USO DE FERRAMENTAS LIVRES EM MODELOS ESPACIALMENTE EXPLÍCITOS E MODELOS BASEADOS NO INDIVÍDUO

Aluno: Daniel Vieira Franzolin – IMECC/Unicamp

Orientadora: Sônia Ternes – Embrapa Informática Agropecuária
Co-orientação: Rafael Vilamiu - Embrapa Informática Agropecuária

11 de Julho de 2011

RESUMO

Este trabalho aborda o uso de uma ferramenta de software livre na modelagem baseada no indivíduo, mais especificamente em autômato celular (AC). A ferramenta escolhida para análise detalhada é o software Golly, por ter se mostrado eficiente na execução de regras gerais de AC. A ferramenta é caracterizada de forma a mostrar em quais aspectos ela é eficiente, bem como quais as principais dificuldades encontradas. Para avaliação prática da ferramenta é usado um modelo de espalhamento de fogo construído sob a abordagem de automato celular, encontrado na literatura. Avaliou-se também a utilização da ferramenta para a análise da evolução de modelos de AC mais simples como o Jogo da Vida, e outros modelos disponíveis em sua biblioteca de código-fonte.

1. INTRODUÇÃO

Autômato celular (AC) é uma das técnicas de modelagem baseada no indivíduo (MBI), constituída por um sistema dinâmico discreto composto de uma grade regular de células, cada uma possuindo um número finito de possíveis estados, que variam de acordo com um conjunto de regras de transição. Num dado momento, o estado de cada célula é determinado a partir do estado anterior das células na sua vizinhança, que é definida de acordo com cada regra podem ser desde as células em torno da referencial até a grade toda.

Os ACs tem grande potencial de uso em áreas diversas de conhecimento como nos estudos da dinâmica de mudança de uso do solo (Soares-Filho et al., 2002), avaliação do impacto de políticas públicas (Oguz et al., 2007) e estudos de dispersão de doenças de impacto para a agricultura (Ferreira & Ferreira, 2005).

Para a implementação de AC é importante o uso de ferramentas de software já disponíveis no mercado e bem validadas, de modo a ganhar-se em eficiência em aspectos computacionais como a alocação e uso de memória e processamento, que são problemas frequentes quando do desenvolvimento de AC para modelagem de fenômenos mais realistas. Dessa forma, o usuário pode focar seus esforços no desenvolvimento das regras de evolução dos AC, sem se preocupar com os aspetos computacionais.

Este trabalho tem como objetivo avaliar ferramentas para implementação e simulação de modelos baseados em AC, preferencialmente disponíveis sob uma licença de software livre.

Dentre os softwares pesquisados, o Golly demonstrou ter o maior potencial de uso para implementação de regras complexas de evolução de AC. O software Golly é uma ferramenta sob licença de software livre (GNU, 2011), tendo assim seu código fonte aberto para ser alterado conforme as necessidades do usuário, o que permite flexibilidade de uso e facilita a sua avaliação. O Golly tem por objetivo a exploração e a aplicação da modelagem segundo a abordagem de automato celular, trazendo em sua biblioteca de funções alguns autômatos clássicos já implementados, como o Jogo da Vida de Conway (Berlekamp et al., 1982) e as regras de Von Neumann (Berlekamp et al., 1982).

2. MATERIAL E MÉTODOS

AC é constituído por um sistema dinâmico discreto composto de uma grade regular de células, cada uma possuindo um número finito de possíveis estados (exemplo “on” e “off”, ou 0 e 1), que variam de acordo com um conjunto de regras de transição. Num dado momento, o estado de cada célula é determinado a partir do estado anterior das células em sua vizinhança.

A partir de ferramentas indicadas em artigos científicos, que possibilitam a implementação de AC, foi feita uma vasta busca na Internet, visando avaliar a usabilidade das ferramentas. No processo de caracterização dos softwares foram considerados: sistema operacional (Linux ou Windows), licença de uso sob a qual o software está disponível, tipo de interface, flexibilidade de definição e implementação de regras e evolução do AC, disponibilidade de documentação, última atualização e versão disponível.

Paralelamente à caracterização foi feito o download e a instalação destas ferramentas para qualificá-las conforme os critérios pré estabelecidos. Dentre estas foram testadas o Golly[1] (disponível no repositório Ubuntu), Cellulate[2] e MASyV2[3]. Para efeito de uma avaliação aprofundada o Golly e o Cellulate demonstraram ser mais recomendáveis, pois percebeu-se pelas informações disponíveis em seus sites Web que aparentavam apresentar um grande potencial de uso (Franzolin & Ternes, 2010).

Na primeira etapa do trabalho, para construção e testes de AC usando os softwares selecionados foi usada a regra conhecida como Jogo da vida, desenvolvida pelo britânico John Conway (Berlekamp et al., 1982), e encontrada nas bibliotecas básicas de grande parte dos softwares que manipulam Acs.

Em uma segunda etapa de testes objetivou-se implementar uma regra inédita na ferramenta Golly, usando-se como regra um modelo elaborado para análise de espalhamento de incêndios florestais (Louzada & Ferreira, 2008). De posse dos resultados contidos neste artigo, é possível compará-los com os implementados e gerados pelo Golly, possibilitando uma avaliação qualitativa da ferramenta.

Como fonte de documentação do software para auxiliar a sua avaliação foi utilizada a disponível no sitio on-line do software Golly (GOLLY, 2011) Todos os testes foram realizados sob o sistema operacional Ubuntu 11.04 (UBUNTU, 2011).

2.1 FUNCIONAMENTO DO GOLLY

A estrutura do Golly se baseia em três arquivos, além de contar com a interface gráfica. Trataremos neste trabalho apenas os arquivos que se relacionam com a implementação das regras de evolução de um AC. Nestes encontramos as regras pré implementadas de AC, o tipo de simetria utilizado na execução do modelo, o estado inicial, as variáveis que são os possíveis estados que cada célula pode admitir e dependem da regra que está sendo usada. O nome dos arquivos que contém os códigos citados são *Rules*, o *Patterns* e o *ReadRuletable*, respectivamente.

Primeiramente vamos caracterizar o tipo de arquivo *Patterns*, que contém a dimensão inicial da grade de evolução do AC, ou seja, número de linhas e de colunas . A Figura 1 apresenta um exemplo do código contido no arquivo *Patterns* com os respectivos valores de linhas e colunas (x e y) e a sequência que define o

[1]http://golly.sourceforge.net/
[2]http://cellulate.sourceforge.net/
[3]http://sourceforge.net/projects/masyv/files/MASyV/

estado inicial (“b” significa estado igual a 0, “o” significa estado igual a 1, “$” significa mudança de linha e os números na frente de cada um destes caracteres são o número de vezes que o estado se repete). A Figura 2 mostra a imagem gerada por este código.

A regra usada neste exemplo é o Jogo da Vida do Conway (Golly, 2011) que é denotada por B3/S23 onde “B” significa “birth” e “S” “survival”. A regra B3/S23 significa que se uma célula tem três vizinhos vivos ela nasce (passa de estado 0 para 1), se a célula tem dois ou três vizinhos ela sobrevive (mantém o estado 1) .

```
#C Runs for 23334 gens.  Initial pop = 12.  Final pops = 2898/2895
#C May be the longest-lived 12-cell pattern within a 12x12 square.
#C Emits a LWSS in gen 13811.
#C See also http://www.radicaleye.com/DRH/methuselahs.html
#C Found by Tomas G. Rokicki, some time before Feb 21, 2005.
x = 8, y = 5, rule = B3/S23
4b3o$3bo$o5bo$b2o2bobo$bobo!
```

Figura 1: Arquivo Patterns

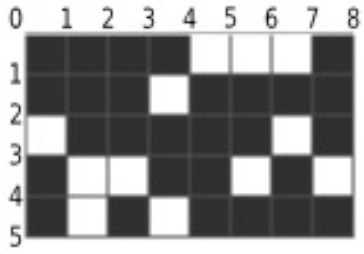

Figura 2: Estado inicial obtido com as instruções do arquivo Patterns

Os arquivos do tipo *Rules* contém a regra de atualização dos estados do AC (as regras já implementadas no Golly são: vonNeumann, Moore, hexagonal, triangularVonNeumann, triangularMoore, oneDimensional), o tipo de simetria da regra, as variáveis da regra e quais os estados que cada uma pode ter.

O terceiro arquivo chama-se *ReadRuleTable,* é escrito na linguagem de programação Python e possui a implementação das regras citadas acima juntamente com as simetrias que cada regra pode admitir na execução. Basicamente este arquivo faz a leitura do *Rules*, o que implica em, alocar variáveis, construir o estado inicial, encontrar a biblioteca da regra e da simetria que será usada na evolução do AC, e por fim é responsável pela permutação, ou seja, a evolução de estados de cada célula no modelo.

A estrutura das regras dentro destes arquivos funciona como um dicionário dentro de dicionário[4]. No Golly o primeiro dicionário tem como chave os nomes de cada regra já implementada e em cada regra existe um dicionário cujas chaves são o tipo de simetria que a regra pode ter existindo pelo menos duas chaves, o “none” que seria a regra com sua simetria mais simples e o “permute”, que é a forma como ocorrerá as permutações durante a execução do modelo.

2.2 PROBLEMA DO ESPALHAMENTO DE FOGO EM AUTOMATO CELULAR

A ideia do modelo criado por Louzada & Ferreira, (2008) foi verificar o quão próximo o uso de automato celular com uma coletânea de regras simples se aproxima do processo de queimadas encontrado na natureza. Com o uso das regras foi feita uma avaliação do surgimento, da propagação e da manutenção das queimas.

Como parâmetros do modelo tem-se a quantidade de pontos iniciais (QuantQueima), o número de graduação de queima (LQ) e o número de graduações para a idade de uma floresta (LR). A caracterização de cada célula é dada pelo par (Tipo,Idade), estas são as duas variáveis usadas no modelo, onde o tipo é 0 se a célula for uma floresta ou 1 se for fogo. A idade varia dentro do intervalo de 1 até o limite máximo de graduações de queima ou floresta, dependendo do tipo 0 ou 1.

A regra de mudança de estado aplicada em todo o sistema é:

- Uma célula tipo 1 se torná do tipo 0 com idade=1 se, na iteração anterior, sua idade era igual a LQ. Caso contrário a idade será incrementada em 1, com limite máximo estabelecido por LQ.

- Uma célula do tipo 0 se tornará do tipo 1, seguindo a regra de probabilidade $P = \frac{Idade}{LR} \cdot \frac{MediaV}{LQ}$, onde MédiaV representa a média de idade das queimadas vizinhas à célula.

Assim com o aumento da temperatura tem-se o aumento da probabilidade de queima da célula, floresta.

3. RESULTADOS E DISCUSSÃO

Como resultado do trabalho realizado tem-se uma levantamento de softwares como Cellulate, MASyV, Golly, Glife entre outros com suas respectivas características, segundo os seguintes critérios: sistema operacional (Linux ou Windows), licença de uso sob a qual o software está disponível, tipo de interface,

[4]Dicionário é uma estrutura de dados dinâmica formadas por objetos (exemplo: vetores, listas, structs) que são identificados por uma chave, esta podendo ser um número real, uma palavra, etc. Para acessar algum objeto é necessário saber apenas a chave que o referencia. Esta estrutura pressupõe que as chaves são retiradas de um conjunto ordenado podendo ser números reais, palavras, ou outro tipo (Cormen et al., 2001)

flexibilidade de definição e implementação de regras e evolução do AC, disponibilidade de documentação, última atualização e versão disponível.

Não foi possível implementar no Golly a regra relacionada ao modelo de incêndios florestais (Louzada & Ferreira, 2008). Isso se deu pois o caminho de pesquisa que foi tomado para implementar e entender o funcionamento do Golly teve como norteador o entendimento de seu código fonte, ou seja, o modo de leitura dos arquivos R*ule* e P*attern*. Para implementar uma nova regra seria necessário modificar a estrutura de dicionário do arquivo *ReadRuleTable* e analisar a forma como implementar a regra de transição no Golly para se criar uma nova. Este procedimento de implementação de regra não é o suportado pelo Golly, pois seria necessário implementar funções na biblioteca do software, além de ser necessário criar arquivos na pasta *Rules* e *Patterns* referentes a nova função.

De todo modo, para que se pudesse testar o algoritmo de Louzada & Ferreira (2008) visando a familiarização com o conceito de AC, implementou-se no software Matlab o código-fonte relativo ao algoritmo. A Figura 3 apresenta o resultado da execução do código relativo a 32 iterações após o início do primeiro foco de incêndio e a Figura 4 mostra o estado final obtido pela execução. A graduação de cores indica a idade em cada célula, sendo que quanto mais próximo do vermelho maior a idade, o que implica no aumento da probabilidade de queima da célula.

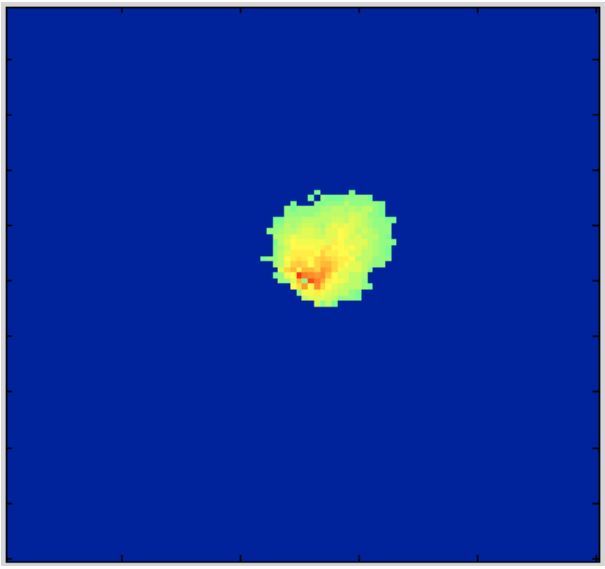

Figura 3: Estado com 32 iterações

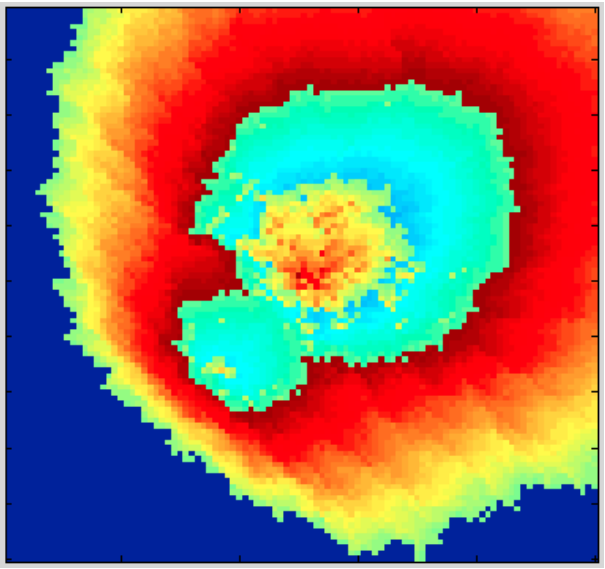

Figura 4: Estado após 104 iterações

Já próximo ao período de finalização dos estudos foram encontrados comentários feitos dentro do código fonte do Golly que indicam que existe um arquivo (*make-ruletable.cpp*) que tem por objetivo auxiliar na implementação de novas regras de AC, não previstas originalmente na biblioteca do Golly. Este arquivo funciona da seguinte forma: tendo já uma função de transição implementada em C/C++ faz-se a compilação usando o *gcc* e o *make-ruletable.cpp;* assim produz-se automaticamente um arquivo de tabelas de regras do Golly, podendo ser usado juntamente com o *Rule* e *Pattern* relacionando a esta regra.

Finalmente, para consolidar os conhecimentos do funcionamento do arquivo *Patterns* implementou-se um modelo simples de AC, o Jogo da vida de Conway. Criou-se um novo arquivo *Patterns* onde foram definidos os valores de x e y, além do estado inicial.

4. CONCLUSÃO

O Golly demonstrou ter a possibilidade de implementar diferentes estados iniciais de AC o que facilita teste de modelos baseados em regras já disponíveis em sua biblioteca. Existe implementado no Golly uma série de scripts feitos em Python e Perl, ambas linguagens de programação com grande uso, que obtêm informações do automato que está em execução como a sua densidade (razão entre as células com valores e o valor total de células). Há também script para mostrar uma geração pré determinada durante a execução do automato, entre vários outros.

Quanto à velocidade de execução o Golly demonstrou ser eficiente e otimizado para as regras contidas em sua biblioteca (vonNeumann, Moore, hexagonal, triangularVonNeumann, triangularMoore, oneDimensional), o tornando assim uma excelente escolha para a evolução de modelos que envolvem estas regras. Percebe-se também um bom gerenciamento de memória pelo software, pois mesmo em AC com uma grade regular grande, por exemplo da ordem de 10^3 , o software não apresenta perda de desempenho.

Um aspecto negativo do software é a dificuldade na implementação de regras complexas de evolução de AC, não presentes originalmente em sua biblioteca. Sugere-se uma análise mais detalhada do arquivo *make-ruletable.cpp* para verificar a forma que ele trata uma regra de AC qualquer, e após a compilação deste arquivo, validar esta nova implementação.

5. AGRADECIMENTOS

O autor agradece ao CNPq pela concessão da bolsa PIBIC, sob a qual o trabalho está sendo desenvolvido.

6. REFERÊNCIAS BIBLIOGRÁFICAS

CORMEN, T.; LEISERSON, C.; RIVEST, R.; STEIN, C.. **Introduction to Algorithms.** 2nd ed. Cambridge, Massachusetts: The MIT Press, 2001. 197-198p

LOUZADA, V.; FERREIRA, W.. **Incêndios florestais em Autômatos Celulares, simples e grandes queimadas**, 2008. 12 p. Dissertação – Instituto de matemática e computação cientifica, Universidade Estadual de Campinas, Campinas.

BERLEKAMP, E. R.; GUY, R. K.; CONWAY, J. H. **Winning ways:** for your mathematical plays, Volume 2: Games in Particular. London: Academic Press INC, 1982. 850 p.

FERREIRA, I.E.P.; FERREIRA, C.P. **Modelagem matemática para dispersão de fungos patogênicos no campo via autômatos celulares**. 2005. In: Anais do IV Congresso de Física Aplicada a Medicina. Instituto de Biociências, Unesp, Botucatu, 2005. Livro de resumos. p.43-44.

OGUZ, H.; KLEIN, A.G.; SRINIVASAN, R. 2007. **Using the Sleuth Growth Model to simulate the impacts of future policy scenarios on urban land use in the Houston-Galveston-Brazoria CMSA**. Research Journal of Social Sciences 2, p.72-82.

SOARES-FILHO, B.S.; PENNACHIN, C.; CERQUEIRA, G.. **DINAMICA** – a stochastic cellualr automata model designed to simulate the landscape dynamics in an Amazonian colonization frontier. 2002. Ecological Modelling, (154): 217-235.

FRANZOLIN, D. V.; TERNES, S.. In: MOSTRA DE ESTAGIÁRIOS E BOLSISTAS DA EMBRAPA INFORMÁTICA AGROPECUÁRIA, 6., 2010, Campinas, SP. **Implementação de autômatos celulares com o uso de software livre:** resumos. Campinas, SP: Embrapa Informática Agropecuária, 2010. p. 27-31.

GNU – Gnu is not unix. GNU General Public License, version 1, 2011. <http://www.gnu.org/licenses/old-licenses/gpl-1.0.html>. Acesso em 3 de junho 2011.

GOLLY, 2011. <http://golly.sourceforge.net/>. Acesso em 3 de junho 2011.

UBUNTU, 2011. <http://www.ubuntu.com/download/ubuntu/download>. Acesso em 3 de junho 2011.