Um Modelo Computacional Paralelo Baseado em Programação de Expressão Genética Aplicado ao Problema de Indução de Regras de Autômatos Celulares

Wagner Rodrigo Weinert[1], **Heitor Silvério Lopes**[2]

[1] Instituto Federal de Educação Ciência e Tecnologia do Paraná (IFPR) – Câmpus Paranaguá
R. Antônio Carlos Rodrigues, 453 – 83215-750 – Paranaguá – PR – Brasil

[2]Universidade Tecnológica Federal do Paraná (UTFPR)
R. Sete de Setembro, 3162 – 80230-901 – Curitiba – PR – Brasil

wagner.weinert@ifpr.edu.br,hslopes@utfpr.edu.br

Abstract. *This paper presents an parallel evolutionary model based on gene expression programming. This methodology can be generalized to solve the problem of cellular automata rule induction, independently the nature of the problem. The model is able to induce the transition rule representing the proposed dynamic. Three parameters must be defined: initial, final and neighborhood cellular automata configuration. Four experiments are performed to evaluate the model,of which three were controlled by a known rule. The results show that the model is capable of inducing correctly rules in experiments with known rules, and confirms the generality of the model to problem induction with unknown rule.*

Resumo. *Este trabalho apresenta um modelo evolucionário paralelo baseado em programação de expressão genética. Esta metodologia pode ser generalizada para a solução do problema de indução de regras de autômatos celulares, não obstante a natureza do problema a ser resolvido, enquanto muitas das propostas reportadas na literatura são específicas. O modelo é capaz de induzir a regra de transição que representa a dinâmica proposta a partir das configurações inicial, final e de vizinhança do autômato. Foram realizados quatro experimentos para avaliar o modelo, destes, três eram controlados com uma regra conhecida. Os resultados obtidos atestam que o modelo é capaz de induzir regras corretamente em experimentos com regras conhecidas, bem como confirma a generalidade do modelo ao realizar a indução no problema com regra desconhecida.*

1. Introdução

Um Autômato Celular (AC) é um sistema dinâmico discreto que evolui pela iteração de uma regra de transição de estado, sendo que os valores das variáveis do sistema mudam em função de seus valores correntes [Wuensche and Lesser 1992]. Quando a regra de transição de estado é conhecida, pode-se simular o comportamento dinâmico de um AC [Wolfram 1984]. No entanto, esta situação não ocorre para a maioria das aplicações de

ACs. A tarefa de encontrar regras de transição de estado que simulem determinado comportamento dinâmico de um AC pode ser generalizada como um problema de indução de regras para ACs.

A maioria dos problemas modelados através de ACs apresentam um espaço de busca de soluções intratável para algoritmos de busca baseados em estratégias determinísticas, o que torna difícil o problema de indução de regras para ACs. Por exemplo, para um AC unidimensional de raio 5 tem-se um espaço de busca de $3,2 \times 10^{616}$ regras. Para um AC bidimensional de mesmo raio o espaço de busca é de $2^{26 \times 10^{35}}$ e para um AC tridimensional é de $2^{4 \times 10^{100}}$ [Wolfram 1984].

Esforços têm sido gastos na busca de novas abordagens eficientes e capazes de simular o comportamento desejado na tarefa de indução de regras para ACs. A maioria destas abordagens [Richards et al. 1990, Andre et al. 1996, Juillé and Pollack 1998, Morales et al. 2001, Oliveira et al. 2001, Ferreira 002b, Oliveira et al. 2002a, Oliveira et al. 2002b, Oliveira et al. 2007] utilizam técnicas baseadas em algoritmos evolucionários, como por exemplo: Algoritmos Genéticos [Goldberg 1989, Holland 1975, Coley 1997, Sivanandam and Deepa 2007] e Programação Genética [Koza 1992, Langdon and Poli 2002, Poli et al. 2008]. Estas técnicas evoluem uma população de indivíduos (no caso, regras) por *n* gerações. Em cada geração um conjunto de operadores genéticos atua sobre a população de indivíduos a fim de melhorar a qualidade do material genético. Esta qualidade é avaliada pela função de *fitness*, e normalmente o processo de evolução é encerrado quando uma solução satisfatória é encontrada ou quando algum critério de término é alcançado, por exemplo, um número máximo de gerações.

Em geral, no problema de indução de regras para ACs utiliza-se como função de *fitness* uma medida obtida através da análise do comportamento dinâmico do sistema, onde cada uma das possíveis soluções (regras) é submetida a um processo de simulação da dinâmica. A regra atua sobre uma configuração inicial de AC por *t* iterações, obtendo-se a configuração final que é comparada à configuração esperada. Se ambas forem iguais então a regra é solução do problema, senão uma medida de similaridade quantifica a qualidade da regra. Nesta abordagem o cálculo do *fitness* tem um elevado custo computacional, o que por vezes, inviabiliza o processo.

Este trabalho apresenta um modelo computacional paralelo para indução de regras de ACs. O modelo proposto baseia-se no modelo construído por Weinert e Lopes (2011) - PRIGEP (*Parallel Rule Induction with Gene Expression Programming*) - para indução de regras em processo de mineração de dados. Neste trabalho o modelo é modificado para aplicação no problema de indução de regras de ACs. O PRIGEP é um modelo baseado no algoritmo de programação de expressão genética originalmente proposto por [Ferreira 2001].

2. PRIGEP

O PRIGEP foi totalmente desenvolvido em linguagem C padrão ANSI sobre a plataforma Linux [Weinert and Lopes 2011]. O paralelismo foi implementado através da biblioteca MPICH2 [Gropp et al. 1999][1] que codifica os conceitos do MPI [Snir et al. 1997]. Esta

[1]Disponível em: http://www.mcs.anl.gov/research/projects/mpich2/

biblioteca cria um ambiente virtual composto por diversas unidades de processamento. Um conjunto de funções de comunicação e sincronismo permite que tais unidades de processamento se comuniquem entre si.

O PRIGEP implementa uma arquitetura lógica de dimensionamento dinâmico onde um processo mestre coordena *n* processos escravos. Um conjunto formado por um processo mestre e *n* processos escravos definem uma ilha de processamento. O PRIGEP também tem a propriedade de trabalhar com múltiplas ilhas que cooperam entre si para a solução do problema.

Na abordagem adotada o processo mestre tem a incumbência de executar o algoritmo evolucionário de programaçao de expressão genética, implementado em [Weinert and Lopes 2006], bem como controlar todo o processo de comunicação entre ele e seus respectivos processos escravos. Todo processo escravo executa a mesma tarefa que, neste caso, é o cálculo da função de *fitness*. Caso opte-se por uma arquitetura composta por mais de uma ilha cria-se um processo colaborativo. Todas as ilhas executam exatamente as mesmas tarefas. No entanto, de tempos em tempos, os processos mestres de cada uma das ilhas comunicam-se entre si trocando informações que podem melhorar a solução de determinado problema.

3. Modelo para Indução de Regras de Autômatos Celulares

O modelo para indução de regras de ACs também utiliza regras na forma "SE *A* ENTÃO *C*". O antecedente "*A*" pode ser formado por um conjunto de condições e "*C*" representa o consequente da regra.

As condições das regras são *t*-uplas da forma $A_i Op V_{ij}$, onde A_i é o *i*-ésimo atributo, Op é um operador relacional (normalmente =, $\neq$, $>$ ou $<$) e V_{ij} é o *j*-ésimo valor que pertence ao domínio do atributo A_i. Para fazer a combinação das possíveis condições são utilizados operadores lógicos (AND, OR ou NOT).

O consequente da regra consiste de uma simples condição na forma $< M_i = V_{ij} >$, onde M_i é um possível atributo meta e V_{ij} é um dos possíveis valores do domínio do atributo selecionado.

Este modelo vislumbra a generalidade, ou seja, deve ser capaz de induzir regras para diferentes problemas modelados sobre o conceito de ACs. A idéia central consiste em, induzir, dado um estado inicial e um estado final para um determinado AC, bem como sua configuração de vizinhança, uma regra de transição que represente tal comportamento. Os estados inicial e final do AC são introduzidos no sistema por arquivos tipo texto.

Neste modelo a codificação dos indivíduos da população segue a abordagem de Pittsburgh [Freitas 2002]. Segundo esta abordagem um indivíduo representa um conjunto de regras dentro do sistema de classificação. Sendo assim, cada indivíduo passa a ter mais de um cromossomo, onde cada cromossomo codifica uma regra, como explicado na sequência.

Como já mencionado, o objetivo é encontrar uma regra que represente o comportamento desejado. Para exemplificar o procedimento, utiliza-se um AC unidimensional onde o conjunto de estados possíveis para cada célula do AC é representado por Σ = 1, 2, 3 e a vizinhança a ser considerada, tem raio igual a 1. Assim, o número de combinações

de vizinhança é 27, e o número de regras que se aplicam a tal configuração, é de aproximadamente 7×10^{12} [Wolfram 1984].

A partir destes dados torna-se possível construir a tabela de transição de estados (Tabela 1) aplicável a este exemplo. A coluna "regra" ainda encontra-se indefinida, pois depende da regra codificada pelo indivíduo apresentado na Figura 1 .

Tabela 1. Vizinhanças para uma regra de transição ainda desconhecida.

transição	vizinhança	regra
	a b c	
0	111	?
1	112	?
2	113	?
...	...	...
24	331	?
25	332	?
26	333	?

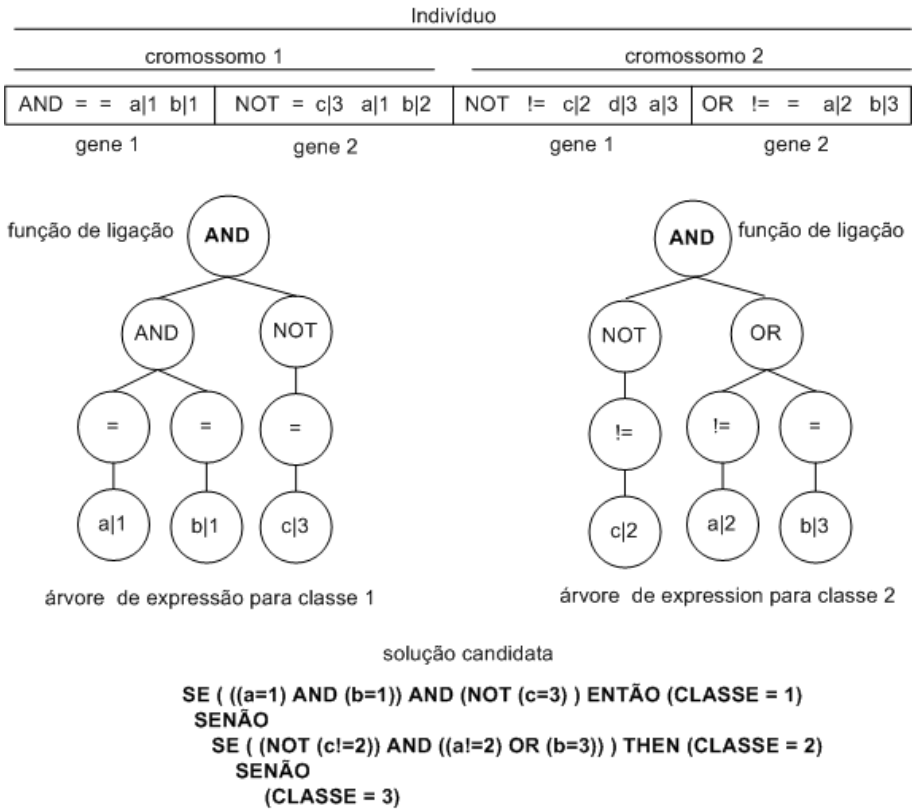

Figura 1. Exemplo de codificação para indução de regras de transição em ACs.

Neste exemplo, Σ é composto por 3 elementos. Cada elemento é definido como uma classe. Ou seja, tem-se um problema de classificação hierárquica composto por 3 classes. Diferentemente dos problemas de classificação, os problemas de indução de regras de ACs não possuem casos de *fitness*. Assim, a idéia central consiste em se partir

da regra codificada no indivíduo e preencher a coluna "regra" da Tabela 1 segundo a abordagem de Pittsburgh.

A codificação dos indivíduos no PRIGEP depende da definição do conjunto de terminais e seus respectivos domínios. Os terminais representam a vizinhança. Neste exemplo cada vizinhança é composta por 3 elementos, assim serão utilizados 3 terminais. Dá-se um rótulo para cada terminal, por exemplo: a, b e c (Tabela 1). Todos os terminais possuem o mesmo domínio. Este domínio é definido pelos estados que as células do AC podem assumir, neste exemplo: 1, 2 e 3.

As classes do problema também são representadas pelos estados que as células do AC podem assumir. Neste exemplo, tem-se um indivíduo que codifica 3 regras de maneira hierárquica, sendo a primeira responsável por cobrir combinações de vizinhança que mapeiam o estado 1. A segunda, executada somente se a primeira não for aplicável, é responsável por cobrir combinações de vizinhança que mapeiam o estado 2. A terceira regra, executada somente se a segunda não for aplicável, é a regra padrão, ou seja, cobre todas as combinações de vizinhança que não foram cobertas pela primeira e segunda regras. A regra padrão não é codificada no indivíduo. A ordem hierárquica de cobertura dos estados das células é inserida como dado de entrada no sistema.

A Figura 1 apresenta um indivíduo codificado. O processo de construção e interpretação deste modelo pode ser consultado em Weinert e Lopes (2006). Cada cromossomo que compõe o indivíduo é mapeado para sua respectiva árvore de expressão. Neste exemplo, os cromossomos são formados por 2 genes, mas este parâmetro é variável, bem como o tamanho da cabeça, que está com valor 3. A solução candidata, a qual vai permitir o preenchimento da coluna "regra" na Tabela 1 apresenta-se como uma regra hierárquica, onde o primeiro nível representa o primeiro cromossomo e o segundo nível o segundo cromossomo. Por fim, adiciona-se a regra padrão que cobre a classe 3.

A Tabela 2 apresenta a regra (terceira coluna) que foi obtida a partir da aplicação da soluções candidata sobre as diferentes combinações de vizinhança, que será utilizada para simular a dinâmica do AC.

Tabela 2. Determinação da regra de transição.

transição	vizinhança	regra
	a b c	
0	111	1
1	112	1
2	113	2
...	...	...
24	331	3
25	332	2
26	333	3

A função de *fitness* implementada precisa ser genérica e aplicável a qualquer dimensão de AC independentemente do alfabeto utilizado. A função proposta que é definida pela Equação 1 utiliza a mesma idéia da medida da distância de Hamming [Peterson and Weldon 1972]. Ela quantifica o número de semelhanças entre dois ACs, célula a célula. Os ACs devem ter a mesma dimensão e o mesmo número de células.

Então divide-se este número pelo número total de células de um dos ACs. Se o resultado da divisão for 1 tem-se uma semelhança completa, se for 0 a semelhança inexiste, e um valor entre 0 e 1 indica o grau de semelhança entre os dois ACs. No caso, os dois ACs são representados pela configuração de AC desejada e a configuração obtida pelo sistema a partir de uma dada configuração inicial, no processo de simulação da dinâmica, tal que

$$fitness = \frac{\sum_{i=1}^{d_3}(\sum_{j=1}^{d_2}(\sum_{k=1}^{d_1}(SE(ACD_{i,j,k} = ACO_{i,j,k}; 1; 0))))}{d_3 * d_2 * d_1} \quad (1)$$

onde,

- d_3, d_2 e d_1: tamanho da dimensão 3, 2 e 1 respectivamente;
- ACD: autômato celular desejado;
- ACO: autômato celular obtido;
- SE: função que retorna 1 caso o teste seja verdadeiro e 0 caso contrário;
- i, j, k: índices que endereçam as células dos ACs ACD e ACO.

A priori o sistema não sabe quantas iterações serão necessárias para que determinada regra simule o comportamento desejado. O que o sistema faz é utilizar um limiar máximo de iterações. A cada iteração ele executa a função de *fitness* e se esta retornar valor 1 ele encerra o processo. Senão, na iteração referente ao limiar utilizado retorna um valor de semelhança pertencente ao intervalo [0..1].

A implementação atual do PRIGEP permite a manipulação de ACs unidimensionais de raio 1, 2 e 3 e ACs bidimensionais que utilizam vizinhaça de Neumann e de Moore [Wolfram 2002]. O sistema também está apto para trabalhar com ACs tridimensionais. No entanto, ainda não foi implementado um modelo de vizinhança para o mesmo. A adição de novos modelos é relativamente simples, visto que o sistema foi construído de maneira modular, já com intuito de facilitar a adição de novas funcionalidades.

4. Resultados

Numa primeira etapa foram realizados 3 experimentos controlados. Cada experimento consiste da indução de *n* regras para uma mesma configuração inicial de AC. Assim, cada regra é obtida em uma rodada independente do algoritmo. A configuração final desejada para o AC é o único parâmetro que se altera em cada rodada, e é representada pela configuração do AC em uma determinada iteração. Os experimentos são ditos controlados porque, primeiro, sabe-se a priori qual a regra se deseja obter. Segundo, sabe-se o número de iterações necessárias para conduzir a configuração inicial do AC até a configuração final desejada durante o processo de simulação da dinâmica. No entanto, é importante lembrar que para problemas reais estas duas informações (regras e número de iterações) são desconhecidas.

Para os experimentos controlados utiliza-se um AC binário de 149 células, inicializado aleatoriamente com distribuição uniforme. A vizinhança é formada por um raio igual a 1. A configuração deste AC é apresentada na Figura 2 onde células com estado 0 aparecem em br'anco e células com estado 1 em preto.

Os experimentos controlados não utilizam o conceito de co-evolução através do trabalho em múltiplas ilhas. Basicamente, os parâmetros evolucionários utilizados são:

Figura 2. Configuração inicial do AC de 149 células.

- número de indivíduos na população: 50;
- número de genes por cromossomo: 3;
- tamanho da cabeça de cada gene: 15;
- conjunto de funções: AND, OR e NOT;
- função de ligação das AEs: AND;
- método de seleção: torneio contemplando 10% dos indivíduos da população;
- mutação: 2% de probabilidade;
- recombinação: 80% de probabilidade;
- transposição IS: 70% de probabilidade;
- transposição RIS: 70% de probabilidade;
- transposição gênica: 70% de probabilidade;
- clonagem (elitismo): habilitada;
- número de gerações: 30;
- número de processos escravo por processo mestre: 10;
- migração: um indivíduo, de cada ilha, selecionado pelo método do torneio a cada 5 gerações, migra para ilha vizinha substituindo o pior indivíduo desta ilha.

O primeiro experimento controlado avalia o desempenho do algoritmo em induzir a regra elementar 95. Esta regra apresenta um comportamento dinâmico de ciclo duplo segundo a classificação de [Li and Packard 1990]. O sistema apresenta-se estável a partir da primeira iteração. A Figura 3 demonstra a dinâmica do sistema por 20 iterações. Este experimento foi segmentado em 4 rodadas. As rodadas contemplam as configurações finais obtidas nas iterações 1, 2, 3 e 10. Em todas rodadas o sistema foi capaz de induzir a regra 95. A Tabela 3 resume algumas informações a respeito dos resultados obtidos. Cada linha corresponde a uma rodada. Na segunda coluna tem-se informação a respeito do comportamento dinâmico do sistema até aquele momento. Esta questão será discutida na próxima seção. A terceira coluna indica a iteração utilizada como determinante do estado final do AC desejado, e a quarta coluna apresenta o número de gerações necessárias para que a solução fosse encontrada.

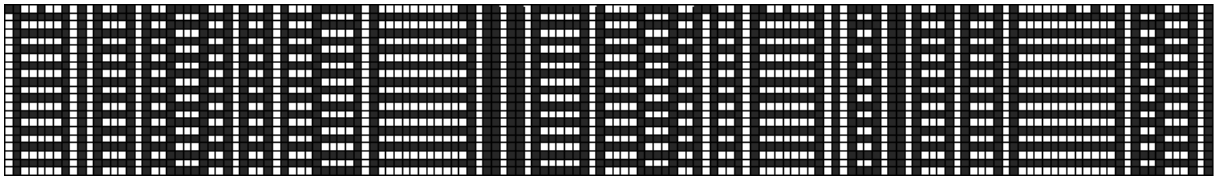

Figura 3. Simulação da dinâmica da regra 95 por 20 iterações.

O segundo experimento controlado avalia o desempenho do algoritmo em induzir a regra elementar 183. Esta regra apresenta um comportamento dinâmico caótico segundo a classificação de [Li and Packard 1990]. O sistema não estabiliza. A Figura 4 demonstra a dinâmica do sistema por 20 iterações. Este experimento foi segmentado em 3 rodadas. As rodadas contemplam as configurações finais obtidas nas iterações 1, 10 e 20. O algoritmo evolucionário configurado com 30 indivíduos não retornou a regra desejada em 350

Tabela 3. Resultados para regra 95.

Rodada	Comportamento	Iteração	Nº de Gerações
1	Ciclo Duplo	1	16
2		2	14
3		3	13
4		10	5

gerações. Então rodou-se novamente o algoritmo com um tamanho de população igual a 100 indivíduos. Com este valor a regra 183 foi encontrada para as rodadas 1 e 2. A rodada 3 somente atingiu seu objetivo com uma população de 200 indivíduos. A Tabela 4 resume algumas informações a respeito dos resultados obtidos.

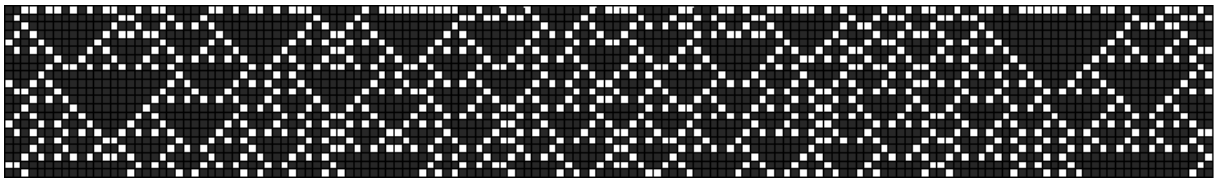

Figura 4. Simulação da dinâmica da regra 183 por 20 iterações.

Tabela 4. Resultados para regra 183.

Rodada	Comportamento	Iteração	Nº de Gerações
1	Caótico	1	184
2		10	187
3		20	16

O terceiro experimento controlado avalia o desempenho do algoritmo em induzir a regra elementar 124. Esta regra apresenta um comportamento dinâmico complexo segundo a classificação de [Li and Packard 1990]. O sistema estabiliza após milhares de iterações. A Figura 5 demonstra a dinâmica do sistema segmentada em duas regiões, da iteração 1 até a 20 (Figura 5(a)) e da iteração 3980 até a 4000 (Figura 5(b)). Este experimento foi segmentado em 2 rodadas. As rodadas contemplam as configurações finais obtidas nas iterações 1 e 4000. O algoritmo evolucionário volta a utilizar 30 indivíduos. A Tabela 5 resume algumas informações a respeito dos resultados obtidos.

Tabela 5. Resultados para regra 124.

Rodada	Comportamento	Iteração	Nº de Gerações
1	Período Transiente	1	81
2	Complexo	4000	27

Concluída a etapa de experimentos controlados apresenta-se um experimento com um problema hipotético. Neste problema deseja-se descobrir uma regra que seja capaz de conduzir o AC bidimensional C_{5x5} apresentado pela Figura 6(a) ao AC apresentado pela Figura 6(b). A única definição existente é quanto a configuração de vizinhança,

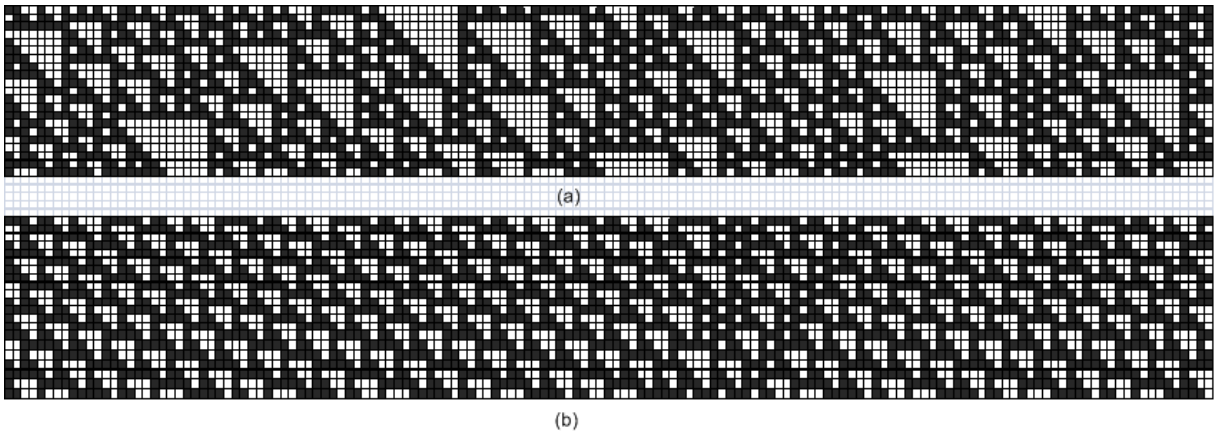

Figura 5. Simulação da dinâmica da regra 124 por 4000 iterações. (a) Apresenta as primeiras 20 iterações e (b) as últimas 20.

que deve ser a de Neumann. Na vizinhança de Neumann para cada célula $c_{i,j}$ de um arranjo bidimensional C_{axb} consideram-se as células: $c_{i-1,j}, c_{i,j-1}, c_{i,j}, c_{i,j+1}, c_{i+1,j}$ como vizinhança.

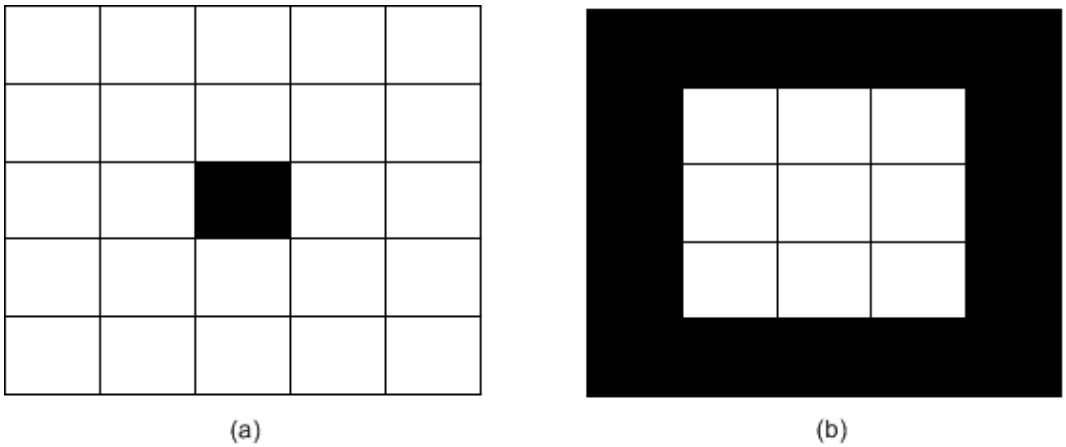

Figura 6. Configurações inicial (a) e final (b) do AC bidimensional.

Para resolução deste problema, foi utilizado o modelo insular composto por 5 ilhas. Foram utilizados os seguintes parâmetros:

- número de indivíduos na população: 1000;
- número de genes por cromossomo: 2;
- tamanho da cabeça de cada gene: 10;
- conjunto de funções: AND, OR e NOT;
- função de ligação das AEs: AND;
- método de seleção: torneio contemplando 10% dos indivíduos da população;
- mutação: 20% de probabilidade;
- recombinação: 80% de probabilidade;
- transposição IS: 70% de probabilidade;
- transposição RIS: 70% de probabilidade;
- transposição gênica: 70% de probabilidade;

- clonagem (elitismo): habilitada;
- gerações: 350;
- número de processos escravo por processo mestre: 20;
- migração: indivíduo selecionado pelo método do torneio a cada 5 gerações;
- número máximo de iterações do AC: 30.

O algoritmo encontrou a solução na geração 103, sendo a regra: 10100000101001010010000000100011. Esta regra conduz o AC da Figura 6(a) ao AC da Figura 6(b) em duas iterações.

5. Discussão e Resultados

O modelo proposto para indução de regras de ACs induz regras para diferentes configurações de ACs. Basicamente, informa-se ao sistema a configuração inicial do AC, a configuração final do AC, o tipo de vizinhança a ser considerado no processo iterativo, bem como o número máximo de iterações aceito para levar o AC inicial ao AC final no processo de simulação da dinâmica.

Inicialmente, com intenção de verificar se o sistema estava realmente fazendo o que se propunha a fazer, foram realizados alguns experimentos controlados. Este termo foi utilizado pelo fato de que nestes experimentos as regras desejadas já eram previamente conhecidas.

Foram realizados 3 experimentos controlados com as regras elementares: 95, 183 e 124. Estas regras apresentam, respectivamente, os comportamentos: ciclo duplo, caótico e complexo. A configuração inicial do AC foi a mesma para todos os experimentos. Para cada experimento, diferentes configurações finais foram testadas. E sempre determinou-se o número exato de iterações necessárias para se conduzir a configuração inicial à configuração final desejada.

Para a regra 95 (Tabela 3) o algoritmo evolucionário não teve dificuldade na busca pela solução. Em poucas gerações as regras foram encontradas. Para a regra 183 (Tabela 4) de comportamento caótico, foi necessário aumentar a diversidade genética para que fosse possível alcançar a solução. Isto ocorre, primeiramente, devido ao comportamento caótico ser minoria dentro das regras que compõem o espaço de busca. Segundo, que no comportamento caótico o sistema não se estabiliza. Para a regra 124 (Tabela 5) de comportamento complexo, 30 indivíduos foram suficientes, uma vez que este comportamento estabiliza após um longo processo de simulação da dinâmica.

Os experimentos controlados apesar de muito simples do ponto de vista do algoritmo evolucionário, devido ao tamanho do espaço de busca, foram de suma importância para a compreensão do funcionamento do sistema como um todo. Foi possível observar diferentes comportamentos dinâmicos e entender que a busca de soluções pertencentes a períodos transientes pode ser muito difícil. Na verdade, esta é a realidade da grande maioria das aplicações reais para as quais este sistema se destina. Como são introduzidos apenas estados iniciais e finais dos ACs, o sistema não sabe nada a respeito da dinâmica que ele deve produzir para solucionar o problema de indução.

Na sequência realizou-se um experimento que avalia o potencial do algoritmo evolucionário. Este experimento consiste em encontrar uma regra para simulação da dinâmica apresentada pela Figura 6. Neste experimento o espaço de busca é formado

por 4294967296 regras. A regra foi encontrada na geração 103 no modelo mestre-escravo de 5 ilhas. Ou seja, após avaliar, aproximadamente, 615000 regras o algoritmo encontrou a solução. Isto equilave a examinar apenas 0,014% do espaço de busca, o que mostra a eficiência do método utilizado.

Entende-se que existem pontos a serem explorados na continuidade deste trabalho. No entanto, todos os objetivos iniciais foram satisfatoriamente alcançados. O modelo paralelo proposto pode ser aplicado a diferentes problemas complexos de indução de regras para ACs. Dentre estes problemas pode-se destacar os relacionados a bioinformática como: modelagem de reações enzimáticas, simulação do crescimento de tumores, espalhamento de infecções, alinhamento de sequências de DNA, dobramento de proteínas, entre outros. Bem como, também ser aplicado a uma generalidade de problemas de classificação, onde sua principal vantagem em relação a outros métodos de indução de regras apresenta-se na simplicidade e consequente compreensibilidade das regras obtidas.

Referências

Andre, D., Bennett, III, F. H., and Koza, J. R. (1996). 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 - GECCO '96*, pages 3–11, Cambridge, MA, USA. MIT Press.

Coley, D. A. (1997). *An Introduction to Genetic Algorithms for Scientists and Engineers*. World Scientific Publishing Company, Singapore.

Ferreira, C. (2001). Gene expression programming: a new adaptative algorithm for solving problems. *Complex Systems*, 13(2):87–129.

Ferreira, C. (2002b). Discovery of the boolean functions to the best density-classification rules using gene expression programming. *Lecture Notes in Computer Science*, 2278:51–60.

Freitas, A. A. (2002). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, Berlin.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, USA.

Gropp, W., Lusk, E., and Thakur, R. (1999). *Using MPI-2: Advanced Features of the Message-Passing Interface*. MIT Press, Cambridge, MA, USA.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor.

Juillé, H. and Pollack, J. B. (1998). Coevolving the ideal trainer: application to the discovery of cellular automata rules. In Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., and Riolo, R., editors, *Proceedings of the Third Annual Conference in Genetic Programming*, pages 519–527, San Francisco. Morgan Kaufmann.

Koza, J. R. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, USA.

Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming*. Springer, New York.

Li, W. and Packard, N. (1990). The structure of elementary cellular automata rule space. *Complex Systems*, 4(3):281–297.

Morales, F. J., Crutchfield, J. P., and Mitchell, M. (2001). Evolving two-dimensional cellular automata to perform density classification: a report on work in progress. *Parallel Computing*, 27(5):571–585.

Oliveira, G. M. B., Asakura, O. K. N., and de Oliveira, P. P. B. (2002a). Dynamic behaviour forecast as a driving force in the coevolution of one-dimensional cellular automata. In *Proceedings of the VII Brazilian Symposium on Neural Networks*, pages 98–103, Los Alamitos, CA, USA. IEEE Computer Society.

Oliveira, G. M. B., Bortot, J. C., and de Oliveira, P. P. B. (2002b). Multiobjective evolutionary search for one-dimensional cellular automata in the density classification task. In *Proceedings of the* 8^{th} *International Conference on Artificial Life, ALIFE VIII*, pages 202–206, Sydney. Cambridge: MIT Press.

Oliveira, G. M. B., Bortot, J. C., and de Oliveira, P. P. B. (2007). Heuristic search for cellular automata density classifiers with a multiobjective evolutionary algorithm. In *Proceedings of VI Congress of Logic Applied to Technology LAPTEC 2007*, volume 1, pages 1–12, Santos, SP. CD-ROM Paper ID:213.

Oliveira, G. M. B., Omar, N., and de Oliveira, P. P. B. (2001). Definition and application of a five-parameter characterization of one-dimensional cellular automata rule space. *Artificial Life*, 7(3):277–301.

Peterson, W. W. and Weldon, E. J. (1972). *Error-correcting codes*. MIT Press, Cambridge.

Poli, R., Langdon, W. B., and McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises, United Kingdom.

Richards, F. C., Meyer, T. P., and Packard, N. H. (1990). Extracting cellular automaton rules directly from experimental data. *Physica D*, 45(1-3):189–202.

Sivanandam, S. N. and Deepa, S. N. (2007). *Introduction to Genetic Algorithms*. Springer, New York.

Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J. (1997). *MPI: The Complete Reference*. MIT Press, Cambridge, 2^{nd} edition.

Weinert, W. R. and Lopes, H. S. (2006). GEPCLASS: a classification rule discovery tool using gene expression programming. *Lecture Notes in Computer Science*, 4093:871–880.

Weinert, W. R. and Lopes, H. S. (2011). Data mining with a parallel rule induction system based on gene expression programming. *Int. J. Innovative Computing and Applications*, 3:136–143.

Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica D*, 10(1-2):1–35.

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media, Champaign.

Wuensche, A. and Lesser, M. (1992). *The Global Dynamics of Cellular Automata*. Addison Wesley, Reading, USA.