Paralelização Eficiente na Simulação da Elevação do Nível do Mar em Áreas de Reentrâncias Maranhenses

Francisco Borges Carreiro Filho[1], **Helder Pereira Borges**[2], **Omar Andres Carmona Cortes**[2]

[1]Bacharelado em Sistemas de Informação
Departamento de Computação (DComp)
Instituto Federal do Maranhão (IFMA)
São Luis – MA – Brasil

[2]Departamento de Computação (DComp)
Instituto Federal do Maranhão (IFMA)
São Luis – MA – Brasil

{francisco}@acad.ifma.edu.br, {helder, omar}@ifma.edu.br

Abstract. *Modeling actual phenomena in computer systems is a complex activity that usually demands processing a massive amount of data. In this context, high-performance computing is an attractive option to reduce the application's execution time. Thus, we investigate parallel applications through multicore (CPU) and many-core (GPU) computing for simulating the rising tide, a significant problem caused by climatic changes, especially in mangrove areas. We simulated the application using cellular automata in three different cenários: sequential, parallel using MPI, and parallel using C-CUDA. Results have shown that parallel versions presented expressive gains, especially in the GPU architecture, reaching a speedup of 2.88 using MPI and 253,03 using a C-CUDA.*

Resumo. *A representação computacional de fenômenos do mundo real é uma atividade complexa, normalmente exigindo o processamento de uma grande quantidade de dados. Neste contexto, a computação de alto desempenho se apresenta como uma solução na redução do tempo de execução nas aplicações. Assim, neste trabalho investiga-se o uso de paralelismo usando multi núcleos (CPU) e muitos núcleos (GPU) na simulação da elevação do nível do mar, um problema sério causado pelas mudanças climáticas, especialmente em regiões de mangue. A simulação foi realizada através do uso de autômatos celulares em três cenários: sequencial, paralelo usando MPI e paralelo usando C-CUDA. Os resultados demonstraram um ganho de performance expressivo nas versões paralelizadas, em especial em GPU, chegando a um speedup de 2,88 e 253,02 em MPI e C-CUDA, respectivamente.*

1. Introdução

O manguezal está presente em praticamente toda a região litorânea brasileira, sendo o estado do Maranhão o maior detentor deste biossistema na região norte-nordeste [Alves 2001]. Dessa forma, a região de estudo encontra-se em uma área denominada de Reentrâncias Maranhenses, no Maranhão, Brasil. Esta área possui 2.680,911,2 ha

(dois milhões, seiscentos e oitenta e um mil, novecentos e onze vírgula dois hectares) de extensão (BRASIL, DECRETO nº 11.901 – 11/06/1991).

Não obstante, é necessário ressaltar que os manguezais desempenham diversas funções naturais de grande importância financeira e ecológica, dentre as quais destacam-se: a proteção da linha costeira atuando contra a ação erosiva das ondas e marés, retenção de sedimentos carreados pelos rios, filtro biológico ao agir contra partículas contaminantes como os metais pesados entre outras [Alves 2001].

O contato imediato dos mangues é com o mar. Por esse motivo, o aumento do nível do mar é um dos mais graves impactos nos manguezais resultantes das mudanças climáticas. Este fenômeno traz efeitos devastadores sobre as áreas de manguezal, afetando a biodiversidade, segurança alimentar e bem-estar de várias comunidades que vivem nesses locais [Vermeer and Rahmstorf 2009]. Nesse contexto, utilizar modelos que simulem a elevação do mar é essencial para prever os impactos dessa elevação de maneira geral.

Em termos de simulação, um modelo denominado BR-MANGUE [Bezerra 2014] foi desenvolvido utilizando um banco de dados com referências espaciais da Ilha Upaon-Açu no estado do Maranhão e um conjunto de regras que simulam a resistência ou a inundação/erosão do mangue em relação ao aumento do nível das marés. Assim, a simulação do aumento do nível do mar nas Reentrâncias Maranhenses visa prover um mapa de dados da região, possibilitando deduções que auxiliem preventivamente em tomadas de decisões quanto à preservação deste tipo de ambiente. Além disso, este modelo implementa conceitos encontrados nos Autômatos Celulares idealizado por John Von Neumann e o matemático Stanislaw Ulam em 1966.

Como a simulação pode utilizar autômatos celulares, estes acabam por ser transformados em matrizes que descrevem os detalhes do local. Em outras palavras, quanto mais detalhes e quanto maior a precisão requerida, mais a computação de alto desempenho se faz necessária para processar essas matrizes. Para lidar com esse requisito, umas das soluções é utilizar a computação de alto desempenho disponibilizadas tanto pelas CPUs com múltiplos núcleos quanto pelas *Graphical Processing Units* (GPUs), sendo estes últimos essencialmente diferentemente dos processadores convencionais, pois uma GPU consegue invocar centenas, senão milhares, de recursos (*threads*) em paralelo [Kirk 2007].

Alguns estudos tem utilizado a simulação para estudar o impacto da mudança climática no Brasil. Em [Soares 2009] desenvolve-se um modelo conceitual para a resposta do mangue a elevação do nível do mar. Em [Faraco et al. 2010] faz-se a análise de vulnerabilidade das mudanças climática no sistema ecológico-social dos mangues brasileiros. Porém, o único trabalho que foi encontrado que utiliza a computação paralela em simulações do impacto da elevação do nível do mar nos mangues é o trabalho de [Jesus et al. 2018] que utilizou MPI na paralelização da referida simulação, obtendo speedup de 1,23 em um modelo usando 97.401 células, speedup de 1,87 em um modelo com 504.064 células e speedup super linear de 7,31 com 3.648.064 de células, sendo todas as execuções com apenas dois processos. Por outro lado, o trabalho não alcançou bons resultados usando 4 processos. Dessa forma, este trabalho melhora a implementação em MPI chegando a ser executado em 8 processos e estende sua implementa para ser executado em uma GPGPU (GPU de propósito geral).

Nesse contexto, este artigo investiga o uso de paralelismo multi núcleo (CPU) e muitos núcleos (GPU) para o processamento de autômatos celulares com o objetivo de simular o aumento do nível do mar em áreas de mangues nas reentrâncias maranhenses, conhecida como ilha de São Luís. O trabalho está dividido da seguinte forma: a Seção 2 apresenta os conceitos básicos sobre autômatos celulares; a Seção 3 detalha como é feita a simulação e as respectivas paralelizações; a Seção 4 mostra os resultados obtidos pela paralelização, apresentando também o resultado da simulação via mapa; finalmente, a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Autômatos Celulares

Os autômatos celulares foram introduzidos por John Von Neumann e Stanislaw Ulam em 1966 como modelos simples, cujo objetivo era estudar processos biológicos, como por exemplo, a autorreprodução. De maneira geral, um autômato celular é composto por um arranjo de células, na qual cada uma é composta por um conjunto de estados finitos e seus vizinhos.

Na verdade, qualquer sistema com muitos elementos discretos idênticos passando por interações locais determinísticas pode ser modelado como um autômato celular [Wolfram 1982], sendo que suas propriedades fundamentais são: o espaço celular, a vizinhança de cada célula, o estado e as regras de transição [Weimar 2003].

A vizinhança de uma célula depende da regra utilizada para obter os vizinhos mais próximos e de sua forma. A vizinhança determina com quais células pode ocorrer a interação no espaço celular, sendo que a vizinhança é definida de acordo com o tipo de problemas sendo simulado. Os tipos mais comuns são a vizinhança de Neumann e a de Moore. Na vizinhança de Neumann cada célula só possui quatro vizinhos, que são o de cima, de baixo, da esquerda e da direita. Na vizinhança de Moore estende-se os vizinhos para as quatro diagonais interagindo com um total de oito células.

O espaço celular pode ser representado de diversas formas, sendo as mais básicas uma grade unidimensional ou bidimensional. Neste trabalho será utilizada a representação bidimensional, pois tem representação direta através de matrizes, ou seja, consiste em células arranjadas em um espaço de m linhas por n colunas, ou seja, $m \times n$.

O estado de uma célula é um valor dentro de um conjunto de possíveis estados em um determinado instante de tempo. Assim, dado um determinado estado de uma célula, as regras de transição ditam qual será seu próximo estado de acordo com uma ”função de transição“, que pode variar de uma equação matemática até uma simples tabela de transições.

3. Modelagem e Simulação

Como previamente mencionado, neste trabalho utiliza-se o modelo específico chamado de BR-MANGUE, levando em consideração quatro componentes: (i) elevação do nível médio do mar (NMRM); (ii) formas de uso e ocupação do solo; (iii) restrições ambientais; e, (iv) dinâmica do manguezal. Cada componente corresponde a um módulo do modelo, com a finalidade de definir um escopo, detalhar e unir características em comum para cada variável do subsistema.

O NMRM é usado como referência topográfica no processo de inundação de áreas adjacentes. O seu valor, que pode ser um número em ponto flutuante, representa a altura

da coluna de água e, quando há o avanço do mar em direção ao continente, as áreas de manguezal e outras formas de cobertura de solo poderão ser inundadas/erodidas, sendo que a elevação média global do mar impacta negativamente sobre o manguezal, uma vez que pode reduzir a área de colonização do ecossistema [Agrawala et al. 2003].

As formas de uso e ocupação do solo representam os estados que uma célula pode assumir. Assim, esta variável permite simular a resistência do mangue, pois o seu valor influencia diretamente na dinâmica, tornando-se uma possível barreira frente à inundação ou uma área propensa à colonização dos manguezais.

As restrições ambientais devem representar o potencial limitante das classes de solo. Por exemplo, as classes que não permitem a colonização pelo manguezal são consideradas restritivas. Outros tipo, tais como, processos de modificação do sedimento, tais como bancos de lama (acreção longitudinal) e o aumento na altura do banco de lama (acreção vertical) também estão presentes neste componente, sendo que a sedimentação tende a influenciar no NMRM e no componente de uso e ocupação do solo, pois uma vez que há acreção vertical, a altura do banco de lama pode se contrapor à elevação da coluna d'água, minimizando o seu impacto.

Finalmente, a dinâmica do manguezal representa os impactos resultantes das interações entre o aumento do NMRM sobre o terreno, o padrão de ocupação do solo e as restrições ambientais à colonização do manguezal. Destas interações têm-se as condições para a resistência, migração ou perda de áreas de manguezal.

3.1. Simulação do Avanço do Nível do Mar

Para a simulação do avanço do nível do mar, uma célula pode ter onze estados possíveis como mostra a Tabela 1 com suas referidas cores em RGB. Como vizinhança utiliza-se a vizinhança de Moore, sendo que cada célula possui os seguintes atributos: linha, coluna, altitude, uso, solo, usoAntigo, soloAntigo e isNull.

Tabela 1. Estados de uma célula no BR-MANGUE

id	Estado	Cor (RGB)
1	Mangue	(0, 100, 0)
2	Vegetação Terrestre	(154, 205, 50)
3	Mar	(0, 0, 255)
4	Área Antropizada	(240, 230, 140)
5	Solo Descoberto	(210, 180, 140)
6	Mangue Inundado	(255, 0,0)
7	Vegetação Terrestre Inundada	(152, 251, 152)
8	Solo Descoberto Inundado	(184, 134, 11)
9	Área Antropizada Inundada	(189, 183, 107)
10	Mangue Migrado	(60, 179, 113)
11	Leito Canal	(0, 255, 255)

As regras de transição são as seguintes:

1. A NMRM é inicializada com a taxa de elevação do nível do mar;

2. Caso a célula atual possua a altimetria inferior aos seus vizinhos e os seus vizinhos são água ou encontram-se em estado inundado, esta célula passa a ter seu estado como erodido/inundado. Caso contrário, o seu estado permanece inalterado;
3. Caso a célula atual seja mangue e o valor do AIM (Área de Influência das Marés) seja superior a altimetria das células adjacentes, a célula corre o risco de sofrer uma migração. Caso contrário, o mangue permanece sem alteração;
4. Caso não existam barreiras à sua colonização, por exemplo, construções antrópicas, o manguezal poderá migrar em direção ao continente. Outra possibilidade é do mangue resistir ao aumento do NMRM em situações em que ocorra intensa acreção vertical e acreção longitudinal de sedimentos, e que estas excedam as taxas de elevação da coluna d'água [McIvor et al. 2013];
5. O ecossistema manguezal oferece proteção à linha de costa contra a ação erosiva das marés, e caso a altura da maré seja superior a altimetria do mangue, este sofrerá migração.

A Figura 1 mostra de forma detalhada as transições possíveis dadas as regras de transição recém apresentadas.

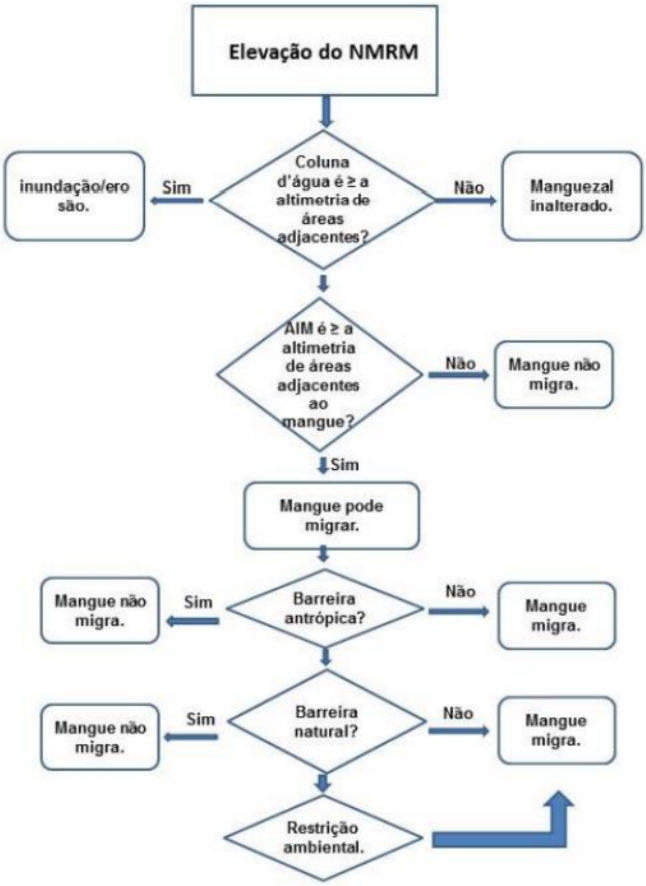

Figura 1. Fluxograma de transições do modelo BR-MANGUE

3.2. Implementação Paralela em MPI

Nesta abordagem utiliza-se o modelo de programação paralela conhecida como SPMD (*Simple Program Multiple Data*) usando *Message Passing Interface* (MPI), no qual o

mesmo código é executado em diferentes partes dos dados. Assim, o autômato celular é dividido em pedaços e distribuído entre os processos através uma instrução de comunicação coletiva *MPI_Scatter*. Quando cada processo chega ao fim de seu processamento invoca-se uma barreira (*MPI_Barrier*) para garantir que todos os processos terminaram sua simulação antes de juntar os resultados. Finalmente, os dados são juntados por uma instrução *MPI_Gather*. A Figura 2 mostra o processo de divisão de um espaço celular em quatro processos.

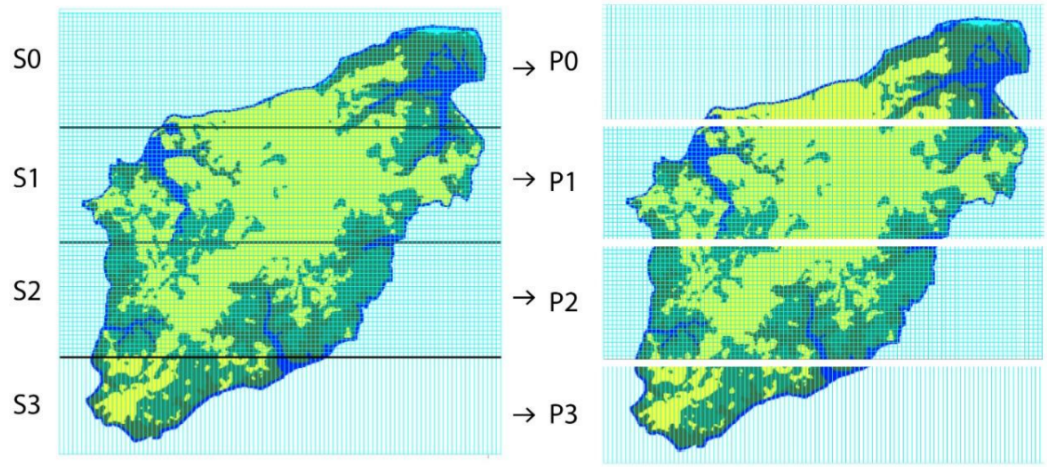

Figura 2. Divisão de um espaço celular em quatro processos

É importante notar que como o espaço celular é dividido, uma cópia das bordas devem ser enviada para o processo que recebeu a parte vizinha. Na Figura 2, P0 deve mandar uma cópia de sua borda para P1, P1 deve mandar uma cópia de suas bordas para P0 e P2, e assim por diante. Esse envio só é feito quando se encontra uma instrução *MPI_Request* sendo executada pelas funções *MPI_Recv* e *MPI_Send*.

3.3. Implementação Paralela em GPU

A vantagem da paralelização em GPU é que o espaço celular não precisa mais ser dividido, pois será executado como uma matriz na qual cada célula será tratada por um *thread* que aplicará as regras de transição. Como são executadas várias iterações sem a necessidade de comunicar a GPU com a CPU, sua execução passa a ser bastante eficiente. A única ressalva ocorre quando a quantidade de células for superior à quantidade de núcleos CUDA. Nesse caso, o programa executará dois ou mais *kernels* com base na quantidade restante de células para completar a matriz. Uma alternativa é escolher um número de blocos correspondente ao tamanho da matriz dividido pela quantidade máxima de *threads* que cada bloco pode assumir na GPU. Desta forma, a GPU invocará *threads* que processarão cada célula da matriz, porém, em diversos blocos até que o processamento seja concluído.

Após o processamento de todas as *threads* o comando *__syncthreads()* funciona como uma barreira para um bloco, garantindo que no final da execução, todas as *threads* do bloco estejam sincronizadas. Com a barreira estabelecida, a matriz pode ser devolvida para a memória RAM e visualizada caso seja necessário. É importante ressaltar que a comunicação entre GPU e CPU é lenta e só deve ser feita quando realmente necessária.

4. Simulação Computacional

Nos testes computacionais foram utilizados três cenários com diferentes tamanhos de espaços celulares. O primeiro é um espaço real contendo 82412 células. Os demais são extensões do original duplicando ou quadruplicando suas células. A Tabela 2 apresenta a quantidade de células e seu respectivo uso, sendo BD 1 a base de dados original.

Tabela 2. Bases de dados e suas células

Uso	BD 1	BD 2	BD 3
Mangue	16.720	66.880	267.520
Vegetação Terrestre	31.256	125.024	500.096
Mar	33.760	135.040	540.160
Antropizada	652	2.608	10.432
Solo Descoberto	24	96	384
Total	82.412	329.648	1.318.592

As simulações foram executadas em um processador i7 4.7Ghz, com 32 GB de RAM, 6 núcleos físicos e hiper *threading* (12 *threads*), sistema operacional Windows 10 versão Home de 64 bits e uma placa de vídeo NVIDIA GeForce 1080ti arquitetura Pascal com 3584 núcleos CUDA. O programa foi implementado usando o Microsoft Visual Studio 2019, Cuda Toolkit 11 e Microsoft MPI (MSMPI) v10.1.2.

A Tabela 3 mostra os resultados do tempo médio de execução em segundos e o *speedup* alcançados pela implementação em MPI em 2, 4 e 8 processos. Como pode ser observado o caso com 82412 células apresenta um melhor *speedup* com 2 processos. Como a matriz é a menor de todas, acredita-se que o tempo de comunicação acabou influenciando no processo todo já que há um aumento no tempo de execução a medida que se aumenta a quantidade de processos. O que não acontece nos dois maiores cenários, nos quais o processamento domina o tempo de execução.

Não obstante, em todos os cenários o *speedup* não se altera significativamente com a quantidade de processos, o que gera um queda significativa da eficiência. Por exemplo, a eficiência do cenário 1 (82412 células) cai de 77% com dois processos para 15% com 8 processos. Além disso, observa-se um ligeiro *speedup* superlinear nos cenários com mais células com 2 processos. Isso pode se dever ao fato da arquitetura compartilhar memória e os dados estarem em cache compartilhada dentro do processador.

Tabela 3. *Speedup* - MPI

Células	Sequencial (s)	2P (s)	Sp	4P (s)	Sp	8P (s)	Sp
82412	18,7	12,1	1,54	14.8	1,26	15,1	1,23
329648	46,05	19,4	**2,88**	17,6	2,61	17,2	2,67
1318592	64,8	24,9	2,60	24,3	2,66	23,6	2,74

A Tabela 4 apresenta os resultados da execução em GPU usando 256, 512 e 1024 *threads* pro bloco. Observa-se nos resultados que a execução em GPU é mais rápida que a execução com múltiplo núcleos em CPU, sendo que o *speedup* chega a 152,03 no primeiro cenários com 82412 células e com 1024T. O melhor resultado foi obtido utilizando-se 512 *thread* no segundo cenário com 329648 células.

Tabela 4. *Speedup* - GPU

Células	Sequencial (s)	256T (s)	Sp	512T (s)	Sp	1024T (s)	Sp
82412	18,7	0.153	122	0.132	141,6	0.123	152,03
329648	46,05	0.183	251	0.182	**253,02**	0,192	220
1318592	64,8	0.4	162	0.389	166,58	0.47	137

De maneira geral o tempo de execução em GPU é sempre melhor do que em CPU com múltiplos núcleos. Isso se dá pelo fato da GPU realizar 256, 512 ou 1024 operações em cada ciclo de clock da GPU sem comunicação com a CPU. Dessa forma, a diferença é visivelmente significativa entre MPI vs GPU, e um teste-t ratifica essa significância.

A simulação utilizou o modelo previamente apresentado e proposto em [Bezerra 2014]. A Figura 3 mostra o resultado da simulação no inicio, meio e fim totalizando 120 anos (ano zero, 60 anos e 120 anos) a frente. Os resultados mostram uma erosão/migração das áreas de mangue especialmente na parte superior esquerda da ilha e nas áreas indicadas pelas setas, ocasionadas pelo aumento do nível do mar. Observa-se também um aumento significativo de áreas que antes eram vegetação terrestres e foram transformadas em mangue inundado.

5. Conclusões

Este artigo mostrou a paralelização em CPU usando MPI e em GPU usando C-CUDA da simulação da elevação do nível do mar através do uso de autômatos celulares. Através da paralelização obteve-se um *speedup* máximo de 2,88 no segundo cenário utilizando dois processos com a paralelização em MPI. Com relação a GPU, obteve-se um *speedup* de 253,02 em GPU usando C-CUDA também no segundo cenário, sendo essa diferença entre MPI e C-CUDA significativa em termos de desempenho. Em outras palavras, o uso da GPU gera um ganho significativo no tempo de execução de uma simulação complexa.

Como trabalhos futuros, ficam as propostas de: (i) aumentar a quantidade de dados, executando o algoritmo em um ambiente com múltiplas GPUs; (ii) utilizar OpenACC para verificar se o ganho em termos de *speedup* se equivale a programação em C-CUDA; e, (iii) tentar um mix de OpenMP mais GPU tentando extrair o máximo possível de paralelismo da simulação;

Referências

Agrawala, S., Ota, T., Ahmed, A., Smith, J. B., and van Aalst, M. K. (2003). Development and climate change in Bangladesh: focus on coastal flooding and the Sundarbans. Technical report, Head of Publications Services, OECD, France.

Alves, J. R. P. (2001). *Manguezais: educar para proteger*. FEMAR: SEMADS.

Bezerra, D. d. S. (2014). Modelagem da dinâmica do manguezal frente à elevação do nível do mar.

Faraco, L. F. D., Andriguetto-Filho, J. M., and Lana, P. C. (2010). Methodology for assessing the vulnerability of mangroves and fisherfolk to climate change. *Pan-American Journal of Aquatic Sciences*, 5(2):205–223.

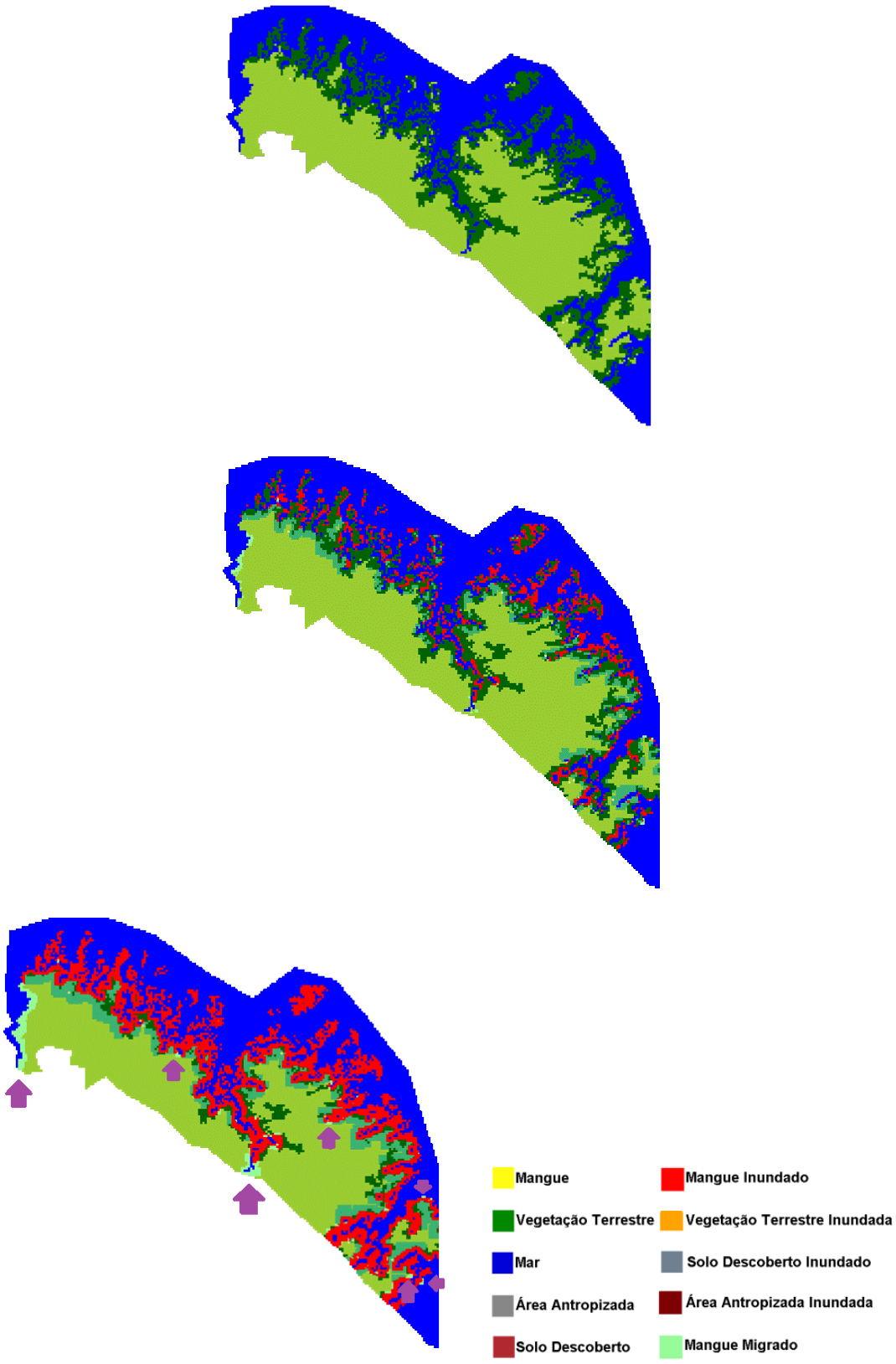

Figura 3. Simulação no inicio, meio e fim

Jesus, E. M. d., Santos, A. L. S., Cortes, O. A. C., and Borges, H. P. (2018). Aplicação de autômatos celulares paralelos para simulação do impacto causado pela elevação do nível do mar. In *Anais do Simpósio de Sistemas Computacionais de Alto Desempenho (WSCAD), Workshop de Iniciação Científica (WIC)*, pages 538–543.

Kirk, D. (2007). Nvidia cuda software and gpu parallel computing architecture. In *Proceedings of the 6th international symposium on Memory management*, volume 7, pages 103–104.

McIvor, A., Spencer, T., Möller, I., and Spalding, M. (2013). The response of mangrove soil surface elevation to sea level rise. Technical report, The Nature Conservancy and Wetlands International.

Soares, M. L. G. (2009). A conceptual model forresponses of mangrove forest to sea level rise. *Journal of Coastal Research*, 56:267–271.

Vermeer, M. and Rahmstorf, S. (2009). Global sea level linked to global temperature. *Proceedings of the National Academy of Sciences*, 106(51):21527–21532.

Weimar, J. R. (2003). *Simulation with Cellular Automata*. Logos Verlag Berlin.

Wolfram, S. (1982). Cellular automata as simple self-organizing systems. Technical report, Calif. Inst. Technol., Pasadena, CA.