Efficient Solutions of the Density Classification Task in One-Dimensional Cellular Automata: Where Can They Be Found?

Zakaria Laboudi
RELA(CS)^2 Laboratory, University of Oum El-Bouaghi
Oum El-Bouaghi, Algeria
laboudizak@gmail.com / laboudi.zakaria@univ-oeb.dz

The density classification task is one among other benchmark problems for studying the ability of cellular automata to solve problems through emergent collective computations. The density classification task attempts to find a local rule that can perform majority voting in an arbitrary initial configuration of a cellular automaton. Solutions for this problem were designed by means of several training mechanisms, more particularly optimization algorithms, due to the lack of standard procedures for selecting suitable local rules. In this paper, we propose new investigations into density determination in one-dimensional cellular automata of radius $r = 4$. We show that our proposal allows retaining a considerable number of unknown rules, some of which may outperform the current efficient solutions. Moreover, we give explanations about the computational mechanisms making global computations emerge so that the considered problem is solved. This is a key element for improving both the general understanding of the way in which computational tasks are solved by emergence and the selection of suitable local state-transitions rules.

Keywords: complex systems; emergent global computation; cellular automata; density determination; symmetry property, number-conserving property; quantum-inspired evolutionary algorithm

1. Introduction

Complex systems are distinguished by a large number of components structured in some form that describes the system at distinct levels ranging from microscopic to macroscopic scales. These components communicate locally in such a way that global computations emerge at the macroscopic level without resorting to any centralized control mechanism. A good example of such observations is the behavior of ants that build their anthills, feed, protect themselves and even attack other insects' colonies; and yet there are neither ant chiefs nor centralized control mechanisms [1].

One of the most important aims for which we study such systems is to try to locate the origin of emerging global computations. This helps to understand their behaviors, predict them and finally perform computations using artificial complex systems. The difficulty of dealing with complex systems stems from the fact that one has to study them in their entirety because, by simplifying them, their intelligibility would be destroyed. That is why we usually rely on modeling to simulate and analyze their behaviors.

Although there exist several models for complex systems, cellular automata (CAs) are favored due to their abstraction level, local interactions, behavioral qualities and computational abilities [1]. In that context, some benchmark tasks were considered in order to find solutions to them using CAs. We cite as examples: the firing squad synchronization problem [2], the parity determination [3], the synchronization task [4] and the density classification task [4–25]. Here, the successfulness of computation depends on the ability to reach some target global states by communicating information across the dimensions of space and time. Such tasks are nontrivial for CAs given that each cell can only perceive a local part of the automaton, depending on its neighboring cells.

In this paper, we consider solving the density classification task (DCT) under its standard version by one-dimensional (1D) CAs. Initial solutions for this task were proposed using diverse methods varying from manual to programmed approaches [10–15]. But, since further properties were put forward for consideration, in particular the symmetry and number-conserving ones, several efficient solutions could be discovered [5, 16–25]. Nevertheless, most problem-solving approaches try to conceive solutions without setting out solid strategies, which—in many cases—weakens the design process. Indeed, local state transitions are often seen as difficult to write so as to allow effective problem solving. This comes back to the lack of clear understanding of computations' nature within CAs, hence the difficulty of finding standard procedures for selecting local rules leading to emerging global computations [3, 24].

The current paper extends the work presented in [25] by completing some unfinished parts of a two-step procedure for retaining solutions of the DCT in 1D CAs of radius $r = 4$. The first step of the procedure allows pinpointing the subpart of the rules' space where the more efficient solutions are most likely found. The second step focuses on using search methods (quantum-inspired evolutionary algorithms) to retain them. As a result, we could find a considerable number of unknown rules that may outperform the current known solutions. Undoubtedly, these findings help, on the one hand, to enhance our general understanding about the way in which CAs produce emerg ing global computations and on the other hand, to reinforce our knowledge and skills for selecting local state-transition rules to perform computational tasks.

We start with some definitions and properties of CAs in Section 2. Then, we provide an overview of the DCT, examine some of its known solutions and highlight their characteristics in Section 3. After that, we give details about our proposal for retaining efficient solutions of the DCT in Section 4. Next, we summarize, analyze and discuss the experiments and the obtained results in Section 5. Finally, we provide a summary of the main issues discussed, some concluding remarks and suggestions for future work in Section 6.

2. Basic Concepts

Next, some definitions and properties of CAs are introduced.

2.1 Cellular Automata

A cellular automaton (CA) consists of a grid of N cells, each in one of q discrete states of set $V = \{0, 1, .., q-1\}$. When the states are assigned to the cells of the grid, we obtain a configuration. The automaton starts with an initial state (initial configuration) at time $t = 0$ and then evolves following some local rule. This latter updates the state of each cell at time t depending on the states of n neighboring cells, including the specified cell, at time $t-1$. Formally, state transition rules are defined as functions from V^n to V. They are also expressed as lookup tables that associate each possible neighborhood's configuration with the state's update of the cell in the center [7]. Table 1 gives an example of a 1D binary rule of radius $r = 1$ (i.e., neighboring cells' number on either side of each cell), denoted by ϕ_{examp}. These lookup tables are often encoded as strings and converted into decimals. For instance, the binary and decimal values of rule ϕ_{examp} are 11100010 and 226, respectively.

Neighborhood configuration	$0\underline{0}0$	$0\underline{0}1$	$0\underline{1}0$	$0\underline{1}1$	$1\underline{0}0$	$1\underline{0}1$	$1\underline{1}0$	$1\underline{1}1$
Output	0	1	0	0	0	1	1	1

Table 1. A typical lookup table for rule ϕ_{examp}.

Let $w \in V$, $v \in V^n$ and ϕ be a cell's state at time t, a neighborhood configuration and a CA local rule, respectively. State w is said to be *quiescent* if $w = 0$, or *active* otherwise. v is said to be a *generation process* (*annihilation process*, respectively) if the cell in the center is *activated* (*deactivated*, respectively) at time $t+1$, that is, $\phi(v) \neq 0 \wedge v\left[\left(n/2\right)\right] = 0$. Also, v is said to be a *preservation process* if the state of the cell in the center is *preserved* at time $t+1$—that is,

$\phi(v) = v[(n/2)]$ [15, 22]. For example, the generations, annihilations and preservations processes of rule ϕ_{examp} are respectively given as follows: $\text{gen} = \{(0, 0, 1), (1, 0, 1)\}$, $\text{ann} = \{(0, 1, 0), (0, 1, 1)\}$ and $\text{pres} = \{(0, 0, 0), (1, 0, 0), (1, 1, 0), (1, 1, 1)\}$.

2.2 The Symmetry Property

For any CA rule ϕ, there exists a set of equivalent rules distributed in classes so that they all lead to the same behavior. Equivalence classes are obtained through transformations of states swapping, directions swapping and the combination of both. For 1D binary CAs, three combinations are distinguished: black-white transformation, left-right transformation and the combination of the two [5, 18]; Figure 1 shows an example of a binary CA rule of radius $r = 1$. We call a symmetric rule any 1D binary CA whose local state transitions verify conditions of the combined transformation. For instance, rule ϕ_{examp} is symmetric. Table 2 gives rules of the typical CA as well as the resulting transformations. In Figure 1, only the colored cells in the table are plotted for explanation.

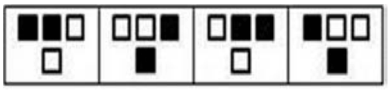

Figure 1. Transformations of a typical 1D binary CA rule.

Neighborhood configuration	$0\underline{0}0$	$0\underline{0}1$	$0\underline{1}0$	$0\underline{1}1$	$1\underline{0}0$	$1\underline{0}1$	$1\underline{1}0$	$1\underline{1}1$
Typical rule	0	1	1	0	1	0	0	1

Table 2. Lookup table used for Figure 1.

2.3 The Number-Conserving Property

For CA rules, the property of number conserving implies the ability to maintain the sum of states in the automaton during any temporal evolution [24]. For binary CAs, this means that the initial number of cells in each state is preserved over time; rule ϕ_{examp} is number conserving.

3. The Density Classification Task

3.1 Overview

Solving the DCT by binary CAs implies finding a local state-transition rule ϕ that classifies configurations according to the current density of cells in each state. Thus, the automaton has to determine for each initial configuration (IC) whether the number of cells in state $w = 1$—which we refer to as black cells—is greater than for cells in state $w = 0$—which we refer to as white cells. When there is a majority of black cells compared to white ones, the CA should reach a homogenous fixed-point pattern (HFP) where all the cells become black; if white is the majority color, all cells become white. We point out that there exist other formulations for the DCT [24]; however, in this paper, we limit ourselves to the standard one for which the considered task is solved by 1D CAs using one basic single rule.

In order to measure the ability of any given solution ϕ, the performance P_N^K is calculated. It consists in determining the rate from K ICs of size N for which rule ϕ gives the right answer. Usually, tests are conducted on random ICs, generated following uniform Bernoulli distribution (UBD) with parameter $\rho \approx 0.5$; ρ refers to the density of black cells in ICs [5, 15]. This represents the most difficult test case. In fact, as it is not always possible to test a whole space of ICs, in particular when N grows, tests rely on approximate performances by considering a large number of ICs.

The DCT allows dealing with complex systems by studying the mechanisms of emerging global computations at a high level of abstraction (i.e., cells are considered as *don't care entities*). Also, it allows assessing the ability of CAs to perform such a kind of computation. In practical terms, the DCT is involved in several real-world decentralized applications imposing that a decision be made by majority voting, such as in multi-agents systems, collaborative robots, fleet of drones, and others. Finally, the DCT plays a role like for the knapsack and the traveling salesman problems in the optimization field, especially evolutionary approaches in CAs.

3.2 Related Works

Actually, the authors in [26] proved that the perfect determination of density in CAs is impossible without committing any errors. As a result, much research has focused on finding imperfect solutions that can classify a wide range of ICs. For this purpose, several methods were proposed, which we could divide into three main classes: naïve methods, methods based on optimization algorithms and finally, methods based on derivation of rules.

3.2.1 Naïve Methods

In this class of methods, local transition rules are written by hand through an iterative process, in which rules are tested and altered as necessary until the desired behavior is achieved. Theoretically, the human design of rules (i.e., handwritten solutions) is the best way for conceiving them. However, in practical terms, this process is difficult to implement, hence the use of naïve assumptions for selecting local rules. As a consequence, solutions designed by naïve methods are restricted to a small number of rules whose performances are not sufficiently satisfactory.

3.2.2 Methods Based on Optimization Algorithms

These methods try to explore a whole space of rules using optimization algorithms, and more particularly evolutionary algorithms. The number of efficient solutions discovered by these methods is significant, especially when the symmetry property was introduced [5]. Nevertheless, most of them search blindly for solutions in a structureless space of rules, which may have negative side effects on the design process.

3.2.3 Methods Based on Derivation

These methods are based on deriving solutions from arbitrarily selected rules. The difference between them lies in the way in which sets of working rules are constructed as well as in the way in which solutions are deduced from the target workspaces. The number of efficient solutions discovered by these methods is also significant, in particular when the number-conserving property was introduced [18, 19]. The main disadvantage of such methods is that they adopt intuitive principles instead of well-defined processes, whether for determining workspaces of rules or for the derivation task.

Table 3 gives the performances and characteristics of some known rules.

	Problem-Solving Method			CA-Radius r		Performance (%)
Rule ID	Naïve	Optimization	Derivation	$r=3$	$r=4$	$P_{N=449}^{K=100.000}$
ϕ_{GKL} [10]	X	-	-	X	-	77.73
ϕ_{Davis95} [11]	X	-	-	X	-	78.57
ϕ_{Das95} [11]	X	-	-	X	-	78.24
ϕ_{DMC} [12]	-	X	-	X	-	73.49
ϕ_{COE1} [13]	-	X	-	X	-	81.56
ϕ_{COE2} [13]	-	X	-	X	-	80.94
ϕ_{GP1995} [11]	-	X	-	X	-	77.76
ϕ_{GEP1} [14]	-	X	-	X	-	78.31
ϕ_{GEP2} [14]	-	X	-	X	-	78.53
ϕ_{MM401} [15]	-	-	X	X	-	79.01
ϕ_{MM0711} [20]	-	-	X	X	-	80.04
ϕ_{MM0802} [16]	-	-	X	X	-	80.36
ϕ_{B33} [21]	-	X	X	X	-	84.15
ϕ_{WdO1} [5, 18]	-	X	X	X	-	85.80

Rule ID	Problem-Solving Method			CA-Radius r		Performance (%)
	Naïve	Optimization	Derivation	$r = 3$	$r = 4$	$P_{N=449}^{K=100.000}$
ϕ_{LCL} [21]	-	X	X	X	-	85.67
ϕ_{ALTER1} [18, 19]	-	X	X	X	-	85.86
$\phi_{\text{M\_ALTER1}}$ [22]	-	-	X	-	X	87.40
ϕ_{R4B1} [22]	-	-	X	-	X	88.69
ϕ_{R4B2} [22]	-	-	X	-	X	87.99
ϕ_{R4B3} [22]	-	-	X	-	X	88.06
ϕ_{R4B4} [22]	-	-	X	-	X	87.96
ϕ_{R4B5} [22]	-	-	X	-	X	87.90
ϕ_{R4B6} [22]	-	-	X	-	X	88.05
ϕ_{R4BL1} [23]	-	-	X	-	X	89.27
ϕ_{R4BL2} [23]	-	-	X	-	X	88.65
ϕ_{R4BL3} [23]	-	-	X	-	X	88.33
ϕ_{R4BZL} [25]	-	-	X	-	X	89.99

Table 3. Statistics on some known solutions of the DCT.

3.3 The General Characteristics of Efficient Solutions

In fact, since some research has confirmed the existence of a strong correlation between the ability to solve the DCT and the symmetry and number-conserving properties [5, 16, 18, 19], a significant number of efficient rules can be designed using CAs of radius $r = 3$. Solutions that we refer to as ϕ_{B33} [21], ϕ_{WdO1} [5], ϕ_{LCL} [21] and ϕ_{ALTER1} [18] are good examples. In spite of that number of solutions, no evidence was given so as to determine the real connection of density classification with such properties. Hence, the authors in [22] managed to provide elucidations about the way in which these properties contribute to performing the DCT. In short, the decomposition of some efficient solutions—such as rules ϕ_{WdO1} and ϕ_{ALTER1}—into minimal components (subrules) showed that they comprise three kinds of subrules: minimal number-conserving (min-NC) subrules that maintain the symmetry of patterns, min-NC subrules that do not maintain the symmetry of patterns and convergence-conditions subrules (i.e., local state-transitions rules leading ICs to reach HFP), which are modified min-NC subrules. As a result, these mechanisms could be replicated in order to design new efficient solutions using CAs of radius $r = 4$ [22, 23, 25]. Table 4 shows the minimal components of rule ϕ_{R4BZL} [25]. Note that the processes (generations and annihilations) containing common parts are organized into segments; for more details about that representation, one should refer to [22]. Also, asymmetrical exchanges figure in patterns that do not contain symbols “[” and “]”.

Symmetric min-NC Subrules	
min-NC1 ♯, 1, 0, [1, $\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$, 0], 0, 1, ♯ ♯, 0, 1, [1, $\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$, 0], 1, 0, ♯	min-NC10 ♯, 0, 0, 0, [$\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$], 0, 1, 0, ♯ ♯, 1, 0, 1, [$\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$], 1, 1, 1, ♯
min-NC2 ♯, [1, 0, 1, $\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$, 0, 1, 0], ♯	min-NC11 ♯, [1, 0, 0, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 1, 1, 0], ♯
min-NC3 ♯, [0, 1, 1, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 0, 0, 1], ♯	min-NC12 ♯, 0, 0, 0, [$\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$], 0, 1, 1, ♯ ♯, 0, 0, 1, [$\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$], 1, 1, 1, ♯
min-NC4 ♯, 0, [0, 1, $\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$, 0, 1], 0, ♯ ♯, 1, [0, 1, $\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$, 0, 1], 1, ♯	min-NC13 ♯, 0, 0, 0, [$\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$], 0, 0, 1, ♯ ♯, 0, 1, 1, [$\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$], 1, 1, 1, ♯
min-NC5 ♯, 0, [1, 1, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 0, 0], 0, ♯ ♯, 1, [1, 1, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 0, 0], 1, ♯	min-NC14 ♯, 1, 0, [0, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 1], 0, 0, ♯ ♯, 1, 1, [0, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 1], 1, 0, ♯
min-NC6 ♯, [0, 1, 1, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 0, 1, 1], ♯ ♯, [0, 0, 1, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 0, 0, 1], ♯	min-NC15 ♯, 1, 0, [0, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 1], 0, 1, ♯ ♯, 0, 1, [0, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 1], 1, 0, ♯
min-NC7 ♯, [0, 0, 1, $\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$, 0, 1, 1], ♯	min-NC16 ♯, ♯, ♯, [1, $\underline{\mathbf{1}}$, 1 | 0, $\underline{\mathbf{0}}$, 0], ♯, ♯, ♯
min-NC8 ♯, [1, 0, 0, $\underline{\mathbf{1}}$ | $\underline{\mathbf{0}}$, 1, 1, 0], ♯	min-NC17 ♯, ♯, 0, 0, [$\underline{\mathbf{1}}$, 0, (1 | 0)∗, 1, $\underline{\mathbf{0}}$], 0, ♯, ♯, ♯ ♯, ♯, 0, 0, [$\underline{\mathbf{1}}$, 0, (1 | 0)∗, 1, $\underline{\mathbf{0}}$], 1, 1, ♯, ♯ ♯, ♯, ♯, 1, [$\underline{\mathbf{1}}$, 0, (1 | 0)∗, 1, $\underline{\mathbf{0}}$], 0, ♯, ♯, ♯ ♯, ♯, ♯, 1, [$\underline{\mathbf{1}}$, 0, (1 | 0)∗, 1, $\underline{\mathbf{0}}$], 1, 1, ♯, ♯
min-NC9 ♯, [0, 0, 0, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 1, 1, 1], ♯	
Asymmetric min-NC Subrules	
min-NC18 0, 0, 0, 1, $\underline{\mathbf{1}}$, 0, $\underline{\mathbf{0}}$, 0, ♯, ♯, ♯ ♯, 1, 0, 0, [$\underline{\mathbf{1}}$, 1 | 0, $\underline{\mathbf{0}}$], 0, ♯, ♯, ♯ ♯, ♯, 1, 0, [$\underline{\mathbf{1}}$, 1 | 0, $\underline{\mathbf{0}}$], 0, ♯, ♯, ♯ ♯, ♯, ♯, 1, $\underline{\mathbf{1}}$, 1, $\underline{\mathbf{0}}$, 0, 1, 1, 1 ♯, ♯, ♯, 1, [$\underline{\mathbf{1}}$, 1 | 0, $\underline{\mathbf{0}}$], 1, 1, 0, ♯ ♯, ♯, ♯, 1, [$\underline{\mathbf{1}}$, 1 | 0, $\underline{\mathbf{0}}$], 1, 0, ♯, ♯	min-NC19 ♯, 0, 0, 0, $\underline{\mathbf{0}}$, 0, $\underline{\mathbf{1}}$, 1, 0, ♯, ♯ 1, 0, 0, [0, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 1], 0, ♯, ♯ ♯, ♯, 1, 0, $\underline{\mathbf{0}}$, 1, $\underline{\mathbf{1}}$, 1, 1, 1, ♯ ♯, ♯, 1, [0, $\underline{\mathbf{0}}$ | $\underline{\mathbf{1}}$, 1], 1, 1, 0

Convergenceconditions
Modified min-NC1
0, 0, 0, 0, $\underline{1}$, 0, 0, 0, 0
1, 1, 1, 1, $\underline{0}$, 1, 1, 1, 1

Table 4. Symmetrical and asymmetrical cells' exchanges in rule ϕ_{R4BZL}.

Remarks

- The underlined states denote central cells in generation and annihilation processes.
- Symbols "[" and "]" refer to delimiters outlining symmetric motifs, while symbol "|" denotes their centers.
- Symbol "#" means that the specified cell can be either in state 0 or in state 1.

The symmetric min-NC subrules lead ICs to undergo a series of transformations. The aim is to produce some target motifs such that when they arise, convergence conditions emerge by applying asymmetric min-NC subrules. Thus, the computation is carried out only when such a pattern appears, since the CA considers it as a criterion for determining the current density of cells. The more slowly the automaton produces convergence conditions, the more it efficiently identifies the density of cells in configurations of large size (i.e., when $N \gg$), because this allows enough time for self-organization before the CA starts the actual computation.

4. The Proposed Approach

Now, we give details about the proposed approach for retaining efficient solutions of the DCT.

4.1 Context and Motivation

Despite the abilities of CAs to carry out computations, there is still a general dearth of knowledge about what is needed so that a given rule ϕ would be able to reach an intended purpose [3, 24]. For instance, the examination of the different methods for retaining DCT solutions shows that they try to design them without adopting a well-defined process. In this paper, we try to complete the unfinished parts of the work presented in [25], in which a two-step procedure for retaining DCT solutions of a neighborhood with radius $r = 4$ is proposed. The design process deals with a reduced subspace of rules where efficient solutions are most likely found instead of using naïve, blind and purely intuitive principles. We mention that although it is expected to obtain better results using CAs of radius $r = 4$, the current efficient solving methods (see for instance [5, 18, 22]) cannot be reused in such a case, since the search space of rules is larger. Consequently, working on CAs of radius $r = 4$ seems a good fit and provides an assessment of the ability to effectively discover new solutions, in particular since most of the solutions of radius $r = 3$ are already known. The aim of our proposal is threefold:

1. To check the relevance of the suggestions and assumptions on which the approach is founded.
2. To strengthen the knowledge base regarding the selection of appropriate local rules.
3. To deepen the general understanding of emerging global computations within CAs.

4.2 Details of the Proposed Approach

The adopted approach for retaining efficient DCT solutions comprises two stages. In the first stage, a reduced search space (workspace) is constructed by combining some minimal subrules into sets. In the second stage, search methods are involved in order to explore the list of potential solutions resulting from the merging of certain elements of the workspace.

4.2.1 Stage 1: Collecting Minimal Subrules Sets

In this stage, we construct sets of subrules in such a way that, when merged, they generate potential solutions for the DCT. As explained above, three types of subrules are needed: symmetric min-NC subrules, asymmetric min-NC subrules and convergence conditions.

(a) *Generating the symmetric min-NC subrules*: To generate that type of subrule, it is necessary to enumerate all symmetric neighborhood configurations and then to determine all possible exchanges between the symmetrical cells. For each symmetrical exchange, a min-NC is generated and stored in a set denoted by *S*; Table 5 shows a typical example. Note that in the case of CAs of radius $r = 4$, the number of such rules is much greater than for the ones of radius $r = 3$.

Symmetrical Pattern	Typical Exchanges	Resulting min-NC
		min-NC1
(1, 1, 1, 1 | 0, 0, 0, 0)	(1, 1, 1, $\underline{1}$ | $\underline{0}$, 0, 0, 0)	♯, 1, 1, 1, $\underline{1}$, $\underline{0}$, 0, 0, 0, ♯
		min-NC2
	(1, 1, $\underline{1}$, 1 | 0, $\underline{0}$, 0, 0)	♯, ♯, ♯, 1, $\underline{1}$, 1, 0, $\underline{0}$, 0, ♯, ♯, ♯

Table 5. A typical processed symmetrical exchange.

(b) *Generating the asymmetric min-NC subrules*: To generate that type of subrule, it is necessary to specify a target pattern for which the exchange between the cells does not maintain the symmetry. The aim is to make convergence conditions appear. The loss of symmetry should be in patterns where the determination of density is clear. Moreover, it is recommended to limit as far as possible the use of that kind of subrule. Once such patterns are defined, we proceed to the specification of all possible asymmetrical exchanges between cells such that for each of them a min-NC subrule is generated and stored in a set denoted by *A*. Table 6 shows a typical example.

(c) *Generating convergence-conditions subrules*: All that remains is to specify convergence conditions that allow reaching HFP and then to put the result in a set denoted by *C*. To do this, one should target patterns where the density determination is as clear as possible. Table 7 shows typical convergence conditions.

Target Pattern	Typical Exchanges	Typical min-NCs
		min-NC1
		♯, ♯, 0, 1, $\underline{1}$, 0, $\underline{0}$, 0, 0, ♯, ♯
		♯, ♯, 1, 1, $\underline{1}$, 1, $\underline{0}$, 0, 1, ♯, ♯
	(0, 0, 1, $\underline{1}$ | 0, $\underline{0}$, 0, 0)	
	(1, 1, $\underline{1}$, 1 | $\underline{0}$, 0, 1, 1)	min-NC2
		♯, ♯, 0, 1, $\underline{1}$, 0, $\underline{0}$, 0, 0, 0, ♯
		♯, ♯, 0, 1, $\underline{1}$, $\underline{0}$, 0, 0, 0, 1
		♯, 1, 1, 1, $\underline{1}$, 1, $\underline{0}$, 0, 1, ♯, ♯
		0, 1, 1, 1, $\underline{1}$, $\underline{0}$, 0, 1, ♯, ♯
(0, 0, 1, 1 | 0, 0, 0, 0)		
(1, 1, 1, 1 | 0, 0, 1, 1)		min-NC3
		♯, ♯, 0, 0, $\underline{0}$, 0, $\underline{1}$, 1, 0, ♯, ♯
		♯, ♯, 1, 0, $\underline{0}$, 1, $\underline{1}$, 1, 1, ♯, ♯
	(0, 0, $\underline{0}$, 0 | $\underline{1}$, 1, 0, 0)	
	(1, 1, 0, $\underline{0}$ | 1, $\underline{1}$, 1, 1)	min-NC4
		♯, 0, 0, 0, $\underline{0}$, 0, $\underline{1}$, 1, 0, ♯, ♯
		1, 0, 0, 0, $\underline{0}$, $\underline{1}$, 1, 0, ♯, ♯
		♯, ♯, 1, 0, $\underline{0}$, 1, $\underline{1}$, 1, 1, 1, ♯
		♯, ♯, 1, 0, $\underline{0}$, $\underline{1}$, 1, 1, 1, 0

Table 6. A typical processed asymmetrical exchange.

Target Patterns	Original min-NC	Modified min-NCs
	min-NC	modified min-NC1
(0, 0, 0, 0, 1, 0, 0, 0, 0)	(♯, 0, 0, 0, $\underline{1}$, $\underline{0}$, 0, 0, 0, ♯)	0, 0, 0, 0, $\underline{1}$, 0, 0, 0, 0
(1, 0, 0, 0, 1, 0, 0, 0, 0)	(♯, 1, 1, 1, $\underline{1}$, $\underline{0}$, 1, 1, 1, ♯)	1, 1, 1, 1, $\underline{0}$, 1, 1, 1, 1
(1, 1, 1, 1, 0, 1, 1, 1, 1)		
(1, 1, 1, 1, 0, 1, 1, 1, 0)		modified min-NC2
		♯, 0, 0, 0, $\underline{1}$, 0, 0, 0, 0
		1, 1, 1, 1, $\underline{0}$, 1, 1, 1, ♯

Table 7. Typical convergence conditions.

4.2.2 Stage 2: Retaining Solutions through Search Methods

In this stage, the elements of sets S, A and C are merged to obtain potential solutions for the DCT.

(a) *Generating and preprocessing the power sets of sets S, A and C*: The sets of all subsets of sets S, A and C are enumerated and stored; we refer to the resulting sets as $P(S)$, $P(A)$ and $P(C)$, respectively. Next, the Cartesian product of sets $P(S)$, $P(A)$ and $P(C)$ is calculated; we refer to the resulting set as CP_{SAC}. Set CP_{SAC} is preprocessed by removing all inconsistent elements. Any element $e \in CP_{SAC}$ is said to be *inconsistent* if and only if merging min-NC subrules of its components into one single rule does not lead to a number-conserving CA. This is caused by overlapping min-NC subrules (i.e., those having some generations and annihilations in common). In fact, the authors in [22] showed that inconsistent subrules may have a negative effect on the performances of rules. For example, although rules $\min - \mathrm{NC}_{i=1..\times 2}$ in Table 5 are min-NC rules, their merging into one single rule does not produce a number-conserving one. We refer to the set of consistent components from Set CP_{SAC} as $CP_{\text{consistent\_}SAC}$. Note that set $CP_{\text{consistent\_}SAC}$ can also be generated using a low-cost backtracking algorithm in order to gather only the consistent elements from sets S, A and C.

(b) *Retaining efficient solutions*: for each element $e \in CP_{\text{consistent\_}SAC}$, it is possible to get a potential solution for the DCT by merging the subrules of its components into one single rule and thus constructing a whole space of potential solutions that can be tested.

5. Experiments

This section presents the experimental part and thus provides, discusses and analyzes the results obtained.

5.1 Technical Aspects

Next, we outline some technical aspects that we use in the experimental part. Intuitively, there are two methods for exploring the elements of set $CP_{\text{consistent\_}SAC}$: exhaustive and approximate search methods.

5.1.1 Retaining Solutions through Exhaustive Search

By using an exhaustive search algorithm that visits all the elements of set $CP_{\text{consistent\_}SAC}$, it is possible to retain all efficient solutions. Theoretically, this is the best way to find such rules, provided that the cardinality of $CP_{\text{consistent\_}SAC}$ does not lead to combinatory explosion and the fitness function should not be CPU time consuming.

5.1.2 Retaining Solutions through Approximate Search

Approximate methods allow finding solutions that are as close as possible to the optimal ones. This mechanism is relevant in cases where it is impossible to find optimal solutions within a reasonable time (i.e., $\left|CP_{\text{consistent\_}SAC}\right| \gg$). On each maximal element $m = (s, a, c) \in CP_{\text{consistent\_}SAC}$, it is possible to apply a local search method (approximate or exact). m is said to be *maximal* if and only if there exists no element $m' = (s', a', c') \in CP_{\text{consistent\_}SAC}$ such that $(s \subset s' \wedge a \subset a' \wedge c \subset c')$. We refer to the set of all maximal elements as M. Indeed, by removing some subrules from any element $m \in M$, the remaining part remains a potential solution for the DCT. Thus, it is possible to target a subset $M' \subseteq M$ that can be chosen randomly or arbitrarily based on preselected properties, in order to check its elements.

5.2 Generating Subrules Sets

Sets S, A and C were manually generated due to the lack of algorithms that carry out this task. The statistics presented in Table 8 show that sets CP_{SAC} and $CP_{\text{consistent\_}SAC}$ could not be generated due to the combinatory explosion. Concerning the elements of set M, it is possible to generate them using the algorithm of enumerating cliques known in graph theory. All that is needed is to consider the min subrules as nodes and the consistency between them as a relation (vertices between nodes). Any element $m \in M$ is a clique in the consistency graph. Likewise, we could check that their number is very large and thus they are not generated.

Set	Cardinality
S	88
A	8
C	3
CP_{SAC}	$2^{\lvert S\rvert}$ x $2^{\lvert A\rvert}$ x $2^{\lvert C\rvert}$
$CP_{\text{consistent\_}SAC}$	?
M	?

Table 8. Statistics on sets S, A, C, CP_{SAC} and $CP_{\text{consistent\_}SAC}$.

5.3 Protocols for Retaining Efficient Solutions

According to Table 8, it seems very difficult to rely on exhaustive search methods. As a result, only approximate search methods are used: quantum-inspired evolutionary algorithms (QIEA) [27, 28]. These algorithms combine principles inspired from evolutionary algorithms (EA) and quantum computing such as qubits and superposition of states. They manipulate populations of individuals (candidate solutions) that are often represented as quantum registers. At each time step, quantum operators as the measure and the interference are applied on individuals in order to generate a new population. We mention that adopting QIEAs instead of conventional genetic algorithms (CGAs) comes from the fact that they show more flexibility, since they allow making both local and global search over solutions' spaces using populations of size ranging from one to many individuals. In addition, some studies showed that QIEAs may outperform CGAs with a low cost in terms of efficiency, number of operators and populations' sizes [27, 28]. For more details about these algorithms, the reader should refer to references [27, 28]. The following pseudocode illustrates the general structure of a standard QIEA.

Pseudocode for QIEAs

```
1. Generate the initial quantum-population $Q(t = 0)$;
2. Generate $P(t)$ by measuring $Q(t)$;
3. Evaluate $P(t)$;
4. Save the best solution in $b$;
5. Update $Q(t)$ using quantum-gate rotation to get $Q(t + 1)$;
6. $t = t + 1$;
7. If the termination criterion is not reached, then Go to 2;
8. End
```

For each element $m \in M$, we store the min subrules of its components in a separated vector v and then use an instance of a QIEA to locally search for potential efficient solutions. Parameters settings are given in Table 9. As in any other meta-heuristic, it is necessary to encode individuals and specify the fitness function.

Parameter	Value
population size	20
interference operator	the same as in [27]
ICs size N	149
threshold for reevaluation F	0.8
size of P_{small}	100
size of P_{large}	20.000

Table 9. Parameters settings.

- *Solutions encoding*: Each vector v is encoded as a binary string and thus a corresponding quantum register, of length l; l is the size of vector v. Each bit $b_{i=0..l}$ indicates the presence/absence of the i^{th} min subrule in vector v. By merging all elements where the bits are set to 1, we obtain an element $e \in CP_{\text{consistent\_}SAC}$ and thus a potential solution for the DCT.

- *The fitness function.* The performance of each visited element $e \in CP_{\text{consistent}\_SAC}$ is calculated through a low-cost fitness function that uses a small pool of ICs, denoted by P_{small}, generated following UBD. When the fitness value exceeds some threshold F, it is recalculated again by considering a larger pool of ICs, denoted by P_{large}, also generated following UBD. Efficient rules are deduced by ranking the retained solutions according to their performances.

5.4 Results, Analysis and Discussion

Now, we need to test out our approach where we make experiments, gather the obtained results and analyze them. Tests were implemented in Java and run on two *HP Z640* stations, each endowed with a *Xeon* processor having 20 cores and 64 GB of memory, and eight workstations, each endowed with an *i7* processor composed of eight cores and 8 GB of memory. Also, we used some dedicated tools to speed up the execution because such experiments are time-consuming operations.

5.4.1 Results

To validate the proposed approach, we constructed a subset $M' \subseteq M$ composed of some hundreds of elements with the goal of making a local search for solutions through QIEAs. The elements of set M' were generated according to three different methods: using the neighborhood of rule ϕ_{R4BZL} [25] since it is a high-performing solution, arbitrarily fixing some min-NC subrules that are selected based on experience, and finally random elements. The aim is to ensure a minimum coverage of the search space. As a result, a large number of efficient solutions could be found, where the performances of some of them could exceed 93.4% by considering ICs of size $N = 149$, while the fitness of rule ϕ_{R4BZL} [25] is about 92.4%. These findings suggest that efficient solutions for the DCT are really most likely found in set $CP_{\text{consistent}\_SAC}$.

5.4.2 Analysis of the Discovered Rules

Once the efficient solutions are retained, we need to analyze them by extracting their properties. For this purpose, we rely on two kinds of analysis: structural and behavioral. Structural analysis is based on decomposing rules into minimal components. Behavioral analysis focuses on the ability of rules to perform the density determination according to configurations of size N and the reachability to HFP.

By comparing the structures of the discovered rules, it was found that they show a strong conceptual similarity by sharing many of their subrules. This observation was also reported in [22, 25] while analyzing rules ϕ_{B33}, ϕ_{Wd01}, ϕ_{LCL}, ϕ_{ALTER1}, $\phi_{\mathrm{R4BL}_{i}=1..3}$, $\phi_{R4B_{i}=1..6}$ and ϕ_{R4BZL}. For instance, rules $\phi_{R4\mathrm{BZL}}$ and ϕ_{R4B1} are only two subrules of rule ϕ_{R4BL1}. Also, by removing separately subrules $\min - \mathrm{NC}_{i=7,12}$ from Table 4, we obtain two high-performing solutions. This explains why the performances of efficient solutions are close to each other. Hence, the determination of the best one among them may depend on small details.

Furthermore, some experiments showed that solutions of the DCT are sensitive to variations in the size of ICs [21, 24]. Therefore, we tested some of the discovered rules on ICs of both small and large sizes. As the size of ICs N was smoothly increased, it was observed that:

- Some rules recorded poor performances on small-sized ICs but they gradually improved with the growth of size N. The behavior of such rules on small-sized ICs is close to a number-conserving CA rather than the one leading to HFP.
- Some rules recorded high performances on small-sized ICs but they gradually worsened with the growth of size N. These rules are then suitable only for small-sized ICs.
- Some rules have maintained their rank in the forefront of the order regardless of size N variations. These rules represent the scalable solutions and thus show more stability with respect to ICs size N.

5.4.3 Computational Mechanisms of Efficient Solutions

According to the structure of the subrules shown in Table 4, the central cells in the generations and annihilations processes exchange their states, provided that the segment connecting them is symmetric with respect to the center, except for subrules min-$\mathrm{NC}_{i=18,19}$ that make convergence conditions appear. Also, most of the symmetrical exchanges in these subrules try to produce the pattern $(1, 0)^*$. This is an attractor that helps to gather information about the global state of the lattice, because it allows making symmetrical exchanges between cells beyond the neighborhood's boundaries, depending on the length of the pattern (see $\min - \mathrm{NC}_{17}$ in Table 3). It is noteworthy to mention that although the CAs currently considered efficient for the DCT differ in the attractors targeted, they all use a similar kind of mechanism.

The described mechanism is analogous to the one adopted by the perfect solution published in [29], in which two elementary CA rules (i.e., rules of radius $r = 1$) are applied in sequence: traffic rule ϕ_{traffic}, which is a number-conserving rule, followed by majority rule ϕ_{maj} that leads the CA to reach fixed-point patterns according to the density of cells. Both rules are applied for $N/2$ time step, where N is the size of ICs. Rule ϕ_{traffic} transforms each IC in such a way that the patterns [(1, 0)*, 0] and [1, (1, 0)*] appear, which is, in fact, interpreted as a characterization mark for the density determination. Rule ϕ_{maj} performs computations in order to reach the right HFP, starting only from regions where the characterization marks are found.

In the same way, rule ϕ_{R4BZL}—and similar rules—is itself composed of two subparts that play a role similar to rules ϕ_{traffic} and ϕ_{maj}. But since these subparts cannot be disjointedly applied at each time step t, the CA attempts not only to delay the apparition of convergence conditions but also to generate them according to the density of cells. That is why the loss of symmetry should be limited as much as possible and applied to patterns whose rates of occurrence are correlated with the current density, in particular when density $\rho \approx 0.5$. Clearly, this clarifies why using only isolated cells as a convergence condition is a good strategy, since it has the lowest rate of occurrence over ICs space.

Certainly, solving the DCT is a typical case of the emergence of global computations. Indeed, the symmetry and number-conserving properties are exploited at the microscopic level as a self-organization mechanism toward an attractor that allows perceiving the overall state of the lattice at the macroscopic level.

6. Conclusion

In this paper, an effective approach was proposed for solving the density classification task (DCT) by one-dimensional cellular automata (1D CAs) with a neighborhood of radius $r = 4$. The experiments carried out allowed retaining many new efficient solutions that may outperform the existing solutions. Also, the analysis of these rules showed that they all use the symmetry and number-conserving properties at the microscopic level to make the cellular automaton (CA) converge toward a target attractor. This latter allows perceiving the global state of the lattice in order to make the right decision. Certainly, these findings are a key element in deepening our general knowledge about both computations within CAs and the way in which local rules would be selected.

Although the proposed approach is founded on a well-defined process and provides better results compared with the other solving approaches, there are, nevertheless, some gaps to be addressed. Indeed, we still lack an effective method for automatically generating sets S and A as well as the convergence-conditions rules for which we used some intuitive assumptions. Maybe formal methods can play a primordial role in such cases. Also, it should be noted that, even if the experiments carried out relied on some dedicated configurations for parallelism, it seemed to us that is necessary to work on high-performance computing infrastructures such as clusters, grids and the cloud in order to overcome the problems stemming from running time and memory requirements. This is essential for developing effective algorithms that search for solutions in set $CP_{\text{consistent\_}SAC}$.

As a future project, we will study the mechanisms by which other tasks such as parity and synchronization are solved, in order to improve our general understanding of computations within CAs.

References

[1] A. G. Hoekstra, J. Kroc and P. M. A. Sloot, eds., *Simulating Complex Systems by Cellular Automata*, New York: Springer, 2010.

[2] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002 p. 1035.

[3] H. Betel, P. P. B. de Oliveira and P. Flocchini, "Solving the Parity Problem in One-Dimensional Cellular Automata," *Natural Computing*, **12**(3), 2013 pp. 323–337. doi:10.1007/s11047-013-9374-9.

[4] G. M. Oliveira, L. G. Martins, L. B. de Carvalho and E. Fynn, "Some Investigations about Synchronization and Density Classification Tasks in One-Dimensional and Two-Dimensional Cellular Automata Rule Spaces," *Electronic Notes in Theoretical Computer Science*, **252**, 2009 pp. 121–142. doi:10.1016/j.entcs.2009.09.018.

[5] D. Wolz and P. P. B. De Oliveira, "Very Effective Evolutionary Techniques for Searching Cellular Automata Rule Spaces," *Journal of Cellular Automata*, **3**(4), 2008 pp. 289–312.

[6] T. de Mattos and P. P. B. de Oliveira, "Guided Evolutionary Search for Boolean Networks in the Density Classification Problem," in: *Distributed Computing and Artificial Intelligence, 15th International Conference (DCAI 2018)* (F. De La Prieta, S. Omatu and A. Fernández-Caballero, eds.), Cham, Switzerland: Springer, 2019 pp. 69–77. doi:10.1007/978-3-319-94649-8_9.

[7] M. Mitchell, P. T. Hraber and J. P. Crutchfield, "Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations," *Complex Systems*, **7**(2), 1993, pp. 89–130. complex-systems.com/pdf/07-2-1.pdf.

[8] W. D. Abilhoa and P. P. B. de Oliveira, "Density Classification Based on Agents under Majority Rule: Connectivity Influence on Performance," in *Distributed Computing and Artificial Intelligence, 16th International Conference (DCAI 2019)* (F. Herrera, K. Matsui and S. Rodríguez-González, eds.), Cham, Switzerland: Springer, 2020 pp. 163–170. doi:10.1007/978-3-030-23887-2_19.

[9] M. Montalva-Medel, P. P. B. de Oliveira and E. Goles, "A Portfolio of Classification Problems by One-Dimensional Cellular Automata, over Cyclic Binary Configurations and Parallel Update," *Natural Computing*, **17**(3), 2018 pp. 663–671. doi:10.1007/s11047-017-9650-1.

[10] P. Gacs, G. Kurdyumov and L. Levin, "One-Dimensional Homogeneous Media Dissolving Finite Islands," *Problems of Information Transmission*, **14**(3), 1978 pp. 92–96.

[11] D. Andre, F. H. Bennett, III and J. R. Koza, "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 First Annual Conference on Genetic Programming*, Stanford University, 1996 (J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo, eds.), Cambridge, MA: MIT Press, 1996 pp. 3–11. dl.acm.org/doi/10.5555/1595536.1595538.

[12] R. Das, M. Mitchell and J. P. Crutchfield, "A Genetic Algorithm Discovers Particle-Based Computation in Cellular Automata," in *International Conference on Parallel Problem Solving from Nature (PPSN 1994)* (Y. Davidor, H. P. Schwefel and R. Männer, eds.), Berlin, Heidelberg: Springer. 1994 pp. 344–353. doi:10.1007/3-540-58484-6_278.

[13] H. Juille and J. B. Pollack, "Coevolving the 'Ideal' Trainer: Application to the Discovery of Cellular Automata Rules," in *Genetic Programming 1998: Proceedings of the Third Annual Conference* (J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba and R. Riolo, eds.), University of Wisconsin, Madison, WI, San Francisco: Morgan Kaufmann, 1998.

[14] C. Ferreira, "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems," *Complex Systems*, **13**(2), 2001 pp. 87–129. complex-systems.com/pdf/13-2-1.pdf.

[15] M. Márques, R. Manurung and H. Pain, "Conceptual Representations: What Do They Have to Say about the Density Classification Task by Cellular Automata?," *Computational Mechanics*, 2006 pp. 1–15.

[16] M. Marques-Pita, and L. M. Rocha, "Conceptual Structure in Cellular Automata: The Density Classification Task," in *Proceedings of the Eleventh International Conference on Artificial Life (Alife)*, Cambridge, MA: MIT Press, 2008 pp. 390–397.

[17] P. P. B. de Oliveira, J. C. Bortot and G. M. Oliveira, "The Best Currently Known Class of Dynamically Equivalent Cellular Automata Rules for Density Classification," *Neurocomputing*, **70**(1), 2006 pp. 35–43. doi:10.1016/j.neucom.2006.07.003.

[18] J. Kari and B. Le Gloannec, "Modified Traffic Cellular Automaton for the Density Classification Task," *Fundamenta Informaticae*, **116**(1–4), 2012 pp. 141–156. doi:10.3233/FI-2012-675.

[19] B. Le Gloannec, “Around Kari’s Traffic Cellular Automaton for the Density Classification,” Project report, École Normale Supérieure de Lyon, Lyon, France, 2009.

[20] M. Marques-Pita, M. Mitchell and L. M. Rocha, “The Role of Conceptual Structure in Designing Cellular Automata to Perform Collective Computation,” in *Unconventional Computing (UC 2008)* (C. S. Calude, J. F. Costa, R. Freund, M. Oswald and G. Rozenberg, eds.), Berlin, Heidelberg: Springer, 2008 pp. 146–163. doi:10.1007/978-3-540-85194-3_13.

[21] Z. Laboudi, S. Chikhi and S. Lakhdari, “Scalability Property in Solving the Density Classification Task,” *Journal of Information Technology Research*, **10**(2), 2017 pp. 60–76. doi:10.4018/JITR.2017040104.

[22] Z. Laboudi and S. Chikhi, “Computational Mechanisms for Solving the Density Classification Task by Cellular Automata,” *Journal of Cellular Automata*, **14**(1–2), 2019 pp. 69–93.

[23] Z. Laboudi and S. Chikhi, “New Solutions for the Density Classification Task in One Dimensional Cellular Automata,” in *Modelling and Implementation of Complex Systems (MISC 2018)* (S. Chikhi, A. Amine, A. Chaoui and D. Saidouni, eds.), Cham, Switzerland: Springer, 2019 pp. 93–105. doi:10.1007/978-3-030-05481-6_7.

[24] P. P. B. De Oliveira, “On Density Determination with Cellular Automata: Results, Constructions and Directions,” *Journal of Cellular Automata*, **9**(5–6), 2014 pp. 357–385.

[25] Z. Laboudi, “An Effective Approach for Solving the Density Classification Task by Cellular Automata,” in *2019 4th World Conference on Complex Systems (WCCS)*, Ouarzazate, Morocco, 2019, pp. 1–8. doi:10.1109/ICoCS.2019.8930805.

[26] M. Land and R. K. Belew, “No Perfect Two-State Cellular Automata for Density Classification Exists,” *Physical Review Letters*, **74**(25), 1995 pp. 5148–5150. doi:10.1103/PhysRevLett.74.5148.

[27] K.-H. Han and J.-H. Kim, “Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization,” *IEEE Transactions on Evolutionary Computation*, **6**(6), 2002 pp. 580–593. doi:10.1109/TEVC.2002.804320.

[28] Z. Laboudi and S. Chikhi, “Comparison of Genetic Algorithm and Quantum Genetic Algorithm,” *International Arab Journal of Information Technology*, **9**(3), 2012 pp. 243–249.

[29] H. Fuks, “Solution of the Density Classification Problem with Two Cellular Automata Rules,” *Physical Review E*, **55**(3) 1997 pp. 2081–2084. doi:10.1103/PhysRevE.55.R2081.