	110	110	100	101	100	010	000
Rule 226	1	1	1	0	0	0	1	0
Fundamental rules	$X/Y/Z$	$X/Y/Z^c$	$X/Y^c/Z$	$X^c/Y/Z$	$X/Y^c/Z^c$	$X/Y^c/Z$	$X^c/Y^c/Z$	$X/Y/Z$

TABLE 4
Shows the CA rule 226 obtained by considering Translation as well as Density Preservation property of 3 fundamental rules X, Y or Z.

(1) Cells	(2) If the left neighbour of $x_1 = 0$ then apply any rule to x_1	(3) If the left neighbour of $x_1 = 1$ then apply any rule to x_1	(4) Rules for all other cells x_2 to x_{n-1}	(5) If the right neighbour of $x_n = 0$ then apply any rule to x_n	(6) If the right neighbour of $x_n = 1$ then apply any rule to x_n
Rules	14, 30, 46, 62, 78, 94, 110, 126, 142, 158, 174, 190, 206, 222, 238, 254	224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239	226	64, 66, 72, 74, 96, 98, 104, 106, 192, 194, 200, 202, 224, 226, 232, 234	128, 129, 132, 133, 144, 145, 148, 149, 192, 193, 196, 197, 208, 209, 212, 213

TABLE 5
Rules used to solve 1-D DCT problem.

$0x_1x_2$	111	110	101	100	011	010	001	000
Requirement	-	-	-	-	011	010	010	000
16 Rules	0/1	0/1	0/1	0/1	1	1	1	0

TABLE 6
Shows the CA rules that can be used for the boundary cell x_1 if its left neighbour value is 0.

four situations 011, 010, 001, and 000 where first bit is "0" will be valid for x_1 as shown in table 6. Out of this four, as the value of the first cell $x_1 = 1$ for the neighborhood situation 011 and 010 thus for density preservation there is no need to translate this 1 towards left. So for these neighborhoods 011 and 010 we need a CA rule such that after applying the rule the requirements should be again 011 and 010 respectively. But for the string 001 the right side 1 can move to the cell position x_1. So, for this neighborhood after applying the rule the requirement should be 010. Similarly, after applying the CA rule to the neighborhood 000 the requirement should obviously be 000. Thus from the requirement row given in table 6, the CA rule can be computed by assigning the values 1, 1, 1 and 0 respectively corresponding to that four neighborhoods and the other four neighborhood situations 111, 110, 101, and 100 where the left neighbor of x_1 is 1 can be assigned by 0 or 1. In this way 16

possible CA rules can be obtained and some of these rule vectors are 00001110 = 14, 00011110 = 30 and so on. The complete list of these rules is given in column (2) of table 5.

Table 7 shows 16 other rules that can be obtained by fixing the left neighbor of $x_1 = 1$. The complete list of the rules in Wolfram's naming convention is given in column (3) of table 5.

Similarly all other rules applicable for the boundary cell x_n can be found out by considering the right neighbour value of x_n either 0 or 1 as shown in table 8 and table 9 respectively. The complete list of the rules are given in column (5) and (6) of table 5.

Thus, the main result of the above discussion gives a set of rules that are listed in table 5 and that can be used in the pre-processing phase of the 1-D DCT algorithm to reach at a configuration like fig-8.

After pre-processing phase is over, depending on the length of CA is either even or odd, a decision is made in the decision phase. For an odd length CA, only two outputs are possible, which is either dense in 0's or dense in 1's. In this case, only looking the value of the middle cell $x_{(n+1)/2}$(or $x_{\lceil n+1 \rceil/2}$) is either 0 or 1, it is possible to take the decision of classification. In case of an even length CA, looking two cells value $x_{(n/2)}$ and $x_{(n/2)+1}$ situated at the middle and the (middle+1) positions respectively, one can take the decision for their densities. It may be mentioned here that, in the decision phase, two types of outputs are possible for DCT. In first type, one can print the output as: the

$0x_1x_2$	111	110	101	100	011	010	001	000
Requirement	111	110	110	100	-	-	-	-
16 Rules	1	1	1	0	0/1	0/1	0/1	0/1

TABLE 7
Shows the CA rules that can be used for the boundary cell x_1 if its left neighbour value is 1.

$x_{n-1}\,x_n0$	111	110	101	100	011	010	001	000
Requirement	-	110	-	010	-	010	-	000
16 Rules	0/1	1	0/1	1	0/1	1	0/1	0

TABLE 8
Shows the CA rules that can be used for the boundary cell x_n if its left neighbour value is 0.

$x_{n-1}\,x_n1$	111	110	101	100	011	010	001	000
Requirement	111	-	011	-	011	-	001	-
16 Rules	1	0/1	1	0/1	1	0/1	0	0/1

TABLE 9
Shows the CA rules that can be used for the boundary cell x_n if its left neighbour value is 1.

starting CA configuration is dense in 0's or the configuration is dense in 1's for an odd length CA, and in case of even length CA, in addition to the previous, another possibility of equal density can also be printed. This particular way of output is required for the DCT problem defined by Capcarrere et al. [5]. In the second type, if the CA is either dense in 0's or dense in 1's trivial rules: Rule 0 or Rule 255 respectively may be applied to get an output of 1-D DCT defined by Gacs et al. [16] and Packard [26]. Thus, using the proposed method, it is possible to solve both the original DCT defined by Gacs et al. [16] and Packard [26] as well as the DCT defined by Capcarrere et al. [5].

The proposed algorithm is versatile as much as it can handle different boundary conditions such as null, periodic, fixed, adiabatic etc. For example, in case of null boundary condition where the extreme cells are connected to logic-0 states only the CA rules given in Column (2), (4) and (5) of table 5 will be used. Similarly for periodic boundary condition (where the extreme cells are connected to each other) depending upon the neighboring values of x_1 and x_n different rules from table 5 can be selected. In addition to solve the 1-D DCT problem the proposed 1-D algorithm can act as a basis to develop the 2-D DCT algorithm.

The proposed two-phase algorithm to solve 1-D DCT problem using the CA rules from table 5 is as follows:

(A) Two-Phase Algorithm to solve 1-D DCT

*/*Pre-processing Phase: Reaching to an intermediate configuration like fig-7*/*

1. *Input: An initial 1-D CA configuration* $(x_1, x_2, x_3, ..., x_{n-1}, x_n)$ *of length n*
2. $i = 1$
3. *while* $(i < \text{n})$
4. {

*/*Rules for boundary cells in different boundary conditions as shown in table 5*/*

// Rules for the first cell x_1

5. *if (the left neighbor of* $x_1 = 0$*) then*
6. *Apply any rule to* x_1 *out of 16 possible rules given in column (2) of table 5*
7. *else if (the left neighbor of* $x_1 = 1$*) then*
8. *Apply any rule to* x_1 *out of 16 possible rules given in column (3)*

// Rules for the last cell x_n

9. *if (the right neighbor of* $x_n = 0$*) then*
10. *Apply any rule to* x_n *out of 16 possible rules given in column (5)*
11. *else if (the right neighbor of* $x_n = 1$*) then*
12. *Apply any rule to* x_1 *out of 16 possible rules given in column (6)*

// Rules for all other cells from x_2 *to* x_{n-1}

13. *Apply Rule 226 uniformly to all the cells from* x_2 *to* x_{n-1} *(refer column 4)*
14. *if (no updating taking place in all the cells) then exit from the while loop*
15. *else* $(i = i + 1)$ *and continue*
16. }

*/*Decision Phase: For deciding the density of the configuration*/*
17. if (n is odd)
18. {
19. if (the bit at position $\lceil n/2 \rceil = 1$*)then*
20. Apply Rule 255 uniformly to all the cells
21. Print "the CA is dense in 1's"
22. else if (the bit at position $\lceil n/2 \rceil = 0$*)then*
23. Apply Rule 0 uniformly to all the cells
24. Print "the CA is dense in 0's"
25. }
26. else if (n is even)
27. {
28. if (the two consecutive bit at positions (n/2) and (n/2) $+ 1 = 00$*) then*
29. Apply Rule 0 uniformly to all the cells
30. Print "the CA is dense in 0's"
31. else if (the two consecutive bit at positions (n/2) and (n/2) $+ 1 = 10$*) then*
32. Print "equal densities in 0's and 1's"
33. else if (the two consecutive bit at positions (n/2) and (n/2) $+ 1 = 11$*) then*
34. Apply Rule 255 uniformly to all the cells
35. Print "the CA is dense in 1's"
36. }

(B) Explanation of the algorithm

Step 1 to Step 16 of the algorithm is the pre-processing phase and generates the required intermediate image like fig-7. Starting from the initial configuration of length n, in the worst case CA is evolved up to $(n-1)$ times by step 3. In each evolution, Rule 226 is applied to all the cells except the boundary. To the boundary cells (x_1 and x_n), depending on the neighborhood values, rules from table 5 can be selected and shown in steps 5 to 8 of the algorithm.

For an odd length CA, only two outputs are possible (Steps 17 to 25), which could either be dense in 0's or dense in 1's. In this case only reading the middle cell value of the vector $(x_1, x_2, x_3, ..., x_{n-1}, x_n)$ one can take the decision of their densities.

For an even length CA, the decision for density classification is made on reading two cell values situated at the middle and the (middle+1) positions of the vector $(x_1, x_2, x_3, ..., x_{n-1}, x_n)$ (Steps 26 to 36).

It can be proved that, after the pre-processing phase of the 1-D DCT algorithm, the string of two consecutive bits in the middle and (middle+1) position of the vector $(x_1, x_2, x_3, ..., x_{n-1}, x_n)$ can never be "01". If so then depending on the cell value situated at (middle-1) position of the vector either 0 or 1, the (middle) cell whose value is now "0" and its right neighbor (middle+1) value is "1" the neighborhood situation is either 001 or 101. Thus by step 13 when Rule 226 (see table 8 and table 9) is applied to the (middle) cell it will update its own state from 0 to 1. Thus by steps 14 and 15 of the

algorithm at least one more evolution takes place, which tells that, the pre-processing phase is not yet over and contradicts the starting assumption.

(C) Time complexity of 1-D DCT algorithm

The worst case time complexity of the above algorithm is clearly O(n) because maximum $(n-1)$ sequential evolutions are required in the pre-processing phase to get an intermediate configuration, after which another evolution is required in the decision phase.

(D) Experimental result

The step by step configurations obtained on using the 1-D DCT algorithm for an initial 1-D lattice of length 8 is shown in fig-8. Similarly, fig-9 and fig-10 demonstrate the outputs obtained by the 1-D DCT algorithm by considering a different initial 1-D CA configuration of length 8. In these cases only the intermediate output is shown based on which a decision can be made.

Consider another intermediate configuration as shown in fig-11 which is the rotational symmetry of fig-7 where 1's, if any, in the initial CA configuration are translated from left to right instead of right to left.

1 0 1 0 0 1 0 1 —*Applying 1−D DCT Algorithm*→ 1 1 1 1 0 0 0 0

FIGURE 9

Shows the output of the pre-processing phase of 1-D DCT algorithm. As the middle two bits are "10" so by step 31 and 32 the decision is that both 0's and 1's are of equal density.

1 0 0 0 0 1 0 1 —*Applying 1−D DCT Algorithm*→ 1 1 1 0 0 0 0 0

FIGURE 10

Shows the output of the pre-processing phase of 1-D DCT algorithm. As the middle two bits are "00" so by step 28 and 29 of the algorithm, Rule 0 is applied and the algorithm will print the CA is dense in 0's.

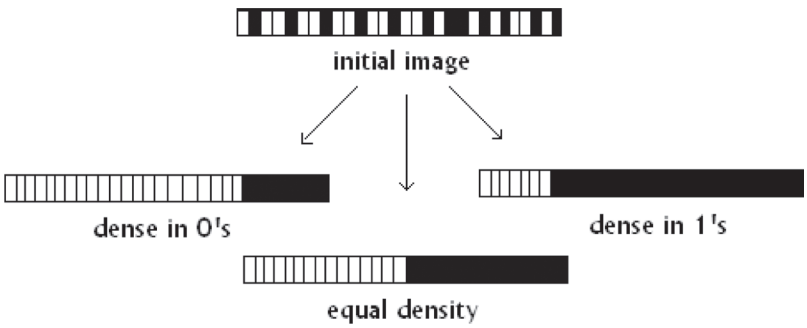

FIGURE 11

Possible intermediate outputs for taking decision in 1-D DCT algorithm.

Like fig-7 and table 5, for fig-11, another list of rules can be found out which can also be used in the pre-processing phase of the proposed 1-D DCT algorithm. These rules can be similarly obtained as before and the complete list is given in table 15. Here 1-D CA Rule 184 can be obtained for the cells x_2 to x_{n-1} as opposed to that of the CA Rule 226 earlier used for fig-7.

Table 11 and table 12 shows 16 CA rules obtained by fixing the left neighbor of x_1 either 0 or 1 and the complete list of these rules in Wolfram's naming scheme is given in column (3) of table 14. Similarly, the rules applicable for the boundary cell x_n can be found out by considering its right neighbor value either 0 or 1 and are shown in table 13 and table 14 and the complete list of these rules are given in column (5) and (6) of table 15.

On comparision with the works of others it is found that, in different boundary conditions such as null, periodic, adiabatic etc. there are many non-uni-

XYZ	111	110	101	100	011	010	001	000
Requirement	111	101	011	010	011	001	001	000
Rule 184	1	0	1	1	1	0	0	0

TABLE 10
Shows the CA rule 184 obtained by considering translation as well as density preservation and is the rotational symmetry of rule 226.

$0x_1x_2$	111	110	101	100	011	010	001	000
Requirement	-	-	-	-	011	001	001	000
16 Rules	0/1	0/1	0/1	0/1	1	0	0	0

TABLE 11
Shows the CA rules that can be used for the boundary cell x_1 if its right neighbour value is 0.

$1x_1x_2$	111	110	101	100	011	010	001	000
Requirement	111	101	101	100	-	-	-	-
16 Rules	1	0	0	0	0/1	0/1	0/1	0/1

TABLE 12
Shows the CA rules that can be used for the boundary cell x_1 if its right neighbour value is 1.

$x_{n-1}\,x_n0$	111	110	101	100	011	010	001	000
Requirement	-	110	-	010	-	010	-	000
16 Rules	0/1	1	0/1	1	0/1	1	0/1	0

TABLE 13
Shows the CA rules that can be used for the boundary cell x_n if its right neighbour value is 0.

$x_{n-1}\, x_n 1$	111	110	101	100	011	010	001	000
Requirement	111	-	011	-	011	-	001	-
16 Rules	1	0/1	1	0/1	1	0/1	0	0/1

TABLE 14
Shows the CA rules that can be used for the boundary cell x_n if its right neighbour value is 1.

(1) Cells	(2) If the left neighbour of $x_1 = 0$ then apply any rule to x_1	(3) If the left neighbour of $x_1 = 1$ then apply any rule to x_1	(4) Rules for all other cells x_2 to x_{n-1}	(5) If the right neighbour of $x_n = 0$ then apply any rule to x_n	(6) If the right neighbour of $x_n = 1$ then apply any rule to x_n
Rules	8, 24, 40, 56, 72, 88, 104, 120, 136, 152, 168, 184, 200, 216, 232, 248	128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143	184	84, 86, 92, 94, 116, 118, 124, 126, 212, 214, 220, 222, 244, 246, 252, 254	168, 169, 172, 173, 184, 185, 188, 189, 232, 233, 236, 237, 248, 249, 252, 253

TABLE 15
Rules used to solve 1-D DCT problem.

form rules (table 5 and table 15), which exactly solve the 1-D DCT problem as opposed to the uniform rules obtained by Capcarrere et al. [5]. For example, in our solution as given in table 15, Capcarrere's uniform rule 184 under null boundary condition, can not be applied to the right most cell x_n. This is because in null boundary condition, the Rule 184 shifts the 1's to the right in a CA configuration. If it is applied to the right most cell of the configuration containing 1 under null boundary condition then in the next configuration the number of 1's will be less than the number of 1's in the preceding configuration. Thus the density preservation condition is violated. In our approach the constraint of preserving the density does not allow the application of Rule 184 on the right most cell of any arbitrary CA configuration under null boundary condition. The rules shown in column (5) of table 15 when applied to the right most cell of the configuration along with Rule 184 at other cells preserve the density.

As mentioned in proposition 1 by Fuks [14] that repeated application of two uniform rules 184 and 232 in sequence gives the 1-D DCT solution. On the other hand, the algorithm and therefore the set of non-uniform rules obtained by us are quite different.

5.2 General 1-D DCT Solution

The 1-D DCT algorithm discussed above can be slightly modified so that the general 1-D DCT problem for arbitrary critical density ρ_c for $0 < c \leq 1$ can be solved. After the pre-processing phase is over i.e. gathering the group of 1's in the left corner of the 1-D row of cells and if the length of the CA n is a multiple of $1/c$ in the proposed 1-D DCT algorithm, then by looking at the $(nc)^{th}$ position of the vector $(x_1, x_2, x_3, ..., x_{n-1}, x_n)$ either 0 or 1, one can take the decision about their densities. For example, if c = 1/3 and say $n = 15$ which is a multiple of 3, then finding the value of $x_{n/3} = x_5 = 0$ or 1, one can take the decision of densities. That is, if $x_{n/3} = x_5 = 0$ then number of 1's are definitely less than $(1/3)^{rd}$ and the number of 0's are more than $(2/3)^{rd}$ of the total population. On the other hand, if $x_{n/3} = x_5 = 1$ then the decision is that, atleast (n/3) number of 1's are present in the total population. Thus, the value c acts as a scale of measurement for taking the decision to solve the general DCT problem.

If n is not a multiple of $1/c$ then density decision can be made by looking two integral cell values at positions $\lceil nc \rceil$ and $\lceil nc \rceil + 1$ of the vector $(x_1, x_2, x_3, ..., x_{n-1}, x_n)$ either 00 or 10 or 11. After the pre-processing phase is over as all 1's are moved towards the left corner of the configuration, so the possibility of 01 will never arise. So, in the decision phase,

a) If the values in the above said positions are 00 then the density of 1 is less than $1/c^{th}$ of the total population.
b) If the values in the above said positions are 10 then the number of 1's are exactly equal to $\lceil nc \rceil$ and the number of 0's are equal to $(n - \lceil nc \rceil)$.
c) If the values in the above said positions are 11 then the density of 1 is more than $1/c^{th}$ of the total population.

One such example for the general density classification problem of density ρ_c where c = 1/3 is the electrolysis process of H_2O that is the molecular formula of Water. During the process of electrolysis $(2/3)^{rd}$ Hydrogen molecules and $(1/3)^{rd}$ Oxygen molecules are stored in anode and cathode respectively. Thus the pre-processing phase embedded in the definition is a natural way to solve a general 1-D DCT problem. That is two corners of the initial 1-D array can be thought of as two electrodes as cathode and anode and the charge particles –ve and +ve charges (say 1's and 0's respectively), are attracted by their corresponding electrodes so that, all the +ve ions are stored in the anode and the –ve charges are stored in the cathode. Similar analogy can be made for mag-

netism considering two corners as two poles North Pole and South Pole of a magnet. Above all the main objective of solving the DCT problem is to better understand the limitations of CA.

6. SOLUTION OF 2-D DCT USING PROGRAMMABLE CELLULAR AUTOMATA (PCA)

6.1 Algorithm to solve 2-D Density Classification Problem

In this section an algorithm is proposed using PCA rules to solve 2-D DCT problem. The nine fundamental rules given in fig-3 are used to achieve both translations of images as well as the density preservation throughout the CA evolution. This algorithm is also a two-phase procedure similar to 1-D DCT algorithm given in section 5. The technique used in the pre-processing phase is a slight modification of Sweeper's algorithm reported in [10]; the algorithm is so named because it is very similar to the way a sweeper sweeps and puts the dust in one corner of a room. The cells which are in the boundary region towards the direction of translation are taken as the points of destination and on repeated application of fundamental 2-D CA rules, all 1's comes closer to these points. For example in the pre-processing phase to achieve a configuration like fig-7 in a $(n \times n)$ matrix, the boundary cells lie in the cell position $(i,1)$ or $(1,j)$ for all $1 \leq i, j \leq n$ can be considered as the points of destination.

Let the variable K be a fundamental 2-D CA rule such as 2, 4, 8, 16 and K^T be its corresponding transpose rule like 32, 64, 128, 256 respectively. From table 3, Rule 1 can be used as density preservation where as other 8 rules can be used for translation of 1's and 0's in different directions but the directions of translation for Rule K and K^T are opposite to each other. It can

64	128	256
32	1	2
16	8	4

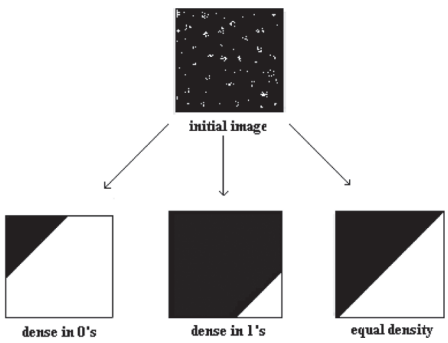

FIGURE 12
Shows an intermediate configuration for 2-D DCT solution using PCA rules 1, 2, 4, 8, 16, 32, 64, 128 and 256.

be noticed that Rules (1, K, K^T) in fig-12 always lie in 1-D. So, the combined effect of these three fundamental rules (1, K, K^T) can be iteratively applied to any 1-D configuration to get an image like fig-7. As any 2-D matrix can be divided into several 1-D arrays either in rows, columns, or in diagonals, the pre-processing phase of 2-D DCT algorithm as shown in fig-12 can be visualized as several pre-processing figures like fig-2 used for 1-D DCT. A general code for these three fundamental rules (1, K, K^T) applied to a 2-D matrix of size ($n \times n$) is shown below. This code is valid for all boundary conditions; null, periodic, reflexive, adiabatic etc.

```
/* Input: A 2-D binary image of size (n × n) just like 1-D, 3-neighborhood
CA, where Y be the cell under consideration whose left and right neighbors
are X and Z respectively*/
ApplyRules (1, K, K^T)
 1.  {
// Boundary cells in the translation direction of Rule K
 2.  if (Y = 1) then
 3.      ApplyRule 1
 4.   else if (Y = 0) then
 5.      ApplyRule K
// All Other Cells
 6.      if (the 3-neighbors in the translation direction of Rule K is 110)
     then
 7.      ApplyRule 1                          /*Density preservation*/
 8.  else if (the 3-neighbors in the translation direction of Rule K is 011)
     then
 9.   ApplyRule K^T      /*0's moves in the translation direction of Rule K^T*/
10.  else
11.   ApplyRule K       /*1's moves in the translation direction of Rule K*/
12.  }
```

An example is shown in fig-13, where the rules 1, 4, and 64 are applied to an arbitrary 2-D binary image of size (6×6). Here $K = 4$ and so $K^T = 64$ and the translation direction of Rule 4 is shown in fig-13 (a); that is diagonally from right-bottom to left-top. For boundary cells in the translation direction of Rule 4 only two border cells depending on the values of Y as 0 or 1, either Rule 1 or Rule 4 are applied and the output is shown in fig -13 (b). For other cells, either Rule 1 or Rule 4 or Rule 64 is applied by looking the three neighborhood values "XYZ" in the translation direction of Rule 4. All rules are applied simultaneously to all the cells of the (6×6) lattice and the final output of the function *ApplyRules* (1,4,64) is shown in fig-13 (c).

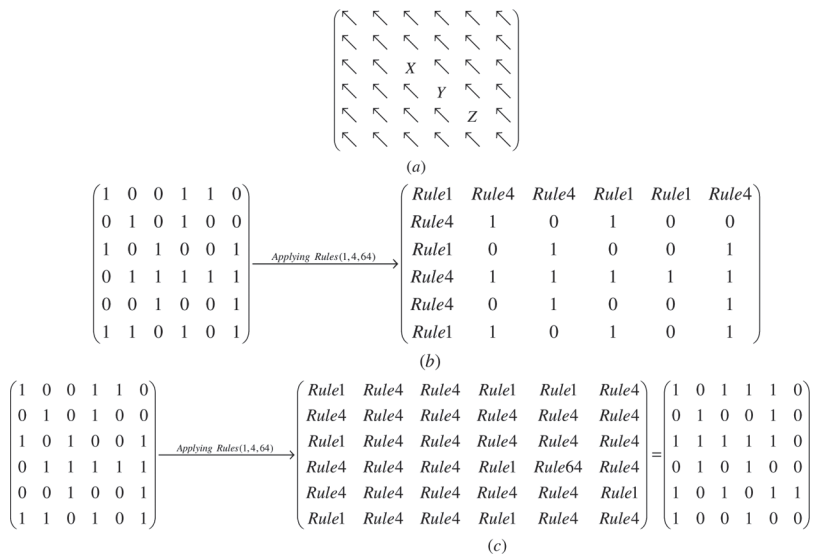

FIGURE 13
(a) Shows the translation direction of Rule 4 and a particular cell Y, its left neighbor X and its right neighbor Z. (b) shows an intermediate output, after step 2 of the algorithm is over. Here either Rule 1 or Rule 4 is applied only to two boundary cells and other cell values are unchanged. (c) Shows the final output of the function *ApplyRules* (1,4,64) to an arbitrary 2-D image of size (6×6) and as a result 1's are translated towards the translation direction of Rule 4.

The algorithm using the above functions for solving 2-D DCT problem corresponding to the intermediate configuration as shown in fig-12 is as follows.

(A) Algorithm to solve 2-D DCT

*/*Pre-processing Phase: Reaching to an intermediate configuration like fig-12*/*

1. *A 2-D binary image of size (n × n) is taken as the input and just like 1-D, 3-neighborhood CA, where Y be the cell under consideration whose left and right neighbors are X and Z respectively*/*
2. *for* i ← 1 *to n*
3. }
4. *ApplyRules* (1,4,164)
5. *ApplyRules* (1,16,256)
6. }
7. *for* i ← 1 *to n*
8. {
9. *ApplyRules* (1,16,256)
10. *ApplyRules* (1,8,128)

```
11. ApplyRules (1,2,32)
12. }
13. for i ← 1 to n
14. {
15. ApplyRules (1,16,256)       /*Rules only applied to the decision line*/
16. }

/*Decision Phase: For deciding the density of the configuration*/
17. if (n is odd)
18. {
19.   if (the bit at position ((n + 1)/2, (n + 1)/2) = 1)then
20.      Apply Rule (2^512 − 1) uniformly to all the cells
21.      Print "the 2-D lattice is dense in 1's"
22.   else if (the bit at position at ((n + 1)/2, (n + 1)/2) = 0) then
23.       Apply Rule 0 uniformly to all the cells
24.       Print "the 2-D lattice is dense in 0's"
25. }
26. else if (n is even)
27. {
28.   if (the bit at positions ((n/2) + 1, n/2) = 0) and (n/2), (n/2) + 1)= 0 then
29.      Apply Rule 0 uniformly to all the cells
30.      Print "the 2-D lattice is dense in 0's"
31. else if (the bit at positions ((n/2) + 1, n/2) = 1) and (n/2), (n/2) + 1)= 0)
    then
32.      Print "equal densities in 0's and 1's"
33. else if (the the bit at positions ((n/2) + 1, n/2) = 1) and (n/2, (n/2) + 1)= 1
    then
34.       Apply Rule (2^512 − 1) uniformly to all the cells
35.       Print "the 2-D lattice is dense in 1's"
36. }
```

(B) Explanation of the algorithm

The proposed algorithm for both density preservation and translation of 1's towards one half of the region uses the following logic.

If the boundary cells towards the translation direction are 0 then only the CA rule is applied. For example in step 4 of the pre-processing phase when the function *ApplyRules* $(1,K,K^T)$ is called for Rules 1, 4 and 64 then Rule 4 is applied to the boundary cells (the cell position $(i, 1)$ or $(1, j)$ for all $1 \leq i, j \leq n$), where the cell values are 0. In addition to this, Rule 4 is also applied to all other cells except the cases where the left neighbor X, the cell Y and the right neighbor Z is "110" or "011". In these two cases, if the neighborhood value XYZ is "110" then Rule 1 is applied whereas, in case of "011" Rule 64

(transpose of Rule 4) is applied. All these situations in 2-D nine-neighborhood CA for Rule 1, Rule 4 and Rule 64 are shown in fig-14.

In the pre-processing phase, 2-D nine-neighborhoods can be considered as four 1-D 3-neighborhoods in different orientations as shown in fig-15. Thus the rules (32, 1, 2), (64, 1, 4), (128, 1, 8), and (16, 1, 256) are iteratively applied to a 2-D binary image as shown from step 2 to step 12 of the algorithm to get an intermediate configuration like fig-12. In steps 13 to 16 the function *ApplyRules* (1,16,256) is called and the rules are applied only to the cells lies in the diagonal line (decision line) after which a decision can be made in the decision phase like 1-D solution. Different intermediate outputs of this algorithm given in fig-16, fig-17, and fig-18 self explains how this sweeper's algorithm works.

It can be noted that, a 2-D lattice is a combination of several rows (or column or diagonal) of 1-D lattices so, the above 1-D DCT algorithm can be applied to all the rows (or column or diagonal) to obtain a pattern as shown in fig-19 and fig-20. So, the pseudocode of the function *ApplyRules* (1,K,K^T) is equivalent to the 1-D CA rules given in table 5 and in table 15. Therefore, instead of using many 2-D nine-neighborhood CA rules (1, 16, 256), (1, 64, 4), (1, 128, 8), and (1, 16, 256) in the pre-processing phase of the above 2-D DCT algorithm, a single 1-D three-neighborhood CA Rule 226 can be applied to each cell one after the another except, two boundary cells present at first row and first column positions. Here the three neighborhoods XYZ are choosen in the translation direction of *Rule K*. And to the boundary cells any one of the 16 possible rules given in table 5 and table 15 is applied. Finally, a trivial rule: either Rule 0 or Rule ($2^{512} - 1$) is applied to get the desired solution of 2-D DCT in the form of all 0's or all 1's.

X	**0/1**	**0/1**
0/1	**Y**	**0/1**
0/1	**0/1**	**Z**

(a)

1	**0/1**	**0/1**
0/1	**1**	**0/1**
0/1	**0/1**	**0**

(b)

0	**0/1**	**0/1**
0/1	**1**	**0/1**
0/1	**0/1**	**1**

(c)

FIGURE 14
(a) Shows 2-D nine- neighborhood CA and Rule 4 is applied to all possible nine- neighborhoods except the string XYZ is 110 or 011. In (b) Rule 1 is applied if XYZ is 110 and in (c) Rule 64 is applied if XYZ is 011.

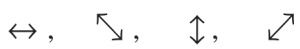

FIGURE 15
Shows 2-D nine- neighborhoods can be considered as 1-D three-neighborhood CA with different orientations.

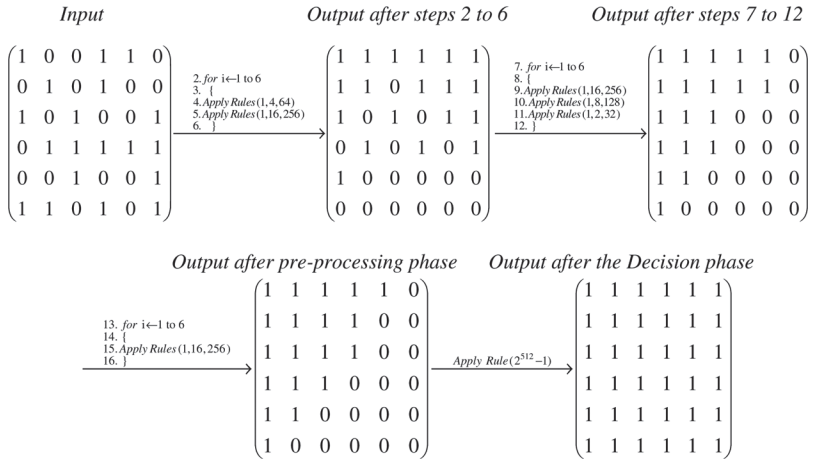

FIGURE 16
In the pre-processing phase output, the bits at cell position (4, 3) = 1 and (3, 4) = 1. So by step 34 of 2-D DCT algorithm the initial image is dense in 1's.

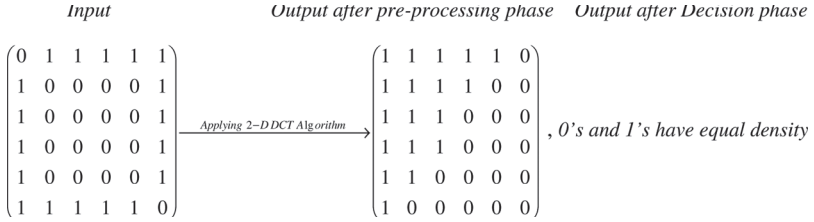

FIGURE 17
After the pre-processing phase, the bits at cell position (4, 3) = 1 and (3, 4) = 0. So by step 32 of 2-D DCT algorithm the density of 0's and 1's are same.

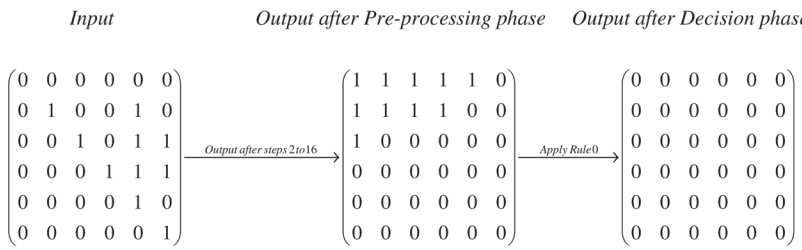

FIGURE 18
After the pre-processing phase, the bits at cell position (4, 3) = 0 and (3, 4) = 0. So by step 30 of 2-D DCT algorithm the initial image is dense in 0's.

(C) Time complexity of 2-D DCT algorithm

The time complexity of the above 2-D DCT algorithm using PCA is linear i.e. O(n) since, CA rules are applied parallel to each cell of the square matrix itera-

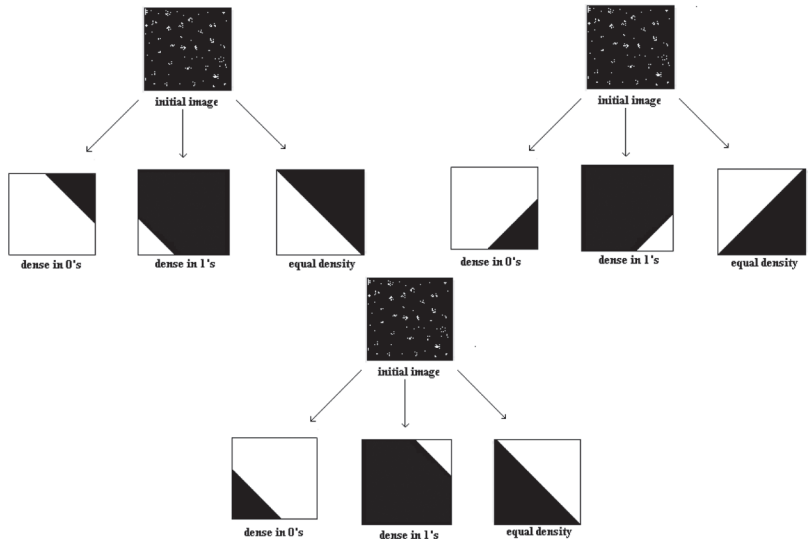

FIGURE 19
Shows other intermediate configurations that can be used in the pre-processing phase of the 2-D DCT solution.

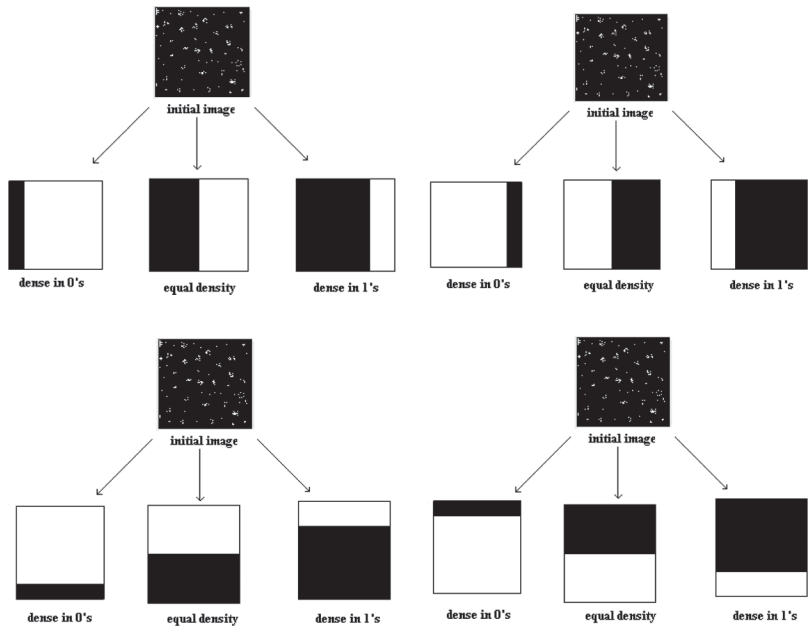

FIGURE 20
Shows other intermediate configurations that can be used in the pre-processing phase of the 2-D DCT solution.

tively for n times. So, the proposed algorithm is computationally better than the sequential algorithm time $O(n^2)$ which first counts the number of 1's and 0's in the initial configuration and then takes the decision of their densities.

(D) Experimental Results

The following figures demonstrate the three possible outputs for 2-D DCT algorithm by considering different initial configurations of size (6×6). In fig-16 it has been shown the output of different intermediate steps after which a decision is made in the decision phase.

The figures, fig-17 and fig-18 are the output after the pre-processing phase is over and it also shows the outputs of the decision phase.

Similar to 1-D, other 2-D images can be used as the output of the pre-processing phase as shown in fig-19 and fig-20; after which a decision of density can be made in the decision phase. Similar algorithms like the previous 2-D DCT algorithm can also be designed for these figures. It may be noted that for a square image of size $(n \times n)$ all these figures can able to solve the 2-D DCT problem. For an arbitrary rectangular image of size $(m \times n)$ only the figures given in fig-20 can be used as an output of the pre-processing phase because for a rectangular image a diagonal region for taking the decision of classification cannot be obtained.

On comparision with the works of Morales et al. [23]; Reynaga and Amthauer [27]; Oliveira and Siqueira [24] it is found that, there are many rules which exactly solve the 2-D DCT problems as opposed to the approximate solution in the literature.

6.2 2-D DCT Problems: Some Variations

In this section some variants of 2-D DCT problem are defined. Their applications and the solutions using PCA with the similar kind of logic are also discussed.

The conventional 2-D DCT problem is defined on a rectangular or a square shaped initial CA configuration. But it can also be defined on various symmetric images like triangular, circular, parallelogram etc. The PCA and the two-phase method used previously to solve the conventional 2-D DCT problem can be similarly used to solve such problems. It is also possible to define the 2-D DCT problem on an asymmetric initial CA configuration. But for its solution, the algorithm can be modified by dividing the region with a curve into two regions of equal area each, and then the fundamental rules are selected to sweep all the 1's to bring it to a single region after which a decision of classification can be made.

So far the DCT has been defined over a continuous image. Now another variant of DCT is to compute the density on a discontinuous image. The application of this problem might be, computing the density of literate and illiterate people of a place, computing the total density of male and female of

a country on knowing individual density of each state and so on. As exact number of literate and illiterate people in different regions are generally unknown apriori, so it may be a difficult task to solve this type of problem using standard CA. But, the pre-processing phase used in this paper can act as a scale of measurement to find the exact density in percentage and can solve this type of problem. In our future research we will be dealing these new kinds of DCT in detail.

7. CONCLUSION

Density Classification Task (DCT) is a fundamental problem in the domain of Cellular Automata (CA). In this paper, we have reviewed the literature on the problem of 1-D and 2-D DCT and its solutions using CA. Our method uses non-uniform CA for solving the problem of DCT. The approach adopted in our method involves two-phases namely, a pre-processing phase and a decision phase. In the pre-processing phase a set of non-uniform CA rules are applied iteratively to an initial CA configuration to reach at a state. Then in the decision phase, the decision about the densities of 0's and 1's are taken. The proposed DCT algorithm accommodates all the cases under different boundary conditions such as fixed, null, periodic, intermediate, adiabatic, reflexive etc. and it can also generalisable to the other kinds of boundaries. This is because, for two boundary positions, the non-uniform CA rules are calculated following the same principle of preserving the density of 0's and 1's as well as looking the translation direction of rules for which all 1's are grouped together so that observing the middle position(s), density decision can be obtained for both 1-D and 2-D lattices. The problems of DCT for arbitrary 2-D lattices of various shapes and sizes are introduced for the first time along with their solutions, and applications.

ACKNOWLEDGEMENT

The authors would like to thank Prof. Pedro P. B. de Oliveira, Universidade Presbiteriana Mackenzie, Brazil, for providing helpful comments and valuable criticisms on the original version of the manuscript, which have greatly improved the presentation of this paper.

REFERENCES

[1] Andre, D., Bennett III, F., Koza, J., 1996. Discovery by Genetic Programming of a Cellular Automata Rule that is Better than any Known Rule for the Majority Classification Problem. ln J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (eds.), Genetic Pro-

gramming 1996: Proceedings of the First Annual Conference, July 28-31, Stanford University, MIT Press, pp. 3–11.

[2] Bossomaier, T. R., Cranny, T. R., Schneider, D., 1999. A new paradigm for evolving cellular automata rule, in: Proc. Congress on Evolutionary Computation, Washington DC, pp. 169–176.

[3] Bossomaier, T., Punnett, L. S., Cranny, T., 2000. Basins of Attraction and the Density Classification Problem for Cellular Automata, Heudin, J.-C. (Ed.): VW 2000, LNAI 1834, Springer-Verlag Berlin Heidelberg, pp. 245–255.

[4] Busic, A., Fates, N., Mairesse, J., Marcovici, I., 2013. Density classification on infinite lattices and trees. Electronic Journal of Probability 18(51), pp. 1–22.

[5] Capcarrere, M. S., Sipper, M., Tommasini, M., 1996. Two-state, r=1 Cellular Automaton that Classifies Density. Physical Review Letters 77, pp. 4969–4971.

[6] Capcarrere, M. S., Sipper, M., 2001. Necessary conditions for density classification by cellular automata. Physical Review E, vol. 64, no. 3, pp. 036113/1–036113/4.

[7] Crutchfield, J. P., Mitchell, M., 1995. The evolution of emergent computation. PNAS, vol. 92 , no. 23, pp. 10742–10746.

[8] Chau, H. F. , Siu, L. W, .and Yan, K. K., 1999. One dimensional n-ary density classification using two cellular automaton rules, I*nt. J. Mod. Phys. c* **10**, 883, doi: 10.1142/s0129183199000681.

[9] Christopher, S. and Larry, B., 2009. Evolution of cellular automata with memory: The Density Classification Task, Biosystems, vol. 97, Issue 2, pp. 108–116.

[10] Choudhury, P. P., Nayak, B.K., Sahoo, S., Rath, S.P., 2008. Theory and Applications of Two-dimensional, Null-boundary, Nine-Neighborhood, Cellular Automata Linear rules, available in arXiv:0804.2346, cs.DM; cs.CC; cs.CV.

[11] de Oliveira, P. P. B., Bortot, J. C., and Oliveira, G. M. B., 2006. The best currently known class of dynamically equvalent cellular automata rules for density classification, Neurocomputing, 70: pp. 35–43.

[12] de Oliveira, P. P. B., 2013. Conceptual connections around density determination in cellular automata, Lecture Notes in Computer Science, 8155, pp. 1–14.

[13] Fates, N. A., 2013. Stochastic cellular automata solutions to the density classification problem- When randomness helps computing. Theory of Computing Systems, 53(2), pp. 223–242.

[14] Fuks, H., 1997. Solution of the Density Classification Problem with Two Cellular Automata Rules. Phys. Rev.: E55, R2081-R2084, Issue 3. Also available in arXiv:comp-gas/9703001v17.

[15] Gabriele, A. R., 2005. The Density Classification Problem for Multi-states Cellular Automata. Capcarrere, M. et al. (Eds.): ECAL 2005, LNAI 3630, Springer-Verlag Berlin Heidelberg., pp. 443–452.

[16] Gacs, P., Kurdyumov, G. L. and Levin, L. A., 1978. One-dimensional uniform arrays that wash out finite islands. Probl. Perdachi. Inform., 14: pp. 92–98.

[17] Jullie, H., Pollack, J. B. 1998. Coevolving the ideal trainer: Application to the discovery of Cellular Automata Rules. Genetic programming *1998*: Proceeding of the Third Annual Conference, San Francisco, CA, Morgan Kaufmann pp. 519–527.

[18] Land, M., Belew, R. K., 1995. No Perfect Two-State Cellular Automata For Density Classification Exists. Physical Review Letters 74(25), pp. 5148–5150.

[19] Maiti, N., Munshi, S., Chaudhuri, P. P., 2006. An Analytical formulation for Cellular Automata (CA) based Solution of Density Classification task (DCT). ACRI 2006, LNCS 4173, pp. 147–156.

[20] Mitchell, M., Hraber, P., Crutchfield, J., 1993. Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations. Complex Systems, 7: pp. 89–130.

[21] Mitchell, M., Crutchfield, P., Hraber, P. T., 1994. Evolving cellular automata to perform computations. Mechanisms and Impediment. Physica D 75: pp. 361–391.

[22] Mitchell, M., Crutchfield, J. P., Das, R., 1998. Evolving cellular automata to perform computations, in: Back, T., Fogel, D., Michalewicz, Z. (Eds.), Handbook of Evolutionary Computation. Oxford: Oxford University Press.

[23] Morales, F. J., Crutchfield, J. P., Mitchell, M., 2001. Evolving two-dimensional cellular automata to perform density classification: A report on work in progress. Parallel Computing, 27: pp. 571–585.

[24] Oliveira, G. M. B. and Siqueira, S. R. C., 2006. Parameter Characterization of Two-Dimensional Cellular Automata Rule Space. Physica D, 217(1): pp. 1–6.

[25] Oliveira, G. M. B., Martins L. G. A., de Carvalho, L. B. and Fynn, E., 2009. Some Investigations About Synchronization and Density Classification Tasks in One-dimensional and Two-dimensional Cellular Automata Rule Spaces. Electronic Notes in Theoritical Computer Science, vol. 252: pp. 121–142.

[26] Packard, N. H., 1988. Adaptation toward the edge of chaos. In: Dynamic Patterns in Complex Systems, Kelso, J. A. S., Mandell, A. J., Shlesinger, M. F. (Eds.), World Scientific, pp. 293–301.

[27] Reynaga, R., Amthauer, E., 2003, Two-dimensional cellular automata of radius one for density classification task $\rho = 1/2$", Pattern recognition Letters 24, pp. 2849–2856.

[28] Sahoo, S., Choudhury, P. P., 2008. Issues on drawing the State Transition Diagram for arbitrary Cellular Automata, arXiv: 0811.1513, nlin.CG.

[29] Sipper, M., Capcarrere, M. S., Ronald, E., 1998. A simple cellular automata that solves the density and ordering problems. International Journal of Modern Physics C 97, pp. 899–902.

[30] von Neumann, J., 1966. The Theory of Self-Reproducing Automata, Burks, A.W. (Ed.), Univ. of Illinois Press, Urbana and London.

[31] Wegener, I., 1987. The complexity of Boolean Functions, Wiley, New York.

[32] Wolfram, S., 1986. Theory and Application of Cellular Automata. World Scientific.

[33] Wolfram, S., 2002. A New Kind of Science, Wolfram Publisher.

[34] Wolz, D. and de Oliveira, P. P. B., 2008. Very effective evolutionary techniques for searching cellular automata rule spaces. Journal of Cellular Automata, 3: pp. 289–312.