Algoritmos Genéticos em Ambientes Paralelos

Michele Alves de Freitas Batista
Instituto Nacional de Pesquisas Espaciais
michele.afreitas@gmail.com

Lamartine Nogueira Frutuoso Guimarães
Instituto Nacional de Pesquisas Espaciais
guimarae@ieav.cta.br

Resumo

Este trabalho tem por objetivo estudar os conceitos de processamento paralelo e a técnica de Message Passing (Troca de Mensagens) aplicado a um algoritmo genético híbrido para encontrar estados estacionários de sistemas dinâmicos. O desenvolvimento de uma ferramenta computacional em ambiente paralelo visa aumentar o desempenho do processamento de cálculos paralelos realizados em múltiplos processadores.

***Palavras-chave**: algoritmos genéticos, MPI, paralelismo, sistemas dinâmicos.*

1. Introdução

Os Algoritmos Genéticos representam, atualmente, uma poderosa ferramenta para busca de soluções de problemas complexos de otimização. Essa foi a solução encontrada na dissertação de Felipe Medeiros "Algoritmo genético híbrido como um método de busca de estados estacionários de sistemas dinâmicos", em que um algoritmo genético é utilizado para encontrar estados estacionários de sistemas.

Assim, um sistema dinâmico representado através de sistemas de equações não-lineares, permite a busca do novo estado estacionário, pela evolução da equação no tempo até que todas as variáveis de estado da solução apresentem um comportamento estacionário. Essa maneira é custosa do ponto de vista computacional.

Na dissertação de Medeiros este método foi utilizado aplicado a modelos de geradores de vapor. No caso estes modelos possuíam respectivamente três, cinco, nove e dezesseis variáveis de estado. Naquele trabalho ficou claro que um grande benefício poderia ser acrescentado à ferramenta caso esta fosse desenvolvida em um ambiente de computação de alto desempenho (para cálculos paralelos realizados em múltiplos processadores).

O processamento paralelo aumenta o desempenho da ferramenta computacional, reduzindo o tempo de processamento, através da decomposição de uma grande tarefa em tarefas menores. Um ambiente paralelo é definido por vários processsadores interligados em rede e modelos de programação paralela.

Para este desenvolvimento se irá utilizar o sistema de computadores Beowulf de processamento paralelo de alto desempenho. Este sistema de computadores faz parte do Laboratório de Engenharia Virtual (LEV), das Divisões de Energia Nuclear (ENU) e Física Aplicada (EFA) do Instituto de Estudos Avançados (IEAv) do Centro Técnico Aeroespacial (CTA) localizado em São José dos Campos, SP.

O modelo de programação paralela selecionado foi o Message Passing e a biblioteca Message Passing Interface (MPI).

Na seção 2 é apresentada a definição de Algoritmos Genéticos (AGs) e explicitado como é feita implementação, mostrando as principais etapas do funcionamento, resssaltando os conceitos dos operadores genéticos crossover e mutação.

Na seção 3 é feita uma apresentação dos sistemas dinâmicos mostrando sua representação através de sistemas de equações diferenciais ordinárias não-lineares.

Na seção 4 são descritos os conceitos de paralelismo e a técnica Message Passing.

Na seção 5 é detalhada a metodologia que será aplicada para a realização desse trabalho.

A seção 6 apresenta as referências bibliográficas e os sites relacionados às principais áreas de interesse.

2. Algoritmos genéticos (AGs)

Os Algoritmos Genéticos formam a parte da área dos Sistemas Inspirados na Natureza; simulando os processos naturais e aplicando-os à solução de problemas reais. São métodos generalizados de busca e otimização que simulam os processos naturais de evolução, aplicando a idéia darwiniana de seleção.

De acordo com a aptidão e a combinação com outros operadores genéticos, são produzidos métodos de grande robustez e aplicabilidade. Estes algoritmos estão baseados nos processos genéticos dos organismos biológicos, codificando uma possível solução a um problema de "cromossomo" composto por cadeia de bits e caracteres. Estes cromossomos representam indivíduos que são levados ao longo de várias gerações, na forma similar aos problemas naturais, evoluindo de acordo com os princípios de seleção natural e sobrevivência dos mais aptos, descritos pela primeira vez por Charles Darwin em seu livro "Origem das Espécies".

Emulando estes processos, os Algoritmos Genéticos são capazes de "evoluir" soluções de problemas do mundo real.

2.1. Implementação de um algoritmo genético

A implementação de um algoritmo genético começa com uma população aleatória de cromossomos. Essas estruturas são, então, avaliadas e associadas a uma probabilidade de reprodução de tal forma que as maiores probabilidades são associadas aos cromossomos que representam uma melhor solução para o problema de otimização do que àqueles que representam uma solução pior.

A função-objetivo de um problema de otimização é construída a partir dos parâmetros envolvidos no problema. Ela fornece uma medida da proximidade da solução em relação a um conjunto de parâmetros. Os parâmetros podem ser conflitantes, ou seja, quando um aumenta o outro diminui. O objetivo é encontrar o ponto ótimo.

A função objetivo permite o cálculo da aptidão bruta de cada indivíduo, que fornecerá o valor a ser usado para o cálculo de sua probabilidade de ser selecionado para reprodução.

As etapas do funcionamento de um AG podem ser descritas da seguinte forma:

1ª. Gera-se a população inicial de programas de forma aleatória;

2ª Avaliação dos indivíduos (função de aptidão);

3ª. Com base nos valores de aptidão, programas sofrem ação de operadores genéticos: reprodução, cruzamento, mutação;

4ª. O processo é repetido até que se obtenha indivíduos com aptidão satisfatória ou por um número pré-definido de iterações.

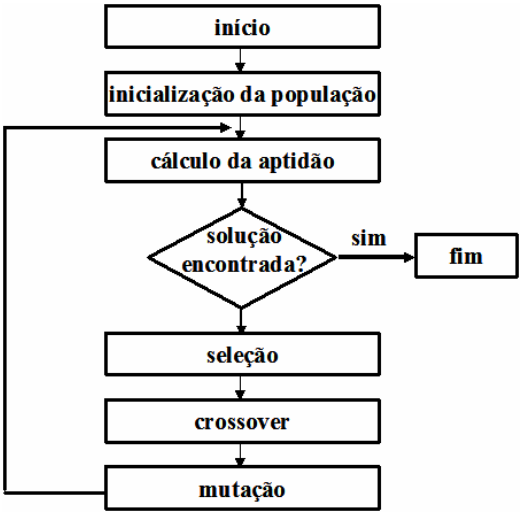

Figura 1. Estrutura de um AG

Para a inicialização, uma população de n indivíduos é gerada aleatoriamente. Cada um dos indivíduos da população representa uma possível solução para o problema, ou seja, um ponto no espaço de soluções.

Geralmente a aptidão do indivíduo é determinada através do cálculo da função objetivo, que depende das especificações de projeto. Ainda nesta fase os indivíduos são ordenados conforme a sua aptidão.

Na seleção, os indivíduos mais aptos da geração atual são selecionados. Esses indivíduos são utilizados para gerar uma nova população por cruzamento. Cada indivíduo tem uma probabilidade de ser selecionado proporcional à sua aptidão.

O operador de crossover ou recombinação cria novos indivíduos através da combinação de dois ou mais indivíduos. A idéia intuitiva por trás do operador de crossover é a troca de informação entre diferentes soluções candidatas. No algoritmo genético clássico é atribuída uma probabilidade de crossover fixa aos indivíduos da população.

O operador de crossover mais comumente empregado é o crossover de um ponto. Para a aplicação deste operador, são selecionados dois indivíduos (pais) e a partir de seus cromossomos são gerados dois novos indivíduos (filhos). Para gerar os filhos, seleciona-se um mesmo ponto de corte aleatoriamente nos cromossomos dos pais, e os segmentos de cromossomo criados a partir do ponto de corte são trocados.

Considere, por exemplo, dois indivíduos selecionados como pais a partir da população inicial de um algoritmo genético e suponhamos que o ponto de corte escolhido (aleatoriamente) encontra-se entre as posições 4 e 5 dos cromossomos dos pais:

PAI #1:	1	0	1	1	1	0	0	1	0	1
PAI #2:	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**0**

Após o crossover, teremos os seguintes indivíduos-filho:

FILHO #1:	1	0	1	1	1	0	**1**	**1**	**1**	**0**
FILHO #2:	**0**	**1**	**1**	**0**	**1**	**0**	0	1	0	1

O operador de mutação modifica aleatoriamente um ou mais genes de um cromossomo. A probabilidade de ocorrência de mutação em um gene é denominada taxa de mutação. Usualmente, são atribuídos valores pequenos para a taxa de mutação. A idéia intuitiva por trás do operador de mutação é criar uma variabilidade extra na população, mas sem destruir o progresso já obtido com a busca.

Considerando codificação binária, o operador de mutação padrão simplesmente troca o valor de um gene em um cromossomo. Assim, se um gene selecionado para mutação tem valor 1, o seu valor passará a ser 0 após a aplicação da mutação, e vice-versa.

PAI:	1	0	1	1	1	0	**0**	1	0	1
FILHO:	1	0	1	1	1	0	**1**	1	0	1

3. Sistemas Dinâmicos

Em geral, os sistemas dinâmicos podem ser representados através de sistemas de equações diferenciais ordinárias não-lineares escritas da seguinte forma:

$$\frac{dy}{dt} = A \times y - b. \qquad (1)$$

Onde y é o vetor que contém as variáveis de estado, A é a matriz de estado e b é o vetor de entradas. Estados estacionários são necessários para se estudar a evolução dinâmica do sistema dada uma perturbação introduzida pelo vetor b, uma vez que os mesmos são utilizados como condições iniciais do processo de evolução dinâmica. Uma forma, óbvia, de obtenção destes estados é resolver o sistema acima na condição $\frac{dy}{dt} = 0$. Desta forma a equação (1) pode ser resolvida de forma analítica da seguinte maneira

$$A \times y = b. \qquad (2)$$

De tal forma que $y = A^{-1} \times b$. Infelizmente, a forma analítica não é eficiente e na maioria dos casos de interesse a solução numérica é a mais indicada.

Simuladores de sistemas dinâmicos trabalham exatamente com a equação (1). Nestes dispositivos muitas vezes é necessário alterar-se valores de parâmetros dentro da matriz de estado A. Isto significa dizer que um novo estado estacionário deve ser calculado. De outra forma, muitas vezes parâmetros internos da matriz de estados se altera devido ao próprio uso do sistema. É o caso, por exemplo, dos coeficientes de transferência de calor quando se trata de sistemas do tipo geradores de vapor, caldeiras, trocadores de calor, núcleo de reatores, entre outros. A maneira numérica como se encontra o novo estado estacionário é simplesmente deixar (numericamente) a equação (1) evoluir no tempo até o momento em que todas as variáveis de estado da solução apresentem um comportamento estacionário.

4. Processamento Paralelo

O processamento paralelo é usado em problema computacionais grandes e complexos para obter resultados mais rápidos, dividindo-os em tarefas pequenas que serão distribuídas em vários processadores para serem executadas simultaneamente. Esses processadores se comunicam entre si para que haja coordenação (sincronização) na execução das diversas tarefas em paralelo. [1]

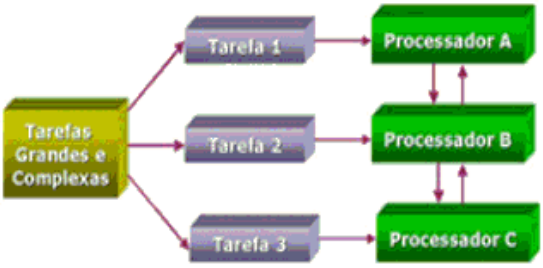

Figura 2. Paralelismo [1]

Como ocorre uma redução no tempo de processamento, o desempenho aumenta.

Em ferramentas não automáticas, o programador é diretamente responsável pelo paralelismo.

Um fator que merece atenção na programação paralela é a sincronização entre os processos. As tarefas executadas em paralelo, num determinado instante, aguardam a finalização mútua para coordenar os resultados ou trocar dados e reiniciar novas tarefas em paralelo.

É necessário que haja a coordenação dos processos e da comunicação entre eles para evitar

que a comunicação seja maior do que o processamento e que consequentemente haja uma queda no desempenho dos processos em execução. [1]

A granularidade é uma medida usada para indicar a relação entre o tamanho de cada tarefa e o tamanho total do programa, ou seja, é a razão entre computação e comunicação. Quanto mais grossa é a granularidade, maior é o tamanho da tarefa dividida. Na granularidade fina, o processamento é menor do que a comunicação e há uma tendência a diminuir o desempenho do programa, enquanto na granularidade grossa, o processamento é maior do que a comunicação e há menor custo no processamento.

4.1. Tipos de paralelismo

4.1.1. Paralelismo de dados

O processador executa as mesmas instruções sobre dados diferentes. É aplicado, por exemplo, em programas que utilizam matrizes imensas e para cálculos de elementos finitos. Os dados são decompostos.

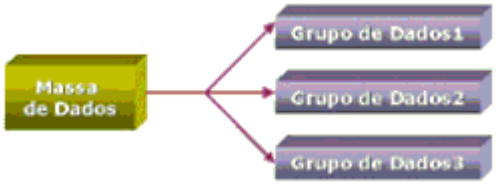

Figura 3. Palelismo de dados [1]

4.1.2. Paralelismo funcional

O processador executa instruções diferentes que podem ou não operar sobre o mesmo conjunto de dados. É aplicado em programas dinâmicos e modulares onde cada tarefa será um programa diferente. O algoritmo é decomposto.

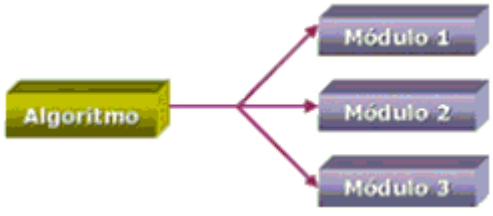

Figura 4. Palelismo funcional [1]

4.1.3. Paralelismo de objetos

Este é o modelo mais recente, que se utiliza do conceito de objetos distribuídos por uma rede (como a Internet), capazes de serem acessados por métodos (funções) em diferentes processadores para uma determinada finalidade.

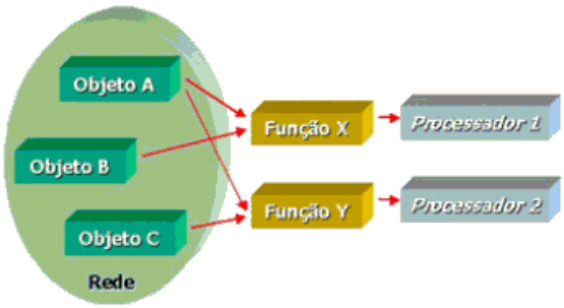

Figura 5. Palelismo de objetos [1]

4.2. Message Passing

O modelo de Troca de Mensagens (Message Passing) é um conjunto de processos que possuem acesso a memória local. As informações são enviadas da memória local do processo para a memória local do processo remoto. A comunicação dos processos é baseada no envio e recebimento de mensagens.

A grande vantagem do Message Passing é que se pode construir um mecanismo de passagem de mensagens para qualquer linguagem, como biblioteca. Outro fator importante, e para este trabalho, fundamental, é a adequação à ambientes distribuídos. Um fator inconveniente é que o programador é diretamente responsável pela paralelização.

As duas principais bibliotecas de Message Passing da atualidade são PVM e MPI.

4.2.1. PVM (Parallel Virtual Machine)

Possui como característica, o conceito de "máquina virtual paralela", dentro da qual processadores recebem e enviam mensagens, com finalidades de obter um processamento global.

4.2.2. MPI (Message Passing Interface)

É um padrão de troca de mensagens, para facilitar o desenvolvimento de aplicações paralelas e de bibliotecas. Tem como características: eficiência, facilidade, portabilidade, transparência, segurança e escalabilidade.

O MPI está se tornando um padrão na indústria. MPI é uma biblioteca de Message Passing desenvolvida para ambientes de memória

distribuída, máquinas paralelas massivas, NOWs (network of workstations) e redes heterogêneas.

MPI define um conjunto de rotinas para facilitar a comunicação (troca de dados e sincronização) entre processos paralelos.

A biblioteca MPI é portável para qualquer arquitetura, tem aproximadamente 125 funções para programação e ferramentas para se analisar a performance.

A biblioteca MPI possui rotinas para programas em Fortran 77 e ANSI C, portanto pode ser usada também para Fortran 90 e C++. Os programas são compilados e linkados a biblioteca MPI.

Todo paralelismo é explícito, ou seja, o programador é responsável por identificar o paralelismo e implementar o algoritmo utilizando chamadas aos comandos da biblioteca MPI. [1]

5. Metodologia

A implementação do algoritmo genético híbrido para resolver o problema de determinação de estados estacionários de sistemas dinâmicos não lineares foi realizada na dissertação intitulada "Algoritmo Genético Híbrido como um método de busca de estados estacionários de sistemas dinâmicos", de Felipe Leonardo Lôbo Medeiros.

O primeiro passo foi a familiarização com o algoritmo genético. Como este foi implementado em na ferramenta C++ (windows), tornou-se necessária a migração do programa para compilação em g++ (linux).

Foi necessário estudar as bibliotecas (stdlib.h, conio.h, dos.h) e a substituição de algumas funções (ex.: random(n), c++ por rand()%n, g++).

No momento estão sendo feitas análises detalhadas das alterações.

A próxima etapa corresponde à programação paralela. Após análise do algoritmo e dos dados obtidos, é necessária a decomposição do algoritmo (paralelismo funcional) e/ou decomposição dos dados (paralelismo de dados).

Através da biblioteca MPI será possível distribuir as tarefas e/ou dados entre vários processadores que trabalharam simultaneamente e coordenar os processos em execução e a comunicação entre os processadores (sincronização).

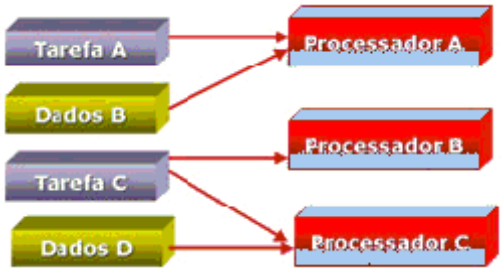

Figura 6. Distribuição de dados e/ou tarefas [1]

6. Referências

[1] Apostila "Curso de MPI" dado pelo CENAPAD/NE.

[2] Man, K. F. & Tang, K. S. & Kwong, S. Genetic Algorithms, Springer, 1999.

[3] Medeiros, Felipe L. L., "Algoritmo Genético Híbrido como um Método de Busca de Estados Estacionários de Sistemas Dinâmicos", Dissertação defendida em 2002, INPE.

[4] Mitchell, M. Introduction to Genetic Algorithms, Mit Press, 1996.

Sites relacionados

Algoritmos genéticos
http://www.din.uem.br/ia/geneticos/
http://laqqa.iqm.unicamp.br/algoritmosgeneticos.htm
http://www.freedom.ind.br/otavio/trab/introducao_algoritmos_geneticos.pdf;
http://www.gta.ufrj.br/~marcio/ genetic.html;
http://www.geocities.com/Athens/Sparta/1350/ia/ag_aplica.html;
http://eotd12dsl.no.sapo.pt/ai.htm
http://www.geocities.com/igoryepes/ visualizar.htm
http://laseeb.isr.ist.utl.pt/publications/tfc/sisapr/pg.html
www.inf.ufpr.br/~aurora/disciplinas/ PGII.ppt
www.gene-expression-programming.com/webpapers/GEPPort.pdf
http://www.sbc.org.br/reic/edicoes/2003e2/cientificos/Chameleon_uma_ferramenta_de_programacao_genetica_orientada_a_gramaticas.pdf
http://algos.inesc-id.pt/tfcs/2004/ node18.html
http://www.geocities.com/igoryepes/visualizar2.htm

Sistemas dinâmicos
http://www.niee.ufrgs.br/SBC2000/eventos/wie/wie040.pdf

Paralelismo
http://www.cbpf.br/cat/sun/SUNHPC.html
http://www.ime.usp.br/~song/mac5705-03.html

http://atlas.ucpel.tche.br/~barbosa/sist_dist/mpi/mpi.html
http://www.inf.pucminas.br/professores/cota/papers/wscad2004b.pdf
http://www.cesup.ufrgs.br/FAQ/ProcessamentoParalelo.html
http://www.unesp.br/prope/projtecn/Educa/Educa34b.htm
http://www.ime.usp.br/~song/papers/jai01.pdf
http://twiki.im.ufba.br/bin/view/MAT151/TrabalhoMvp