Paralelização de um Algoritmo Genético em Problema Inverso de Condução de Calor

Sabrina Bergoch Monteiro Sambatti
Haroldo Fraga de Campos Velho
Laboratório Associado de Computação e Matemática Aplicada(LAC)
Instituto Nacional de Pesquisas Espaciais (INPE)
sabrina@lac.inpe.br, haroldo@lac.inpe.br

Resumo

Este trabalho tem como objetivo comparar duas estratégias de implemtação paralela de um algoritmo genético, modelo Ilha e Stepping Stone, aplicado a solução de um problema inverso em condução de calor.

1. Introdução

A teoria do problema inverso, no sentido amplo, teve início na geofísica. Isto porque geofísicos procuram estudar as camadas interiores do planeta utilizando dados coletados na superfície [9]. Atualmente suas aplicações estenderam-se para diversas áreas, tais como processamento de imagem e aerodinâmica.

O professor russo Oleg Mikailivitch Alifanov deu uma das melhores definições de problemas inversos, enunciando que "a solução de um problema inverso consiste em determinar causas desconhecidas baseando-se nas observações de seus efeitos" (http://www.me.ua.edu/whatis.html).

O problema inverso pode ser formulado como um problema de otimização não linear com restrições. A solução do problema de otimização pode ser obtida através de métodos determinísticos ou métodos estocásticos. Em alguns casos este método implícito pode requerer milhões de iterações, demandando muito tempo de cálculo. Assim, a computação paralela é uma estratégia adequada para tratar problemas que demandam uso intensivo de computação.

Neste trabalho foi usado como caso teste o problema inverso em uma barra unidimensional [7, 8], que trata da determinação da temperatura no estado inicial através de dados obtidos em instantes posteriores. Aqui, o problema de otimização é solucionado pelo uso de Algoritmo Genético (método estocástico).

O algoritmo genético, inspirado nos mecanismos de evolução de população de seres vivos onde os mais aptos sobrevivem, é altamente paralelizável possuindo várias estratégias de implementação. Nesse trabalho foram utilizadas duas estratégias de algoritmos genéticos paralelos (AGP) com o objetivo de realizar um estudo sobre o tempo de processamento e a velocidade de convergência.

2. Processamento de Alto Desempenho

A demanda por computadores de alto desempenho é cada vez maior nas áreas de aplicações científicas e de engenharia, pois muitos desses problemas possuem simulações com características que facilitam a paralelização.

O avanço no processamento de alto desempenho está atrelado na velocidade da transferêncai de dados, o que torna viável a interconexão de vários computadores – podendo até superar o desempenho dos supercomputadores. Este tipo de máquina é conhecida como sistema distribuído, sendo que cada processador (ou nó) possui memória local.

Devido a diversidade de arquiteturas paralelas, a implementação de algoritmos necessita de recursos específicos. Assim, linguagens, compiladores e bibliotecas foram propostos com a finalidade de explorar o máximo da capacidade de processamento, incluindo mecanismos de sincronização e comunicação.

Existem várias formas de programação paralela, e a escolha dependerá da arquitetura que está sendo utilizada. Nesse trabalho foi utilizado um sistema de computação distribuída, onde a comunicação entre os nós é feita através de troca de mensagens realizadas por chamadas explícitas às rotinas da biblioteca MPI(*Message Passing Interface*).

Os algoritmos podem ser implementados utilizando diversas estratégias de paralelização, como por exemplo na dinâmica molecular, que tem como paralelismo natural os cálculos das forças e as atualizações das velocidades/posições que podem ser feitos simultaneamente para todos os átomos.

Da mesma forma, a implementação do algoritmo genético paralelo admite várias estratégias de paralelização, simulando a evolução independente de várias subpopulações.

Para quantificar a melhora obtida pela paralelização de

um código utiliza-se medidas de desempenho por ter uma forte ligação com o tempo de execução.

O *Speedup* é definido como a razão entre o tempo execução de um algoritmo em um único processador (T_1) e o tempo de execução em "p" processadores. Seja T_p o tempo gasto na execução de uma aplicação paralela com p processadores, o valor do *speedup Sp* é dado por:

$S_p = \frac{T_1}{T_p}$

A eficiência Ep de um algoritmo é definida como a razão entre o *speedup* e o número p de processadores, mostrando o quanto o paralelismo foi explorado no algoritmo. Essa razão deve ser menor ou igual a 1:

$E_p = \frac{S_p}{p} \leq 1$

Na programação paralela muitos códigos não são totalmente paralelizáveis, possuindo partes intrinsicamente seqüenciais. Gene Amdahl desenvolveu uma técnica, conhecida por *Lei de Amdahl*, que calcula quanto do programa pode ser executado em paralelo. Assim, a razão entre o tempo paralelo e o seqüencial é estabelecido, mostrando que o *speedup* não é linear, o que aconteceria em condições ideais, num programa totalmente paralelo [3].

3. Problemas Inversos

Existem dois motivos para o estudo de problemas inversos: quando se deseja conhecer estados passados ou parâmetros de um sistema físico e quando se deseja saber como influenciar um sistema através de seu estado presente ou através dos parâmetros, com o objetivo de dirigilos para que se obtenha um estado desejado no futuro [9].

Formalmente o problema direto pode ser expresso na forma matricial:

$$d = G(m) \tag{1}$$

onde $d = [d_1 d_2 ... d_D]^T$ é o vetor/função de dados (causas), $m = [m_1 m_2 ... m_M]^T$ é o vetor/função de parâmetros (efeitos) a ser determinado, e G é uma matriz/operador (modelo matemático).

Existem alguns métodos para solucionar o problema 1, os mais usados são: inversão direta, mínimos quadrados (garante existência da solução, mas é instável na presença de ruídos), métodos de regularização e outros tipos (molificação, filtros digitais, filtro de Kalman, Redes Neurais, etc).

O Método de regularização altera o problema original através da adição de um termo, de maneira a transformar o problema *mal-posto* num problema *bem-posto*.

Os tipos mais conhecidos de regularização são o Princípio da Máxima Entropia e a Regularização de Tikhonov [10]. Na análise inversa o problema é alterado da seguinte forma:

$$min\{\|d - Gm\|_2^2 + \alpha \|Bm\|_2^2\} \tag{2}$$

O regularizador de Tikhonov pode ser expresso por:

$$B[m] = \sum_{k=0}^{p} \mu_k \|m^{(k)}\|_2^2 \tag{3}$$

A escolha do parâmetro de regularização é a maior questão da regularização, pois pode-se suavizar muito ou pouco a solução. Existem alguns critérios para a escolha do parâmetro, os mais usados são: discrepância de Morozov e o critério de Hansen.

Como visto na definição de problemas inversos, a solução é obtida através da minimização de uma forma funcional, isto é, a solução de um problema de otimização. Dentre os vários métodos existentes para solucionar os problemas de otimização, aqui o algoritmo genético é empregado.

4. Metodologia

Este trabalho teve como objetivo investigar duas estratégias de paralelização do algoritmo genético aplicado na solução de um problema inverso de determinação de condição inicial em condução de calor unidimensional, definido pela equação da difusão :

$$\frac{\partial^2(x,t)}{\partial x^2} = \frac{1}{\alpha}\frac{\partial T(x,t)}{\partial t}, \quad \text{para } 0 \leq x \leq L \text{ e } t > 0, \tag{4}$$

onde as condições de contorno são dadas por:

$$\frac{\partial(x,t)}{\partial x} = 0 \ \text{ em } x = 0 \ \text{ e } x = L, \tag{5}$$

e a condição inicial é definida como:

$$T(x,t) = f(x) \ \text{ em } t = 0 \tag{6}$$

A solução do problema direto é explicitamente obtida por [11]:

$$T(x,t) = \sum_{m=0}^{+\infty} e^{-\beta_m^2 t}\frac{1}{N(\beta_m)}X(\beta_m, x) \times \int_0^1 X(\beta_m, x^{'})f(x^{'})dx^{'} \tag{7}$$

onde $X(\beta_m, x) = \cos(\beta_m x)$ são as auto-funções associadas ao problema, $\beta_m = m\pi$ os autovalores e $N(\beta_m)$ as normas correspondentes.

Podendo ser expresso de forma compacta por:

$$T_j = \sum_{i=1}^{I}[f_i C_{ij} + f_{i+1} C_{ij}], \tag{8}$$

Portanto, o perfil de temperatura fica determinado através da solução do sistema matricial

$$T^{t_f} = C^{t_f} f \tag{9}$$

A solução do problema inverso baseado nessa formulação é fortemente mal-posto, ou seja, os mínimos quadrados não determinam uma solução adequada [8].

A expressão (2) fornece um método adequado para a solução, mesmo com dados contaminados por ruído. Assim sendo, a função custo do presente problema é formulada como:

$$J_\gamma(f) = \sum_{j=1}^{M} [T_j^{exp} - T_j^{mod}(f)]^2 + \gamma\Omega(f) \qquad (10)$$

onde $\Omega(f) \equiv \| f \|_2^2$ representa o termo de regularização e γ é o parâmetro de regularização, que foi determinado pelo método da discrepância.

No presente trabalho foi empregada a regularização de Tikhonov de ordem zero. O problema de otimização foi resolvido pelo algoritmo genético, que tem sido efetivo para a solução deste problema [1]

4.1. Algoritmo Genético

Algoritmo Genético é um método de otimização inspirado nos mecanismos da evolução de populações de seres vivos. Em 1975, John Holland [5] desenvolveu uma teoria que deu origem a primeiro algoritmo genéico (AG).

As técnicas de busca de otimização, que consistem em achar a solução que corresponda ao ponto máximo ou mínimo da função objetivo (conforme o problema em questão), apresentam espaço de busca (onde estão todas as possíveis soluções do problema) e função de aptidão (corresponde a nota que o indivíduo irá receber, e por onde será julgado)

O pseudo-código, algoritmo (1), representa um algoritmo genético padrão. Seja *Pop(t)* a população de cromossomos da geração *t*:

Algorithm 1 Algoritmo Genético padrão

```
inicializar Pop(t)
avaliar Pop(t)
while critério de parada não for satisfeito do
  t ← t + 1
  selecionar Pop(t) de Pop(t-1)
  aplicar crossover em Pop(t)
  aplicar mutação em Pop(t)
  avaliar Pop(t)
end while
```

Primeiramente, é gerada uma população inicial, formada por um conjunto aleatório de cromossomos, que podem ter representação real ou binária, dependendo da aplicação. Cada cromossomo ou indivíduo representa uma possível solução, a população é avaliada e cada indivíduo recebe uma nota denominada função de aptidão, que normalmente corresponde ao valor da função objetivo, refletindo todas as características desejáveis para a solução do problema. Com base nesses valores, é aplicado um operador de seleção, onde os mais aptos sobrevivem (elitismo) [6], evitando a perda da melhor solução. Os indivíduos selecionados podem sofrer alterações através da aplicação dos operadores genéticos *crossover* e mutação, gerando descendentes para a nova geração. Esse processo é repetido até que o critério de parada seja satisfeito.

4.1.1. Implementação Paralela

Um dos principais aspectos do algoritmo genético é sua capacidade de paralelização, pois na realidade, a evolução natural acontece com toda uma população e não somente com um único individuo, reafirmando a existência da natureza paralela nesse processo [4].

O algoritmo genético paralelo (AGP) consiste na distribuição das tarefas de um algoritmo genético simples por diferentes processadores, sendo que a idéia básica é que cada processador possa criar novos indivíduos e calcular suas aptidões em paralelo.

Um dos fatores para melhorar a eficiência na busca da solução ótima em um AG é determinado pelo tamanho da população, quanto maior a população mais memória será necessária para armazenar os indivíduos, implicando no aumento do tempo de execução para que o algoritmo convirja.

O mesmo problema pode ser implementado em diferentes ambientes de hardware, software e estratégia de paralelismo. Da mesma maneira os algoritmos genéticos paralelos possuem diversas formas de implementação que depende de alguns elementos:

- método de avaliação da aptidão e da aplicação do operador de mutação;
- utilização de uma única população ou várias sub-populações;
- no caso de várias sub-populações, qual a regra utilizada para a troca de indivíduos entre elas;
- estratégia da aplicação do operador de seleção: global ou local;

Com base nas condições anteriores pode-se dividir o AGP em três classes distintas: global, granularidade fina, e a classe que foi utilizada nesse trabalho, granularidade grossa, que compreendem os modelos Ilha e Stepping Stone:

Modelo ilha

Neste modelo a população é dividida em pequenas sub-populações, que são distribuídas entre os processadores, simulando a natureza. Seguindo a mesma idéia, alguns indivíduos podem migrar para qualquer outra sub-população. A migração é permitida entre todas as sub-populações: cada processador se comunica com os

demais processadores do sistema.

Modelo *Stepping Stone*

Aqui a população também é particionada, como no modelo ilha, mas a migração fica restrita somente às sub-populações vizinhas. Uma estratégia para implementar essa idéia é visualizar o conjunto de processadores com na forma de um anel, permitindo que cada processador comunique somente com os dois processadores vizinhos imediatos ao anel.

5. Resultados Preliminares

Para resolver o problema inverso em condução de calor com algoritmo genético paralelo, foi utilzado como condição inicial de temperatura a seguinte função:

$$f(x) = \begin{cases} 2x, & 0 \leq \text{x} < 0.5; \\ 2(i-x), & 0.5 \leq \text{x} < 1; \end{cases} \qquad (11)$$

Foram feitas simulações com dois diferentes tamanhos de população: 336 e 1008 indivíduos. As tabelas 1, 2, 3 e 4 mostram os tempos total, o tempo gasto com a rotina "otimiza" (algoritmo genético), *Speedup* e eficiência para os modelos Ilha e Stepping Stone. Pode-se verificar que a "otimiza" ocupa a maior parte do tempo total da simulação, justificando sua paralelização.

Tabela 1: Modelo Ilha - versão 1 - 336 indivíduos

N.Proc.	T.Otimiza	T.Total	S	E
1	670	670	1.00	1.00
2	338	338	1.98	0.99
4	173	174	3.86	0.96
6	116	117	5.74	0.96
8	93	94	7.11	0.89

Tabela 2: Modelo Stepping Stone - 336 indivíduos

N.Proc.	T.Otimiza	T.Total	S	E
1	668	668	1.00	1.00
2	350	351	1.90	0.95
4	173	174	3.84	0.96
6	117	118	5.66	0.94
8	89	89	7.50	0.94

Nas figuras 1 e 2 são mostrados as curvas de*speedup*, e nas figuras 3 e 4 são mostradas as soluções do problema de estimação da condição inicial obtida com as estratégias Ilha e Stepping Stone.

No decorrer do experimento verificou-se que o modelo Ilha poderia ser implementado de forma diferente. Na primeira versão, foi implementado de forma centralizada [2], onde o processador mestre gerencia a comunicação e a escolha do melhor indivíduo, a segunda versão seguiu

Tabela 3: Modelo Ilha- versão 1 - 1008 indivíduos

N.Proc.	T.Otimiza	T.Total	S	E
1	2808	2809	1.00	1.00
2	1449	1449	1.94	0.97
4	733	734	3.83	0.96
6	487	488	5.76	0.96
8	376	376	7.47	0.93

Tabela 4: Modelo Stepping Stone - 1008 indivíduos

N.Proc.	T.Otimiza	T.Total	S	E
1	2791	2791	1.00	1.00
2	1480	1480	1.89	0.94
4	744	745	3.75	0.94
6	489	489	5.70	0.95
8	375	378	7.39	0.92

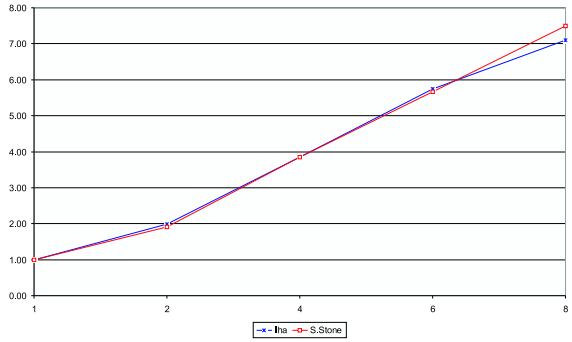

Figura 1: Speedup dos dois Modelos: Ilha e Stepping Stone, para 336 indivíduos

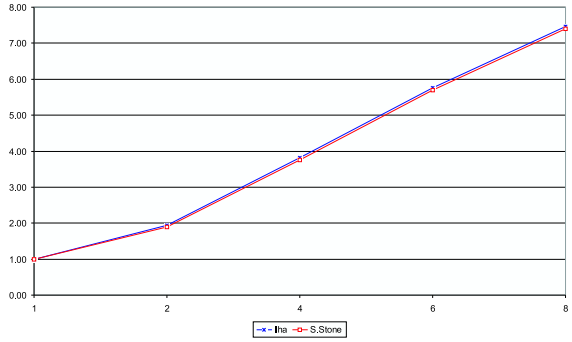

Figura 2: Speedup dos dois Modelos: Ilha e Stepping Stone, para 1008 indivíduos

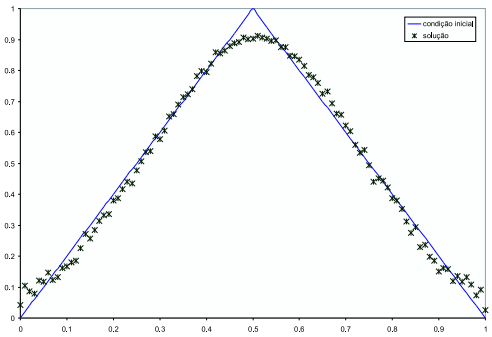

Figura 3: Solução Modelo: Stepping Stone, 1008 indivíduos - 4 processadores

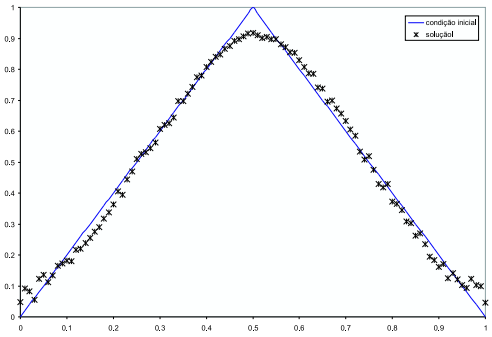

Figura 4: Solução Modelo: Ilha, 1008 indivíduos - 4 processadores

literalmenteo modelo ilha, ou seja, descentralizado, onde cada processador envia para todos os outros o seu melhor indivíduo. Além da estratégia de implementação, outra mudança foram as diretivas utilizadas para comunicação, na primeira versão utilizou-se *MPI_SEND* e *MPI_RECV* e na segunda apenas *MPI_BCAST*. As tabelas 5 e 6 apresentam os resultados para as duas versões do modelo Ilha:

Tabela 5: Modelo Ilha(versão 1 e 2)-336 indivíduos

	N.Proc-1		N.Proc-8	
	v.1	v.2	v.1	v.2
T.Otimiza	670	665	93	89
T.Total	670	665	94	90
S	1.00	1.00	7.11	7.47
E	1.00	1.00	0.89	0.93

Tabela 6: Modelo Ilha(versão 1 e 2)-1008 indivíduos

	N.Proc-1		N.Proc-8	
	v.1	v.2	v.1	v.2
T.Otimiza	2808	2895	376	369
T.Total	2809	2895	376	370
S	1.00	1.00	7.11	7.83
E	1.00	1.00	0.89	0.98

6. Conclusão

Analisando as curvas de *Speedup* dos dois modelos, observa-se pouca diferença entre elas com comportamento próximo do linear, tanto na simulação com 336 quanto na com 1008 indivíduos. Pode-se verificar que a rotina “otimiza“ ocupa a maior parte do tempo, justificando sua paralelização.

Com os resultados obtidos para as duas versões do modelo ilha, foi possível avaliar a influencia no resultado final percebendo-se que houve um pequeno ganho na segunda versão.

Agradecimentos

Os autores agradecem ao Dr. Stephan Stephany (LAC-INPE) por sua ajuda e sugestões e a Leonardo D. Chiwiacowsky (CAP-INPE) por ter disponibilizado seu código computacional.

Referências

[1] L.D. Chiwiacowsky and H. F. de Campos Velho. Different approaches for the solution of a backward heat conduction problem. *Inverse problem in engineering*, 2002.

[2] L.D. Chiwiacowsky, H.F. de Campos Velho, A.J. Preto, and S. Stephany. *Identifying Initial Condition in Heat*

Conduction Transfer by a Genetic Algorithm: A Parallel Approach, volume CD-Rom: trabalho codigo cil261-37. XXIV Iberian Latin-american Congress on Computational Methods in Engineering (CILAMCE-2003), Ouro Preto (MG), Brasil, Outubro 2003.

[3] Kevin Dowd and Charles Severance. *High Performance Computing*. O'Reilly, Sebastopol, 2 edition, July 1998.

[4] Xavier Hüe. Genetic algorithms for optimization. Technical report, The University of Edinburgh, February 1997.

[5] J. H. Holland. *Adaptation in natural and artificial system*. MIT Press, 1975.

[6] E. G. M. Lacerda and A. C. P. L. F. de Carvalho. Introdução aos algoritmos genéticos. *XIX Congresso Nacional da Sociedade Brasileira de Computação*, II:51–126, julho 1999.

[7] W. B. Muniz, F. M. Ramos, and H. F. de Campos Velho. Entropy and a tikhonov regularization techniques applied to the backwards heat equation. *Computers and mathematics with application*, 40(8/9):1071–1084, 2000.

[8] W. B. Muniz, F. M. Ramos, and H. F. Campos Velho. A comparison of some methods for estimating the initial condition of the heat equation. *Journal of Computational and Applied Mathematics*, 1(103):153–171, 1999.

[9] Tarantola. *Inverse Problem Theory*. Elsevier, Amsterdam, 1987.

[10] A. V Tikhonov and V. Y Arsenin. *Solutions of Ill-posed Problems*. Winston and Sons, New York, 1977.

[11] M. N. Özisik. *Heat conduction*. Wiley Interscience, USA, 1980.