RECIT
Revista Eletrônica Científica Inovação e Tecnologia

Revista Eletrônica Científica Inovação e Tecnologia
ISSN: 2175-1846

https://periodicos.utfpr.edu.br/recit

Uma solução paralela de um modelo multiagente para simulação computacional da propagação de hipotéticas doenças

Wesley Luciano Kaizer
kaizerwesley@gmail.com
Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, Paraná, Brasil.

Rogério Luís Rizzi
rogeriorizzi@hotmail.com
Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, Paraná, Brasil.

Claudia Brandelero Rizzi
claudia_rizzi@hotmail.com
Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, Paraná, Brasil.

Guilherme Galante
gui.galante@gmail.com
Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, Paraná, Brasil.

RESUMO

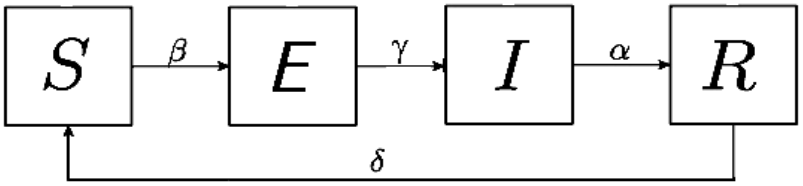
Neste trabalho são apresentados e discutidos os encaminhamentos dados à realização de uma solução paralela para um sistema multiagente que visa simular a propagação de doenças baseadas em modelo compartimental tipo SEIRS. A solução foi obtida empregando programação com múltiplas threads em memória compartilhada via OpenMP. A paralelização foi aplicada às três operações principais da dinâmica de propagação: a rotina de movimentação de agentes, a rotina de contato entre agentes e a rotina de transição de estados dos agentes. Experimentos computacionais realizados indicam que a dinâmica da propagação da doença, assim como a eficiência computacional dos resultados obtidos em programação paralela, são influenciadas significativamente pela quantidade de indivíduos, pela dimensão do ambiente de resolução, pela quantidade de threads empregada na solução, pela distribuição dos tipos de agentes, entre outros fatores. Conclui-se, dentre outros aspectos, que o caminho a uma solução paralela não é a simples paralelização do código sequencial.

PALAVRAS-CHAVE: Modelo multiagentes probabilístico, solução multi-thread, OpenMP.

INTRODUÇÃO

Um modelo tipo Suscetível, Exposto, Infectado, Recuperado e Suscetível (SEIRS) é utilizado para descrever a transmissão de doença entre indivíduos de uma população humana sem dinâmica vital e migração e, portanto, de tamanho fixo, N (DALEY; GANI; 1999). Cada indivíduo pode estar em somente um dos quatro estados: suscetível (S), exposto (E), infectante (I) e recuperado (R). O estado suscetível indica que o indivíduo ainda não contraiu a doença e está apto a adquiri-la. O estado exposto indica que o indivíduo contraiu a doença, mas não é capaz de transmiti-la. O estado infectante indica que o indivíduo está com a doença, sendo capaz de transmiti-la para outros em estado suscetível. O estado recuperado indica que o indivíduo já foi infectado com a doença e não pode se contaminar novamente nem transmiti-la para outros em tal período. Após o período de recuperação ou a imunidade do indivíduo terminar, ele é novamente inserido como um indivíduo suscetível na população (VYNNYCKY; WHITE, 2010). O fluxo de transição de estado dos indivíduos no modelo é mostrado na Figura 1.

Figura 1 – Fluxo de transição de estados dos indivíduos no modelo SEIRS



Fonte: Autoria Própria (2016).

Na Figura 1, β designa a taxa de transmissão da doença no encontro de indivíduos saudáveis com infectantes. O indivíduo fica um período γ no estado exposto. O parâmetro α designa a taxa de recuperação em que indivíduos doentes se recuperam, estando imunes àquela doença por outro período. Este período é caracterizado pelo parâmetro δ que modela qual é a taxa de perda de imunidade de indivíduos que estejam recuperados, passando-os novamente para o estado suscetível.

Considerando-se uma população fixa no tempo, pode-se descrever matematicamente o fluxo de transição dos possíveis estados dos indivíduos no modelo SEIRS através de um sistema de equações diferenciais ordinárias, EDOs, em que N=S(t)+E(t)+I(t)+R(t) designa o tamanho fixo da população, onde as quantidades de indivíduos nos diferentes estados variam no tempo. Assim, obtêm-se sistemas de EDOs não lineares, que não têm solução explícita conhecida (BOYCE; DIPRIMA, 2010). As equações para os casos contínuos e discretos são como apresentados em (1).

$$\begin{cases} \frac{d(S(t))}{dt} = -\frac{\beta}{N} I(t)S(t) + \delta S(t) & ; S(o) = S_o \\ \frac{d(E(t))}{dt} = +\frac{\beta}{N} I(t)S(t) - \gamma E(t) & ; E(o) = E_o \\ \frac{d(I(t))}{dt} = +\gamma E(t) - \alpha I(t) & ; I(o) = I_o \\ \frac{d(R(t))}{dt} = +\alpha\, I(t) - \delta S(t) & ; R(o) = R_o \end{cases} \begin{cases} S^n = S^{n-1} - \frac{\beta}{N} I^{n-1} S^{n-1} + \delta S^{n-1} & ; S^o = S_o \\ E^n = E^{n-1} + \frac{\beta}{N} I^{n-1} - \gamma E(t)^{n-1} & ; E^o = E_o \\ I^n = I^{n-1} + \gamma S^{n-1} - \alpha I(t)^{n-1} & ; I^o = I_o \\ R^n = R^{n-1} + \alpha\, I^{n-1} - \delta S^{n-1} & ; R^o = R_o \end{cases} \quad (1)$$

Nas formulações apresentadas em (1) S_0, E_0, I_0 e R_0 representam, respectivamente, as distribuições iniciais das populações de suscetíveis, expostos, infectantes e recuperados. A solução analítica do PVI contínuo não é conhecida ou não é possível de ser obtida. Resta, pois, obter numericamente sua solução resolvendo a formulação discreta, em que *n* e *n-1* denotam os diferentes níveis de tempo que são avaliados numericamente às quantidades envolvidas no modelo.

Modelagem multiagente

Os modelos em EDOs são à base do modelo baseado em agentes, que foi desenvolvido e utilizado nas simulações computacionais (WOOLDRIDGE, M.; JENNINGS, N. R, 1995), (RUSSELL; NORVIG, 2004). A modelagem empregada para simular o espalhamento de hipotética doença de transmissão direta em indivíduos considera agentes baseados em modelos que são definidos espaço-temporalmente, especificando-se como ocorre a transição do seu estado num intervalo de tempo e seu movimento no ambiente, de uma posição para outra no passo de tempo. Um passo de tempo é especificado como um ciclo de transição.

A especificação formal de um agente é realizada através de operador de evolução que define o estado atual do agente quando interagindo com o ambiente. Esse operador decorre da composição entre os operadores de transição temporal, que realiza uma transição do estado interno do agente considerando sua interação com outros agentes e com o ambiente, e o operador de transição espacial, que movimenta o agente de sua posição para outra, considerando-se os atributos de conectividade e mobilidade.

Formalmente, um agente $\chi(t)$ é definido espaço-temporalmente especificando como ocorre a transição do seu estado num intervalo de tempo e seu movimento no espaço. Neste modelo existem agentes que estão nos estados suscetíveis ou expostos ou infectantes ou recuperados, e qualquer um desses agentes é especificado como $\chi(t)=(P,C,E)$. Nesta especificação, P é a sua posição no ambiente, E designa o seu estado interno e C um registro para o contador de ciclos que controla as transições de estados de acordo com as taxas de exposição, ou de recuperação ou de perda de imunidade de indivíduos (RIZZI, RIZZI, KAIZER, 2015).

Cada agente implementa uma operação de evolução λ que atualiza o estado atual do agente quando interagindo com o ambiente, definido como $\lambda(\chi(t))= \sigma(\mu\,(\chi(t)))$, que decorre da composição entre os operadores μ e σ. O operador de conecto-mobilidade, μ, movimenta o agente de sua posição

considerando-se os atributos de conectividade e mobilidade, e o operador σ realiza a transição do estado interno do agente considerando-se sua interação com outros agentes e o ambiente.

O operador espaço-temporal $\lambda(\chi(t))$ realiza as operações do agente $\chi(t)$ movimentando-o da posição (i,j) para uma posição (ξ,η) no ciclo de tempo atual, t, para o ciclo de tempo t + 1. Formalmente o estado resultante das operações no agente $\chi(t)$ é representado como em (2):

$$\lambda(\chi(t)^{(i,j)}) \equiv \sigma(\mu\ (\chi(t+1))^{(\xi,\eta)}) \qquad (2)$$

Para a dinâmica populacional são considerados três tipos de operações, as de movimentação, contato e de transição de estados. Nas operações de movimentação, os indivíduos são movimentados dentro de um ambiente virtual com topologia matricial através de suas vizinhanças de Moore às posições escolhidas aleatoriamente, respeitando os limites do ambiente. Nas operações de contato ocorre, probabilisticamente, a transmissão da doença através dos indivíduos infectados para os indivíduos suscetíveis que ocupam uma mesma posição no ambiente. Nas transições de estados, ocorre a passagem de estados dos indivíduos para expostos, e depois de expostos para infectantes e então de infectantes para recuperados. Por fim de recuperados para suscetíveis. Tais operações são realizadas na sequência em que foram apresentadas e uma vez a cada ciclo, que consiste na aplicação dos operadores sobre a população de indivíduos e geração de arquivos de saída. Uma simulação é composta por vários ciclos.

Para especificar algoritmicamente tal operador é necessário determinar configurações e parâmetros que estabelecem quais as condições do ambiente e do estado interno do agente. O ambiente é composto por uma estrutura matricial com L linhas e C colunas, onde se considera uma estrutura de vizinhança tipo Moore para efeitos de conecto-mobilidade de um agente que está numa posição (i,j) a uma posição (ξ,η) vizinha. Quando da distribuição inicial dos agentes ou mesmo da sua movimentação em qualquer ciclo da simulação, não há restrições à quantidade máxima que ocupam uma mesma posição.

Os parâmetros das quantidades iniciais de agentes e as taxas são definidos em faixas, o que significa que cada parâmetro possui um valor mínimo e máximo. Nas operações que dependem das taxas ou na movimentação, é randomizado um valor dentro da sua faixa ou da posição a ser movimentada para estabelecer o caráter probabilístico do modelo. A Tabela 1 apresenta os parâmetros utilizados.

Tabela 1 – Lista dos parâmetros necessários à simulação

Nome	Descrição	Mínimo	Máximo
Simulação	Simulações executadas para uma simulação Monte Carlo	MC	MC
Ciclos	Ciclos de simulação que serão executados em cada simulação	Tempo	Tempo
Linhas	Quantidade de linhas da matriz base do ambiente	#L	#L
Colunas	Quantidade de colunas da matriz base do ambiente	#C	#C
Agentes suscetíveis	Quantidade inicial de agentes no estado suscetível	#S	#S
Agentes expostos	Quantidade inicial de agentes no estado exposto	#E	#E
Agentes infectantes	Quantidade inicial de agentes no estado infectado	#I	#I
Agentes recuperados	Quantidade inicial de agentes no estado recuperado	#R	#R
Taxa de exposição	Taxa de infecção pela doença do agente suscetível em contato com um agente infectado	β_{min}	β_{max}
Ciclos de latência	Ciclos o agente fica no estado exposto	γ_{min}	γ_{max}
Ciclos de infectância	Ciclos o agente fica no estado infectado	α_{min}	α_{max}
Ciclos de recuperação	Ciclos o agente fica no estado recuperado	δ_{min}	δ_{max}

Fonte: Autoria Própria.

Na Tabela 1, os parâmetros da distribuição inicial, da especificação do domínio e das taxas ou períodos às operações definem as quantidades ou faixas que são empregadas nos experimentos computacionais. Os agentes transitam probabilisticamente nos estados em faixas que variam neste trabalho em $10=\beta_{min}\leq\beta\leq\beta_{max}=15$, $70=\gamma_{min}\leq\gamma\leq\gamma_{max}=85$, $20=\alpha_{min}\leq\alpha\leq\alpha_{max}=25$ e $60=\delta_{min}\leq\delta\leq\delta_{max}=65$. Mas essa escolha pode ser modificada por faixas adequadas às doenças em que tais parâmetros sejam conhecidos. Para a manutenção da densidade dos diferentes estados dos agentes no ambiente de simulação, deve-se considerar que sua parametrização, especificada por #L×#C, depende das quantidades dos agentes nos estados suscetível, exposto, infectante e recuperado, #S, #E, #I, #R, respectivamente.

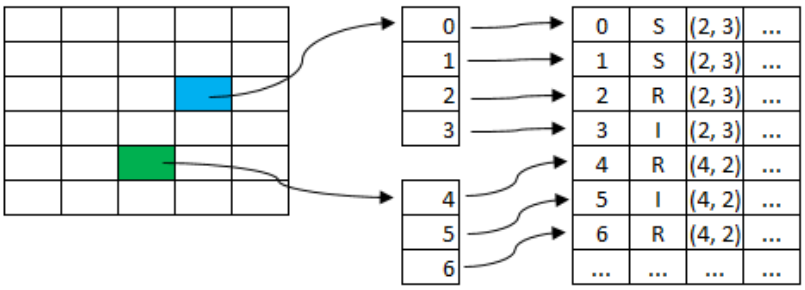
A metodologia tipo Monte Carlo adotada neste trabalho, consiste na execução de determinada quantidade de simulações com mesma configuração inicial às quantidades de agentes e à dimensão do ambiente (CHWIF; MEDINA, 2014). Porém, pela natureza probabilística do modelo, as simulações têm, eventualmente, diferentes taxas ou períodos de infecção, recuperação e imunização. O valor médio, VM, de cada variável ou estado em cada passo de tempo, em uma simulação Monte Carlo é especificada pela razão VM/MC, onde MC designa a quantidade de simulações executadas. Mas esse valor pode ser configurado para explorar a distribuição dos parâmetros e a eficiência computacional, entre outros fatores da simulação.

Estratégia de implementação

Para implementar os algoritmos de movimentação de agentes, contato entre agentes e transição de estados dos agentes, utilizou-se estruturas de dados puramente vetoriais ou matriciais, que reduzem significativamente o uso de ponteiros para acesso às estruturas, melhorando notavelmente o desempenho das simulações. A vetorização de todas as estruturas de dados foi possível graças à simplicidade do agente e do ambiente modelados na simulação, que permitem expressar completamente seu significado através de vetores homogêneos e de tamanho fixo.

A Figura 2 representa a estrutura base do ambiente de simulação que consiste em uma matriz de *n* linhas e *m* colunas, onde em cada posição é armazenada um vetor contendo os endereços dos agentes que a ocupam. Como o ambiente e a matriz de agentes são estruturas matriciais, este vetor armazena endereços para determinadas linhas da matriz de agentes, sendo que cada linha caracteriza um agente.

Figura 2 – Ilustração da estrutura matricial empregada para implementação do ambiente e dos agentes



Fonte: Autoria Própria (2016).

A estrutura matricial do ambiente tem papel fundamental na simulação, principalmente na operação de contato entre agentes, permitindo otimizar o tempo de execução para trechos de códigos custosos computacionalmente. A implementação realizada utilizou fortemente esta estrutura de ambiente, o que diminuiu significativamente o processamento necessário nas rotinas que utilizavam a matriz de agentes, pois para cada posição da matriz do ambiente, sabia-se previamente quais agentes a ocupavam, sendo desnecessário realizar uma coleta e processar toda a matriz de agentes inúmeras vezes.

Implementação em paralelo: OpenMP

À implementação do modelo foi empregada a linguagem de programação *C*, com a posterior paralelização de rotinas críticas utilizando a *API* para programação *multi-thread* com memória compartilhada *OpenMP* (CHAPMAN; JOST; VAN DER PAS, 2007). A *API OpenMP* viabiliza a paralelização de programas escritos em *C*, *C++* e *Fortran*, através da inclusão de diretivas de programação ou *pragmas* diretamente no código-fonte. A inclusão de *pragmas* é mais comum em

estruturas de laços de repetição controlados por contador. Utilizando a *API OpenMP*, a implementação realizada foi paralelizada em suas rotinas críticas: as rotinas de movimentação de agentes, de contato entre agentes e transição de estados dos agentes.

As rotinas foram escolhidas para paralelização, pois demandam muito processamento, principalmente a de contato entre agentes, pois trata diretamente com a representação matricial do ambiente e com a matriz de agentes utilizados na simulação. Como a matriz do ambiente pode ser dividida em linhas, sendo essas passíveis de processamento paralelo e não apresentando dependência de dados entre si, a estratégia utilizada foi o processamento de cada linha da matriz do ambiente por uma *thread* diferente, de forma que o ambiente é então dividido em porções que contêm várias linhas da matriz base e que são escalonadas entre as *threads*. Esse escalonamento de tarefas entre as *threads* é transparente ao programador, pois a *API OpenMP* realiza automaticamente este processo.

As Figuras 3, 4, e 5 ilustram, respectivamente, os pseudocódigos das implementações realizadas para as operações de movimentação de agentes, de contato entre agentes e de transição de estados dos agentes realizadas durantes as simulações. Destaca-se na cor vermelha, em cada figura, o local exato da inserção da diretiva de compilação relacionada à paralelização do código utilizando a *API OpenMP*.

Figura 3 – Ilustração do pseudocódigo da operação de movimentação dos agentes

```
#PRAGMA OMP PARALLEL FOR
Para todos os agentes indivíduos na simulação faça
        direção = randomiza_numero_inteiro_entre(0, 7);
        Se direção igual a 0 então
                subtrai_um_posicao_x(indivíduo);
        Fim se
          Se direção igual a 1 então
                soma_um_posicao_x(indivíduo);
                                                    Fim se
          Se direção igual a 2 então
                subtrai_um_posicao_y(indivíduo);
        Fim se
          Se direção igual a 3 então
                soma_um_posicao_y(indivíduo);
        Fim se
          Se direção igual a 4 então
                subtrai_um_posicao_x(indivíduo);
                    subtrai_um_posicao_y(indivíduo);
        Fim se
          Se direção igual a 5 então
                subtrai_um_posicao_x(indivíduo);
                soma_um_posicao_y(indivíduo);
        Fim se
          Se direção igual a 6 então
                soma_um_posicao_x(indivíduo);
                    subtrai_um_posicao_y(indivíduo);
        Fim se
          Se direção igual a 7 então
                soma_um_posicao_x(indivíduo);
                    soma_um_posicao_y(indivíduo);
        Fim se
Fim para
```

Fonte: Autoria Própria (2016).

Figura 4 – Ilustração do pseudocódigo da operação de contato entre os agentes

```
#PRAGMA OMP PARALLEL FOR
Para cada posição x, y do ambiente faça
        Se posição x, y do ambiente contém indivíduos infectados então
                Para cada indivíduo suscetível na posição x, y faça
                        Se percentual_randomico <= taxa_de_infecção então
                         Muda_estado_do_indivíduo_para_infectado();
                        Fim se
                Fim para
        Fim se
Fim para
```

Fonte: Autoria Própria (2016).

Figura 5 – Ilustração do pseudocódigo da operação de transição de estados dos agentes

```
#PRAGMA OMP PARALLEL FOR
Para todos os agentes indivíduos na simulação faça
    Se indivíduo está em estado infectado então
        Se contador de períodos do indivíduo é maior ou igual ao período de
infectância então
            Muda_estado_do_indivíduo_para_recuperado();
            Zera_contador_de_periodos_do_indivíduo();
        Senão
            Soma_um_contador_de_periodos_do_indivíduo();
        Fim se
    Fim se
    Se indivíduo está em estado recuperado então
        Se contador de períodos do indivíduo é maior ou igual ao período de
recuperação então
            Muda_estado_do_indivíduo_para_suscetivel();
            Zera_contador_de_periodos_do_indivíduo();
        Senão
            Soma_um_contador_de_periodos_do_indivíduo();
        Fim se
    Fim se
Fim para
```

Fonte: Autoria Própria (2016).

RESULTADOS E DISCUSSÃO

À manutenção da densidade dos indivíduos no ambiente, considerou-se o caso base de uma matriz com #L=#C=65. Neste caso, à distribuição inicial em que $\#N_0$=1000, $\#S_0$=10 e $\#R_0$=0, observa-se que a razão $\#N_0/\#L^2$=1000/4225≅0,237. Assim, tomou-se, por aproximação, que a dimensão do ambiente a essa representação matricial é calculada como $\#L_{0=}(\#N_0/0,23)^{1/2}$. A Tabela 2 apresenta as pertinentes informações a respeito das dimensões das matrizes e das quantidades de indivíduos, e apresentam os tempos de execução, em segundos, da execução sequencial e com 4 *threads*. A coluna “Ambiente” apresenta o valor utilizado para as dimensões #L e #C do ambiente, as colunas #N, #S e #I indicam as quantidades totais de agentes, de agentes suscetíveis e de agentes infectantes, respectivamente. As colunas “Serial”, “Paralelo” e “Speedup” indicam os tempos de execução, em segundos, para a execução sequencial, paralela e o *speedup* obtido para o caso de teste, respectivamente.

Tabela 2 – Dados da distribuição inicial e os respectivos resultados

Teste	Ambiente	#N	#S	#I	Serial	Paralelo	Speedup
1	65	1000	990	10	0,22	0,55	0,4
2	114	3000	2970	30	0,69	1,68	0,410714
3	208	10000	9900	100	2,63	6,36	0,413522
4	294	20000	19800	200	6,71	13,2	0,508333
5	466	50000	49500	500	24,2	40,11	0,603341
6	659	100000	99000	1000	52,81	80,94	0,652459
7	807	150000	148500	1500	66,43	110,17	0,602977
8	932	200000	198000	2000	114,43	151,99	0,752878
9	1142	300000	297000	3000	152,09	227,73	0,667852
10	1318	400000	396000	4000	199,5	309,07	0,645485
11	1474	500000	495000	5000	250,77	388,38	0,645682
12	1546	550000	544500	5500	279,96	428,04	0,654051
13	1865	800000	792000	8000	399,26	608,19	0,656472
14	2085	1000000	990000	10000	493,19	752,34	0,655541

Fonte: Autoria Própria (2016).

Observando os speedups e tempos de execução obtidos com os testes apresentados, conclui-se que a simples paralelização de trechos de códigos nem sempre conduz a ganhos imediatos de desempenho, sendo necessário atentar às estruturas de dados utilizadas na implementação, bem como às densidades e proporções de cada tipo de agente na população, que influenciam fortemente os resultados finais obtidos. Sendo a densidade populacional muito baixa, a doença propaga-se rapidamente na população de agentes, levando sua dinâmica a um fim prematuro e acarretando resultados impróprios. Os experimentos realizados também mostraram que o tempo de execução do algoritmo sequencial é igual ao tempo de execução do algoritmo paralelo com uma única *thread*.

Da Tabela 2 é possível observar que, em geral, os tempos sequenciais são menores que os tempos obtidos com o emprego de 4 *threads*. Algumas possíveis justificativas a esses resultados, ainda em investigação, decorrem do fato que a carga de processamento não é suficiente para compensar a criação e execução do problema em 4 *threads*. Adicionalmente destaca-se que foi empregada uma estrutura de dados já otimizada, dentre várias pesquisadas, sendo que esta mostrou ser altamente eficiente. Trabalhos em andamento investigam essas questões, configurando problemas com diferentes quantidades de agentes e ambiente com maior dimensão.

CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado um modelo baseado em indivíduos para a simulação computacional à propagação de hipotéticas doenças. Deu-se enfoque à solução paralela de um modelo multiagente empregando programação *multi-thread* em memória compartilhada via OpenMP.

Foi utilizado a *API OpenMP* na paralelização das três principais rotinas da simulação: a rotina de movimentação de agentes, a rotina de contato entre

agentes e a rotina de transição de estados dos agentes. Essas rotinas são críticas pois computam diretamente sobre a estrutura matricial do ambiente, sobre as listas dos agentes em uma determinada posição do ambiente e sobre a matriz que contém a população dos agentes, demandando bastante tempo de processamento na conclusão de suas operações.

Pelos resultados exibidos na Tabela 2 conclui-se que, embora aplicado métodos à paralelização da implementação, não foi possível obter ganho de desempenho, que seriam notáveis caso a implementação em paralelo tivesse alcançado tempos de execução menores às da implementação sequencial. Experimentos computacionais indicaram que a eficiência computacional é influenciada significativamente pela configuração do problema, sobretudo no que se refere à quantidade de agentes, a dinâmica de interação entre eles, assim como a dimensão do ambiente. Para trabalhos futuros pretende-se investigar os reais motivos do decréscimo de desempenho obtido com a solução paralela, estudando mais detalhadamente o problema e as estruturas de dados utilizadas. Pretende-se desenvolver novas versões utilizando outras metodologias de paralelização como o *MPI* e *CUDA*, utilizando estruturas de dados baseadas na manipulação direta de *bits* (*bitstring*).

A parallel solution of a mult agent model for computational simulation of hypothetical disease spreading

ABSTRACT

In this work we present and discuss the results of a parallel solution for a SEIRS model disease simulation. The solution was obtained using multi-thread programming in shared memory architectures using OpenMP API. The results show that the serial and parallel numerical solution are similar, considering the Monte Carlo strategy for obtaining the mean values of simulations that is relevant to validate the results. Computational experiments indicate that the dynamics of disease spreading and the computational efficiency obtained in parallel version are significantly influenced by the amount of individuals, the size of the environment, the amount of threads used, the distribution of types of agents, among others factors. We conclude that the path to parallel solutions is not the simple parallelization of sequential code, without considering such aspects.

KEYWORDS: Probabilistic multi-agent model, multi-threaded programming, OpenMP.

REFERÊNCIAS

BOYCE, William E.; DIPRIMA Richard C. **Equações Diferenciais Elementares e Problemas de Valores de Contorno**. Editora LTC, 9.ed, 2010.

DALEY, D. J.; GANI J. **Epidemic Modelling – An Introduction**. Editora Cambridge University, 1.ed, 1999.

CHAPMAN, Barbara; JOST, Gabriele; VAN DER PAS, Ruud. **Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)**. The MIT Press, 2007.

CHWIF, L.; MEDINA, A. C. **Modelagem e Simulação de Eventos Discretos: Teoria e Aplicações**. 4ª edição. São Paulo: Elsevier Brasil, 2014.

WOOLDRIDGE, M.; JENNINGS, N. R. **Intelligent Agents – Theory and Practice. The Knowledge Engineering Review, Inglaterra, 1995.**

RIZZI, R. L.; RIZZI, C. B.; KAIZER, W. L. **Um Modelo Multiagentes para a Dinâmica de populações de Aedes Aegypti com População Humana Acoplada com a Coexistência de Múltiplos Sorotipos de Dengue e a Presença ou não da Bactéria Wolbachia**. Relatório de Pesquisa não publicado. Universidade Estadual do Oeste do Paraná. 2015. 135 p.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**. 2.ed. São Paulo: Campus, 2004.

VYNNYCKY, Emilia, WHITE, Richard. G**. An Introduction to Infectious Disease Modelling**. Oxford University Press. New York. 2010.

Recebido: 25 jul. 2016.
Aprovado: 23 nov. 2016.
DOI:
Como citar: KAISER, W. L.; RIZZI, R. L.; RIZZI, C. B.; GALANTE, G. Uma solução paralela de um modelo multiagente para simulação computacional da propagação de hipotéticas doenças. R. Eletr. Cient. Inov. Tecnol., Medianeira, v. 2, n. 14, p. 114-126, jul./dez. 2016. Disponível em: <https://periodicos.utfpr.edu.br/recit>. Acesso em: XXX.
Correspondência:
Wesley Luciano Kaiser
Rua Otelo Celestino de Castilhos, 827, Vale do Sol, Cascavel, Paraná, Brasil.