Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2006) 35–43
NEUROCOMPUTING
www.elsevier.com/locate/neucom

The best currently known class of dynamically equivalent cellular automata rules for density classification

Pedro P.B. de Oliveira[a,*], José C. Bortot[b], Gina M.B. Oliveira[c]

[a]Universidade Presbiteriana Mackenzie - Faculdade de Computação e Informática, Rua da Consolação 896, Consolação; 01302-907 São Paulo, SP, Brazil
[b]Fundação Armando Álvares Penteado - Faculdade de Computação e Informática, Rua Alagoas 903, Higienópolis; 01242-902 São Paulo, SP, Brazil
[c]Universidade Federal de Uberlândia - Faculdade de Computação, Av. João N. de Ávila 2160, Bloco B, Campus Santa Mônica; 38400-902 Uberlândia, MG, Brazil

Available online 18 August 2006

Abstract

The possibility of performing computations with cellular automata (CAs) opens up new conceptual issues in emergent computation. Driven by this motivation, a recurring problem in this context is the automatic search for good one-dimensional, binary CA rules that can perform well in the density classification task (DCT), that is, the ability to discover which cell state outnumbers the other state. In the past, the most successful attempts to reach this target have relied on evolutionary searches in the space of possible rules. Along this line, a multiobjective, heuristic evolutionary approach, implemented as a distributed cooperative system, is presented here, which yielded outstanding results, including a rule that led to the characterisation of a class of four equivalent rules, all of them with the best performance currently available in the literature for the DCT.

Keywords: Cellular automata; Evolutionary multiobjective optimisation; Density classification task; Nondominated sorting genetic algorithm; Dynamic behaviour; Emergent computation

1. Introduction

The possibility of performing computations with cellular automata (CAs) opens up new conceptual issues in emergent, artificial life type forms of computation, and opens up alleys of novel technological advances. Regardless of more than four decades of cellular automata research, not many methods have been developed to make CA programming possible. Nevertheless, chiefly among them, evolutionary computation techniques have been used in the design of CA rules that perform a predefined computation over an initial condition given to it as input.

In this context, a well studied computation is the *density classification task* (DCT), for which various evolutionary computation techniques have been used to look for CA rules that could solve it [1,20,15,35,24]; basically, this task is to find out in a binary sequence which cell state outnumbers the other state, a trivial task for any system with centralised control, but a daunting one for a CA, because of its totally decentralised, local and parallel processing nature. Even with evolutionary techniques, the enormous size of CA rule spaces is a serious burden to be overcome, since searches in them become slow and limited.

In [24] a set of static parameters (i.e., directly derived from the CA rule table) was proposed, aiming at the reduction of the latter problem, as they are used as an auxiliary metric to guide the processes underlying the genetic search. In this approach the parameter-based heuristic was incorporated into the GA in two aspects: in the fitness function and in the genetic operators (of crossover and mutation). The fitness function of a cellular automaton rule was made by the weighed sum between a fitness component due to its efficacy in a sample of initial configurations (ICs), with a second fitness component that represents the bias due to the parameter-based heuristic [24]. Although this solution yielded an improvement on the genetic search, the weight of the heuristic in the rule evaluation interferes on this performance.

*Corresponding author. Tel.: +55 11 3236 8711; fax: +55 11 3236 8600.
E-mail addresses: pedrob@mackenzie.br (P.P.B. de Oliveira), jbortot@uol.com.br (J.C. Bortot), gina@facom.ufu.br (G.M.B. Oliveira).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.07.003

Instead of using a weighted sum to evaluate the rule, we developed later a multiobjective evolutionary approach [23]. Accordingly, the heuristic value and the efficacy in the IC sample were kept separate, as independent objectives to be followed by the genetic search. With such a setting we were then able to achieve rules with higher performance than we had been able until then, using population size of 200 individuals.

The use of multiobjective evolutionary search in the present context extends the latter effort with the same algorithm, but now with populations of 1000 individuals, made possible by a distributed and cooperative computational architecture that is introduced herein, as a way to support highly demanding evolutionary searches.

If separately considered, the bits and pieces of the whole endeavour have already been shown not to lead to the best possible result of the problem being tackled. Thus, originality herein resides in the actual computational architecture, in the heuristic information obtained from the several very good rules we found for the DCT, and on the analyses made to discriminate between the top ones. In fact, it is the joint operation of them all that shows off their real thrust; after all, the main result of the paper is that our approach allowed the remarkable achievement of characterising the best CA rules currently known for the density classification task, a problem that had been proposed in 1978, whose best rule known until now, due to Juillé and Pollack [15], is shown herein to be just one out of four equivalent possibilities.

The remainder of the paper goes as follows. The next section introduces the notions associated with the multiobjective evolutionary algorithm we used, the nondominated sorting genetic algorithm (NSGA) [23]. The subsequent section then introduces cellular automata and the density classification task, and discusses various issues related to them. Then, the computational architecture we designed is presented, and the experiments carried out are described. Finally, after showing the results, which make evident its very significant overall performance, we make the case about our claim that the best rule we found in the experiments defines, indeed, a class of four equivalent rules with the best performance currently available in the literature.

2. Multiobjective evolution with NSGA

The NSGA was proposed by Srinivas and Deb [30] as a way to tackle optimisation problems with multiple objectives. It is based on the concept of nondomination. Suppose that there are N objectives $f_1, f_2, \ldots, f_N$ to be simultaneously optimised. A solution A is said to be dominated by another solution B, or B dominates A, if B is better than A in relation to at least one of the objectives f_i, and is better than or equal to A in relation to all other objectives $f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_N$. Two solutions A and B are nondominated in relation to each other if A does not dominate B and B does not dominate A. The so-called Pareto optimum is the set of nondominated solutions considering the entire search space, that is, any candidate solution that is better than those from the Pareto optimum with respect to one of the objectives, is guaranteed to be worse with respect to another objective.

The basic difference of NSGA in relation to a simple GA is the way in which the individuals are evaluated. In a nutshell, in order to obtain the fitness value of an individual, instead of using the fitness components associated with each objective involved in the problem, these components are used to rank the individuals according to their degree of domination over the others in the population; it is this measure of domination that is used as the fitness value that guides the action of the genetic operators and the selection process.

Before selection, a classification is performed based on several layers of nondomination. Initially, all individuals in the population that are nondominated are separated as the first, outer layer, and a (dummy) fitness value is assigned to them, whose role is simply to characterise their degree of domination over the others in the population, as mentioned above; for the outer layer, the highest fitness value is assigned to the individuals in it that exhibit the highest degree of domination (the actual value assigned is proportional to the population size; other details do exist but, for brevity, are omitted here). Then, the remaining individuals are classified again, also based on the nondomination criterion, and the second layer is formed with a (dummy) fitness lower than the first one. This process continues until all individuals are classified in their respective layers.

Once the nondomination layers were obtained, in order to maintain the diversity of the individuals inside each layer, a sharing function method is used to assign fitness to each individual. The basic idea of the sharing function is to reduce the fitness of the individuals that have many neighbours in the same layer. The fitness reduction is by a factor that depends on the number and the proximity of neighbouring individuals.

After the nondomination classification and fitness assignment are performed, stochastic remainder proportional selection is used so that the outer the layer an individual is in, the likelier its chance to reproduce. The other steps of the NSGA are very similar to the simple GA [12].

Therefore, no elitist strategy is used in NSGA, in the sense of an external archive that would retain the best nondominated solutions found along the evolutionary process, which would directly pass on to the next generation, regardless of how well fitted those individuals might already be; and this is in contrast with various more recent, elitist, multiobjetive evolutionary methods, such as NSGA-II [7,6], SPEA, SPEA2, PAES and PESA [6].

The fact that we centred the multiobjective approach herein around a nonelitist method, instead of using its more recent, more efficient, less costly, elitist counterpart, NSGA-II [7], derives from the problem being tackled.

The point is that, in the context of evolving CA to perform the DCT, the single-objective genetic algorithm presented in [20]—that has been used to compare the different approaches to the problem [1,20,15,35,24,23]—relies on a very elitist strategy for selection, both for crossover and reinsertion of individuals in the population; hence, to maintain all the basic structure of that reference GA was a natural constraint, at the same time that using a nonelitist multiobjective algorithm seemed a sensible decision, so as not to overload the resulting algorithm as far as elitism is concerned. But now that the simpler, and to some extent naïve multiobjective approach we adopted, has shown to be successful, as is shown in this paper, changing the actual multiobjective algorithm to a more sophisticated and well-established one, is clearly a natural step ahead.

3. Cellular automata

Cellular automata [36] are discrete complex systems possessing both a dynamic and a computational nature, basically consisting of an n-dimensional lattice and a transition rule. The lattice is a regular arrangement of cells, each one with an identical pattern of local connections to other cells, and subjected to some boundary conditions, usually periodic. Each cell can take on one of a discrete set of possible states, and the neighbourhood of a cell is defined as the cell, together with the others that are connected to it.

The transition rule yields the next state for each cell, as a function of its neighbourhood and, at each time step, all cells synchronously have their states updated.

In computational terms, a cellular automaton is, therefore, an array of finite automata, where the state of each automaton depends on the state of its neighbours.

For one-dimensional CAs, the size m of the neighbourhood is usually written as $m = 2r + 1$, where r is called the *radius* of the automaton. In the case of binary-state CAs, the transition rule is given by a state transition table that lists each possible neighbourhood together with its output bit, i.e., the updated value for the state of the central cell in the neighbourhood. Fig. 1 gives an example, with rule 137 of the space of all one-dimensional cellular automata rules with 2 states per cell and radius 1 (the so-called elementary space), in which black cells represent state 1, and white cells represent 0.

The rule is denominated (the decimal number) 137, since it corresponds to the binary number formed by the output bits of its rule table, neighbourhood $111\ldots1$ being on the left-hand side, as shown in the figure; such a naming convention is widely used in the literature, for any radius, so that we preserve it herein.

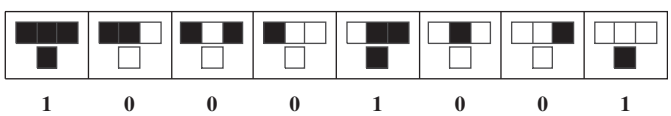


Fig. 1. Example of a cellular automaton rule and the naming convention used here. This is rule 137, since the binary representation for the decimal 137 is 10001001.

The dynamics of a cellular automaton is associated with its transition rule. In order to help forecast the dynamic behaviour of CAs, several parameters have been proposed, directly calculated from their transition table [24,17,18,3].

Along this line, a set of five parameters was used in [24]. In the experiments involving the DCT task described in the next section, two parameters from that set were used: sensitivity [3] and activity propagation [24]. All of them have been normalised between 0 and 1, for one-dimensional CAs with any radius, and have been formally defined in [24]. For the present purpose it suffices to know that these were the only parameters used; the actual details about their usage are being omitted here, but the reader can refer to [24,23] for details.

In respect to the dynamical behaviour of CAs, it is important to notice that the rules of a certain family—say, all possible one-dimensional, binary rules with the same radius—can be partitioned into *classes of equivalent dynamical behaviour*. This is achieved by changing all 0s to 1s in a rule table (the black–white transformation), by reversing all neighbourhoods while preserving the original output bit they originally lead to (the left–right transformation), and by doing the latter two in combination. As a consequence, equivalence classes are formed with 4, 2 or a single rule. Fig. 2 illustrates the dynamical equivalence class for elementary rule 137, formed by rules 110, 193 and 124, obtained after each of the previous transformations, respectively.

4. The density classification task (DCT)

As mentioned earlier, one of the most widely studied computational problems in the context of cellular automata is the *density classification task*. In its standard formulation it states that a binary, one-dimensional CA has to converge to a final configuration of all cells in state 1, when the initial configuration has more 1s than 0s, and to a configuration of all 0s, whenever the initial configuration has more 0s than 1s. The problem usually does not specify what should happen to the CA if the initial configuration has as many 1s as 0s, although one could require (as sometimes happens in the literature) that, in this case, a specific final configuration also has to be achieved, for instance, a sequence of 01 blocks. While solving the DCT is a trivial task for any centralised computational system, it is a daunting task for any fully distributed system, with local processing, as a cellular automaton.

Although the DCT was proposed in 1978 [9], it was not until 1995 that the problem, as formulated, was proven not to be possible [16].

Before the original proof was derived a few groups were trying to find a rule that could solve the DCT. However, with such a possibility precluded the search shifted towards looking for the rule that could solve the DCT as perfectly

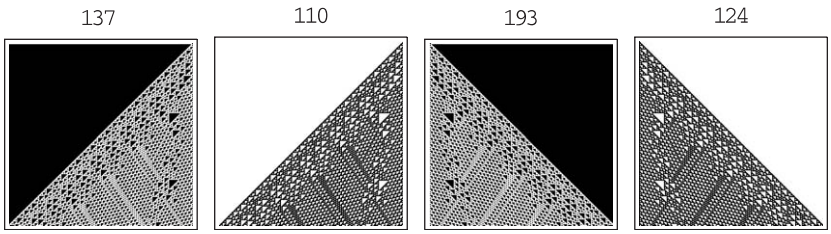


Fig. 2. Dynamical equivalence among CA rules. For example, dynamical equivalent rules for elementary rule 137 are obtained trough the transformations: black–white (110), left–right 193, and the combination of both (124).

as possible. Using different search techniques, mostly evolutionary computation-based methods, various groups have put their methods at test, and over time good rules have been found. For historical reasons, the search methods employed have been scanning the huge radius 3, 2-state CA space, the best rule found so far (a fact that is reappraised herein) being due to Juillé and Pollack [15], with a score of about 86%, considering CA lattices with odd lengths; the second best rule published in the literature is also due to these authors, its score being a bit less than 85%. The fact is that, regardless of empirical and analytical efforts carried out during the last few years, the best imperfect rule for the DCT remains unknown.

5. Experiments with the cooperative NSGA

As hinted at earlier, our main motivation in the present work was the possibility of using a large population size of 1000 individuals, since we were unable to in previous efforts, due to the high computational demand of an evolutionary search of that order. The idea then of devising a multi-processed alternative to the NSGA was a natural step to try.

It is well known that evolutionary algorithms are very amenable to various forms of parallelisation, in particular in the fitness evaluation and in the action of the genetic operators [5,13]. In tune with that, after considering a few possibilities [32,11,21,14], we developed a distributed computational system, named the Cooperative NSGA Environment, as presented in Fig. 3.

Basically, the Cooperative NSGA Environment is composed of three units, described below, and Fig. 4 depicts a flowchart with the interactive actions between two of them, the NSGA Server and Clients.

1. The *NSGA Client* is replicated in all machines in the local area network, as many as possible, so as to yield the required data processing power. The client connects to the NSGA Server, offering its services to the server, mainly the evaluation of an individual against the ICs generated by the NSGA Server, which is the actual bottleneck of the whole process. Optionally, the NSGA Client may also be responsible for the genetic operations of crossover and mutation of the individuals controlled and served by the NSGA Server; this option is defined in the configuration file of a run (kept in the NSGA Server), as represented in Fig. 4, by the dotted area.
2. The *NSGA Server* is responsible for the distribution of tasks to the Clients, as they step forward offering their cooperative processing power. The NSGA Server is also responsible of some centralised processing procedures like the generation of ICs, the selection of individuals for the subsequent generations, and the selection processes for crossover and mutations. Recovery and restart is also the responsibility of the NSGA Server. The NSGA Server has many threads that are responsible for attending requests from the NSGA Clients, and all of them share the cooperative control table (CCT), whose role is to ensure the system robustness, as it logs all tasks being processed in all clients. Then, at each generation, after 90% of all computations have been concluded, the NSGA Server looks up the CCT, trying to identify any client who may have been discontinued and, if this is the case, it redirects the corresponding failed task to the next available client.
3. The *Seed Server* is primarily in charge of generating the random seeds that are used to the Clients. However, it also does various bookkeeping actions related to the experiments, such as keeping track of all good rules obtained during a run.

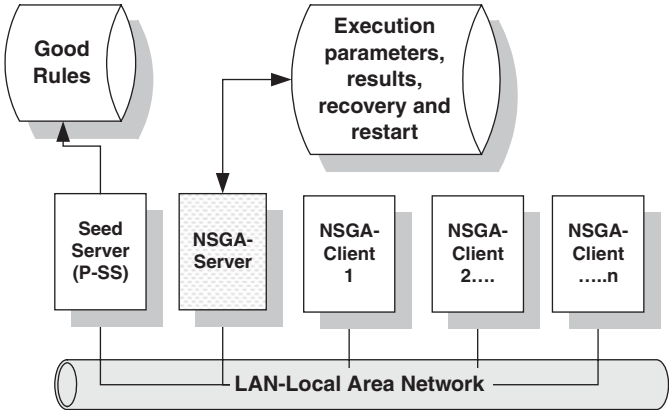


Fig. 3. The NSGA cooperative environment.

With the latter computational environment the NSGA experiments were carried out, following the specifications presented in Table 1. Very significantly, 43 rules were

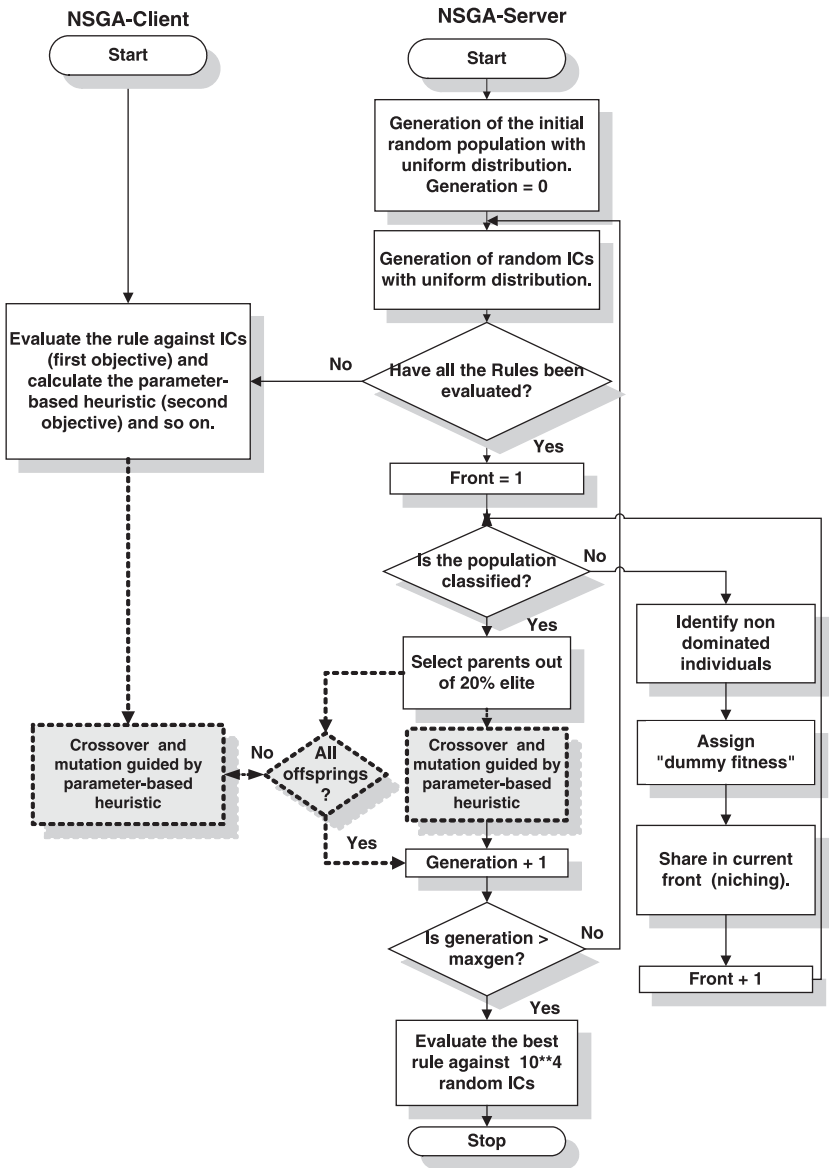


Fig. 4. The NSGA Server and Client.

Table 1
Characterisation of the NSGA experiments

Feature of the GA or of the CAs	Definition
Population size	1000
Length of a GA run	3000 and 5000 generations
Initial configurations per generation	1000 ICs
Initial configurations for final evaluation	10000 ICs
Number of GA runs	20
Elitism	20%
Selection (for crossover)	Randomly from the elite
Crossover	100%
Mutation	2% per bit
Radius (r) of the CAs	3
Lattice size	149 cells
Length of the CA temporal evolutions	300 timesteps
Objectives	2 or 3: fitness + (1 or 2 parameters in the heuristic)

found with efficacy in between the best published rules until then, referred to herein as JP1 and JP2, whose scores are around 86.0% and 85.0%, respectively [15]. In addition, more than 10 rules we found scored higher than 85%; some of them are shown in the upper part of Table 2 (excluding the bottom row), in the decimal notation defined in Section 3. So, although very good rules had been found, none of them had outperformed JP1.

At the vantage point of having such an outstanding set of rules, a comparative analysis was undertaken with them, with a focus on the bits in the binary representation of their numbers. Eventually this led to the observation of some common patterns:

- Some bits were *invariant* in all the rules.
- All bit sequences were *balanced*, that is, the number of 0s was the same as the number of 1s.

Table 2
The best DCT rules and their scores with 10 000 initial conditions, where their number of 1s is randomly generated through Binomial distribution

Rule ID	Rule number	Efficacy (%)
JP1	333572843222032047479422407594398747176	86.09
JP2	311868750708919091743697397865327110272	84.70
BOO2	331865637541090131129871359445180088544	85.97
BOO3	313256455741306110733900351070274060512	85.39
BOO4	330477941466437280680722581515944915968	85.38
BOO5	338729805509742283078328733596895710728	85.35
BOO1	330977303262886499262089477799035316776	86.16

- There was a *high degree of symmetry*, of a certain kind, among the bits. By symmetry in this case we mean the following: assuming that a $(2n)$-bit long, binary, one-dimensional cellular automaton rule is represented by the binary sequence $b_{-n}b_{-(n-1)}\ldots b_{-2}b_{-1}b_1b_2\ldots b_{n-1}b_n$, the symmetry degree can be measured by the number of bit positions in which the bit with the i-index, is opposite to the bit with the $(-i)$-index. Among the good rules we found such a symmetry measure was noticed to be in the range from 45 to 52, where the maximum possibility would be 64.

By analysing the previous information, a heuristic was then conceived that integrated them all. With such a heuristic information, a new version of our NSGA setting was devised and executed, which took into account in the operation of crossover and mutation, by extending their biased implementation (already in use, through the parameter-based heuristic hinted at earlier) with the new heuristic information derived from the very good rules we had found. Such heuristic was embedded in the genetic operators by imposing the additional bias in their action that the offspring allowed to be generated would also have to abide by the three heuristic observations above; those not following the impositions would simply be discarded and the operators would then make another attempt. As a result, a new rule was found, denominated here BOO1, with efficacy of 86.16%, which is also shown in Table 2, at the bottom row. And this rule was suggestive of being the best rule ever found for the density classification task.

6. When two are one: analysis of the best rules found

Although BOO1 outperformed JP1 over 10000 ICs, randomly generated with Binomial distribution of the number of 1s in them, experience shows that this is not a robust sample size. Therefore, in order to trustfully compare the scoring ability of the two rules, a much more significant test suite was conceived and the rules were subjected to them.

Lattice size of 149 cells was used, updated for 400 timesteps in the CA evolution. In addition to the ICs being generated out of a Binomial distribution of their number of 1s, another set was also used, of *nearly-balanced* binary sequences. This means that either the number of 1-bits is outperformed by a single 0, or the number of 0-bits is outperformed by a single 1. This distribution was used because it is known that this is the most difficult type of initial configuration for CA rules in the DCT. In fact, this can be readily realised by the dramatic decrease in performance of both JP1 and BOO1 rules, which dropped to the level of about 57% efficacy.

Furthermore, in addition to using two types of datasets, their cardinalities were also taken very seriously. Hence, instead of the few thousands that are common in the literature, 16 million ICs of each kind (8 million for each) were employed to evaluate each rule, in 16 trials defined by blocks of 1 million, independently generated for each rule.

What has been observed was that the differences in performance between BOO1 and JP1 were small—the values differing in the first or second decimal places—but consistent, indicating the supremacy of BOO1; even when considering the 16 individual blocks of 1 million ICs, the same pattern remained. With these results in mind it became very suggestive that BOO1 was a better rule than JP1 (in fact, we even announced this result officially in a conference [4]).

But another issue came out that had to be verified: it could be that the observation above might have been a statistical artefact derived from the use of distinct blocks of 1 million ICs for each rule evaluation. In order to probe the latter, a new test suite was carried out, now using blocks of 1 million ICs that would be used for all the rules in a trial. This time the tests relied only upon Binomially generated ICs, totaling 10 million of them. But, surprisingly, a new pattern emerged, as JP1 slightly outperformed BOO1.

After such an extensive statistical sampling the fluctuations observed could only be explained by the following, nonmutually exclusive alternatives: the sample size was still smaller than it should be, or there was a strong correlation between the rules, so that only rarely (that is, only for some rare type of IC) they would perform differently, with a consequent small fluctuation in performance.

The former alternative seemed quite reasonable in the present case, since the space of possible ICs is enormous, with $2^{149} \approx 7.1 \times 10^{44}$ possibilities; so, in order to sample such a space, the sample size would definitely have to be very large.

But the latter also appeared possible; after all, in the last experiment and in the previous one, the standard deviations over the set of trials (10 in the latter, and two sets of eight trials in the former) always remained under 650 (usually around 450), thus indicating a very small spread of the results.

By examining the binary representations of BOO1 and JP1 more closely, significant bit symmetries sprung out, which prompted us to check each other's dynamical equivalence class. In doing so, it became apparent that, quite surprisingly, BOO1 and JP1 do belong to the same class of dynamical equivalence. Table 3 clarifies such a

Table 3
BOO1 and JP1 belong to the same dynamical equivalence class

BOO1 class	JP1 class	Rule number
BOO1 ≡	BWLR[JP1]	330977303262886499262089477799035316776
BW[BOO1] ≡	LR[JP1]	313256445605053069064990487414399566048
LR[BOO1] ≡	BW[JP1]	313276250114949209747720368944870863008
BWLR[BOO1] ≡	JP1	333572843222032047479422407594398747176

relation, assuming that, if R_{CA} is the denomination of a one-dimensional, binary CA rule, then BW[R_{CA}], LR[R_{CA}] and BWLR[R_{CA}] represent the rules dynamically equivalent to R_{CA}, derived, respectively, from the black–white transformation of R_{CA}, the left–right transformation of R_{CA}, and the combination of both (as explained in Section 2).

Testing BOO1 and JP1 over all 2^{15} possible initial conditions of size 15, they yielded the very same performance, thus suggesting what is at issue in the case. The point is that, whenever an IC sample is closed in respect to all the binary sequences that can be formed out of the three symmetry transformations (BW, LR and BWLR), all the rules in a dynamical equivalence class have to perform the same over the set of ICs. Naturally, this is what happens when all 2^N possible ICs of size N are taken into account, as in the example above, or when the IC sample is formed out of any group of binary sequences, together with those obtained from the three transformations of the original group (and excluding the occasional repetitions). Further to this rationale, various extensive empirical tests—with several dynamical classes (including all the 88 classes of the elementary space) and types of IC samples—led to the same conclusion. It is worth remarking that, in spite of a number of studies related to the DCT already published, this is the first time such a realisation has been brought about.

Now, once settled that BOO1 and JP1 have the same performance, the question remained of which exactly is that value. Instead of statistically sampling the space, we performed the exhaustive evaluation of one of the rules in the equivalence class, over all possible initial configurations of a given size. The idea was to start with a small lattice size and go on increasing it until convergence in the measured rule performance could be noticed; at this point non-linear regression methods could be used to extrapolate the limit performance. The nice feature of this approach is that the resulting performance would not be constrained by the type of probability distribution assumed for generating the ICs, thus standing a much higher chance of accuracy. Although enumerative searches tend to be disregarded as a possibility, many recent discoveries in emergent computation have been made possible through enumerative methods [36].

Fig. 5 displays the outcome of the experiment. The dots in the graph represent the increase in absolute performance of the rule, over all possible initial conditions of odd length, from 13 to 35 bits. The data was obtained through runs of a *Mathematica* program [37] for about 4 months, half the time under seven 64-bit AMD Opteron processors, family 24x, and the other half (for lattice size 35 alone) under fifteen Opterons.

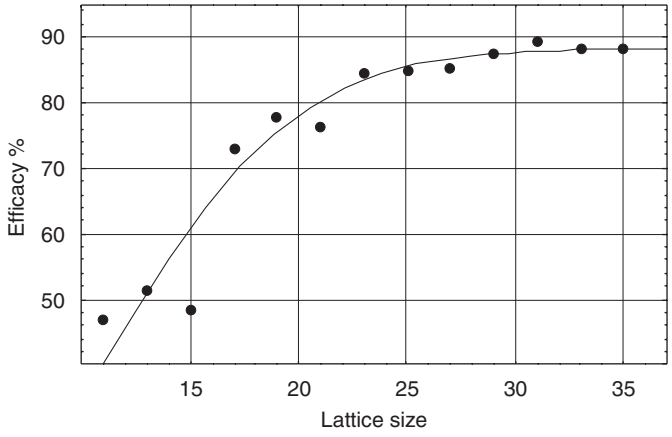


Fig. 5. Limit performance of BOO1's dynamical equivalence class, over all initial configurations of a given lattice size.

Although oscillations can be observed for the smaller lattice sizes, they diminish as the lattice size increases. This is possibly due to a combination of two effects: the preclusion, in the temporal evolution of the CAs, for small lattice sizes, of certain interactions involving particular types of blocks of bits, and the varying degree of effectiveness of the DCT solution strategy employed by BOO1 (or the others in its equivalence class). Explaining the very displaced position of the size 15 performance, for instance, would be an extremely interesting but far from trivial task.

The curve also displayed in the figure represents the inferred approximation of the rule performance and was generated from within *Mathematica*'s own non-linear regression capabilities. Two functions were used to fit the data (a second order polynomial and another, based on hyperbolic tangent, which is the actual curve shown in Fig. 5), both yielding similar results just below 88.3%. This figure can, therefore, be regarded as a fair estimate for the limit performance of BOO1-JP1 class, over all possible initial conditions of any given size.

7. Concluding remarks

The automatic design of cellular automata rules by means of evolutionary algorithms is an active area of research, with efforts aimed at finding CA rules with predefined phenomenological characteristics in their temporal evolution [2,22,27], amenable to perform in specific real-world applications [10,25,31], or that have predetermined computation-theoretic features [8,26,28]; evolutionary algorithms have also been used in the design of non-standard forms of CAs, such as those with asynchronous update [34] or non-local rules [33], and for designing spatial arrangements of CA rules [29] or ways of sequentially combining a set of rules, so that a preconceived end result is achieved [19]. Regardless of the case, the algorithms have varied significantly from one case to another, indicating that the problems and CA spaces concerned have generally imposed their particular demands.

Multiobjective evolutionary optimisation using the nondominated sorting genetic algorithm (NSGA), implemented in the distributed cooperative computational environment introduced herein, was very effective for performing a successful search in the space of possible cellular automaton rules for the density classification task. And this happened even considering that NSGA has already been displaced in the literature by better approaches, such as NSGA-II, the elitist counterpart of the algorithm we used, and natural candidate for replacing it here.

The success of the approach reported was made evident through the various very effective rules that came out of the search, with scores higher than 85%, thus outperforming JP2, the second best rule available in the literature.

With these good rules at hand, it was then possible to use them as guides for the subsequent search that eventually led to BOO1. Contrarily to what statistical analysis initially suggested, BOO1 was shown not to outperform JP1; instead, they belong to the same dynamical equivalence class, thus bringing forward two other very good rules that have not yet been found in the literature.

An estimate of the performance of a given CA rule can be achieved by statistically sampling the space of possible ICs. But while this procedure is reasonable for a rough estimate, a faithful absolute estimate that would allow deciding between rules at a very high performance level can be very misleading. With the experiments described herein, and the emphasis on their historical sequence, we hope to have clarified the latter point, so that future endeavours of a similar kind can handle such an issue much more seriously than it has been accounted for until now in the literature.

This work extended our continuing past efforts in the general issue of devising ways to automatically design cellular automaton rules that are capable of implementing predefined computational tasks. The results we have obtained along time, now crowned by the conceptual achievement of realising the equivalence between the best rules known to this date, thus make us believe that the approach we are pursuing is worth being sustained, even though alternative ways might be thought of [2,22,27,10,25,31,8,26,28,34,33,29,19].

Acknowledgements

We are grateful to MackPesquisa, Instituto Presbiteriano Mackenzie's research funding programme, for their continuing support to this work, in particular through a research grant from Edital 2004. Thanks also to D.A. 'Progdan' Alves for his helpfulness with part of the computational infrastructure we relied upon. P.P.B.O. thanks Wolfram Research for the Mathematica Academic Grant No. 1149, and FAPESP, São Paulo State Foundation for Research Support, for the Grant Proc. 2005/04696-3, and G.M.B.O. thanks CNPq, National Council for Research and Development, for the Grant Proc. 300273/02-9.

References

[1] D. Andre, F. Bennett III, J. Koza, Discovery by genetic programming of cellular automata rule that is better than any known rule for the majority classification problem, in: J.R. Koza, D.E. Goldberg, D.B. Fogel, R.L. Riolo (Eds.), Genetic Programming 96: Proceedings of the First Annual Conference, MIT Press, Cambridge, MA, 1996, pp. 3–11.

[2] E. Bilotta, A. Lafusa, P. Pantano, Searching for complex CA rules with GAs, Complexity 8 (3) (2003) 56–67.

[3] P.M. Binder, A phase diagram for elementary cellular automata, Complex Systems 7 (1993) 241–247.

[4] J.C. Bortot, P.P.B. de Oliveira, G.M.B. Oliveira, Multiobjective, heuristic evolutionary search in a cooperative environment leads to the best cellular automaton rule in the density classification task, in: A. Barros, A. Araújo, H.C. Yehia, R. Teixeira (Eds.), Proceedings of VIII Brazilian Symposium on Neural Networks, IEEE Press/SBC, São Luís, MA, Brazil, CD-ROM:Paper 3565, 2004.

[5] E. Cantú-Paz, A summary of research on parallel genetic algorithms, IlliGal Report 95007, (July, 1995).

[6] C.A.C. Coello, D.A. Van Veldhuizen, G.B. Lamont, Evolutionary Algorithms for Solving Multi-objective Problems, Kluwer Academic Publishers, New York, 2002.

[7] K. Deb, A. Pratab, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[8] C. Ferreira, Discovery of the Boolean Functions to the Best Density-Classification Rules Using Gene Expression Programming, in: Lecture Notes in Computer Science, vol. 2278, 2002, pp. 51–60.

[9] P. Gacs, G.L. Kurdyumov, L.A. Levin, One-dimensional uniform arrays that wash out finite islands, Probl. Peredachi Inf. 14 (1978) 92–98 Apud [20].

[10] N. Ganguly, A. Das, P. Maji, B.K. Sikdar, P.P. Chaudhuri, Evolving cellular automata based associative memory for pattern recognition, in: Proceedings of the International Conference on High Performance Computing, Hydrabad, India, 2001.

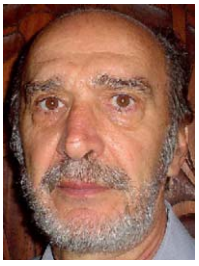
[11] B. Glick, Grid computing could help fight against cancer, ⟨http://www.computing.co.uk/Analysis/1136055⟩. Accessed on Oct-18-2002.

[12] D.E. Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[13] J.J. Grefenstette, Parallel adaptive algorithms for function optmization, Technical Report CS-81-19, Vanderbilt University Computer Science Department, Nashville, 1981.

[14] D. Grune, H.E. Bal, C.J.H. Jacobs, K.G. Langendoen, Modern Compiler Design, Wiley, New York, 2000.
[15] H. Juillé, J.B. Pollack, Coevolving the "ideal" trainer: application to the discovery of cellular automata rules, in: J.R. Koza, W. Banzhaf, K. Chellapilla, M. Dorigo, D.B. Fogel, M.H. Garzon, D.E. Goldberg, H. Iba, L. Riolo (Eds.), Genetic Programming 1998: Proceedings of the Third Annual Conference, Morgan Kaufmann, San Francisco, CA, 1998.
[16] M.W.S. Land, R.K. Belew, No two-state CA for density classification exists, Phys. Rev. Lett. 25 (1995) 5148–5151.
[17] C.G. Langton, Computation at the edge of chaos: phase transitions and emergent computation, Physica D 42 (1990) 12–37.
[18] W. Li, N. Packard, The structure of elementary cellular automata rule space, Complex Systems 4 (1990) 281–297.
[19] C.L.M. Martins, P.P.B. de Oliveira, Evolving Sequential Combinations of Elementary Cellular Automata Rules, in: Lecture Notes in Artificial Intelligence, vol. 3630, 2005, pp. 461–470.
[20] M. Mitchell, P.T. Hraber, J.P. Crutchfield, Evolving cellular automata to perform computations: mechanism and impediments, Physica D 75 (1993) 361–391.
[21] MPI-Forum, MPI Documents, Message Passing Interface Forum. ⟨http://www.mpi-forum.org/docs/docs.html⟩. Accessed on Oct-18-2002.
[22] S. Ninagawa, Evolving Cellular Automata by $1/f$ Noise, in: Lecture Notes in Artificial Intelligence, vol. 3630, 2005, pp. 453–460.
[23] G.M.B. Oliveira, J.C. Bortot, P.P.B. de Oliveira, Multiobjective evolutionary search for one-dimensional cellular automata in the density classification task, in: R.K. Standish, M.A. Bedau, H.A. Abass (Eds.), Artificial Life VIII, Complex Adaptive System Series, MIT Press, Cambridge, MA, 2003, pp. 202–206.
[24] G.M.B. Oliveira, P.P.B. de Oliveira, N. Omar, Definition and applications of a five-parameter characterization of one-dimensional cellular automata rule space, Artif. Life J. (2001) 277–301.
[25] T. Pikula, E, Gramatová, M. Fischerová, Automatic design of cellular automata for generating deterministic test patterns. in: Digest of Papers: IEEE European Test Workshop (ETW), Maastricht, The Netherlands, 2003, pp. 285–286.
[26] E. Sapin, O. Bailleux, J.-J. Chabrier, Research of a Cellular Automaton Simulating Logic Gates by Evolutionary Algorithms, in: Lecture Notes in Computer Science, vol. 2610, 2003, pp. 414–423.
[27] E. Sapin, O. Bailleux, J.-J. Chabrier, Research of Complex Forms in the Cellular Automata by Evolutionary Algorithms, in: Lecture Notes in Computer Science, vol. 2936, 2004, pp. 373–400.
[28] E. Sapin, O. Bailleux, J.-J. Chabrier, P. Collet, A New Universal Cellular Automaton Discovered by Evolutionary Algorithms, in: Lecture Notes in Computer Science, vol. 3102, 2004, pp. 175–187.
[29] M. Sipper, M. Tomassini, Computation in artificially evolved, non-uniform cellular automata, Theor. Comput. Sci. 217 (1999) 81–98.
[30] N. Srinivas, K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, Evol. Comput. (1994) 221–248.
[31] A. Swiecicka, F. Seredynsky, Cellular automata approach to scheduling problem. in: Proc. of the Int. Conference on Parallel Computing in Electrical Engineering (PARELEC 2000), 2000, pp. 29–33.
[32] A.S. Tanenbaum, Computer Networks, fourth ed., Prentice-Hall PTR, Englewood Cliffs, NJ, 2002.
[33] M. Tomassini, M. Giacobini, C. Darabos, Evolution and dynamics of small-world cellular automata, Complex Systems 15 (4) (2005) 1–24.
[34] M. Tomassini, M. Venzi, Evolving robust asynchronous cellular automata for the density task. Complex Systems 13 (3) (2003) 185–204.
[35] J. Werfel, M. Mitchell, J.P. Crutchfield, Resource sharing and coevolution in evolving cellular automata, IEEE Trans. Evol. Comput. 4 (2000) 388–393.
[36] S. Wolfram, A New Kind of Science, Wolfram Media, 2002.
[37] S. Wolfram, The Mathematica Book, fifth ed., Wolfram Media, 2003.

Pedro Paulo Balbi de Oliveira received a degree in Electronics Engineering at the Aeronautics Institute of Technology, Brazil, in 1981, an M.Sc. in Applied Computing, at the Brazilian Institute for Space Research (INPE), in 1986, and a D.Phil. in Cognitive Science, at the University of Sussex, England, in 1994. In the beginning of his career he worked with knowledge-based artificial intelligence, mostly as a researcher at INPE, but for the last 15 years he has been closer to the artificial life community, doing research on computational and dynamic aspects of cellular automata, and applications of evolutionary computation. Since 2001 he has been a faculty member of the School of Computing and Informatics, at Universidade Presbiteriana Mackenzie, in São Paulo, Brazil.



José Carlos Bortot is currently a professor at FAAP university—Fundação Armando Álvares Penteado—in São Paulo, Brazil. After receiving an undergraduate degree in Economics, in 1967, he soon moved to the then fledgeling field of Informatics, where he has been an active participant for about 38 years, both as a technician in the design of many real-world telecommunication, commercial and banking systems, and as a university professor in various topics in computer languages and information systems. In 2003, he received an M.Sc. in Electrical Engineering, at Universidade Presbiteriana Mackenzie, São Paulo, Brazil, with an emphasis on artificial intelligence, the field that remains in the core of his current interests for both research and development.

Gina Maira Barbosa de Oliveira received a B.S. degree in Electrical Engineering at the National University of Uberlândia, Brazil, in 1990, and an M.Sc. and a Ph.D. from the Computer Science Department of the Aeronautics Institute of Technology, Brazil, in 1992 and 1999, respectively. Currently, she is a full-time faculty member of the National University of Uberlândia, with academic activities carried out at the M.Sc. Programme in Computer Science and at the School of Computer Science. Her research interests include cellular automata and evolutionary computation.