COMPUTAÇÃO E EVOLUÇÃO EM AUTÔMATOS CELULARES UNIDIMENSIONAIS

Gina M. B. Oliveira*
Nizam Omar*
Pedro P. B. de Oliveira**

Resumo

Autômatos Celulares (ACs) são sistemas dinâmicos discretos que pertencem à classe dos sistemas complexos. Dentre as capacidades mais pesquisadas dos ACs está a sua habilidade de realizar computações através de processamentos locais e paralelos. Entretanto, o entendimento de como os ACs realizam computações ainda é extremamente vago. Uma das abordagens mais bem-sucedidas é a utilização de programação evolutiva para encontrar ACs capazes de executar uma tarefa computacional. Neste trabalho, fazemos uma revisão sobre computação e evolução em ACs e apresentamos os resultados de nossa pesquisa recente, em que parâmetros de previsão de comportamento dinâmico foram utilizados na evolução de ACs.

Abstract

Cellular Automata (CA) are discrete dynamic systems that belong to the complex systems class. One of the CA capacities that more attract the researchers' interest is its ability to accomplish computations through local and parallel processings. However, the understanding of how the CA do computations is still extremely vague. A well succeed approach is the use of evolutionary programming to find CA capable to execute a computational task. In this work, we present a review of CA computation and evolution. We also present the results of our recent research, in which parameters that helping to forecast dynamic behavior were used in the CA evolution.

* Universidade Presbiteriana Mackenzie. E-mail: gina@mackenzie.com.br.
** Instituto de Pesquisa e Desenvolvimento da Universidade do Vale do Paraíba. E-mail: pedrob@univap.br.

1 INTRODUÇÃO

Autômatos Celulares (ACs) são exemplos de sistemas discretos que se tornaram ferramentas importantes no estudo de sistemas complexos. A exemplo de outros sistemas desta classe, os ACs exibem um comportamento dinâmico variado e imprevisível. Assim como os ACs têm o potencial de modelar o comportamento de sistemas complexos na natureza, eles também têm a habilidade de executar computações complexas com um alto grau de eficiência e robustez. Um aspecto bastante estudado diz respeito a como os ACs realizam essas computações: através de processamentos locais e intrinsecamente paralelos que interagem entre si, emergindo um comportamento global e coordenado.

Vários pesquisadores estão interessados nas relações entre o comportamento dinâmico genérico de um AC e suas habilidades computacionais, como parte de uma questão maior sobre as relações entre a teoria dos sistemas dinâmicos e a teoria computacional.[1] Muitos pesquisadores têm se empenhado em provar o poder computacional dos ACs, através de duas abordagens distintas: a *computação explícita* e a *computação implícita*.[2]

Por se tratarem de estruturas genuinamente paralelas, cujo componente básico (a célula do AC) é de uma lógica extremamente simples, os ACs têm sido cogitados como uma opção para arquiteturas descentralizadas de computadores e que poderiam servir, também, de paradigma para novas tecnologias de hardware, especialmente para a nanotecnologia.[3] No entanto, o entendimento de como os ACs realizam computações ainda é extremamente vago e os pesquisadores vêm se empenhando em buscar métodos para viabilizar sua programação. Uma das abordagens mais bem-sucedidas é a utilização de técnicas de programação evolutiva para encontrar ACs capazes de executar uma tarefa computacional específica.[4]

Um AC é caracterizado por uma regra de transição de estados, que determina qual será o próximo estado do reticulado do AC, a partir do seu estado atual. Ao serem executados, os ACs exibem comportamentos dinâmicos típicos que podem ser agrupados em classes. Vários pesquisadores têm se dedicado ao estudo da dinâmica dos ACs através de indicadores que aproximam a previsão da sua dinâmica. Em um trabalho recente, buscamos estabelecer um conjunto eficaz de parâmetros de previsão de comportamento dinâmico.[1, 5]

Neste trabalho, além de fazermos uma revisão das principais linhas de pesquisa em computação de ACs, relataremos os resultados de experimentos nos quais uma heurística fornecida pelos parâmetros de previsão de comportamento dinâmico selecionados foi aplicada à busca evolutiva dos ACs na tarefa computacional conhecida por *tarefa de classificação da densidade*. Nesses experimentos foi alcançada uma melhoria significativa em relação aos experimentos realizados por outros pesquisadores.

No item 2, apresentaremos alguns conceitos básicos referentes à definição dos ACs. No item 3, mostraremos os principais trabalhos relacionados à computação explícita e implícita dos ACs. No item 4, relataremos a evolução de ACs que computam. No item 5, apresentaremos os resultados de nossos experimentos, utilizando uma busca evolutiva guiada por parâmetros de previsão. Finalmente, apresentaremos as conclusões de nosso trabalho no item 6.

2 AUTÔMATOS CELULARES

Um AC constitui-se basicamente de duas partes: *espaço celular e regra de transição*. Utilizaremos aqui a notação adotada por M. Mitchell.[4]

O *espaço celular* é um reticulado de **N** células idênticas, cada qual com um padrão idêntico de conexões locais para outras células e com condições de contorno. O conjunto de estados possíveis da célula é denotado por Σ e o número de estados desse conjunto é denotado por **k**. Cada célula é denotada por um índice **i** e seu estado a um dado tempo **t** é denotado por $\mathbf{n}_i^t$, onde $\mathbf{S}_i^t \in \Sigma$. O estado $\mathbf{S}_i^t$ da célula **i** e os estados das células às quais a célula **i** está conectada são chamados de vizinhança $\mathbf{n}_i^t$ da célula **i**.

A regra de transição é denotada por $\Phi(\mathbf{n}_i^t)$ que fornece o próximo estado $\mathbf{S}_i^{t+1}$ para cada célula **i**, como uma função de $\mathbf{n}_i^t$. A cada passo de tempo, todas as células atualizam seus estados sincronicamente de acordo com $\Phi(\mathbf{n}_i^t)$.

Para ACs unidimensionais, o tamanho da vizinhança n_i é normalmente escrito como $|n_i| = 2r + 1$, onde r é chamado o raio do AC. No caso de ACs de estados binários, a regra de transição é dada por uma tabela de transições de estado da regra, que lista cada vizinhança possível junto com seu bit de saída, que é o valor da atualização do estado da célula central da vizinhança.

Os 256 ACs unidimensionais, com $k = 2$ e $r = 1$, são chamados Autômatos Celulares Elementares (ACEs). Wolfram[7] propôs um esquema de numeração para os ACEs, no qual os bits de saídas são ordenados lexicograficamente e são lidos da direita para a esquerda para formar um binário entre 0 e 255.

2.1 Classificação dinâmica

Os ACs exibem comportamentos dinâmicos típicos que podem ser agrupados em classes. Mais de uma classificação do espaço de regras é utilizada na literatura, sendo que Wolfram propôs uma classificação qualitativa do comportamento dos ACs que é a mais referenciada na literatura e que agrupa os ACs em quatro classes dinâmicas.[1,7]

Posteriormente, Li e Packard propuseram uma série de refinamentos na classificação original de Wolfram.[8, 9, 10, 11] Adotamos em nosso trabalho um esquema de classificação que divide o espaço de regras em seis classes:[11]

- Regras nulas ou ponto fixo espacialmente homogêneas: a configuração limite é formada tão-somente por 0s ou por 1s.
- Regras ponto fixo espacialmente heterogêneas: a configuração limite é invariante ao reaplicarmos a regra do AC (com um possível deslocamento espacial), excluindo as configurações formadas tão-somente por 0s ou por 1s.
- Regras ciclo duplo: a configuração limite é invariante ao reaplicarmos a regra do AC duas vezes (com um possível deslocamento).
- Regras periódicas: a configuração limite é invariante à aplicação da regra L vezes, com o tamanho do ciclo L ou independentemente do tamanho do sistema.
- Regras complexas ou à beira do caos: embora a dinâmica limite possa ser periódica, o intervalo de transição pode ser extremamente longo e, tipicamente, este intervalo cresce mais que linearmente com o tamanho do sistema.
- Regras caóticas: produzem dinâmicas não periódicas. Estas regras são caracterizadas pela divergência exponencial do comprimento do seu ciclo com o tamanho do sistema e pela instabilidade com respeito a perturbações.

2.2 Parametrizações do espaço de regras

Apesar de sua simplicidade de implementação, a previsão da dinâmica de um AC é um problema indecidível.[12] Vários pesquisadores têm se empenhado em estudar a dinâmica dos ACs através de indicadores que aproximam a previsão do seu comportamento dinâmico.

O caminho mais direto para se analisar a dinâmica de um determinado AC é observar o seu comportamento através do diagrama de padrões espaço-temporais que ele gera a partir de várias inicializações aleatórias. Outra abordagem possível é a análise da tabela de transições da regra associada ao AC, com a vantagem de que a dimensão de uma regra é pequena, comparada ao espaço de padrões. Dessa forma, muito do estudo da dinâmica dos ACs recai no estudo do espaço de regras.[8, 9, 10, 11, 13]

Vários parâmetros calculados diretamente da regra de transição têm sido propostos para prever o comportamento dos ACs, notadamente o parâmetro λ de Langton.[14] A alta cardinalidade do espaço de regras dos ACs torna a sua parametrização uma tarefa difícil, e muitos estudos apontam para a necessidade de se utilizar mais de um parâmetro para se ter uma melhor caracterização das dinâmicas.[10, 13] Em "Dinâmica e Evolução de Autômatos Celulares Unidimensionais" e "Análise Crítica das Parametrizações no Espaço de Regras de Autômatos Celulares Unidimensionais" revisamos e analisamos quatro dos principais parâmetros publicados: o parâmetro λ;[10, 14] os parâmetros de campo médio;[10, 11] o parâmetro sensitividade (μ)[13] e o parâmetro Z.[16] A partir dessa análise, propusemos oito diretrizes básicas que devem ser seguidas no estabelecimento de qualquer parâmetro de previsão de comportamento dinâmico. Em "Análise Crítica das

Parametrizações no Espaço de Regras de Autômatos Celulares Unidimensionais" apresentamos uma avaliação dos parâmetros citados, baseada nessas diretrizes. Uma das principais conclusões dessa análise é o fato de que o parâmetro λ, o mais utilizado e referenciado de todos, possui vários problemas de concepção que limitam a sua eficiência. Outra conclusão é que, da mesma forma que essas diretrizes servem para avaliar parâmetros já propostos, devem também ser utilizadas para direcionar a proposição de novos parâmetros.

Dentre os parâmetros publicados, analisados por nós à luz das diretrizes propostas, selecionamos dois: o parâmetro *sensitividade* e o parâmetro Z. Além desses, elaboramos e testamos vários outros (seguindo as diretrizes) e no final obtivemos um conjunto de cinco parâmetros. Os três novos parâmetros que compõem esse conjunto foram chamados de *atividade absoluta, domínio da vizinhança e propagação da atividade*.[2, 5]

Os parâmetros *sensitividade* e Z nos auxiliam a discriminar relativamente os comportamentos nulo e caótico. Os parâmetros *atividade absoluta e domínio da vizinhança* nos auxiliam a fazer a discriminação relativa dos comportamentos ponto fixo e ciclo duplo. O parâmetro *propagação da atividade* nos auxilia a definir a região caracterizada pelas regras nulas e ponto fixo.

3 COMPUTAÇÃO EM AUTÔMATOS CELULARES

Dentre as capacidades dos ACs que mais atraem o interesse dos pesquisadores está a sua habilidade em realizar computações. Podemos dividir a forma de computação estudada em duas classes: computação explícita e computação implícita.

A computação implícita é aquela executada diretamente da regra do AC, em que a configuração inicial do reticulado é a entrada da computação e a saída é dada pela configuração do reticulado após certo tempo. Já a computação explícita ocorre em um nível mais alto; o papel da regra do AC é prover a dinâmica de sustentação sobre a qual a computação será executada. Nesse caso, a entrada da computação não é toda a configuração inicial do reticulado, mas apenas uma parte dela; assim como a saída é também uma parte da configuração final.[2]

3.1 Computação explícita

Na computação explícita de ACs, busca-se encontrar modelos que associem a dinâmica de ACs com a máquina de Turing. Em geral, os resultados encontrados por esse tipo de abordagem são interessantes apenas como prova do poder computacional universal da arquitetura dos ACs, mas, na prática, nenhuma das construções obtidas são viáveis, visto que são implementações muito pesadas.

Como exemplo desse tipo de abordagem, podemos citar o estudo que mostra que o AC conhecido como *Life* é capaz de realizar computação universal.[4] O *Life*, proposto por John Conway no final dos anos 60, trata-se de um AC bidimensional, com acoplamento dado pela Vizinhança de Moore. Neste tipo de vizinhança, a célula está conectada com suas oito vizinhas que a tocam. A transição do *Life* é dada pela seguinte regra:

- Uma célula que está no estado 1 permanecerá nesse estado se somente 2 ou 3 vizinhos estiverem no estado 1; caso contrário, ela irá para o estado 0.
- Uma célula que está no estado 0 irá para o estado 1 se somente 3 vizinhos estiverem no estado 1; caso contrário, ela permanecerá no estado 0.

O *Life* é constituído de um reticulado bidimensional infinito, começando com uma configuração inicial com um número finito de 1s; todas as outras células recebem o estado 0. Apesar de sua regra de transição simples, este AC com freqüência dá origem a padrões interessantes e complexos. Por exemplo, é muito fácil construir configurações iniciais que irão produzir estruturas localizadas simples e propagantes, chamadas *gliders*. A Figura 1 dá um exemplo de um *glider* que se move um quadrado diagonalmente a cada 4 passos de tempo.

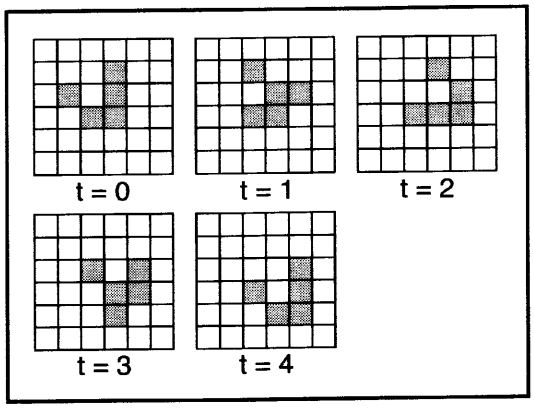

Figura 1 Exemplo de um *glider* de período 4 no *Life*

Outras estruturas encontradas no *Life* são o *glider gun*, uma estrutura estacionária que emite um novo *glider* a cada T passos de tempo e o *eater*, uma estrutura que aniquila *gliders*.[4] *Gliders, glider guns e eaters* são as estruturas-chaves usadas na construção de um computador universal no *Life*. Funções lógicas básicas são construídas a partir de interações entre fluxos de *gliders* disparados por *glider guns*. Por exemplo, em alguns casos, quando dois *gliders* se encontram eles se aniquilam. Esse fato é utilizado para implementar uma porta NOT, conforme a Figura 2 (a).

Um fluxo de bits de entrada A é representado como um fluxo de *gliders*, espaçados de forma que a presença de um *glider* representa um 1 e a ausência, um 0. Esse fluxo colide com um fluxo perpendicular vindo de um *glider gun*. Uma vez que dois *gliders* colidindo se aniquilam, os únicos *gliders* sobreviventes no fluxo vertical serão aqueles que "colidirem" com a ausência de *gliders*, ou "buracos", no fluxo de entrada – os bits 0. O fluxo vertical de saída resultante é o NOT do fluxo horizontal de entrada.

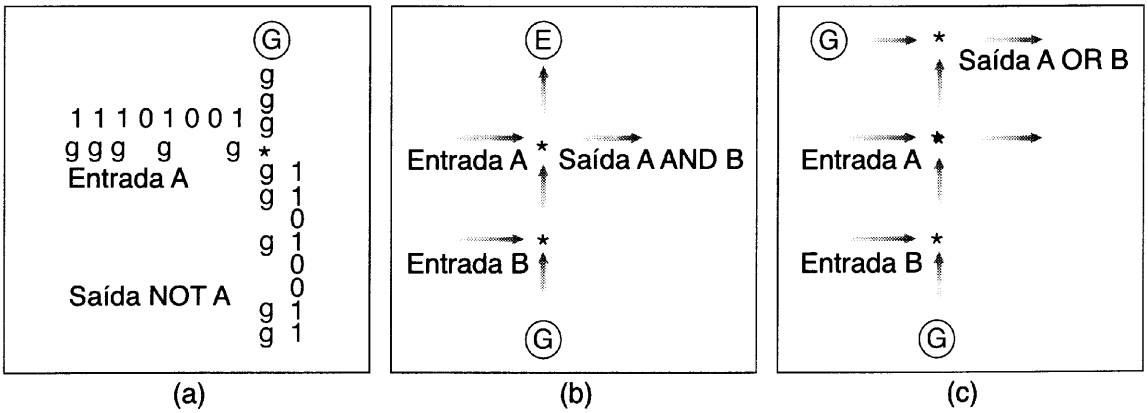

Figura 2 (a) Porta NOT construída a partir de *gliders* (g) e *glider guns* (G) no *Life*. O * representa uma colisão de *gliders*. **(b)** Porta AND construída a partir de *gliders*, *glider guns* e *eaters* (E). **(c)** Porta OR construída a partir de *gliders* e *glider guns*

A porta AND, apresentada na Figura 2 (b), tem dois fluxos de entrada horizontais, A e B, e devolve como saída um fluxo horizontal onde os 1s (*gliders*) correspondem a posições onde A e B são ambos 1. Nessa situação, o *glider* de B aniquila o *glider* do fluxo vertical vindo de G e, portanto, o *glider* de A pode passar pelo fluxo vertical, gerando 1 na saída. É necessário colocar um *eater* para consumir os *gliders* do fluxo vertical que vierem a passar pelas duas colisões – nos casos em que A e B são ambos 0. A porta OR pode ser construída similarmente, como ilustrado na Figura 2 (c).

A construção também inclui formas de se dispor de *gliders* indesejados, copiar fluxos de *gliders*, repor, atrasar ou estreitar fluxos de *gliders* e implementar registros de armazenagem auxiliar via blocos estacionários de células, que podem ser acessados e movidos por frotas de *gliders*. Essas técnicas e as funções lógicas descritas acima podem ser usadas para construir circuitos constituídos de fluxos de *gliders* que podem computar qualquer função recursiva parcial. Dessa forma, demonstra-se que o *Life* é universal. Mas, nessa construção, os recursos são considerados ilimitados. A meta não é construir um computador eficiente no *Life*, mas simplesmente mostrar que, em princípio, o *Life* pode implementar qualquer função computável.

3.2 Computação implícita em ACs

Na computação implícita, a computabilidade de um AC é considerada uma decorrência natural do seu comportamento dinâmico complexo. Assim, as pesquisas nesse tipo de abordagem buscam descobrir como os ACs podem ser utilizados como computadores paralelos rápidos e práticos, sem a preocupação em fazer uma analogia com a Máquina de Turing.

James Crutchfield e James Hanson[4] entendem que o termo computação implícita se refere a estruturas que emergem no comportamento de um sistema e podem ser entendidas em termos computacionais, mesmo na ausência de interpretação sobre o

lugar em que a computação está ocorrendo. O entendimento de como os ACs executam a computação implícita ainda é vago. Hordijk, Crutchfield e Mitchell[17] propõem uma interpretação para esse fenômeno através do conceito de interação entre partículas elementares.

Um exemplo do uso da computação implícita em ACs está na resolução da Tarefa de Classificação da Densidade (TCD).[4] Nesse problema, a meta é encontrar um AC binário unidimensional que decida se uma configuração inicial de 0s e 1s possui, ou não, uma quantidade maior de 1s. Se for maior, o reticulado deste AC deve evoluir para uma configuração ponto fixo de 1s (ou seja, todas as células permanecerão no estado 1); caso contrário, deve evoluir para uma configuração ponto fixo de 0s.

Projetar um algoritmo que execute a TCD é trivial para um sistema com um controlador central ou com armazenagem central, como em um computador com um contador ou em uma rede neural, em que todas as entradas são conectadas a uma unidade central da camada escondida. No entanto, esta é uma tarefa não trivial para ACs de raios pequenos ($r << N$), uma vez que estes se baseiam apenas em interações locais para realizar sua evolução. Para executar bem essa tarefa em um reticulado de tamanho fixo, é requerido muito mais poder computacional do que o que pode ser executado por uma simples célula ou uma combinação linear de células. Como os estados 1s podem estar distribuídos pelo reticulado do AC, este deve ser capaz de transferir informações por grandes distâncias (N). Para conseguir isto, é necessária uma coordenação global das células que estão separadas por grandes distâncias e não podem se comunicar diretamente. Assim, para que a tarefa seja realizada, é necessário explorar o paralelismo intrínseco dos ACs. Foi provado que nenhum AC de estado finito e raio finito, com condições de contorno periódicas, pode executar esta tarefa perfeitamente para todos os tamanhos de reticulados.[18]

A Figura 3 apresenta duas evoluções de uma regra de transição de um AC com vizinhança $r = 3$ e reticulado $N = 149$ células, executando perfeitamente a TCD.[4] Na primeira evolução, a configuração inicial possui uma quantidade de 1s menor do que a de 0s. Na segunda evolução, a configuração inicial possui uma quantidade de 1s maior.

Podemos citar como outros exemplos de computação implícita a Tarefa de Sincronização (TS),[2, 19] a utilização de ACs unidimensionais para o reconhecimento de linguagens formais[4] e o Problema da Sincronização do Disparo do Pelotão (PSDP).[4]

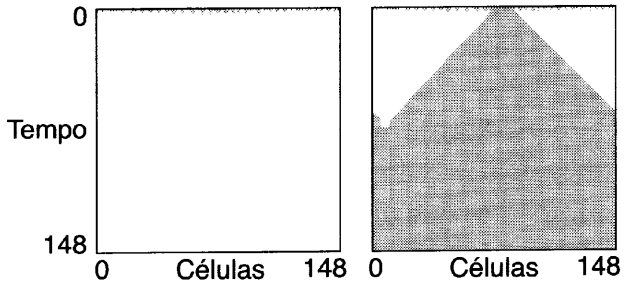

Figura 3 Diagramas espaço-temporais de um AC, classificando corretamente a densidade de 1s dos reticulados iniciais

4 EVOLUÇÃO DE AUTÔMATOS CELULARES

Uma vez definida uma tarefa computacional, não é um problema trivial encontrar um AC que a execute. A programação manual de um AC, além de ser trabalhosa e dispendiosa, possui a desvantagem de a solução encontrada ser muito dependente da abordagem utilizada por quem a propôs, sem a garantia de que tenha sido feita uma análise global das possíveis soluções na tentativa de encontrar a melhor. Por outro lado, uma busca exaustiva no espaço de regras dos ACs torna-se impossível, devido à alta cardinalidade deste espaço. Por exemplo, o espaço de regras dos ACs unidimensionais binários de $r = 2$ é formado por 2^{32} regras e o espaço de regras dos ACs unidimensionais binários $r = 3$ é formado por 2^{128} regras. Uma saída para este problema é a utilização de métodos de busca e otimização, mais especificamente métodos de computação evolutiva, na busca dos ACs que computem uma tarefa determinada.

Packard foi o primeiro pesquisador a publicar resultados utilizando Algoritmos Genéticos (AGs)[20] como ferramenta para encontrar regras de ACs que realizassem uma determinada computação.[21] Ele considerou as regras dos ACs unidimensionais como indivíduos de uma população. Para um AC binário com vizinhança de raio r, o número de entradas de uma tabela de transição é dado por 2^{2r+1} e o espaço de regras é dado por $2^{(2^{2r+1})}$. Tipicamente, uma população de regras representa somente uma amostra minúscula do espaço de todas as regras possíveis. Packard definiu a aptidão de um AC como a habilidade deste em executar uma determinada tarefa computacional. Desta forma, o genótipo de um AC é dado por sua regra de transição e seu fenótipo por sua habilidade em executar a tarefa desejada. O *crossover* entre dois ACs é dado pela geração de duas novas tabelas de transição (filhos), a partir de segmentos de outras duas tabelas de transição (pais). A mutação é dada pela alteração aleatória do *bit* de saída de uma das transições da tabela. Packard utilizou AGs evoluindo ACs para resolver a TCD, apresentada no item 3.

Vários trabalhos têm sido publicados com resultados de evoluções de ACs. Mitchell, Crutchfield e Das[22] apresentam os resultados encontrados para a TCD, assim como um método de análise das estratégias adotadas pelas melhores regras alcançadas. Juntamente com Hordijk, Crutchfield e Mitchell,[17] os autores propõem um modelo formal de estratégias computacionais em ACs utilizando o conceito de partículas. Em dois artigos[19, 22] encontramos os resultados dos experimentos dos autores aplicando AGs para encontrar regras ACs que executam uma outra tarefa computacional conhecida como *tarefa de sincronização*.

Trabalhos que utilizam outras técnicas evolutivas na busca de ACs que computem também têm sido publicados. Andre, Bennet III e Koza[23] descrevem suas experiências com a evolução de ACs utilizando *programação genética* na resolução da TCD. Sipper[24] descreve seu trabalho com ACs não uniformes (uma variação do conceito original dos ACs, em que cada célula pode obedecer a uma regra de transição diferente), evoluídos

a partir de um algoritmo co-evolutivo, também aplicados à TCD. Em outro artigo de Sipper,[3] a co-evolução é aplicada a outras duas tarefas computacionais: a Tarefa de Sincronização e a Tarefa de Ordenação. Juillé e Pollack[25] descrevem seus experimentos utilizando um algoritmo co-evolutivo e uma estratégia de avaliação denominada *compartilhamento de recursos* para encontrar a melhor regra já publicada para a TCD.

4.1 Tarefa de Classificação da Densidade

Neste item, faremos um retrospecto dos principais experimentos e resultados publicados para a Tarefa de Classificação da Densidade, utilizando técnicas evolutivas ou não. Esses experimentos foram realizados em ACs unidimensionais, binários, uniformes e de raio 3. A Tabela 1 concentra os dados das principais regras publicadas para a solução desse problema. Cada regra está referenciada pelas iniciais de seus descobridores, na primeira coluna desta tabela. Temos nas demais colunas a regra em hexadecimal, o ano em que ela foi publicada, os autores do trabalho, o método utilizado para encontrar a regra e o seu desempenho. O desempenho de uma regra para esta tarefa foi medido em amostras de 10^4 configurações iniciais de reticulados diferentes, geradas aleatoriamente. Estes casos são os mais difíceis de classificar, pois possuem aproximadamente 50% de 0s e 50% de 1s.

Em 1978, Gacs, Kurdyumov e Levin[23] apresentaram uma regra de um AC que resolve esta tarefa com razoável desempenho e que durante anos perdurou como a melhor regra para a solução deste problema. Essa regra foi obtida manualmente.

TABELA 1

Regras publicadas para a tarefa de classificação da densidade

Nome	Regra em hexadecimal (ACs de raio 3)	Ano	Autores	Método	Desempenho (em 10^4)
GKL	005F005F005F005F005FFF5F005FFF5F	1978	Gacs, Kurdyumov, Levin	Manual	81,6%
MHC	0504058705000F77037755837BFFB77F	1993	Mitchell, Hraber, Crutchfield	A. Genético	76,9%
DAV	002F035F001FCF1F002FFC5F001FFF1F	1995	Davis	Manual	81,8%
DAS	070007FF0F000FFF0F0007FF0F310FFF	1995	Das	Manual	82,2%
ABK	050055050500550555FF55FF55FF55FF	1996	Andre, Bennet, Koza	Progr. genética	82,3%
JP1	011430D7110F395705B4FF17F13DF957	1998	Juillé e Pollack	A. co-evolutivo	85,1%
JP2	1451305C0050CE5F1711FF5F0F53CF5F	1998	Juillé e Pollack	A. co-evolutivo	86,0%

Em 1988, Packard[21] publicou seus experimentos utilizando AGs para encontrar ACs que executassem essa tarefa. Packard não detalhou a implementação utilizada, nem o desempenho das regras encontradas. Em 1993, Mitchell, Hraber e Crutchfield[6] apresentaram os resultados de seus experimentos utilizando para AGs simples. Eles usaram AGs com tamanho da população $T_P = 100$ e evoluindo para um total de 100 gerações. A cada geração, a avaliação de cada indivíduo era obtida testando o

desempenho do AC em 100 Configurações Iniciais (CIs) de reticulados diferentes. Foram feitas aproximadamente 300 execuções e a melhor regra encontrada é a apresentada na Tabela 1.

Em 1995, Davis e Das construíram, de forma manual e independente, a partir da regra GKL, duas regras que apresentam desempenhos superiores a ela e que são apresentadas na Tabela 1.[23]

Em 1996, Andre, Bennet III e Koza[23] apresentaram os resultados de seus experimentos, utilizando Programação Genética (PG). Eles usaram uma arquitetura paralela de 64 nós e o PG distribuído utilizado tinha um tamanho da população total de T_P = 51.200, sendo que cada nó processava uma população de T_P = 800. O número de CIs testado a cada geração foi 1.000. Foram feitas cinco execuções e a melhor regra encontrada é a apresentada na Tabela 1. Essa foi a primeira regra encontrada através de estratégia evolutiva a exibir um desempenho superior às regras manuais.

Em 1998, Juillé e Pollack[25] divulgaram os resultados de seus experimentos utilizando um algoritmo co-evolutivo e a estratégia conhecida como *compartilhamento de recursos*. Na co-evolução, além de evoluir uma população de regras ACs, existe uma população de CIs que também evolui, ou seja, à medida que o algoritmo interage, as regras dos ACs vão se tornando cada vez mais eficientes em classificar corretamente as configurações iniciais, que, por outro lado, vão se tornando cada vez mais difíceis de ser classificadas. Em uma série de vinte experimentos com tamanho da população T_P = 400, tanto para a população de regras ACs quanto para a população de CIs, a melhor regra encontrada é a apresentada na Tabela 1, com o nome JP1. Em outra série de seis experimentos, utilizando tamanho da população T_P = 1.000, a melhor regra encontrada é a apresentada na Tabela 1, com o nome JP2. Essa é a regra de maior desempenho para a TCD publicada até hoje. Mitchell, Werfel e Crutchfield[26] analisaram os experimentos descritos por Juillé e Pollack;[25] a principal conclusão é que os excelentes resultados obtidos por eles devem-se mais à estratégia *compartilhamento de recursos* do que ao uso do algoritmo co-evolutivo.

5 PARÂMETROS DE PREVISÃO APLICADOS NA EVOLUÇÃO DA TCD

Neste item, apresentaremos os resultados de nosso trabalho recente,[2] no qual utilizamos o poder de previsão do nosso conjunto de parâmetros de previsão, citado no item 2, para obter melhorias significativas na busca evolutiva da Tarefa de Classificação da Densidade.

Inicialmente, calculamos os parâmetros de nosso conjunto selecionado para as regras publicadas, apresentadas na Tabela 1. Na primeira linha da Tabela 2, vemos as faixas dos parâmetros para as regras publicadas da TCD (raio 3). Podemos verificar que os valores dos parâmetros se localizaram em faixas estreitas, com exceção do parâmetro Z. Para efeito de comparação, na segunda linha, apresentamos as faixas esperadas para o comportamento nulo, segundo os valores obtidos no espaço elementar. Essas faixas correspondem aos intervalos dos parâmetros encontrados para todas as regras nulas do espaço elementar (raio 1). Podemos observar que quase todos os parâmetros se enquadram exatamente nas faixas esperadas para dinâmicas nulas. A exceção é o parâmetro atividade absoluta, mas, mesmo neste caso, podemos dizer que esse parâmetro praticamente se enquadra na faixa esperada.

TABELA 2

Faixas dos parâmetros calculados para as regras de ACs de raio 3

	Sensitividade	Domínio da vizinhança	Atividade absoluta	Propagação da atividade	Z
Regras publicadas da tarefa classificação da densidade	0,23 a 0,40	0,84 a 0,91	0,10 a 0,26	0,07 a 0,11	0,24 a 0,6
Regras nulas no espaço elementar	0 a 0,50	0,50 a 0,92	0,12 a 0,50	0 a 0,25	0 a 0,75
95% das regras de raio 3 geradas aleatoriamente	0,45 a 0,55	0,35 a 0,65	0,4 a 0,6	0,15 a 0,35	0,5 a 0,7

Com o objetivo de verificar o quão características desta tarefa seriam as faixas encontradas para as regras publicadas da TCD, confrontamos estes valores com os encontrados para uma amostra de 50.000 regras ACs de raio 3, geradas aleatoriamente. Na terceira linha da Tabela 2, vemos as faixas em que se enquadram os parâmetros da maioria das amostras geradas aleatoriamente (95% das 50.000 regras). Conforme podemos observar, os parâmetros das regras que executam a TCD se enquadram em faixas que não são características da maior porção do espaço de regras, especialmente os parâmetros sensitividade, domínio da vizinhança e atividade absoluta.

Portanto, podemos concluir, a partir da análise da Tabela 2, que as regras publicadas para a Tarefa da Classificação da Densidade realmente se encontram na faixa prevista pelo conjunto de parâmetros para o comportamento nulo e que esta faixa não é trivial para os ACs de raio 3.

5.1 Utilização da heurística dos parâmetros na TCD

De posse destes resultados analíticos tão animadores, passamos à seguinte questão: uma vez que temos a informação da região do espaço dos parâmetros onde boas regras devem ocorrer, não seria possível utilizar esta informação de forma ativa, na busca dos

ACs que executem bem a tarefa? A partir dos resultados apresentados na Tabela 2, nosso objetivo passou a ser incorporar a informação do nosso conjunto de parâmetros como uma métrica auxiliar da otimização feita pelos AGs.

Inicialmente, replicamos um dos experimentos publicados por Mitchell, Hraber e Crutchfield[6] utilizando AGs na busca de ACs que executem bem a TCD e encontramos resultados compatíveis com os publicados pelos autores. Posteriormente, repetimos esses experimentos, inserindo a informação dos parâmetros na função de avaliação dos AGs. Os resultados encontrados para as duas séries de experimentos (100 execuções cada uma) são apresentados na Tabela 3. Cada linha desta tabela apresenta o número de execuções que encontraram desempenhos em determinado intervalo, por experimento. A Tabela 4 apresenta o desempenho das duas melhores regras encontradas em cada um desses experimentos.

Maiores detalhes de como foi feita a incorporação da heurística dos parâmetros nos AGs podem ser encontrados em Oliveira, de Oliveira e Omar.[2, 5]

TABELA 3

Desempenhos encontrados para os experimentos Mitchell e Parâmetros

Faixa de desempenho (%)	Nº regras encontradas em Mitchell	Nº regras encontradas em Parâmetros
< 50	6	0
>50 e < 55	4	0
>55 e < 60	0	0
>60 e <65	16	14
>65 e <70	69	77
>70 e <75	3	7
>75	2	2
Total de experimentos	100	100
Média de desempenho	65%	67,3%

TABELA 4

Desempenhos das duas melhores regras nos experimentos Mitchell e Parâmetros

	Mitchell	Parâmetros
Desempenho 1ª regra	76,1%	80,4%
Desempenho 2ª regra	75,6%	77,2%

Analisando os valores absolutos, com a inserção da informação dos parâmetros (Parâmetros), conseguimos encontrar duas regras com desempenhos superiores ao melhor desempenho encontrado pelo nosso experimento básico (Mitchell). Ressalte-

se ainda que esses resultados são também superiores à melhor regra encontrada pelos pesquisadores que fizeram o experimento original (76,9%, em 300 execuções).[6] Em termos relativos, o desempenho médio da primeira seqüência de experimentos foi 65%, enquanto o da segunda foi 67,3%.

6 CONCLUSÕES

A computação realizada pelos ACs é uma área de interesse crescente, pois estas estruturas são exemplos de sistemas distribuídos, descentralizados e robustos que podem ser utilizados como paradigma para a computação paralela massiva. Além disso, a forma como a computação "emerge" nos ACs serve como modelo para que os cientistas aprofundem seu conhecimento no comportamento de sistemas naturais compostos por componentes simples com comunicação local e que exibem propriedades coletivas emergentes.

Apesar de os primeiros estudos nesta área terem se concentrado na computação explícita dos ACs, cremos que esse tipo de pesquisa já cumpriu seu principal papel, provando o poder de computabilidade universal dos ACs. Vemos os estudos voltados para a computação implícita dos ACs como mais promissores, uma vez que os resultados alcançados por esta abordagem são mais práticos. No entanto, a computação implícita dos ACs ainda é uma área pouco compreendida e que carece de ferramentas que auxiliem na sua programação, visto que a busca manual de regras com habilidade computacional não é factível. Um passo importante nesse sentido foi a utilização, por diversos pesquisadores, de estratégias evolutivas na busca dessas regras.

Acreditamos que a nossa proposta de trabalho – a combinação de parâmetros de previsão de comportamento dinâmico às estratégias evolutivas – pode trazer um novo caminho para o estudo da computação implícita. Para isso, nos baseamos nos resultados significativos obtidos na evolução de ACs em tarefas computacionais. Como podemos observar nas Tabelas 3 e 4, a inserção da informação dos parâmetros conseguiu melhorar substancialmente o desempenho das regras encontradas, tanto em relação aos valores médios quanto aos valores máximos encontrados.[2]

Cabe ressaltar que, além do experimento Parâmetros, descrito no item 5, vários outros experimentos foram realizados, na Tarefa de Classificação da Densidade e em outras duas tarefas computacionais: a Tarefa de Sincronização[2, 27] e a Tarefa de Agrupamento. Podemos afirmar que os resultados demonstraram que sempre existe um ganho no desempenho das regras encontradas, com relação à evolução sem a informação dos parâmetros, ou seja, a heurística dos parâmetros demonstrou uma robustez bastante significativa.[2]

REFERÊNCIAS BIBLIOGRÁFICAS

1. WOLFRAM, S. Computation theory of cellular automata. *Communication is Mathmatical Physics*, v.96, 1984.
2. OLIVEIRA, G.M.B. *Dinâmica e evolução de autômatos celulares unidimensionais*. São José dos Campos: Instituto Tecnológico de Aeronáutica, 1999. (Tese de Doutorado).
3. SIPPER, M. A simple cellular automata that solves the density and ordering problems. *International Journal of Modern Physics*, v.9, n.7, 1998.
4. MITCHELL, M. Computation in cellular automata: a selected review. *Nonstandard Computation*. Weinheim: VCH Verlagsgesellschaft, 1996.
5. OLIVEIRA, G.M.B., de OLIVEIRA, P.P.B., OMAR, N. Evolving solutions of the density classification task in 1D cellular automata, guided by a parameter-based heuristic that forecast their dynamic behavior. *Alife7*, 2000 (to appear).
6. MITCHELL, M., HRABER, P., CRUTCHFIELD, J. Revisiting the edge of chaos: evolving cellular automata to perform computations. *Complex Systems*, v.7, p.89-130, 1993.
7. WOLFRAM, S. Cellular automata. *Los Alamos Science*, v.9, p.2-21, 1983.
8. LI, W., PACKARD, N. The Structure of elementary cellular automata rule space. *Complex Systems*, v.4, p.281-297, 1990.
9. LI, W., PACKARD, N., LANGTON, C. Transition phenomena in cellular Aautomata rule space. *Physica D*, v.45, p.77-94, 1990.
10. LI, W. Parameterizations of cellular automata rule space. Santa Fe Institute. *Preprint*, 1991.
11. LI, W. Phenomenology of nonlocal cellular automata. *Journal of Statistical Physics*, v.68, p.829-882, 1992.
12. CULIK II, K., HURD, L.P., YU, S. Computation theoretic aspects of cellular automata. *Physica D*, v.45, p.357-378, 1990.
13. BINDER, P.M. A phase diagram for elementary cellular automata. *Complex Systems*, v.7, p.241-247, 1993.
14. LANGTON, C.G. Computation at the edge of chaos: phase transitions and emergent computation. *Physica D*, v. 42, 1990.
15. OLIVEIRA, G.M.B., de OLIVEIRA, P.P.B., OMAR, N. Análise crítica das parametrizações no espaço de regras de autômatos celulares unidimensionais – *1º Workshop de Computação* – ITA, 1998.
16. WUENSCHE, A. Complexity in one-D cellular automata: an atlas of basins of attraction fields of one-dimensional cellular automata. *Cognitive Science Research Papers: CSRP 321*. Brighton: University of Sussex, 1994.

17. HORDIJK, W., CRUTCHFIELD, J., MITCHELL, M. Embedded-particle computation in evolved cellular automata. *Proceedings of Physics and Computation*, 1996.

18. LAND, M., BELEW, R. No perfect two-state cellular automata for density classification exists. *Physical Review Letters,* v.74, n.25, p.5148, 1995.

19. DAS, R., CRUTCHFIELD, J., MITCHELL, M., HANSON, J. Evolving globally synchronized cellular automata. *Proceedings of International Conference on Genetic Algorithms.* San Francisco, 6, 1995.

20. GOLDBERG, D.E. *Genetic algorithms in search, optimization and machine learning.* Massachusetts: Addison-Wesley, 1989.

21. PACKARD, N. Adaptation toward the edge of chaos. *Dynamic patterns in complex systems.* Cingapura, p.293-301, 1988.

22. MITCHELL, M., CRUTCHFIELD, J., DAS, R. Evolving cellular automata with genetic algorithms: a review of recent work. *Proceedings of International Conference on Evolutionary Computation and Its Applications.* Moscou, 5, 1996.

23. ANDRE, D., BENNET III, F., KOZA, J. (1996). Evolution of intricate long-distance communication signals in cellular automata using programming. *Proceedings of Artificial Life V Conference.* Japão, 5, 1996.

24. SIPPER, M. Co-evolving non-uniform cellular automata to perform computations. *Physica D*, v.92, p.193-208, 1996.

25. JUILLÉ, H., POLLACK, J.B. Coevolving the "ideal" trainer: application to the discovery of cellular automata rules. *Proceedings of Genetic Programming Conference.* Madison, 3, 1998.

26. MITCHELL, M., WERFEL, J., CRUTCHFIELD, J.P. Resource sharing and coevolution in evolving cellular automata. [on-line.] *EvCA Papers, Coevolution,* 1999. Available at: <http://www.santafe.edu/~evca/Papers/papers.html>

27. OLIVEIRA, G.M.B., OMAR, N., de OLIVEIRA, P.P.B. Parametrização do espaço de regras de autômatos celulares unidimensionais. *Anais do XIII Congresso Brasileiro de Automática – CBA2000,* Florianópolis, 2000.