2005 International Nuclear Atlantic Conference - INAC 2005
Santos, SP, Brazil, August 28 to September 2, 2005
ASSOCIAÇÃO BRASILEIRA DE ENERGIA NUCLEAR - ABEN
ISBN: 85-99141-01-5

DESENVOLVIMENTO DE UM ALGORITMO GENÉTICO PARALELO UTILIZANDO MPI E SUA APLICAÇÃO NA OTIMIZAÇÃO DE UM PROJETO NEUTRÔNICO

Marcel Waintraub[1], Rafael P. Baptista[2], Claudio M. N. A. Pereira[3]
[1] Instituto de Engenharia Nuclear (IEN / CNEN)
Caixa Postal 68550 - Ilha do Fundão, s/n - CEP 21945-970 - Rio de Janeiro, RJ
Tel.: + 55 21 22098180 - email: marcel@ien.gov.br

[2] Universidade Federal do Rio de janeiro (PEN / COPPE)
Caixa Postal 68509 - Ilha do Fundão, s/n
CEP 21945-970 - Rio de Janeiro, RJ
email - rafael@ien.gov.br

[3] Instituto de Engenharia Nuclear (IEN / CNEN)
Caixa Postal 68550 - Ilha do Fundão, s/n
CEP 21945-970 - Rio de Janeiro, RJ
email - cmnap@ien.gov.br

RESUMO

Este trabalho apresenta o desenvolvimento de um algoritmo genético paralelo [1] distribuído aplicado a otimização de um projeto neutrônico. Para a implementação do paralelismo, utilizou-se a biblioteca "Message Passing Interface" (MPI) [2], padrão para computação paralela em ambientes de memória distribuída com intercambiamento de mensagens. Outra característica importante do MPI é sua portabilidade para qualquer arquitetura.

Como principais objetivos deste artigo têm-se: i) validação dos resultados obtidos pela aplicação deste algoritmo na otimização de um projeto neutrônico, através de comparações com resultados apresentados na literatura. [3][4] e ii) teste de performance do cluster do Instituto de Engenharia Nuclear (IEN) [5] em problemas de otimização aplicados ao cálculo de física de reatores.

Os experimentos demonstraram que o algoritmo genético paralelo utilizando a biblioteca MPI, implantado no cluster do IEN, apresentou ganhos significativos nos resultados obtidos, bem como no tempo de processamento. Tais resultados ratificam a utilização do cluster do IEN com algoritmos genéticos paralelos para resolução de problemas de otimização de projetos neutrônicos.

Palavras-chave: Algoritmos genéticos; Processamento distribuído; Cluster; Projeto de Reatores.

1. INTRODUÇÃO

Os algoritmos genéticos (AGs) constituem uma poderosa técnica de otimização devido a sua robustez e fácil customização. Conseqüentemente, eles vêm sendo aplicados com sucesso na busca de soluções aceitáveis para diversos problemas de engenharia, em especial, problemas complexos como aqueles encontrados na engenharia nuclear, tais como: recarga de combustível [6]; política de manutenção [7]; projeto de sistemas de diagnósticos [8]; projeto neutrônico de reatores [4]; dentre outros.

Embora sejam capazes de encontrar boas soluções, para certos problemas os AGs podem necessitar que a função de avaliação seja calculada milhares de vezes. Dependendo do custo computacional de cada avaliação, a obtenção de uma solução aceitável pode tornar-se inviável. Felizmente os AGs trabalham com uma população de soluções independentes, tornando possível distribuir a carga computacional por vários processadores. Apesar de parecer trivial, a paralelização dos AGs esbarra na não-linearidade verificada no seu mecanismo, visto que sua eficiência e qualidade das soluções dependem de vários parâmetros.

Para superar esta dificuldade, adotou-se o modelo de malha grossa para o AGP – Algoritmo Genético com Ilhas (AGI), que comparado ao Algoritmo Genético Simples (AGS), demonstrou ser mais robusto, aumentando assim as chances de se obter melhores resultados. Além disso, o AGI é o modelo mais indicado para aplicações em redes convencionais de computadores, evitando o uso de computadores com arquiteturas paralelas de custo elevado.

A computação paralela em sistemas distribuídos é realizada através da passagem de mensagens que será descrita no capítulo 2. Neste capítulo será descrito o ambiente de passagem de mensagens com plataforma portátil denominado MPI, utilizado neste trabalho.

Neste trabalho focaremos a validação dos resultados obtidos pela aplicação deste AGI na otimização de um projeto neutrônico, através de comparações com resultados anteriores apresentados por Pereira *et al.* e testaremos a performance do *cluster* instalado no Laboratório de Computação Paralela do IEN em problemas de otimização aplicados ao cálculo de física de reatores. Os resultados, apresentados no capítulo 3, mostram a evidente superioridade do AGI sobre o AGS, não somente em relação ao tempo global de otimização, mas também na qualidade das soluções encontradas.

2. O ALGORITMO GENÉTICO PARALELO COM MPI

2.1. Algoritmo Genético Paralelo

Nos últimos anos, o surgimento de computadores mais velozes e com maior capacidade de memória tem propiciado com que várias técnicas computacionais de *software* possam sair de um escopo puramente acadêmico para o mundo real, proporcionando aplicações práticas de

grande valor, nas diversas áreas do conhecimento. A exemplo disto, pode-se citar a Computação Evolucionária, uma poderosa técnica de otimização, aplicável a problemas de difícil solução através de técnicas tradicionais, mas que, em contrapartida, demandam um alto custo computacional.

Hoje em dia já são observadas várias aplicações práticas bem sucedidas de AGs em diversos setores da engenharia. Na Engenharia Nuclear tal fato é também observado [4-8]. No entanto, existem aplicações (dentre elas pode-se citar aquelas relacionados com o projeto e a operação de reatores nucleares) que necessitam da execução de simulações computacionais, cujo custo, por si só, já é relativamente elevado. Nestes casos, como os AGs trabalham com populações de candidatos a soluções, eles esbarram no limite da capacidade de processamento dos computadores geralmente disponíveis (não paralelos).

Uma forma de sobrepujar tal dificuldade está na utilização de arquiteturas computacionais paralelas. No entanto, computadores especialmente projetados para processamento paralelo possuem um custo muito elevado, impondo, assim, uma nova restrição à sua utilização. Uma alternativa mais acessível é a utilização de *clusters* de computadores pessoais (PCs) [11], ou, simplesmente usufruir da cooperação entre os computadores de uma rede comum (neste caso, nem toda implementação paralela torna-se eficiente).

Os algoritmos genéticos paralelos (AGPs) podem ser catergorizados em 3 diferentes aproximações básicas: Mestre-Escravo, Celulares (malha fina), e Ilhas (malha grossa ou distribuídos). O primeiro modelo é a versão paralela do AGS (canônico), conseqüentemente, ele não altera nem restringe os operadores genéticos. Neste modelo, somente a avaliação é distribuída pelas máquinas disponíveis. O controle das gerações, seleção e os operadores genéticos não são paralelizados. A figura 1(a) ilustra esta abordagem.

No AG celular cada indivíduo da população é colocado em um processador (célula). As células são arranjadas geograficamente (a grade 2-D é a maneira mais comum), como pode ser visto na figura 1(b), e o cruzamento fica restrito a vizinhança. Esta abordagem demonstrou proporcionar mais diversidade do que o AGS, resultando numa melhor exploração do espaço de busca.

O modelo de ilhas é uma abordagem multi-populacional. Cada "sub-população" é, então, localizada em um processador (ilha) que tem o seu próprio processo evolutivo independente. A fim de promover cooperação entre as ilhas, um novo operador, chamado migração é criado. De acordo com alguma estratégia predefinida, indivíduos migram de uma ilha para outra. São possíveis várias topologias para o AGI. Nestre trabalho utilizamos a topologia em anel, ilustrada na figura 1(c). Cada ilha inicia o processo de busca a partir de sua própria população inicial e, durante algumas gerações, explora alguma parte do espaço de busca. As migrações ocorrem, informações sobre diferentes regiões do espaço de busca são trocadas entre as ilhas, promovendo maior diversidade na pesquisa.

Dentre os modelos existentes de AGPs, o Modelo de Ilhas tem demonstrado considerável eficiência não só devido aos ganhos referentes a tempo de processamento (no caso de múltiplos processadores), bem como na melhoria de resultados em processos de otimização

complexos. Além disso, este modelo é o mais propício à execução distribuída, adaptando-se bem à redes comuns de computadores.

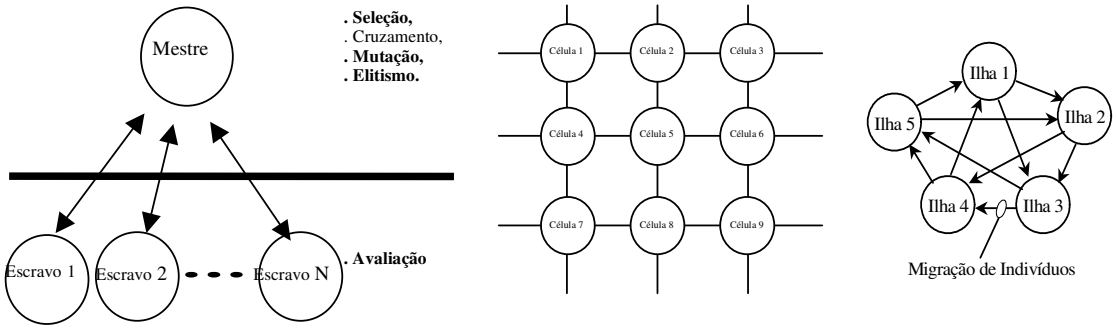

Figura 1 – (a) Modelo Mestre-Escravo, (b) Modelo Celular e (c) Modelo de Ilhas.

2.2. Interface de Troca de Mensagens

Um ambiente de programação via troca de mensagens é formado por: uma linguagem seqüencial (como C e Fortran), que será utilizada na implementação dos programas seqüenciais nos processadores; e a biblioteca de troca de mensagens, a qual fornece as ferramentas necessárias para a ativação e cooperação entre os processos paralelos.

Os ambientes de passagem de mensagens foram desenvolvidos inicialmente para máquinas com processamento maciçamente paralelo (Massively Parallel Processing - MPP), onde, devido a ausência de um padrão, cada fabricante desenvolveu seu próprio ambiente, sem se preocupar com a portabilidade[1] do software gerado. Atualmente, os ambientes de passagem de mensagens foram remodelados considerando-se três grandes objetivos: utilizar o potencial dos sistemas distribuídos no desenvolvimento de aplicações paralelas; permitir a união de plataformas heterogêneas (MPP e/ou redes de estações de trabalho); e permitir a portabilidade das aplicações paralelas desenvolvidas.

Baseados nesses objetivos, vários grupos de pesquisa desenvolveram ambientes de passagem de mensagem independentes da máquina a ser utilizada. Esses ambientes foram chamados de ambientes de passagem de mensagem com plataforma portátil (ou "plataforma de portabilidade") e puderam ser implementados em várias plataformas (*hardware* + sistemas operacionais). Dois exemplos mais importantes desses ambientes portáteis para equipamentos heterogêneos são : PVM (Parallel Virtual Machine) e MPI.

[1] É a possibilidade de executar ou modificar um programa para que possa ser executado em mais de um sistema de computador, ou sob mais de um sistema operacional. Os *softwares* de alta portabilidade podem ser transferidos para outros sistemas sem grandes esforços.

PVM é um conjunto integrado de bibliotecas e ferramentas de software, cuja finalidade é emular um sistema computacional concorrente heterogêneo, flexível de propósito geral. Como seu próprio nome sugere, permite que um conjunto de plataformas paralelas heterogêneas conectadas em rede seja utilizado como uma arquitetura paralela de memória distribuída, formando uma máquina paralela virtual.

O PVM fornece as funções que permitem ao usuário iniciar, comunicar e sincronizar tarefas na máquina virtual. Uma tarefa é definida como uma unidade computacional em PVM análoga aos processos UNIX (freqüentemente é um processo UNIX). O modelo computacional do PVM é, portanto, baseado na noção de que uma aplicação consiste em várias tarefas. Cada tarefa é responsável por uma parte da carga de trabalho da aplicação.

O MPI é uma tentativa de padronização, independente da plataforma paralela, para ambientes de programação via troca de mensagens. Surgiu da necessidade de se resolver alguns problemas relacionados às plataformas de portabilidade, tais como: restrições em relação à real portabilidade de programas, devido as grande número de plataformas; e o mau aproveitamento de características de algumas arquiteturas paralelas.

Uma série de motivos justificam a necessidade de um padrão para esse tipo de sistema, tais como: portabilidade e facilidade de uso (a medida que a utilização do MPI aumentar, será possível portar transparentemente aplicações entre um grande número de plataformas paralelas); fornecer uma especificação precisa (fabricantes de *hardware* podem implementar eficientemente em suas máquinas um conjunto bem definido de rotinas); e crescimento da indústria de *software* paralelo (a existência de um padrão torna a criação de *software* paralelo por empresas uma opção comercialmente viável, o que também implica em maior difusão do uso de computadores paralelos).

É importante ressaltar que, enquanto o PVM é uma implementação, o MPI é apenas um especificação sintática e semântica de rotinas constituintes da biblioteca de comunicação. Sendo assim, uma análise mais completa do MPI deve ser feita levando em consideração a implementação a ser utilizada. A composição do MPI é definida por um conjunto de 129 rotinas, que oferecem os seguintes serviços: comunicação ponto-a-ponto; comunicação coletiva; suporte para grupo de processos; suporte para contextos de comunicação; e suporte para topologia de processos.

Analisando cada um dos tópicos anteriores, tem-se:

1 - Comunicação Ponto-a-Ponto e Coletiva: são responsáveis pela ação básica de uma biblioteca de troca de mensagens, que se trata da transferência de uma mensagem. Cada transferência ponto-a-ponto envolve somente dois processos, um transmissor e um receptor.

Rotina *Send( )*: Responsável pela transmissão de uma mensagem. É definida por:

send(msg_id, msg, tamanho, destino)

Quando ocorre uma operação de transferência de mensagens, os dados que compõem a mensagem são geralmente copiados em um *buffer*, de onde são transmitidos quando

possível. Após a cópia para o *buffer*, as posições de memória (variáveis) onde os dados transmitidos estavam armazenados podem ser reutilizadas sem problemas
A rotina *Send()* possui diversas variações semânticas, são elas:

- *Send( )* síncrono: A rotina só se completa com o recebimento de uma confirmação de que a mensagem foi recebida. Caso contrário, tem-se uma rotina *Send ( )* assíncrona.

- *Send( )* bloqueante: O processo transmissor é bloqueado até que se copie a mensagem para o *buffer* de transmissão.

- *Send( )* não bloqueante: O processo transmissor não espera a cópia para o *buffer*. Então, antes de se reutilizar a variável que está sendo enviada, deve-se testar se a mensagem já foi copiada para o *buffer*.

Rotina *Receive( )*: Responsável pela recepção de uma mensagem. É definida por:

receive(msg_id, msg, tamanho, fonte)

Seguindo as definições do Fórum MPI, pode-se ter:

- *Receive( )* bloqueante: A rotina só se completa se a mensagem ativada for recebida.

- *Receive( )* não bloqueante: Similar a rotina *Send( )* não bloqueante, a mensagem é recebida sem o bloqueio da rotina, de maneira que deve-se testar o término da operação antes de sua utilização.

2 - Suporte para Grupos de Processos: O MPI relaciona os processos em grupos, e esse processos são identificados pela sua classificação dentro desse grupo. Por exemplo, suponha um grupo contendo *n* processos: estes processos serão identificados utilizando-se números entre 0 e *n*-1. Essa classificação dentro do grupo é denominada *rank*. O MPI apresenta primitivas de criação e destruição de grupos de processos. Então, um processo no MPI é identificado por um grupo e por um *rank* dentro deste grupo.

3 - Suporte para Contextos de Comunicação: São escopos que relacionam um determinado grupo de processos. São implementados para garantir que não existam mensagens que sejam recebidas ambiguamente por grupos de processos não relacionados. Assim, um grupo de processos ligados por um contexto não consegue se comunicar com um grupo que esteja definido em outro contexto. Este tipo de estrutura não é visível nem controlável pelo usuário, e o seu gerenciamento fica a cargo do sistema. Para criação de contextos, o MPI se utiliza do conceito de *communicator*, que é um objeto manuseado pelo programador e relaciona um grupo (ou grupos) de processos com um determinado contexto. Se existem, por exemplo, aplicações paralelas distintas executando em um mesmo ambiente, para cada uma delas será criado um *communicator*. Isso criará contextos distintos que relacionarão os grupos de processos de cada aplicação e evitará que estes interfiram entre si.

4 - Suporte para Topologias: O MPI fornece primitivas que permitem ao programador definir a estrutura topológica com a qual os processos de um determinado grupo se relacionarão. Como exemplo de uma topologia, pode-se citar uma malha, onde cada ponto de interseção na malha corresponde a um processo.

2.3. Modelo de Ilhas aplicado a um Algoritmo Genético Paralelo via MPI

A utilização de *clusters* de estações de trabalho ou microcomputadores (PCs) vem recebendo grande aceitação recentemente devido ao custo relativamente baixo, à facilidade de atualizações, à utilização de *hardware* e *software* "abertos" e à independência de fornecedores e licenças de importação.

O Laboratório de Computação Paralela do IEN segue a filosofia dos sistemas de computação paralela e distribuída do tipo *Beowulf* [6]. As principais características dos *clusters* de computadores do tipo *Beowulf* são a utilização de sistema operacional *freeware*, como Linux ou FreeBSD, e o emprego de rede de comunicação dedicada exclusivamente ao sistema. Em geral, a comunicação do sistema computacional com o mundo exterior é feita através de um único nó (*Front End)*. O *cluster* do IEN possui 16 PCs, com processadores Pentium 4 de 3.0 Ghz com tecnologia HT, 16 Gb RAM. O desempenho de pico (*peak performance*) excede 50 Gflop/s (1 Gflop/s equivale a 10^9 operações de ponto flutuante por segundo). Os computadores são conectados em topologia estrela através de uma rede Gigabit Ethernet de 1000 Mb/s. O sistema operacional utilizado é o Linux.

Com relação a quantidade de comunicação requerida por cada um dos modelos apresentados no item 2.1, observa-se que o AGI é o que requer menos (muito menos) troca de informações entre os processadores. Portanto, este modelo é o mais indicado para ser implantado em sistemas distribuídos. Como a plataforma computacional disponível é uma rede local convencional, escolhemos o AGI.

3. SIMULAÇÃO COMPUTACIONAL

3.1. O Problema

Como o principal objetivo deste trabalho é demonstrar a potencialidade da ferramenta computacional disponível no Laboratório de Computação Paralela do IEN, utilizaremos os resultados obtidos num trabalho anterior [3], comparando os modelos AGS e AGI com relação aos tempos de processamento e soluções encontradas. Brevemente, o problema abordado consiste em minimizar o fator de pico médio f_p, de um reator tipo PWR cilíndrico com 3 zonas de enriquecimento, considerando que este deve permanecer crítico ($k_{eff} = 1.0 \pm 1\%$) e submoderado, provendo um dados fluxo médio ϕ_0. Então, o problema de otimização pode ser escrito como:

$$f_p\,(R_f,\ \Delta_c,\ R_e,\ E_1,\ E_2,\ E_3,\ M_f,\ M_c)$$

Sujeito as seguintes restrições:

$$\phi(R_f,\ \Delta_c,\ R_e,\ E_1,\ E_2,\ E_3,\ M_f,\ M_c) = \phi_0; \qquad (1)$$

$$0.99 \leq k_{eff}(R_f, \Delta_c, R_e, E_1, E_2, E_3, M_f, M_c) \leq 1.01; \quad (2)$$

$$\frac{dk_{eff}}{dV_m} > 0\,; \quad (3)$$

$$R_{f\,min} \leq R_f \leq R_{f\,max}; \quad (4)$$

$$\Delta_{c\,min} \leq \Delta_c \leq \Delta_{c\,max}; \quad (5)$$

$$R_{e\,min} \leq R_e \leq R_{e\,max}; \quad (6)$$

$$E_{1\,min} \leq E_1 \leq E_{1\,max}; \quad (7)$$

$$E_{2\,min} \leq E_2 \leq E_{2\,max}; \quad (8)$$

$$E_{3\,min} \leq E_3 \leq E_{3\,max}; \quad (9)$$

$$M_f = \{UO_2, \text{ ou U-metálico}\}; \quad (10)$$

$$M_c = \{\text{Zircaloy-2, Alumínio ou Aço-304}\}; \quad (11)$$

onde V_m é o volume do moderador e os subscritos *min* e *max* referem-se aos limites inferior e superior dos parâmetros, cujas faixas encontram-se na tabela 1.

Tabela 1. Faixas de variação dos parâmetros de projeto

Parâmetro	Símbolo	Faixa
Raio do combustível (pol.)	R_f	0.2-0.5
Espessura do encamisamento (pol.)	Δ_c	0.01-0.1
Espessura do moderador (pol.)	R_e	0.01-0.3
Enriquecimento da zona 1 (%)	E_1	2.0-5.0
Enriquecimento da zona 2 (%)	E_2	2.0-5.0
Enriquecimento da zona 3 (%)	E_3	2.0-5.0
Material do combustível	M_f	UO_2 ou U-metálico
Material do encamisamento	M_c	{Zircaloy-2, Alumínio ou Aço-304}

Para a avaliação da plataforma computacional do IEN, realizamos vários experimentos comparando o AGI com o AGS. Nesta comparação consideramos um número igual de avaliações para os dois modelos, isto é, para um AGS com população de 200 indivíduos, o AGI equivalente terá 4 ilhas contendo uma população de 50 indivíduos cada. Além disso, consideramos um critério de parada de 500 gerações e as ilhas conectadas em anel, com o melhor indivíduo migrando a cada 10 gerações.

Para cada configuração de população, computamos 10 experimentos com diferentes parâmetro genéticos, tais como: cruzamento, taxa de mutação e sementes de geração randômica por ilha.

Na Tabela 2, apresentamos os resultados dos 10 casos, obtidos para os modelos AGS e AGI calculados a partir de uma ferramenta genérica desenvolvida em linguagem *Java*, denominada ProGenIS [12], para arquitetura cliente-servidor através de soquetes, e aqueles obtidos do AGI calculados a partir programa MITIGA com a interface de troca de mensagens MPI (AGIM), que utiliza a biblioteca de classes GACLsLib [13], desenvolvido no IEN em linguagem C^{++}.

Tabela 2. Comparação entre o AGS, o AGI e o AGIM em 500 gerações.

Caso	P=50	P=200			P=400		
	AGS	**AGS**	**AGI**	**AGIM**	**AGS**	**AGI**	**AGIM**
			(4 Ilhas[a])			(8 Ilhas[a])	
1	1,2895	1,3578	1,3322	1,2993	1,2849	1,2993	1,2823
2	1,3596	1,3065	1,2799	1,2992	1,2791	1,2793	1,2782
3	1,4138	1,3057	1,3378	1,3098	1,3169	1,2776	1,2792
4	1,3432	1,3287	1,2835	1,2819	1,3235	1,2996	1,2804
5	1,3662	1,2889	1,2810	1,2804	1,3204	1,2824	1,2801
6	1,2879	1,3020	1,2796	1,2895	1,3102	1,2804	1,2855
7	1,3164	1,3082	1,2784	1,2990	1,3352	1,2788	1,2783
8	1,3404	1,3367	1,3034	1,2846	1,3084	1,2808	1,2782
9	1,3541	1,3214	1,2989	1,2804	1,2993	1,2990	1,2819
10	1,3256	1,3328	1,3170	1,3162	1,3463	1,2839	1,2814
Média	1,3397	1,3189	1,2992	1,2940	1,3124	1,2861	1,2806
Tempo (h)	1,4	6,6	1,4	0,3	14,1	1,4	0,3

Este resultados mostram que além do ganho significativo relativo ao tempo total de processamento para o processo de otimização, as soluções encontradas foram melhores. Isto mostra que a manutenção da diversidade proporcionada pelo modelo de ilhas pode ter refletido na melhoria do resultado da otimização. Observa-se portanto, que o AGI apresenta resultados ligeiramente melhores, que podem ser atribuídos às configurações dos parâmetros genéticos ou polarização da população inicial nos 10 experimentos feitos.

3. CONCLUSÔES

Este trabalho mostra a importância em se ter disponível uma ferramenta computacional como a que se encontra instalada no Laboratório de Computação Paralela do IEN. Tomando como base a utilização da computação evolucionária que para problemas de otimização complexos são grandes demandantes de tempo de processamento, os resultados encontrados demonstram que além do ganho de tempo, houve uma melhora na eficiência do processo de otimização.

AGRADECIMENTOS

Cláudio M. N. A. Pereira e Rafael P. Baptista são patrocinados pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq.

REFERÊNCIAS

1. Cantú-Paz E., "Efficient and Accurate Parallel Genetic Algorithms", *Kluwer Academic Publishers*, ISDN 0792372212, Boston (2000).
2. [W. Gropp, E. Lusk e A. Skjellum, "Using MPI: Portable Parallel Programming with the Message Passing Interface", MIT Press (1994).
3. Pereira C. M. N. A. e Lapa, C. M. F., "Coarse-Grained Parallel Genetic Algorithm Applied to a Nuclear Reactor Core Design Optimization Problem", *Annals of Nuclear Energy*, Vol.30, pp.555-565 (2003).
4. Pereira, C. M. N. A. , Schirru, R. e Martinez, A. S. , "Basic Investigations Related to Genetic Algorithms in Core Designs", *Annals of Nuclear Energy*, Vol.26, n.3, pp.173-193 (1999).
5. Sampaio, P. A. B., Lapa, C. M. F. Jospin, R. J., Pereira, C. M. N. A., Moreira, M. L., Lapa, N. S., Nery, D. S. e Mol A. C. A., "Computação de Alto desempenho no Instituto de Engenharia Nuclear", ENFIR (2000).
6. Chapot J.L.C., Silva F.C., Schirru R., "A New Approach to the Use of Genetic Algorithms to Solve Pressurized Water Reactor´s Fuel Management Optimization Problem", *Annals od Nuclear Energy* Vol. 26(7), pp.641-655 (1999).
7. Lapa C.M.F., Pereira C.M.N.A. e Mol A.C.A., "Maximization of a Nuclear System Availability through Maintenance Scheduling Optimization Using Genetic Algorithm", *Nuclear Engineering and Design*, **Vol.196**, pp. 95-107 (2000).
8. Lapa C.M.F., Pereira C.M.N.A.e Frutuoso e Melo P. F., "An Application of Genetic Algorithms to Surveillance Tests Optimization od a PWR Auxiliary Feed-Water System", *International Journal of Intelligent Systems* **Vol.17 (8)**, pp.813-831 (2002).
9. Alvarenga M.A.B., Martinez A.S., Schirru R., "Adaptive Vector Quantization Optimized by Genetic Algorithms for Real-Time Diagnosis throught Fuzzy Sets", *Nuclear Technology* **Vol.120(3)**, pp.188-197, 1997.
10. Cantú-Paz E. e Goldberg D.E., "Efficient Parallel Genetic Algorithms: Theory and Practice", *Computer Methods in Applied Mechanics and Engineering*, **Vol.186**, pp.221-238 (2000).
11. "The Beowulf Project", http://www.beowulf.org, (2005).
12. Pereira, C. M. N. A. , ProGenIS: Programmable Genetic Island System – Uma Ferramenta Genética para Desenvolvimento de Algoritmos Genéticos Paralelos Utilizando o Modelo de Ilhas: Apresentação do Sistema e Estado de Desenvolvimento Atual, *Relatório Técnico* – IEN-SETER-07/01 (2001).
13. Pereira, C. M. N. A. , GACLSLIB: Genetic Algorithm Class Library – Uma Biblioteca de Classes para Desenvolvimento de Algoritmos Genéticos sobParadigma de Orientação a Objetos, *Comunicação Técnica*, IEN-SETER-01/00 (2000).

.