symmetry

MDPI

Article

Evolution of Hybrid Cellular Automata for Density Classification Problem

Petre Anghelescu

Department of Electronics, Computers and Electrical Engineering, National University of Science and Technology POLITEHNICA Bucharest, 060042 Bucharest, Romania; petre.anghelescu@upb.ro

Abstract: This paper describes a solution for the image density classification problem (DCP) using an entirely distributed system with only local processing of information named cellular automata (CA). The proposed solution uses two cellular automata's features, density conserving and translation of the information stored in the cellular automata's cells through the lattice, in order to obtain the solution for the density classification problem. The motivation for choosing a bio-inspired technique based on CA for solving the DCP is to investigate the principles of self-organizing decentralized computation and to assess the capabilities of CA to achieve such computation, which is applicable to many real-world decentralized problems that require a decision to be taken by majority voting, such as multi-agent holonic systems, collaborative robots, drones' fleet, image analysis, traffic optimization, forming and then separating clusters with different values. The entire application is coded using the C# programming language, and the obtained results and comparisons between different cellular automata configurations are also discussed in this research.

Keywords: cellular automata; density classification problem; dynamical systems; bio-inspired systems; majority problem; clustering algorithms

Citation: Anghelescu, P. Evolution of Hybrid Cellular Automata for Density Classification Problem. *Symmetry* **2024**, *16*, 599. https://doi.org/10.3390/sym16050599

Academic Editor: Wiesław Leonski

Received: 5 April 2024
Revised: 25 April 2024
Accepted: 9 May 2024
Published: 12 May 2024

1. Introduction

Initially introduced by Packard, the Density Classification Problem (DCP), also referred to as the Density Classification Task (DCT), may seem like a straightforward counting problem [1]. Of course, for an electronic system with a processor, controller, or memory, this issue is insignificant. On the contrary, DCP is a significant issue for cellular automata (CA) systems which are characterized by arbitrary size, different neighborhood, any dimension, and without a central processing unit (decentralized control and massive parallelism), because they are based solely on local interactions between cells using evolution rules [2]. The DCP requires some form of global coordination, even though cells with a certain distance between them cannot directly interact. This means that distant cells must somehow influence each other's behavior to achieve the desired outcome. As a result, no cell within the CA can establish immediate global communication, and the macroscopic evolution of the CA is actually a reflection of the microscopic evolution.

The DCP is therefore of great interest to many researchers as a means of testing and measuring the computational power of different computing paradigms, including CA. Although the DCP seems like a simple counting problem, it is surprisingly challenging for CA to solve. This difficulty stems from the limitations of CA: they lack the memory to keep track of a running total and the ability to perceive the entire system at once, making it difficult to accurately assess the relative density of 0 s and 1 s. Due to this challenge, the DCP enables us to evaluate the computational capabilities of CA, particularly their ability to achieve complex global behavior based only on applying local evolution rules. These systems reveal data processing capabilities on a global scale, capabilities that are not explicitly represented in their elementary cells or in their local cell neighborhood interconnections [3,4].

As an example, in the two-state variant of the CA, the objective is to obtain a final CA configuration in which all the cells are mapped to either 0′s or 1′s. This outcome should depend on the initial image configuration: if, initially, there were more 0 s, then all cells should become 0, and vice versa for a majority of 1 s. This process essentially classifies the binary image string based on the density (frequency) of 0 s and 1 s. If we consider the CA as an example of a multicellular system consisting of a large number of cells, we could say that after a predetermined number of generations using different evolution rules to solve the DCP problem, the CA evolution must have reached a dead end. This means that the DCP has been resolved, and the system has been able to classify the initial configuration of the image. Local regions with similar densities cluster and grow over time, ultimately determining the final configuration of the CA used for classification.

From the research reported in paper [5], it appears that it is not possible to create a one-dimensional two-state CA that categorizes binary strings based on their densities of 1 s and 0 s. Based on this observation, the research reoriented to the search for imperfect alternatives that could categorize the initial configuration space.

Unfortunately, finding a CA rule that effectively solves a computational problem such as the DCP remains a significant hurdle even after the problem has been defined. Three main approaches could be considered relevant: direct programming—this is labor-intensive and requires a deep understanding of CA dynamics [6]. Designing rules manually is a complex process that often leads to trial-and-error experimentation until their behavior achieves the desired objective; exhaustive search—investigating every possible CA rule for a specific behavior is computationally infeasible. The large number of candidate rules (the high cardinality of the rule space) makes this approach unsuitable for intricate tasks; heuristic search—a workable alternative which involves the use of search and optimization algorithms, notably emphasizing evolutionary computation techniques [7–11]. These approaches were the key to the discovery of a significant number of solutions, especially with the introduction of the symmetry property [12–14], with the introduction of the conserving property [15,16], asynchronous updating methods [17], and even firing-squad synchronization that aims to establish an evolution rule whereby all cells within a given region enter into a special state after an identical number of steps [18] (p. 1035). Another interesting approach is presented in [19], where symmetry and conserving properties are achieved using CA with neighborhood dimension r = 4.

Moreover, it was also shown in [20] that a solution to the DCP does indeed exist, with a different result from the one found by Packard. Additionally, research carried out in paper [21] showed that a rule addressing the DCP in a one-dimensional CA must fulfill two clear requirements:

1. Density conserving—the initial density, corresponding to the input configuration of the system, should be conserved over time.
2. Balanced rule—the rule table should demonstrate a density of 0.5 and maintain uniformity across the configuration space.

An interesting approach is reported in [22], where a genetic algorithm (GA) evolves a one-dimensional CA to solve the DCP. This paper introduces a multi-states CA for the first time and addresses the DCP. Their results obtained suggest a notable similarity between elementary CA and multi-states ones. Another interesting approach is outlined in [23], wherein a combination of two elementary rules known as the "traffic rule" (also referred to as rule 184) and the "majority rule" (also known as rule 232) is employed to tackle the DCP. Their primary finding, utilizing the preimage counting method for the order parameter, suggests that the DCP can be effectively addressed by employing this pair of rules 184 and 232.

An interesting method is described in [24–26], showcasing an algorithm capable of reversing the operations of the CA. This entails identifying a CA rule wherein the configuration space is neatly partitioned into two separated basins of attraction. One basin encompasses rules with a density surpassing the critical density, c_d, converging towards a state comprising entirely of 1 s. The other basin comprises all remaining configurations, converging towards a state consisting solely of 0 s.

In connection with the relevant CA-based DCP techniques, the main aim of this work is to solve the DCP using two essential features of the hybrid CA: density conserving and translation of the information stored in the cells of the CA through the lattice. The fulfilment of these two conditions, translation of the information across the lattice and density conserving, was accomplished using a dynamical system named CA and using a three-stage algorithm. Firstly, we generate an elementary image of a two-dimensional CA in which each cell is surrounded by 8 neighbors, similar to digital images. Different combinations of evolution rules are applied to different cells of the CA, and the problem of cells localized at the end of the lattice, caused by the fact that there are no other neighbors, was solved by using null boundary conditions, making the solution more effective. Secondly, in stage 2 of the algorithm, we investigate the comportment of the nine possible elementary evolution rules that shift the information in different directions within the CA. We have further analyzed all combinations of each of the three elementary CA rules and selected only those combinations of three rules that allow us to shift the information across the lattice in the direction we need in order to create some clusters in the CA. Thirdly, the density classification decision is made in function of the image dimension, odd or even length. The simulation results indicate a perfect concordance between the theoretical approach and the practical results obtained. Furthermore, the proposed DCP solution based on the hybrid CA method presented here is efficient because it focuses only on a specific set of CA evolution rules, thus streamlining the design process and making it simpler and more efficient. In contrast to existing methods, where increasing the radius leads to a potential explosion of the rule spaces, this approach is characterized by minimizing the number of potential solutions and thus increasing the performance. This not only facilitates parallelization using instruction optimization algorithms [27] and through hardware implementation, such as with Field Programmable Gate Arrays (FPGA) devices, but also enables generalized classification, dimensional scalability, and maintains architectural simplicity.

The contributions stemming from this research, taking into account also the explanation given in the previous paragraph, can be summarized as follows:

1. An efficient hybrid CA-based solution for DCP that can be used in parallel and distributed systems and applied to many real-world decentralized problems that require a decision to be taken by separating clusters with different values is proposed.
2. The technical details and the arguments that illustrate the capability of hybrid CA to combine the two necessary features, density conserving and translation of the information stored in the cells across the lattice, in order to obtain the solution to the DCP are provided.
3. The CA-based method for DCP developed here is a general one and is relatively simple and efficient for implementation, as it can be scaled to a larger or a smaller number of cells.
4. Experimental results, showcased across different working scenarios, confirm and offer relevant insights into how to improve the exploration of the CA search space and also the ways of creating solutions in the CA context.

The paper is structured as follows. Section 1 provides the main aim of this work and a short overview of the DCP-relevant related research. Section 2 is focused on the fundamentals of the problem statement and the design of the hybrid CA-based algorithm for the DCP. Here, we describe the solution to the DCP, which is solved using the two-dimensional hybrid CA structures. Section 3 shows the implementation of the proposed hybrid CA to solve the DCP. Section 4 of this paper presents, with examples, the experimental results obtained on the hybrid CA and the image DCP and evaluates the quality of the results obtained. In Section 5, we assess our approach in comparison to established DCP solving techniques, highlighting the main key potential benefits. Finally, Section 6 discusses the conclusions and perspectives of this work.

2. Materials and Methods

In this section, we describe in detail our approach to solving the DCP using an entirely distributed system with only local processing of information, named CA.

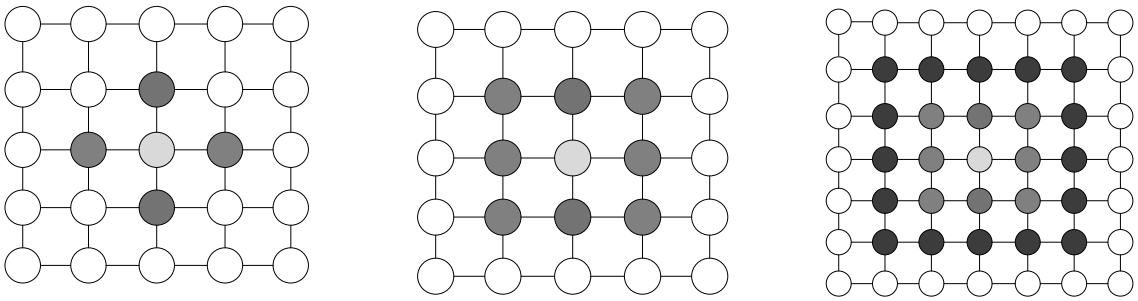
2.1. Fundamentals of the Problem Statement

CA are mathematical models that illustrate physical systems where time and space are discrete, and connections exist only between adjacent cells forming the automaton [18]. Of course, theoretically, each cell can assume a vast number of states [28], but the greater the number of possible states, the more difficult it is to analyze the behavior of the system. In the proposed decision system, however, each cell comprising the automaton can practically hold two states: 0 or 1 (corresponding to binary images) [29]. CAs exhibit significant parallelism as all cells in the automaton can be updated simultaneously within a single time clock. Interactions among cells are strictly local, occurring within the neighborhood of cells [4]. The configuration of neighborhoods is determined by both the topology of the underlying lattice of cells [30] and a specified radius. Common models of neighborhood interactions include:

1. Von Neumann model, radius r = 1, where, for establishing the next state of a central cell, three neighboring cells are used in the one-dimensional CA case (Figure 1a) and five neighboring cells, formed by the central cell and its four adjoining horizontal and vertical neighbors, in the two-dimensional CA case (Figure 2a).
2. Moore model, radius r = 1, in which three neighboring cells are used in the one-dimensional CA case and nine neighboring cells, formed by the central cell and its eight adjoining neighbors, including diagonals, in the two-dimensional CA case (Figure 2b).
3. Extended Moore model, radius r = 2, in which five neighboring cells are used in the one-dimensional CA case (Figure 1b) and twenty-five neighboring cells, formed by the central cell and its twenty-four adjoining neighbors, including diagonals, in the two-dimensional CA case (Figure 2c).

(**a**) (**b**)

Figure 1. (**a**) One-dimensional CA with radius = 1; (**b**) One-dimensional CA with radius = 2.



(**a**) (**b**) (**c**)

Figure 2. (**a**) Two-dimensional von Neumann model; (**b**) Two-dimensional Moore model; (**c**) Two-dimensional Extended Moore model.

The CA evolve at discrete time steps, and the next state of each cell that is part of the CA depends only on its current state and on the states of its neighbors. As a result, each cell is based on a storage circuit (for example, a D flip-flop) and a combinational logic circuit

(CL) that set the evolution functions of the next state of the cell. A general image of a cell that includes different models of neighborhood interactions is shown in Figure 3.

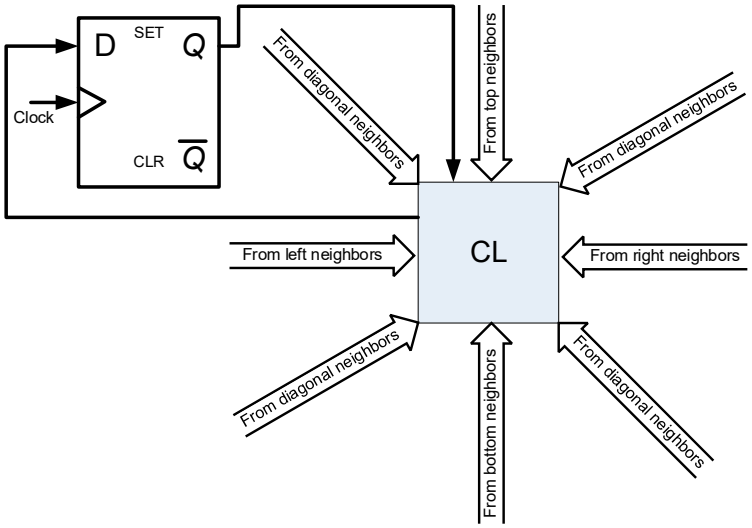


Figure 3. General view of a cell that is part of the CA.

The next-state function implemented (see Figure 3), in CL circuit block, which describes an evolution rule for a cell of CA, can be expressed in the following ways:

1. In the case of one-dimensional CA with radius r = 1:

$$C_i(t+1) = f[C_{i-1}(t), \mathbf{C_i(t)}, C_{i+1}(t)] \quad (1)$$

2. In the case of one-dimensional CA with radius r = 2:

$$C_i(t+1) = f[C_{i-2}(t), C_{i-1}(t), \mathbf{C_i(t)}, C_{i+1}(t), C_{i+2}(t)] \quad (2)$$

3. In the case of two-dimensional CA with radius r = 1 and von Neumann model:

$$C_{i,j}(t+1) = f[C_{i,j-1}(t), C_{i-1,j}(t), \mathbf{C_{i,j}(t)}, C_{i,j+1}(t), C_{i+1,j}(t)] \quad (3)$$

4. In the case of two-dimensional CA with radius r = 1 and Moore model:

$$C_{i,j}(t+1) = f[C_{i,j-1}(t), C_{i-1,j}(t), C_{i,j+1}(t), C_{i+1,j}(t), \mathbf{C_{i,j}(t)}, C_{i-1,j-1}(t), C_{i-1,j+1}(t), C_{i+1,j-1}(t), C_{i+1,j+1}(t)] \quad (4)$$

5. In the case of two-dimensional CA with radius r = 2 and Extended Moore model:

$$\begin{aligned} C_{i,j}(t+1) = f[&C_{i,j-1}(t), C_{i-1,j}(t), C_{i,j+1}(t), C_{i+1,j}(t), C_{i-1,j-1}(t), C_{i-1,j+1}(t), C_{i+1,j-1}(t), C_{i+1,j+1}(t), \\ &C_{i-2,j-2}(t), C_{i-1,j-2}(t), C_{i-2,j-1}(t), C_{i,j-2}(t), \mathbf{C_{i,j}(t)}, C_{i-2,j}(t), C_{i-2,j+1}(t), C_{i-2,j+2}(t), C_{i-1,j+2}(t), \\ &C_{i,j+2}(t), C_{i+1,j-2}(t), C_{i+1,j+2}(t), C_{i+2,j-2}(t), C_{i+2,j-1}(t), C_{i+2,j}(t), C_{i+2,j+1}(t), C_{i+2,j+2}(t)] \end{aligned} \quad (5)$$

In Equations (1)–(5), we assumed that:

- C stands for cell;
- i and j represent the position of a single cell in a two-dimensional lattice of cells;
- t represents the time step;
- $C_i(t+1)$ represents, in the one-dimensional CA, the output state of the central cell at the time step t+1;
- $C_{i,j}(t+1)$ represents, in the two-dimensional CA, the output state of the central cell at the time step t+1;
- f represents the evolution function of the CA.

It is obvious that, for marginal cells of the CA, for example in the case of the one-dimensional CA, cells C_1 and C_n, the evolution function f presented in Equations (1)–(5) cannot be applied directly because there are no other neighbors at the end of the lattice. For this reason, different boundary conditions will be applied there. Some boundary conditions variants in the case of one-dimensional CA with radius 1 are presented below.

Null boundary conditions refer to a scenario in which the marginal cells of the CA are linked to logic 0 states: 0 [C_1, C_2, . . ., C_{n-1}, C_n] 0.

Periodic boundary conditions entail connecting the extreme cells to each other, creating a cyclic arrangement: C_n [C_1, C_2, . . ., C_{n-1}, C_n] C_1.

Fixed boundary conditions involve connecting the marginal cells with preassigned, unchanging logic 0 or 1 states: 0 [C_1, C_2, . . ., C_{n-1}, C_n] 0 OR 0 [C_1, C_2, . . ., C_{n-1}, C_n] 1 OR

$$1\ [C_1, C_2, \ldots, C_{n-1}, C_n]\ 0 \text{ OR } 1\ [C_1, C_2, \ldots, C_{n-1}, C_n]\ 1.$$

Adiabatic boundary conditions involve replicating the values of boundary cells to virtually neighboring states: C_1 [C_1, C_2, . . ., C_{n-1}, C_n] C_n.

Intermediate boundary conditions dictate that the value of the left or right boundary will match the value of the cell located two positions away in the corresponding direction: C_3 [C_1, C_2, . . ., C_{n-1}, C_n] C_{n-2}.

Reflexive boundary conditions entail setting the value of the left or right boundary to match the value of the cell one neighbor away in the corresponding direction: C_2 [C_1, C_2, . . ., C_{n-1}, C_n] C_{n-1}.

For instance, in the context of a one-dimensional CA with radius 1 and two possible states per cell, there are a total of 256 (2^8) potential rules, each of which can be represented by a three-variable Boolean function [18,31]. These rules are enumerated according to Wolfram's naming convention, ranging from rule number 0 to rule number 255. From this perspective, an algebraic expression of three evolution rules is presented in Table 1.

Table 1. CA's rules that determine how the next state of cells will be updated based on the current state of neighboring cells.

Rules [1]	**111**	**110**	**101**	**100**	**011**	**010**	**001**	**000**
30	0	0	0	1	1	1	1	0
90	0	1	0	1	1	0	1	0
204	1	1	0	0	1	1	0	0
226	1	1	1	0	0	0	1	0

[1] Decimal representation of the rules.

Figure 4 depicts the graphical representation of rule 30.

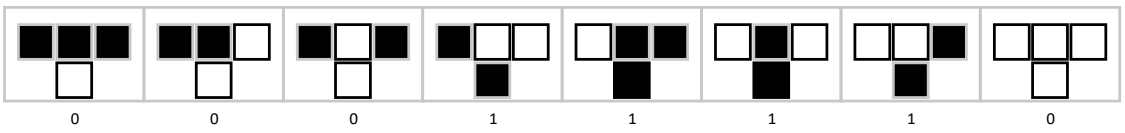


Figure 4. Evolution rule 30, featuring 2 states and radius 1.

Figure 5 illustrates the graphical representation of rule 226.

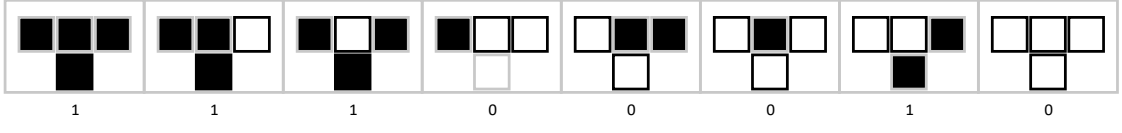


Figure 5. Evolution rule 226, featuring 2 states and radius 1.

The combinational logic expressions for rules 30, 90, 204, and 226 in a CA can be expressed as follows:

1. Rule 30:

$$C_i(t+1) = C_{i-1}(t) \text{ XOR } (C_i(t) \text{ OR } C_{i+1}(t)) \quad (6)$$

2. Rule 90:

$$C_i(t+1) = C_{i-1}(t) \text{ XOR } C_{i+1}(t) \quad (7)$$

3. Rule 204:

$$C_i(t+1) = C_i(t) \quad (8)$$

4. Rule 226:

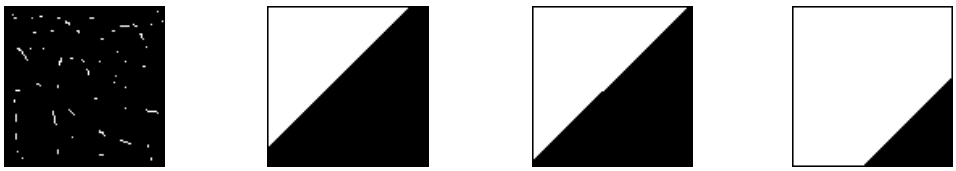
$$C_i(t+1) = C_{i+1}(t) \text{ XOR } (C_i(t) \text{ AND } (C_{i-1}(t) \text{ XOR } C_{i+1}(t))) \quad (9)$$

If a CA's rule employs exclusively XOR logic, it is considered a linear rule and the CA is referred to as linear. Rules incorporating XNOR logic are denoted as complement rules. When a CA utilizes a blend of XOR and XNOR rules, it is termed an additive CA. If all the cells of the CA share the same rule, then the CA is said to be a uniform CA, otherwise, it is categorized as a hybrid CA.

CA find many different practical engineering applications (watermark and image encryption, complex logic puzzles, simulation of complex physical phenomena: solitons, fractals, and chaos; modeling and simulation of population density scenarios and the behavior of crowds, etc.) [32–36], including the use of CA concepts in solving the DCP, as demonstrated in this paper.

2.2. The Hybrid CA-Based Algorithm for the Problem of Density Classification

The DCP method introduced here operates on the premise that in this problem the primary aim is to construct or discover a binary two-dimensional hybrid CA capable of categorizing the density of 1 s and 0 s within the initial lattice configuration. Under this approach, if the initial lattice has a higher density of 1 s than 0 s, the CA is expected to transition towards a final state containing only 1 s after a transition phase. Conversely, if the initial configuration tends towards a higher density of 0 s, the CA should converge to a null configuration consisting only of 0 s. So, in such scenarios, the evolution of the CA predominantly depends on the initial state, and, after applying the evolution rules, the initial image undergoes a translation across the lattice. It is imperative to ensure that the density conserving condition is also satisfied during this process. The fulfilment of these two conditions, translation of the information across the lattice, e.g., towards the bottom right, and density conserving, can be graphically visualized in Figure 6.



(**a**) (**b**) (**c**) (**d**)

Figure 6. (**a**) Original image; (**b**) One possible result: lattice densely populated with 1 s; (**c**) Another possible result: lattice with equal density; (**d**) Yet another possible result: lattice consisting mainly of 0 s.

In two-dimensional CA with radius r =1 and Moore model, as depicted in Figure 2b and detailed in Equation (4), the subsequent state of each cell within the CA is determined

by considering its current state along with the states of the eight cells in its immediate neighborhood. An example of CA rules conventions and the effect of how the application of these rules influence the direction of the image content translation is illustrated in Table 2.

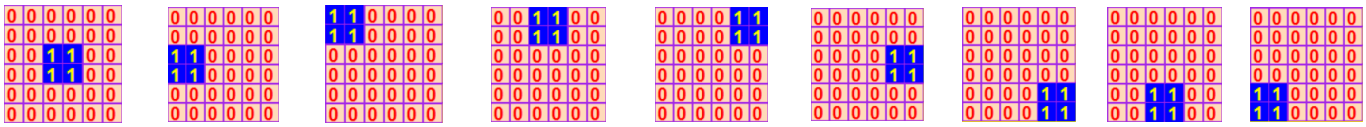
Table 2. CA's rules convention that determines the next state of the central cell and the directions of shifting of the information.

Left Neighbors of the Central Cell	Top and Bottom Neighbors of the Central Cell	Right Neighbors of the Central Cell
64 Shifting of information towards the bottom right.	128 Shifting of information towards the bottom.	256 Shifting of information towards the bottom left.
32 Shifting of information towards the right.	**1** [1] The densities of 0 s and 1 s remain unchanged, yet no translation occurs.	2 Shifting of information towards the left.
16 Shifting of information towards the top right.	8 Shifting of information towards the top.	4 Shifting of information towards the top left.

[1] The CA central cell used to investigate the density conserving and shifting of information.

In the table above, the central cell labelled **1** represents the cell under examination, acting as a variable influenced by itself and eight neighboring variables. Each number listed in Table 2 denotes the CA's rule number that defines the dependency of the current cell solely on a specific neighboring cell. For instance, rule 4 signifies the dependency of the current cell on its bottom-right neighbor, and so forth.

Figure 7 provides an insight into the dynamic behavior of the evolution rules presented in Table 2. It shows the effect of applying each evolution rule individually to a small image (6×6 pixels), with only four pixels in state 1.



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**) (**g**) (**h**) (**i**)

Figure 7. (**a**) Initial image of the CA; (**b**) The effect of the applying of evolution rule 2; (**c**) The effect of the applying of evolution rule 4; (**d**) The effect of the applying of evolution rule 8; (**e**) The effect of the applying of evolution rule 16; (**f**) The effect of the applying of evolution rule 32; (**g**) The effect of the applying of evolution rule 64; (**h**) The effect of the applying of evolution rule 128; (**i**) The effect of the applying of evolution rule 256.

As can be seen in Figure 7, all of the 1 s pixels are moved in the inside of the CA in the direction indicated by the evolution rule selected. Of course, the degree of change inside the CA depends on how many times the evolution rules are applied in succession. In this way, the entire CA lattice of cells evolve at discrete time steps, and also different evolution rules could be applied to different cells at the simultaneous time of evolving, thus forming a hybrid CA system. Obviously, one problem is that at the end of the lattice, we cannot directly apply some of the specified evolution rules (see Figure 7) because there are no other neighbors at the end of the lattice. To solve this problem, in the proposed hybrid CA-based algorithm, we apply the variant with null boundary conditions at the end of the lattice.

The algorithm presented herein for addressing the DCP can be partitioned into three stages: a rule-building stage necessary to test and validate the behavior of the evolution rules presented in Table 2 and Figure 7, a computation stage leading to an intermediate configuration of the CA lattice, and a decision stage necessary to make a density choice.

The first part of the algorithm, known as the rule-building stage, is illustrated in Figure 8.

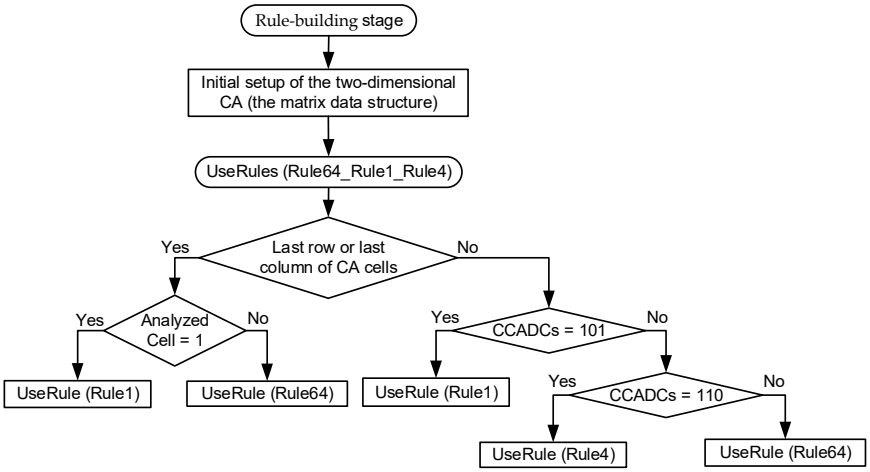


Figure 8. Rule-building stage.

In Figure 8, CCADCs stands for "Central Cell and Diagonal Cells", and the UseRules (Rule64_Rule1_Rule4) block means that each CA central cell (position (i, j) in the lattice) has two neighbors on the diagonal. One neighbor is localized at position (i−1, j−1) in the matrix, and the other is localized at position (i+1, j+1) in the matrix. If we apply the combination of rules 64, 1, and 4 to all the cells of the CA, using the logic described in Figure 8, all values of 1 will be translated to the bottom-right direction. If the value of the cells at the boundary in the direction of translation is zero, in this case the last row or column of the CA cells, then only rule 64 is applied. Otherwise, the cells remain in their previous state—this is equivalent to applying rule 1 to the cell. Therefore, rule 1 can be used for density conserving, and the other two rules, in this case rule 64 and rule 4, can be used to move the 1′s values to the bottom-right direction in the matrix.

The second part of the algorithm, known as the computational stage, is illustrated in Figure 9.

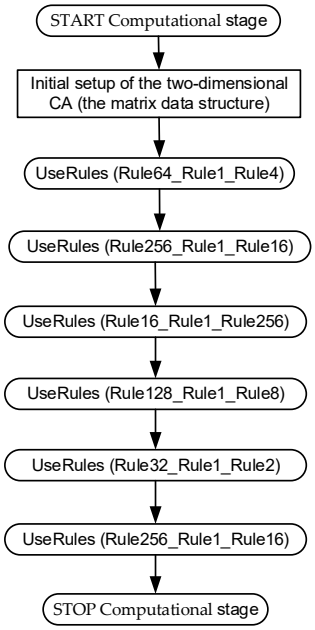


Figure 9. Computational stage.

In Figure 9, in the same way, as described in the previous paragraph and in Figure 8, different combinations of three rules are applied to the CA cells, and the effects to the value of the cells are as follows:

- UseRules (Rule64_Rule1_Rule4)—can be used to move the 1′s values to the bottom-right direction in the matrix (see explanations in previous paragraph).
- UseRules (Rule256_Rule1_Rule16)—can be used to move the 1′s values to the bottom-left direction in the matrix.
- UseRules (Rule16_Rule1_Rule256)—can be used to move the 1′s values to the top-right direction in the matrix.
- UseRules (Rule128_Rule1_Rule8)—can be used to move the 1′s values to the bottom direction in the matrix.
- UseRules (Rule32_Rule1_Rule2)—can be used to move the 1′s values to the right direction in the matrix.

These nine basic CA evolution rules (Rule 1, Rule 2, Rule 4, Rule 8, Rule 16, Rule 32, Rule 64, Rule 128, and Rule 256), and their combinations shown in Figure 8, are used to accomplish both image translation and density conserving throughout the hybrid CA evolution.

After implementing, in the first two stages depicted in Figures 8 and 9, suitable evolution rules on the initial lattice configuration, the hybrid CA generates an intermediate configuration that includes both 0 s and 1 s. The hybrid CA will then make its classification decision on the basis of the intermediate density. This last part of the algorithm, referred to as the decision stage, is depicted in Figure 10.

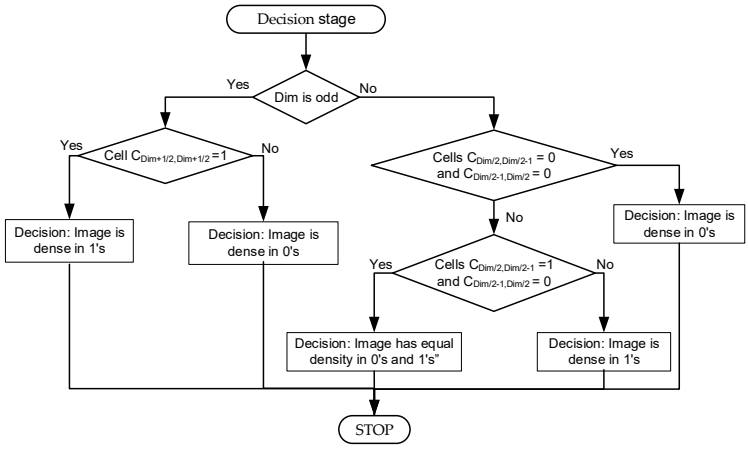


Figure 10. Decision stage.

As shown in Figure 11a, for a hybrid CA with an odd dimension, there are only two viable choices: either the pattern is predominantly dense in 0 s or predominantly dense in 1 s. In this scenario, only by examining the value of the middle cell, $C_{[dim+1/2,\ dim+1/2]}$, can we decide their densities.

(**a**) (**b**)

Figure 11. (**a**) An example of a hybrid CA with an odd number of cells—decision using middle cell value; (**b**) An example of a hybrid CA with an even number of cells—decision using two cells from the secondary diagonal matrix.

For an even dimension hybrid CA, Figure 11b, the density classification decision is made by reading the values of the two cells located in the center of the secondary diagonal matrix data structure. It is easy to prove that the values of two consecutive cells in the center of the secondary diagonal matrix can never be 01 after the computational stage of the DCP algorithm. Based on this observation, the combination of the two cells could be: 00 and in this case the image is dense in 0′s; 10 and in this situation the image has equal density in 0′s and 1′s; 11 and in this case the image is dense in 1′s.

3. Implementation of the Proposed Hybrid CA Solution

In this section, we present the implementation of the already described algorithm based on hybrid CA for solving the DCP.

To verify the behavior of the proposed DCP system, two main modules were implemented, the CA rules simulator and the entire hybrid CA used to solve the DCP. Both modules are coded in the C# programming language available in the Visual Studio Community 2022 integrated development environment (IDE), 64-bit version, and run on a laptop of type ASUS GL552VW with an i7-6700HQ processor (frequency 2.6 GHz) and 16 GB RAM and using a Windows 10 Pro 64-bit operating system.

First of all, we implement the CA rules simulator algorithm in C# in accordance with the logic presented in the rule-building stage shown in Figure 8. As an example, the function named UseRules (Rule64_Rule1_Rule4) implies that rules 64, 1, and 4 are applied to the two-dimensional CA cells (see explanations in the previous chapter). These rules can be used to move the 1′s values to the bottom-right direction in the matrix and, in consequence, the 0′s values will be moved to the top-right direction.

Figure 12 shows the two-dimensional lattice of cells and the variants of applying rules 64, 1, and 4.

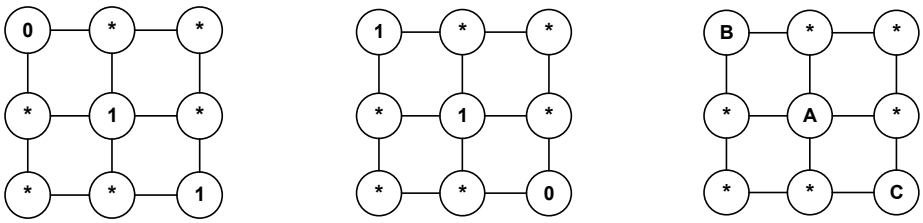


Figure 12. (**a**) CA with a central cell in state 1 and diagonal cells with values 01 implies the use of rule 1; (**b**) CA with a central cell in state 1 and diagonal cells with values 10 implies the use of rule 4; (**c**) CA with diagonal cells ABC other than 101 or 110 implies the use of rule 64.

In Figure 12, the asterisk (*) indicates that the cell could be in any state, which could be either 0 or 1.

The general algorithm used to solve the DCP is given next.

1. Input: Load the hybrid CA with a 2D arbitrary binary image of dimension (dim x dim). The global configuration of the CA represents the initial image to be processed.

2. Rule controller: use general rules according with Figure 12*c and* ***only for boundary cells in the translation direction of rule B:***

If (A=1) then use rule 1 and if (A=0) then use rule B

3. Rule controller: ***all other cells of the hybrid CA***

If (the states of the three neighborhood cells in the translation direction of rule B is 011) then use rule 1. ***That means density conserving.***
Else if (the states of the three neighborhood cells in the translation direction of rule B is 110) then use rule C. ***This means that 0 s will be moved in the translation direction of rule C.***
Else (That means the states of the three neighborhood cells in the translation direction of rule B is 110) then use rule B. ***This means that 1 s will be moved in the translation direction of rule B.***

4. Output: The new matrix data structure representing the new configuration of the cells of the CA (effectively rendering the new image).

For clarity, as an example, let us consider the implementation of the UseRules method, specifically Rule64_Rule1_Rule4. We've encoded the logic described in the general algorithm presented above and depicted in Figure 12, along with the algorithm illustrated in Figure 9. The C# source code for function UseRules64_1_4 is as follows:

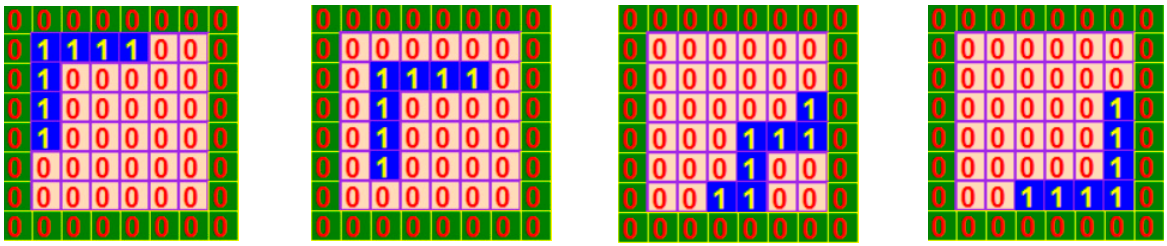
```
private void UseRules64_1_4()
{
//C is the matrix data structure used to store the cells of the CA;
//intermed is the intermediate matrix data structure used to store the intermediate
//state of the CA; dim is the dimension of the CA.
for (int i = 1; i < dim−1; i++)
{
// apply one of the rules 64, 1, or 4 depending on the state of the cells
for (int j = 1; j < dim−1; j++)
{
//for the last line or the last column
if (i == dim−2 || j == dim−2)
{
if (C[i, j] == 1) //apply rule 1
intermed[i, j] = C[i, j];
else
//apply rule 64
intermed[i, j] = C[i−1, j−1];
}
else // for all the other cells of the CA
{
if (C[i + 1, j + 1] == 1 && C[i, j] == 1 && C[i−1, j−1] == 0) {
intermed[i, j] = C[i, j]; //apply rule 1
}
else if (C[i + 1, j + 1] == 0 && C[i, j] == 1 && C[i−1, j−1] == 1) {
intermed[i, j] = C[i + 1, j + 1]; //apply rule 4
}
else {
//apply rule 64 for all other combinations
intermed[i, j] = C[i−1, j−1];
}
} } } }
```

There are several working scenarios between the cells of the CA, but we simulate the comportment of the CA and select for the final implementation only the variant in which the two essential features are achieved: density conserving and translation of the information stored in the cells of the CA through the lattice.

4. Results

In this section, we evaluate the previously described and implemented algorithm based on hybrid CA for solving the DCP by conducting experiments, gathering and interpreting the results obtained.

Figure 13 shows the results of applying UseRules (Rule64_Rule1_Rule4) to an image with a configuration of 6 by 6 cells. The extra border cells are all set to 0—null boundary conditions.

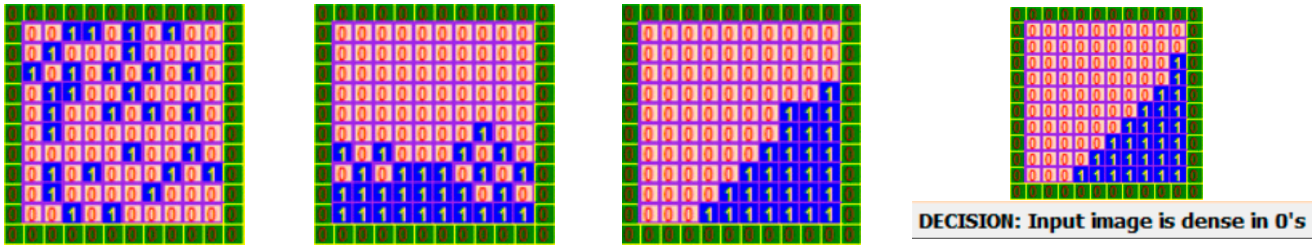


(**a**) (**b**) (**c**) (**d**)

Figure 13. (**a**) Initial configuration (image) of the CA; (**b**) Configuration of the CA after one evolution step; (**c**) Configuration of the CA after three evolution steps; (**d**) Configuration of the CA after five evolution steps (Final image state).

As can be seen in Figure 13, when applying rules 64, 1, or 4, as per the previously outlined algorithm, the initial image undergoes transformation, segregating all occurrences of 1′s from 0′s and relocating them to the bottom right of the image. At the same time, the application of the evolution rules does not change the initial density of 1 s and 0 s but only ensures the translation of the information stored in the lattice. Of course, this is just one possible example, but we ran numerous tests with different initial states of the image, and all the results obtained were correct.

In the same way, all the combinations of rules presented in the computational stage—Figure 9—can be applied. An illustrative example of the application of all the rule combinations described in computational stage is provided in Figure 14.



(**a**) (**b**) (**c**) (**d**)

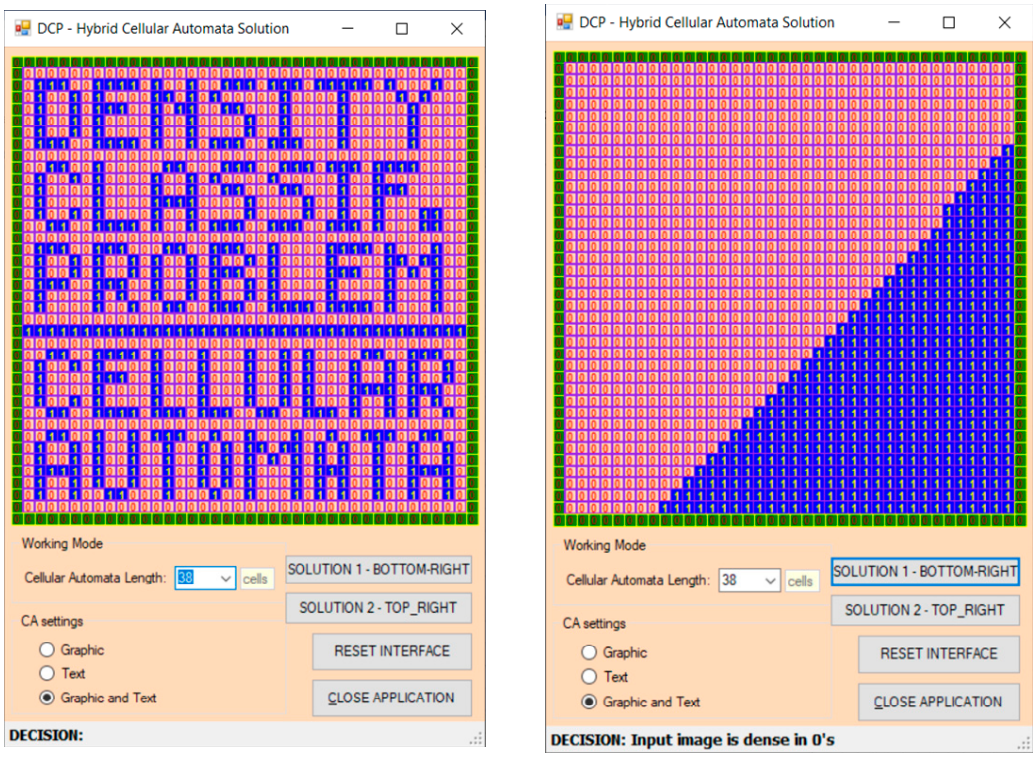
Figure 14. (**a**) Initial image of the CA; (**b**) Image of the CA after an evolution step (computational stage: UseRules (Rule64_Rule1_Rule4) + UseRules (Rule256_Rule1_Rule16)); (**c**) The image of the CA after the second step of evolution (computational stage: UseRules (Rule16_Rule1_Rule256) + UseRules (Rule128_Rule1_Rule8) + UseRules (Rule32_Rule1_Rule2)); (**d**) The image of the CA after the third step of evolution (computational stage: UseRules (Rule256_Rule1_Rule16)) and decision stage.

Figure 14b–d show the output images produced by the two-dimensional CA algorithm assuming an initial CA of size 10. In Figure 14d, at the decision stage, because the values of the two cells located in the center of the secondary diagonal matrix data structure are “00”, the conclusion is that the image is dense in 0′s.

The entire hybrid CA used to solve the DCP uses the logic presented in Figure 9, and each time the evolution rules are applied, a new image is obtained. Different outputs of the CA configuration matrix self-explain how the algorithm works.

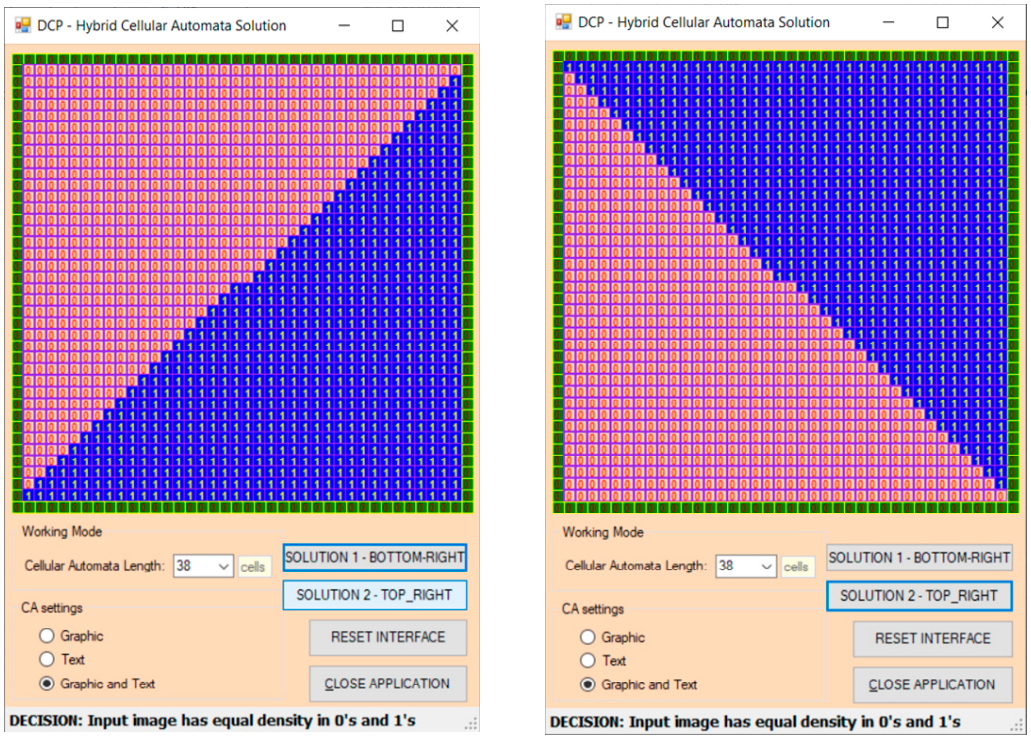
Figure 15 illustrates a relevant example of a demonstrative task applied to a standard input CA image.

Consistent with the algorithm described here, which reflects the bottom-right decision as shown in Figure 16a, the same logic could be applied, this time using the top-right rules, allowing us to classify the images as shown in Figure 16b. The two algorithms, bottom-right and top-right, are equivalent and lead to the same conclusion regarding the density of the original image.



(**a**) (**b**)

Figure 15. (**a**) Initial configuration of the CA (initial image); (**b**) Output after computational stage and decision stage (final image) for the bottom-right solution.



(**a**) (**b**)

Figure 16. (**a**) Final image of the CA with random initial image and decision for the bottom-right solution; (**b**) Final image of the CA with the same initial image as in case (**a**) and decision for the top-right solution.

Figure 16a,b show the perfect correlation between the two CA solutions, bottom-right and top-right, to the DCP.

5. Discussion

In this section, we assess our approach in comparison to established DCP solving techniques, highlighting the primary key potential benefits as outlined below.

Instead of operating over a complete solution space (as presented, for instance, in the references [7,8,12,16,19,37]), the method presented here focuses on a limited set of CA evolution rules (specifically nine rules, detailed in Table 2 and Figure 9) and only six combinations thereof (refer to Figure 9), where optimal solutions are located. Consequently, this streamlines the design process, making it more straightforward and more efficient. Unlike some existing methods, where increasing the radius r (see the neighborhood structure of the CA and the correlation between the size of the CA and the neighborhood dimension) results in a potential exponential expansion of rule spaces (posing the potential of combinatorial explosion), our approach excels by minimizing the number of potential solutions, thereby enhancing performance.

Certainly, examining both individual rules and their combinations enhances our understanding of the computations performed by CA and deepens our understanding of the resulting emergent computations.

Above all, the main objective of solving the DCP is to better understand the limits of the CA. It can be noted that for a square configuration of the CA of size *dimxdim*, the presented algorithm can solve the DCP, but for an arbitrary configuration of the CA (or any other form of image: concave or convex, parallelogram, etc., or even discontinuous images) of size *axb*, the problem has to be studied further, because in these scenarios no diagonal region for taking the decision can be obtained. Therefore, when dealing with any arbitrary configurations of CA, the algorithm outlined in this paper needs to be adapted. One approach could involve partitioning the CA lattice into equally sized regions. Subsequently, the evolution rules can then be selected and applied to these regions to segregate and differentiate the occurrences of $1'$s and $0'$s within each region. Following this segregation process, a classification decision can be made.

6. Conclusions

This paper introduces an approach for the adoption and application of models that rely solely on local interactions and without a central processing unit, as hybrid CA are, dedicated to solving the DCP. The hybrid CA proposed model is a general one because it can be scaled to a larger or a smaller number of cells and is based on the three related stages named: rule building, computation, and decision.

The findings from this study strongly indicate that exploring CA is highly worthwhile and offers a promising avenue for future research endeavors. The methodology used to develop the CA solution for DCP showcased here could, with slight adjustments, prove useful to tackle other mathematical or physical problems, such as object classification (for example, in the case of discontinuous images, this solution could be useful to compute the density of objects on a conveyor belt and then evaluate the density of each configuration and then detect objects), forming and then separating clusters with different values, two-dimensional filtering, clustering noise in different images, and so forth. In addition, another direction is the possibility of creating a fully parallel version of the described hybrid CA algorithm using reconfigurable hardware (e.g., FPGA devices).

Funding: This research was funded by Unitatea Executiva pentru Finantarea Invatamantului Superior, a Cercetarii, Dezvoltarii si Inovarii, a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI–UEFISCDI, project number PN-III-P2-2.1-PED-2021-3669, within PNCDI III, contract number 693PED/28.06.2022.

Data Availability Statement: Since our funding agency retains the rights to the generated data, they are not publicly available. However, some data are included in the article, and upon request, individuals can obtain access to the full data by contacting the corresponding author via email.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Packard, N.H. Adaptation toward the edge of chaos. In *Dynamic Patterns in Complex Systems*; World Scientific: Singapore, 1988; pp. 293–301.
2. Oliveira, G.M.B.; Martins, G.A.; de Carvalho, B.; Fynn, E. Some Investigations About Synchronization and Density Classification Tasks in One-dimensional and Two-dimensional Cellular Automata Rule Spaces. *Electron. Notes Theor. Comput. Sci.* **2009**, *252*, 121–142. [CrossRef]
3. Shao, C.; Shao, F.; Liu, X.; Yang, D.; Sun, R.; Zhang, L.; Jiang, K. A Multi-Information Dissemination Model Based on Cellular Automata. *Mathematics* **2024**, *12*, 914. [CrossRef]
4. Anghelescu, P.; Ionita, S.; Sofron, E. Encryption Technique with Programmable Cellular Automata (ETPCA). *J. Cell. Autom.* **2010**, *5*, 79–105.
5. Land, M.; Belew, R.K. No Perfect Two-State Cellular Automata for Density Classification Exists. *Phys. Rev. Let.* **1995**, *74*, 5148–5150. [CrossRef]
6. Anghelescu, P.; Stirbu, C. Cellular Automata Based Algorithm for Image Density Classification Task. In Proceedings of the 6thInternational Conference on Electronics, Computers and Artificial Intelligence, ECAI 2014, Bucharest, Romania, 23–25 October 2014; Volume 6, pp. 5–9. [CrossRef]
7. Mitchell, M.; Hraber, P.; Crutchfield, J. Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations. *Complex Syst.* **1993**, *7*, 89–130.
8. Mitchell, M.; Crutchfield, P.; Hraber, P.T. Evolving cellular automata to perform computations. *Mech. Impediment Phys. D* **1994**, *75*, 361–391.
9. Jullie, H.; Pollack, J.B. Coevolving the ideal trainer: Application to the discovery of Cellular Automata Rules. In Proceedings of the Third Annual Genetic Programming Conference, San Francisco, CA, USA, 1998; Morgan Kaufmann: Burlington, MA, USA, 1998; pp. 519–527.
10. Andre, D.; Bennett, F.; Koza, J. Discovery by Genetic Programming of a Cellular Automata Rule that is Better than any Known Rule for the Majority Classification Problem. In Proceedings of the Genetic Programming, First Annual Conference, Stanford, CA, USA, 28–31 July 1996; pp. 3–11.
11. Oliveira, G.M.B.; de Oliveira, P.P.B.; Omar, N. Evolving solutions of the density classification task in 1D cellular automata, guided by parameters that estimate their dynamic behavior. In *Artificial Life VII, Proceedings of the Seventh International Conference on Artificial Life*; MIT Press: Cambridge, MA, USA, 2000; pp. 428–436. [CrossRef]
12. Wolz, D.; De Oliveira, P.P.B. Very Effective Evolutionary Techniques for Searching Cellular Automata Rule Spaces. *J. Cell. Autom.* **2008**, *3*, 289–312.
13. Marques-Pita, M.; Rocha, L.M. Conceptual Structure in Cellular Automata-The Density Classification Task. In Proceedings of the ALife XI: Eleventh International Conference on the Simulation and Synthesis of Living Systems, Winchester, UK, 5–8 August 2008; Bullock, S., Noble, J., Watson, R.A., Bedau, M.A., Eds.; MIT Press: Cambridge, MA, USA, 2008; pp. 390–397.
14. Laboudi, Z. An Effective Approach for Solving the Density Classification Task by Cellular Automata. In Proceedings of the 4th World Conference on Complex Systems (WCCS), Ouarzazate, Morocco, 22–25 April 2019; pp. 1–8. [CrossRef]
15. Kari, J.; Le Gloannec, B. Modified traffic cellular automaton for the density classification task. *Fundam. Informaticae* **2012**, *116*, 141–156. [CrossRef]
16. Laboudi, Z.; Chikhi, S. Computational mechanisms for solving the density classification task by cellular automata. *J. Cell. Autom.* **2019**, *14*, 69–93.
17. De Oliveira, P.P.B. On density determination with cellular automata: Results, constructions and directions. *J. Cell. Autom.* **2014**, *9*, 357–385.
18. Wolfram, S. *A New Kind of Science*; Wolfram Media: Champaign, IL, USA, 2002.
19. Laboudi, Z.; Chikhi, S. New Solutions for the Density Classification Task in One Dimensional Cellular Automata. In *Modelling and Implementation of Complex Systems*; Lecture Notes in Networks and Systems; Chikhi, S., Amine, A., Chaoui, A., Saidouni, D., Eds.; Springer: Cham, Switzerland, 2019; Volume 64. [CrossRef]
20. Capcarrere, M.S.; Sipper, M.; Tommasini, M. Two-state, r=1 Cellular Automaton that Classifies Density. *Phys. Rev. Lett.* **1996**, *77*, 4969–4971. [CrossRef] [PubMed]
21. Capcarrere, M.S.; Sipper, M. Necessary conditions for density classification by cellular automata. *Phys. Rev. E* **2001**, *64*, 036113. [CrossRef] [PubMed]
22. Gabriele, A.R. The Density Classification Problem for Multi-states Cellular Automata. In *Advances in Artificial Life*; ECAL 2005. Lecture Notes in Computer Science; Capcarrère, M.S., Freitas, A.A., Bentley, P.J., Johnson, C.G., Timmis, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 443–452. [CrossRef]

23. Fuks, H. Solution of the Density Classification Problem with Two Cellular Automata Rules. *Phys. Rev. E* **1997**, *55*, R2081–R2084. [CrossRef]
24. Wuensche, A.; Lesser, M. *The Global Dynamics of Cellular Automata, An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*; Studies in the Sciences of Complexity; Addison Wesley: Boston, MA, USA, 1992; pp. 22–60.
25. Wuensche, A. Basins of Attraction of Cellular Automata and Discrete Dynamical Networks. In *Cellular Automata*; Encyclopedia of Complexity and Systems Science Series; Adamatzky, A., Ed.; Springer: New York, NY, USA, 2018. [CrossRef]
26. Maiti, N.S.; Munshi, S.; Chaudhuri, P.P. An Analytical Formulation for Cellular Automata (CA) Based Solution of Density Classification Task (DCT). In *Cellular Automata*; ACRI 2006; Lecture Notes in Computer Science; El Yacoubi, S., Chopard, B., Bandini, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4173, pp. 147–156. [CrossRef]
27. Anghelescu, P. Parallel Optimization of Program Instructions Using Genetic Algorithms. *J. Comput. Mater. Contin.* **2021**, *67*, 3293–3310. [CrossRef]
28. Anghelescu, P.; Sofron, E.; Ionita, S. VLSI Implementation of High-Speed Cellular Automata Encryption Algorithm. In Proceedings of the 30th International Semiconductor Conference, CAS 2007, Sinaia, Romania, 15–17 October 2007; pp. 509–512. [CrossRef]
29. Stanica, G.; Anghelescu, P. Cryptographic Algorithm Based on Hybrid One-Dimensional Cellular Automata. *Mathematics* **2023**, *11*, 1481. [CrossRef]
30. Vellarayil Mohandas, N.; Jeganathan, L. Classification of Two Dimensional Cellular Automata Rules for Symmetric Pattern Generation. *Symmetry* **2018**, *10*, 772. [CrossRef]
31. Levina, A.; Mukhamedjanov, D.; Bogaevskiy, D.; Lyakhov, P.; Valueva, M.; Kaplun, D. High Performance Parallel Pseudorandom Number Generator on Cellular Automata. *Symmetry* **2022**, *14*, 1869. [CrossRef]
32. Wang, M.; Chen, M.; Li, J.; Yu, C. 3D Copyright Protection Based on Binarized Computational Ghost Imaging Encryption and Cellular Automata Transform. *Symmetry* **2022**, *14*, 595. [CrossRef]
33. Jeon, J.-C. Multi-Layer QCA Shift Registers and Wiring Structure for LFSR in Stream Cipher with Low Energy Dissipation in Quantum Nanotechnology. *Electronics* **2023**, *12*, 4093. [CrossRef]
34. Chatzinikolaou, T.P.; Karamani, R.E.; Fyrigos, I.A. Handling Sudoku puzzles with irregular learning cellular automata. *Nat. Comput.* **2024**, *23*, 41–60. [CrossRef]
35. Li, G.H.Y.; Leefmans, C.R.; Williams, J. Photonic elementary cellular automata for simulation of complex phenomena. *Light Sci. Appl.* **2023**, *12*, 132. [CrossRef] [PubMed]
36. Bandini, S.; Briola, D.; Dennunzio, A. Distance-based affective states in cellular automata pedestrian simulation. *Nat. Comput.* **2024**, *23*, 71–83. [CrossRef]
37. Montalva-Medel, M.; de Oliveira, P.P.B.; Goles, E. A portfolio of classification problems by one-dimensional cellular automata, over cyclic binary configurations and parallel update. *Nat. Comput.* **2018**, *17*, 663–671. [CrossRef]