ALGORITMOS PARALELOS DE OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS EM PROBLEMAS NUCLEARES

Marcel Waintraub

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Nuclear, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Nuclear.

Orientador(es): Roberto Schirru
Cláudio Márcio do Nascimento Abreu Pereira

Rio de Janeiro
Março de 2009

ALGORITMOS PARALELOS DE OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS EM PROBLEMAS NUCLEARES

Marcel Waintraub

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA NUCLEAR.

Aprovada por:

_____

Prof. Roberto Schirru, D.Sc.

_____

Prof. Cláudio Márcio do Nascimento Abreu Pereira, D.Sc.

_____

Prof. Jose Antonio Carlos Canedo Medeiros, D.Sc.

_____

Prof. Eduardo Gomes Dutra do Carmo, D.Sc..

_____

Prof. Antonio Carlos de Abreu Mól, D.Sc.

_____

Prof. Celso Marcelo Franklin Lapa, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2009

Waintraub, Marcel

Algoritmos Paralelos de Otimização por Enxame de Partículas em Problemas Nucleares / Marcel Waintraub. – Rio de Janeiro: UFRJ/COPPE, 2009.

XII, 97 p.: il.; 29,7 cm.

Orientadores: Roberto Schirru

Cláudio Márcio do Nascimento Abreu Pereira.

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia Nuclear, 2009.

Referencias Bibliográficas: p. 88-97.

1. Algoritmo de Otimização por Enxame de Partículas. 2. Projeto de Reatores. 3. Recarga de Reatores Nucleares. 4. Processamento Paralelo. I. Schirru, Roberto *et al*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Nuclear. III. Titulo.

DEDICATÓRIA

À minha mãe e ao meu pai, sem os quais nada seria possível.

À minha esposa Silvia pelo amor dedicado a mim por todos estes anos de convivência.

Aos meus filhos Tatiana, Mariana e Fabio, que tornam a minha vida mais feliz e compensadora.

AGRADECIMENTOS

Ao Prof. Dr. Roberto Schirru, orientador, pelo apoio e confiança durante a realização deste trabalho de pesquisa;

Ao prof. Dr. Cláudio Márcio N. A. Pereira, orientador e amigo, pela sua dedicação, bom senso e boas idéias, que nortearam todo este trabalho de pesquisa;

Ao Dr. Julio Cezar Suita, Diretor do Instituto de Engenharia Nuclear, pelo apoio e disponibilização da infraestrutura necessária para a realização deste trabalho;

Ao Dr. Edison de Oliveira Martins Filho, pela amizade e incentivo que sempre me dedicou;

Aos servidores do IEN, pelo apoio e colaboração durante a realização deste trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ALGORITMOS PARALELOS DE OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS EM PROBLEMAS NUCLEARES

Marcel Waintraub

Março/2009

Orientadores: Roberto Schirru

Cláudio Márcio do Nascimento Abreu Pereira

Programa: Engenharia Nuclear

O Algoritmo de Otimização por Enxame de Partículas (PSO, do inglês *Particle Swarm Optimization*) é uma metaheurística baseada em populações de indivíduos (MBP), na qual os candidatos à solução evoluem através da simulação de um modelo simplificado de adaptação social. Juntando robustez, eficiência e simplicidade, o PSO tem adquirido grande popularidade. São reportadas muitas aplicações bem sucedidas do PSO nas quais este algoritmo demonstrou ter vantagens sobre outras MBPs bem estabelecidas. Entretanto, o custo computacional continua sendo uma grande restrição para o PSO, assim como para todas as outras MBPs, especialmente em problemas de otimização que envolvam funções objetivo que sejam grandes consumidoras de tempo de processamento. Para superar tal dificuldade, pode-se utilizar a computação paralela. A vantagem imediata no PSO paralelo (PPSO) é a redução do tempo computacional. Abordagens Mestre-Escravo, explorando esta característica são as mais investigadas. Neste trabalho, desenvolvemos diferentes algoritmos PPSO explorando as vantagens de topologias de vizinhança melhoradas, implementadas através de estratégias de comunicação em arquiteturas com multiprocessadores. O PPSO proposto foi aplicado a dois problemas complexos e grandes consumidores de tempo de processamento da engenharia nuclear: i) projeto neutrônico de reatores (PR) e ii) otimização da recarga de combustível nuclear (RC). Após exaustivos experimentos, concluímos que: i) o PPSO ainda continua encontrando soluções melhores após milhares de iterações, tornando proibitiva a utilização do modelo serial (não paralelo) do PSO neste tipo de problemas reais; e ii) o PPSO com estratégias de comunicação mais elaboradas demonstrou ser mais eficiente e robusto que o modelo Mestre-Escravo. Discutiremos aqui com mais cuidado os detalhes de implementação e os resultados.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

PARALLEL PARTICLE SWARM OPTIMIZATION ALGORITMS
IN NUCLEAR PROBLEMS

Marcel Waintraub

March/2009

Advisors: Roberto Schirru

Cláudio Márcio do Nascimento Abreu Pereira

Department: Nuclear Engineering

Particle Swarm Optimization (PSO) is a population-based metaheuristic (PBM), in which solution candidates evolve through simulation of a simplified social adaptation model. Putting together robustness, efficiency and simplicity, PSO has gained great popularity. Many successful applications of PSO are reported, in which PSO demonstrated to have advantages over other well-established PBM. However, computational costs are still a great constraint for PSO, as well as for all other PBM, specially in optimization problems with time consuming objective functions. To overcome such difficulty, parallel computation has been used. The default advantage of parallel PSO (PPSO) is the reduction of computational time. Master-slave approaches, exploring this characteristic are the most investigated. However, much more should be expected. In this work, we develop several different PPSO algorithms exploring the advantages of enhanced neighborhood topologies implemented by communication strategies in multiprocessors architectures. The proposed PPSO has been applied to two complex and time consuming nuclear engineering problems: i) reactor core design (CD) and ii) fuel reload (FR) optimization. After exhaustive experiments, it has been concluded that: i) PPSO still improve solutions after many thousands of iterations, making prohibitive the efficient use of serial (non-parallel) PSO in such kind of real-world problems ii) PPSO with more elaborated communication strategies demonstrated to be more efficient and robust than the master-slave model. Implementation details and results are carefully discussed here.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. Introdução

Durante as últimas décadas, muito se tem estudado sobre conhecimento heurístico direcionado à solução de problemas com elevado nível de complexidade computacional (NP-Completo e NP-Difícil). Apesar do sucesso destes métodos na solução de problemas do mundo real, tornou-se notória a dificuldade de se criar heurísticas de caráter geral que sejam eficientes na solução de uma gama de problemas práticos. Somente por volta dos anos 80, começaram a surgir estudos no sentido de se tentar desenvolver procedimentos heurísticos com uma certa estrutura teórica (tendo como exemplo mais notável o sistema PRODIGY, que através de aprendizagem automatizada, evoluía seus processos de tomada de decisão) (MINTON, 1988), sem, no entanto, prejudicar a sua principal característica, que é a flexibilidade.

Esta meta tornou-se mais realista, a partir da reunião de conceitos das áreas de Otimização Combinatória e Inteligência Artificial, mais especificamente Inteligência Computacional, viabilizando a construção da chamada Melhor Estratégia ou de métodos Inteligentemente Flexíveis, também conhecidos como Metaheurísticas. Estes métodos, situados em domínios possuem como característica estruturas com uma menor rigidez que a dos métodos clássicos de otimização. As metaheurísticas de otimização global têm como um dos objetivos principais buscar mecanismos que evitem a convergência prematura para ótimos locais.

Nesta linha de pensamento, o Algoritmo Genético (AG) (HOLLAND, 1975; GOLDBERG, 1989), uma das mais populares metaheurísticas baseada em população (MBP), têm sido aplicada com sucesso em vários problemas complexos de otimização

na engenharia nuclear, tais como: recarga de combustível nuclear (PARKS, 1996; CHAPOT et al., 1999; KOBAYASHI and AIYOSHI, 2002), projeto neutrônico de reatores nucleares (PEREIRA et al., 1999; SACCO et al., 2004; PEREIRA, 2004), política de manutenção (LAPA et al., 2000; LAPA at al. 2006) e de testes de inspeção (YANG et al., 2000; LAPA et al., 2002; LAPA et al., 2003), identificação de transientes (ALMEIDA et al., 2002), projetos de experimentos em escala reduzida (BOTELHO et al., 2008; LAPA et al., 2004), otimização da extração da turbina (SACCO, et al., 2002), além de outros problemas mais específicos.

Outras metaheurísticas, tais como: Aprendizado Incremental Baseado em Populações (PBIL, do inglês *Population Based Incremental Learning*) (CALDA et al., 2008), Sistemas de Colônia de Formigas (ACS, do inglês *Ant Colony Systems*) (LIMA et al., 2008), Algoritmos de Colisão de Partículas (PCA, do inglês *Particle Collision Algorithms*) (SACCO et al., 2006; SACCO et al., 2008), além do algoritmo Greate Delug (SACCO, 2006), também têm sido aplicados em problemas de engenharia nuclear.

Embora os processadores modernos apresentem alta performance computacional, muitos problemas de engenharia permanecem computacionalmente custosos para as MBPs. Apenas como ilustração: num cálculo simples de otimização da recarga de combustível nuclear (RC), utilizando AG e códigos clássicos de física de reatores, pode-se levar vários dias para se atingir uma boa (mesmo que não ótima) solução. Tais problemas têm motivado a investigação de modelos paralelos (PEREIRA e LAPA, 2003; WAINTRAUB et al., 2005; PEREIRA e SACCO, 2008).

Uma forma de sobrepujar o limite da capacidade de processamento dos computadores geralmente disponíveis (não paralelos) consiste 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), ou, simplesmente usufruir da cooperação entre os computadores de uma rede comum (neste caso, nem toda implementação paralela torna-se eficiente).

O paralelismo computacional tem sido amplamente explorado nos AGs, seguindo basicamente 3 modelos: *Mestre-Escravo*, onde apenas a avaliação da função objetivo é distribuída; *Ilhas* (malha grossa), onde sub-populações evoluem paralelamente trocando informações através de "migrações", segundo uma dada estratégia e *Celular* (malha fina), onde cada candidato (indivíduo) a solução é alocado em um processador/processo (célula) e geralmente seguem uma arquitetura de malha 2D, interligada em forma de toróide. A conexão entre as células define uma vizinhança fixa para os indivíduos.

Recentemente, o Algoritmo de Otimização por Enxame de Partículas (PSO) (KENNEDY e EBERHART, 1995), um MBP inspirada em modelos de adaptação social, tem-se demonstrado como uma boa alternativa para solução de problemas complexos de engenharia, superando outras MBPs em vários problemas de otimização de engenharia nuclear (DOMINGOS et al., 2006; WAINTRAUB et al., 2006; PEREIRA et al., 2007; MENESES et al., 2008; MEDEIROS et al., 2008).

O PSO tem adquirido popularidade devido a sua robustez, eficiência e simplicidade de desenvolvimento. Além disso, geralmente necessita de menos esforço computacional quando comparado a outras MBPs, tais como AGs ou outros algoritmos evolucionários.

Não obstante, da mesma forma que outras MBPs, o PSO ainda pode encontrar problemas com funções objetivo que necessitem a execução de simulações computacionais, cujo custo, por si só, já é relativamente elevado. Nestes casos, os modelos paralelos e distribuídos são muito úteis e, às vezes, praticamente obrigatórios. A vantagem mais imediata do PPSO está na redução do tempo computacional. Tal característica pode ser obtida facilmente com o modelo Mestre-Escravo, no qual somente a avaliação da aptidão de cada partícula (*fitness*) é calculada em paralelo. Entretanto, assim como ocorre com outras MBPs, a utilização de estratégias de comunicação (vizinhança) propicia maior diversidade e conseqüentemente, robustez, melhorando as chances de encontrar resultados melhores.

CHANG *et al.* (2005) propuseram uma abordagem de malha grossa que apresenta ganhos, devido a estratégia de comunicação, sobre o PSO tradicional (serial). Neste trabalho, entretanto, somente funções matemáticas foram utilizadas. Poucas aplicações reais de PPSO foram encontradas na literatura. SCHUTTLE *et al.* (2003) aplicaram uma abordagem Mestre-Escravo a um sistema biomecânico. Encontraram bons resultados do ponto de vista de engenharia. JIN *et al.* (2005) desenvolveram um PPSO para projeto de antenas. Eles também utilizaram um modelo Mestre-Escravo. VENTER *et al.* (2006) focaram no ganho de velocidade decorrente da utilização de um PPSO assíncrono aplicado na otimização da asa de uma aeronave de transporte típica.

LI *et al.* (2007) propuseram um modelo de PPSO de malha fina baseado em Unidades de Processamento gráfico (GPU, do inglês *Graphic Processor Unit*). Realmente, este trabalho aponta para uma nova tendência em computação de alto desempenho. Entretanto, diferentemente da abordagem baseada em multiprocessadores, proposto aqui, as GPUs são arquiteturas paralelas vetoriais (SIMD, do inglês *Single Instruction Multiple Data*) e não são adequadas para as nossas aplicações, nas quais cada simulação de física de reatores deve ser feita em processos separados e com sua própria memória.

Uma característica comum observada nas aplicações do mundo real mencionadas anteriormente foi o foco no ganho de velocidade associado ao processamento paralelo. O ganho decorrente das estratégias mais elaboradas de comunicação não foi investigado nestes casos.

Conseqüentemente, tendo como motivação: i) o bom desempenho apresentado pelo PSO em problemas nucleares, ii) o custo computacional das funções objetivo envolvidas nestes problemas e, principalmente, iii) a melhora observada no PSO com a utilização estratégias de comunicação (CHANG *et. al.*, 2005; JIAN *et al.*, 2004), este trabalho pretende investigar e desenvolver modelos de PPSO com diferentes estratégias de comunicação, para aplicação em problemas reais de engenharia nuclear. O foco aqui é não somente a redução do custo computacional, mas também os ganhos em termos de eficiência e robustez, proporcionados pelo uso das estratégias de comunicação propostas.

Para tal, quatro abordagens de PPSO, percorrendo desde malha grossa até fina, foram desenvolvidos e aplicados a dois problemas clássicos de otimização na área de

reatores nucleares: i) projeto neutrônico de reatores (PR) e ii) otimização de recarga de combustível nuclear (RC).

O foco principal deste trabalho consiste em analisar o desempenho dos PPSOs propostos em problemas reais da engenharia nuclear que tenham custo computacional elevado, investigando as vantagens e dificuldades em cada modelo de PPSO, além de mostrar o ganho decorrente do paralelismo, não somente em termos de velocidade de processamento, como também no próprio resultado da otimização.

O próximo capítulo faz uma apresentação do algoritmo PSO padrão e modelos clássicos de paralelismo. O capítulo 3 descreve os PPSOs propostos, enquanto que o 4 descreve os problemas de otimização adotados. No capítulo 5, são discutidos as aplicações e resultados obtidos pelo PPSO e, finalmente, no capítulo 6 apresentamos as conclusão finais.

2. Algoritmo de Otimização por Enxame de Partículas (PSO)

2.1. PSO Padrão

O algoritmo de Otimização por Enxame de Partículas (PSO), proposto por KENNEDY e EBERHART (1995), foi inspirado no comportamento de enxames biológicos e em aspectos de suas adaptações sociais. Apesar de ser classificado como algoritmo evolucionário, que tradicionalmente têm sua força na competição (competição Darwiniana) entre os indivíduos, o PSO utiliza a estratégia de colaboração para evoluir.

No PSO, simula-se um enxame de estruturas candidatas a solução, chamadas "partículas". Elas "voam" em um espaço *n*-dimensional (o espaço de busca do problema de otimização), atraídas por regiões de alto valor de adaptação, conforme ilustrado na figura 1 (KAEWKAMNERDPONG e BENTLEY, 2005).

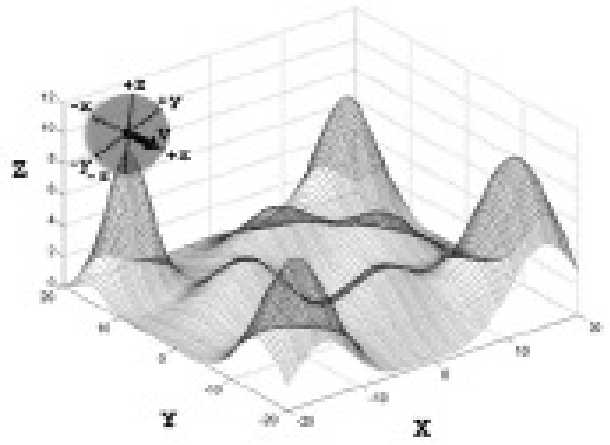


Figura 1– A partícula observando o espaço de busca num problema de otimização**.**

A "posição" da partícula representa a própria candidata a solução, enquanto a topologia do espaço de busca é dada pela função objetivo do problema. A cada partícula também é atribuída uma "velocidade", onde se encontram as informações de direção e taxa de mudança de posição em função do "tempo", e o atributo de *performance* (ou adequação), obtido pela avaliação da função objetivo na posição da partícula.

A mudança da posição da partícula e de sua velocidade é guiada por sua própria experiência (informação histórica das regiões boas e ruins pelas quais a partícula já passou), bem como pela observação de seus vizinhos bem sucedidos.

Sejam $\vec{X}_i(t) = \{x_{i,1}(t), \cdots, x_{i,n}(t)\}$ e $\vec{V}_i(t) = \{v_{i,1}(t), ..., v_{i,n}(t)\}$, respectivamente, a posição (o próprio vetor candidato a solução) e a velocidade (sua taxa de mudança) da partícula *i* no tempo *t*, em um espaço de busca n-dimensional. Considerando-se também $\overrightarrow{pBest_i}(t) = \{pBest_{i,1}(t), \cdots, pBest_{i,n}(t)\}$, a melhor posição já encontrada pela partícula *i* até o tempo *t* e $\overrightarrow{gBest_i}(t) = \{gBest_{i,1}(t), \cdots, gBest_{i,n}(t)\}$ a melhor posição já encontrada pelo enxame até o tempo *t*. As regras de atualização do PSO para a velocidade e posição, no PSO canônico, são dadas por:

$$v_{i,n}(t+1) = v_{i,n}(t) + c_1.r_1.(pBest_{i,n}(t) - x_{i,n}(t)) + c_2.r_2.(gBest_{i,n}(t) - x_{i,n}(t)) \quad (1)$$

$$x_{i,n}(t+1) = x_{i,n}(t) + v_{i,n}(t+1) \quad (2)$$

onde r_1 e r_2 são números randômicos uniformemente distribuídos entre 0 e 1. Os coeficientes c_1 e c_2 são as constantes de aceleração (geralmente chamadas de

aceleração cognitiva e social, respectivamente) relativas a $\overrightarrow{pBest}$ e $\overrightarrow{gBest}$ respectivamente. A contribuição cognitiva (individual) e social (enxame) à movimentação de uma dada partícula pode ser visualizada como um paralelogramo num espaço de busca 2-D, conforme ilustrado na figura 2 (SCHUTTE *et al.*, 2004).

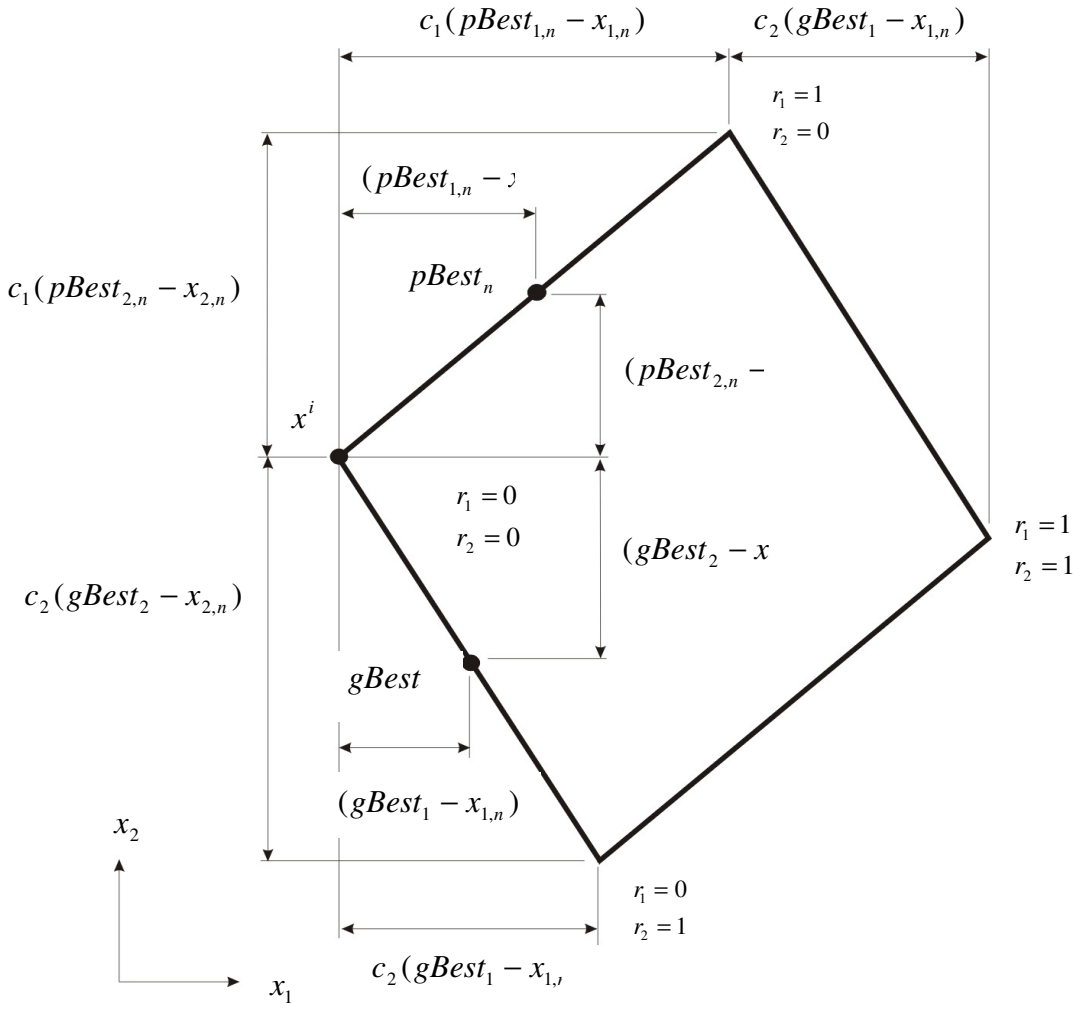


Figura 2– Contribuições cognitiva e social.

Como podemos observar nas equações (1) e (2), não existe um mecanismo que limite a velocidade de uma dada partícula. Esta falta pode resultar em uma baixa

eficiência para o PSO, quando comparados a outras técnicas de computação evolucionária (ANGELINE, 1998). Desta forma, o PSO localiza rapidamente (quando comparado com outras técnicas de EC) a região do ótimo, mas uma vez dentro desta região, ele pode enfrentar dificuldade em ajustar o seu incremento de velocidade para prosseguir numa busca mais refinada.

Para resolver este problema introduz-se um peso para a velocidade anterior da partícula, denominado peso inercial, w, cuja regra é considerada crítica para a convergência do PSO. Com isto, a equação da velocidade da partícula, anteriormente descrita pela equação 1, passa a ser:

$$v_{i,n}(t+1) = w.v_{i,n}(t) + c_1.r_1.(pBest_{i,n}(t) - x_{i,n}(t)) + c_2.r_2.(gBest_{i,n}(t) - x_{i,n}(t)) \qquad (3)$$

Valores elevados de w promovem a exploração e prospecção globais, enquanto que valores baixos conduzem a uma busca local. Uma aproximação comumente utilizada para aumentar a performance do PSO, promovendo um balanço entre a busca global e local, consiste em inicializar w com um valor alto e ir decrescendo (linearmente) durante a execução do PSO (SHI and EBERHART, 1998), conforme mostra a Equação (4):

$$w = \begin{cases} w_{\max} - \dfrac{w_{\max} - w_{\min}}{iter_{max}} iter, & iter \le iter_{max} \\ w_{\min}, & iter > iter_{max} \end{cases} \qquad (4)$$

onde *iter* é a iteração corrente e $iter_{max}$ é o número máximo de iterações.

Valores muito altos de velocidade podem fazer com que a partícula tente "voar" para fora do espaço de busca. Neste caso, algumas estratégias podem ser utilizadas, como "parar" a partícula nos limites do espaço de busca ou, mais natural e eficiente, refletir a partícula para dentro do espaço de busca, conforme ilustrado na Figura 3, que considera busca em 2 dimensões.

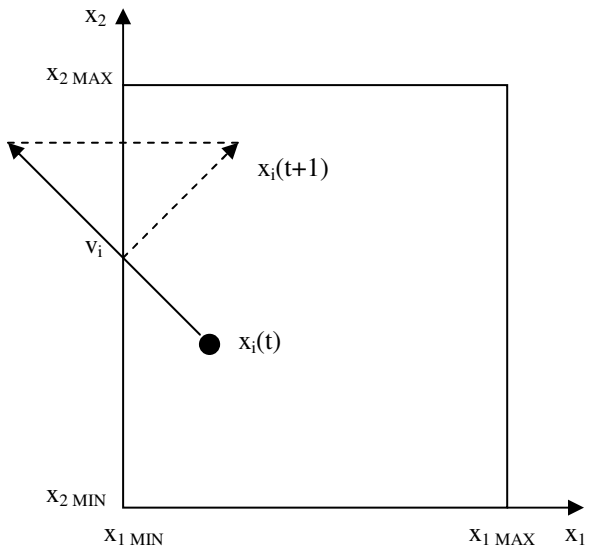


Figura 3– Reflexão de uma partícula nas "bordas" do espaço de busca.

Tal tratamento, entretanto, pode ocasionar a necessidade de várias reflexões em caso de velocidades muito altas, podendo diminuir a eficiência do PSO. Torna-se adequado, então, uma limitação do valor da velocidade da partícula em um valor máximo (V_{MAX}). Quando a velocidade exceder este limite, ela será fixada em V_{MAX}.

No PSO original, todas as partículas "navegam" por um espaço de busca *n*-dimensional, delimitado pelos intervalos de variação dos *n* parâmetros de otimização. O enxame é classificado como sendo de vizinhança global, onde todas as partículas são informadas quando um novo melhor ponto é encontrado no espaço de busca. Denominamos este modelo de PSO-Padrão.

No algoritmo do PSO, o enxame é inicializado randomicamente (posições e velocidades). Então, enquanto o critério de parada (no caso um número máximo de iterações) não é atingido, executa-se um *loop* contendo os seguintes passos:

i. partículas são avaliadas de acordo com a função objetivo, e os valores de aptidão de cada partícula são determinados;

ii. valores de $\overrightarrow{pBest}$ e $\overrightarrow{gBest}$ são atualizados; e

iii. partículas são movimentadas de acordo com as equações de atualização para a velocidade e posição (Equações (1) e (2)).

A Figura 4 mostra o pseudo-código do PSO.

```
Algoritmo PSO
begin
   for i=1 to n_particles do begin
      randomize(Xi); randomize(Vi);
   end;
   for iter=1 to itermax do begin
      for i=1 to n_particles do evaluate (Xi);
      for i=1 to n_particles do update(pBesti,gBest);
      for i=1 to n_particles do begin
         Vi = w*Vi+c1*r1*(pBesti–Xi)+c2*r2*(gBest–Xi);
         Xi = Xi + Vi;
      end;
   end;
end.
```

Figura 4 – Pseudocódigo do PSO.

Existem algumas variações do PSO-Padrão que visam aumentar a diversidade populacional, fato este que geralmente resulta em melhores resultados. Uma estratégia comum consiste em considerar modelos de vizinhanças locais. Alguns modelos de vizinhança consideram "distâncias" no espaço de busca. Neste tipo de abordagem, uma redondeza radial da posição da partícula determina o limite da vizinhança, dentro da qual a adaptação social deve ocorrer. Esta abordagem será denominada PSO-Local. Outra abordagem, muito pouco explorada segundo a literatura atual, é um modelo que considera vizinhanças fixas entre as partículas (modelo de Von Neumman, KENNEDY, 2006). Este modelo tem a característica de demandar menos esforço computacional e ser altamente paralelizável.

2.2. Modelos de Paralelismo Computacional

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.

Dentre as metaheurísticas baseadas em populações (MBPs) onde mais se tem explorado o paralelismo computacional podemos citar os Algoritmos Genéticos Paralelos (AGP) (CANTÚ-PAZ, 2000). Os conceitos utilizados nos AGPs podem, entretanto, ser extrapolados (com adaptações) à outras MBPs. Dependendo da estratégia de comunicação adotada, podemos classificar os modelos paralelos em três tipos: i) mestre-escravo; ii) malha grossa (modelo de ilhas); e iii) malha fina (modelo celular).

No modelo mestre-escravo (veja figura 5a), somente as avaliações da função de adequação (*fitness*) são feitas em paralelo (nos processos ditos “escravos”). O processo dito “mestre” controla a evolução da geração e as operações nos indivíduos da população. Por este ponto de vista, o processo de otimização mantém o algoritmo original inalterado. Neste modelo, o ganho consiste apenas na redução do tempo computacional do processo de otimização.

Os modelos de malhas grossas e finas exploram as restrições de vizinhança decorrentes das estratégias de comunicação entre os processadores (ou processos). Conseqüentemente, a manutenção da diversidade (ponto chave para o sucesso de qualquer MBP) é aperfeiçoada, tornando o algoritmo mais robusto e eficiente.

O modelo de ilha (malha grossa), ilustrado na figura 5b, faz uma abordagem baseada em múltiplas populações. Cada sub-população é alocada em um processador, chamado de "ilha", onde ela sofre um processo adaptativo independente. Então, de acordo com dada estratégia de "migração" pré-definida (normalmente um dado número de gerações), indivíduos são trocados.

O modelo de ilhas é capaz de controlar o intercâmbio de informações globais através da estratégia de migração, desde que parâmetros, tais como a freqüência de migração, sejam adequadamente escolhidos.

Além disso, devemos salientar que as sub-populações devem ter um tamanho representativo. Tamanhos inadequados das sub-populações podem conduzir a convergência prematura. Este fato acarreta na necessidade de grandes populações globais (soma das sub-populações das ilhas), quando comparadas aos modelos mestre-escravo e celular.

No modelo celular (malha fina) os indivíduos, I_i, são dispostos num arranjo matricial de células interconectadas de acordo com a topologia de uma grade 2D

(topologia Von Neumann), formando um toróide, onde cada célula é conectada a outros quatro vizinhos, como pode ser visto na Figura 5c. Tal restrição de vizinhança retarda o fluxo de informação entre processos não vizinhos, aumentando assim a diversidade na busca por soluções.

Como este modelo utiliza uma única população, distribuída pelos processadores disponíveis, populações menores são aceitáveis, quando comparadas ao modelo de ilhas. Por outro lado, a troca de informações através das conexões entre vizinhos é feita permanentemente, o que impõe um aumento de comunicação em relação ao modelo de ilhas. Uma vantagem é que não há a necessidade de definir-se uma estratégia de migração.

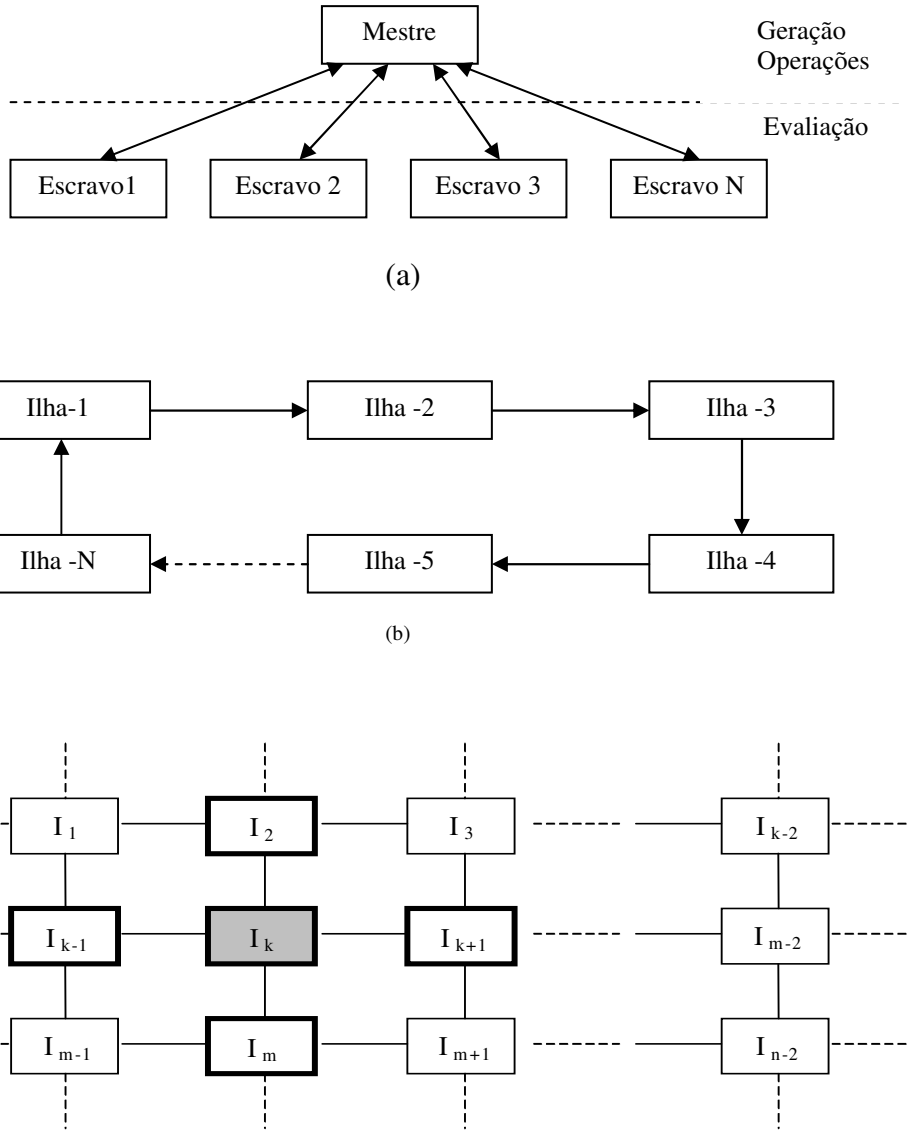


Figura 5 – Modelos Paralelos do AG: a) mestre-escravo; b) ilhas (anel); c) celular

3. Modelos Paralelos Propostos de PSO

Hoje em dia já são observadas várias aplicações bem sucedidas de PSOs em diversos setores da engenharia. Na Engenharia Nuclear tal fato é também observado. Em vários problemas, o PSO tem demonstrado vantagens sobre outras técnicas concorrentes (como por exemplo, os AGs) (WAINTRAUB *et al*., 2006, PEREIRA *et*

al., 2007, MEDEIROS e SCHIRRU, 2008, MENESES *et al*., 2008) tanto em termos de resultado da otimização, como em termos de custo computacional. No entanto, ainda assim, existem aplicações cujo custo computacional das simulações envolvidas, por si só, já é relativamente elevado, impactando na solução prática do problema através de computadores tradicionais.

Conforme mencionado anteriormente, uma forma de contornar tal dificuldade é a utilização de arquiteturas computacionais paralelas.

Três dos modelos paralelos propostos de PSO são inspirados nos modelos paralelos tradicionais do AG (CANTÚ-PAZ, 2000). O PPSO mestre-escravo, o de ilhas (topologia em anel) e o celular foram adaptados do AGP. Também propomos nesta tese uma nova abordagem para PPSOs. A idéia consiste em utilizar o conceito de vizinhança no PSO, para "conectar" as ilhas através de alguns indivíduos (vizinhos), eliminando assim a necessidade de definir o parâmetro do intervalo de migração. Neste trabalho denominamos estes modelos de modelos de ilhas com vizinhança do PPSO.

O modelo PPSO mestre-escravo é o mais simples, onde as avaliações são feitas pelos processos escravos enquanto o controle do PSO é centralizado no processo mestre. No modelo mestre-escravo do PPSO, o algoritmo do PSO original não é alterado e nenhuma melhoria é obtida na solução. Os outros modelos estão descritos nas seções subseqüentes.

3.1. Modelo Paralelo de Ilhas do PSO: PPSO-Ilhas

O modelo desenvolvido para o PPSO de ilhas (PPSO-Ilhas) utiliza a topologia de ilhas em anel, na qual, sub-populações de partículas evoluem separadamente, com trocas de partículas periodicamente através de um processo de "migração". Após um certo número de iterações, a melhor partícula de cada ilha, chamada de ótimo local $\overrightarrow{nBest}$ "migra" para outra ilha, de acordo com a topologia em anel (ver figura 6a). Mais precisamente, ela substitui uma partícula de outra ilha escolhida aleatoriamente.

A topologia em anel foi escolhida devido à facilidade de implementação e os bons resultados obtidos na implementação de Algoritmos Genéticos Paralelos encontrados na literatura (PEREIRA e LAPA, 2003; PEREIRA e SACCO, 2008).

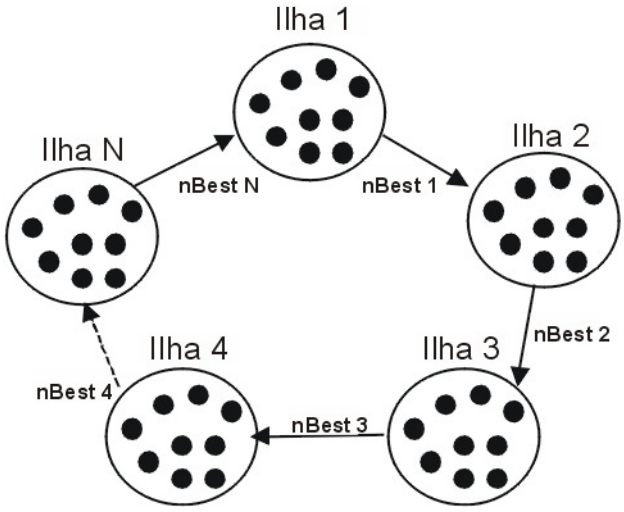


Figura 6a – Modelo de Ilhas: PSO-Ilhas – topologia em anel

A Figura 6b mostra o pseudocódigo do modelo de Ilhas do PSO, realizado em paralelo por cada ilha.

```
Algoritmo PSO-Ilhas {realizado em paralelo por cada ilha}
begin
   for i=1 to n_particles do begin
      randomize(Xi); randomize(Vi);
   end;
   for iter=1 to itermax do begin
      for i=1 to n_particles do evaluate (Xi);
      for i=1 to n_particles do update(pBesti,nBest);
      for i=1 to n_particles do begin
         Vi = w*Vi+c1*r1*(pBesti-Xi)+c2*r2*(nBest-Xi);
         Xi = Xi + Vi;
      end;
      if (iter MOD mig_interval)==0 Realiza_Migração
   end;
end.
```

Figura 6b – Pseudocódigo do PSO-Ilhas.

3.2. Modelo Paralelo Celular do PSO: PPSO-Celular

No modelo PPSO celular (PPSO-Celular) proposto, as partículas são dispostas num arranjo matricial de células interconectadas de acordo com a topologia de uma grade 2D (Von Neumann), formando um toróide, onde cada célula é conectada a outras quatro, como pode ser visto na Figura 7a. Note que as partículas têm uma vizinhança

fixa, ou seja, P_k, por exemplo, tem somente P_{k-1}, P_{k+1}, P_1 e Pm como vizinhos, independentemente de suas posições no espaço de busca. Portanto, ao invés de um $\overrightarrow{gBest}$ (Eq. 1) comum a todo o enxame, cada partícula possui um "$\overrightarrow{gBest}$" (na realidade seria um ótimo local ou $\overrightarrow{nBest}$) individual, selecionado entre seus vizinhos.

A topologia celular promove uma rápida adaptação local (dentro dos nichos formados pelas vizinhanças) e desacelera a circulação da informação globalmente, retardando a convergência e promovendo uma maior diversidade na população.

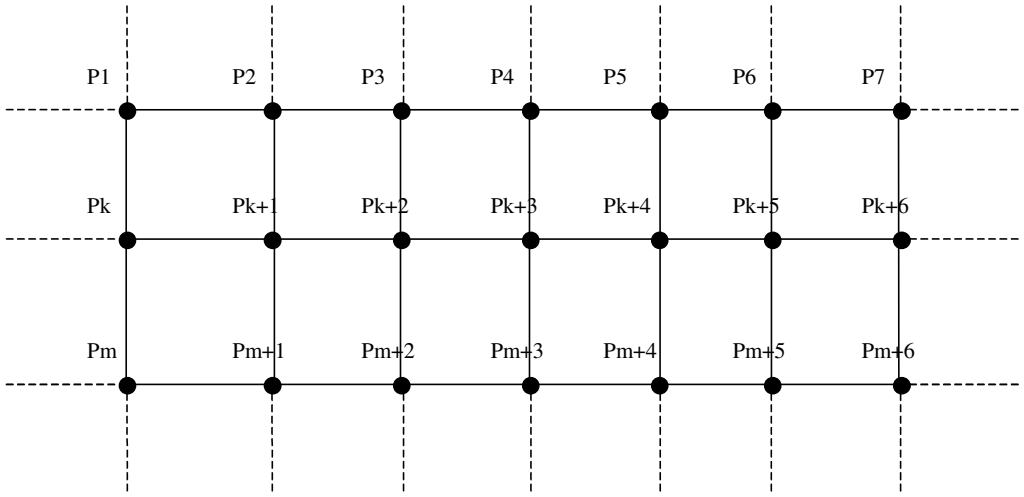


Figura 7a – Modelo Celular: PSO-Celular. OBS: P_k = Partícula k

Ensaios preliminares com *benchmarks* demonstraram que uma arquitetura de vizinhanças como acima descrita leva a resultados superiores aos obtidos por PSO-Original. Entretanto, observa-se ainda que a atração promovida pelo ótimo global ($\overrightarrow{gBest}$) pode promover convergência prematura (ressalta-se que, mesmo assim, os resultados são melhores que os do PSO tradicional). Motivados por esta observação,

propõe-se o estudo e incorporação no proposto PPSO-Celular de novas técnicas de diversificação populacional, visando o desenvolvimento de uma ferramenta ainda mais eficiente e robusta.

```
Algoritmo PSO-Celular
begin
   for i=1 to n_particles do begin
      randomize(Xi); randomize(Vi);
   end;
   for iter=1 to itermax do begin
      for i=1 to n_particles do evaluate (Xi);
      for i=1 to n_particles do update(pBest);
      for i=1 to n_particles do begin
           for j=1 to n_neighbors do update(nBest);
           {n_neighbors = 4 para todas as partículas}
      end;
      for i=1 to n_particles do begin
           Vi = w*Vi+c1*r1*(pBesti-Xi)+c2*r2*(nBest-Xi);
           Xi = Xi + Vi;
      end;
   end;
end.
```

Figura 7b – Pseudocódigo do PSO-Celular.

3.3. Modelos Paralelos de Ilhas com Vizinhança do PSO

O modelo de ilhas com vizinhança do PPSO é uma alternativa ao modelo de ilhas, no qual cada ilha é conectada às outras através de N vizinhos, de acordo com uma dada topologia. Este modelo pode ser visto como uma espécie de hibridização entre os

modelos de ilhas e o celular. A vantagem observada neste modelo é que ele fornece uma interface natural entre as ilhas, na qual a informação é naturalmente passada, dispensando a necessidade de definirmos um "intervalo de migração", parâmetro ao qual o modelo de ilhas pode ser consideravelmente sensível. Neste trabalho, investigamos duas topologias:

i) Uma topologia em anel, na qual cada ilha conecta-se a outras duas através de duas partículas; e
ii) Uma topologia de grade 2D, na qual cada ilha conecta-se a outras quarto através de quatro vizinhos.

A topologia em anel, chamada aqui de PPSO-N2 encontra-se ilustrado na Figura 8a, enquanto que a topologia de grade, chamada aqui de PPSO-N4 está na Figura 9a.

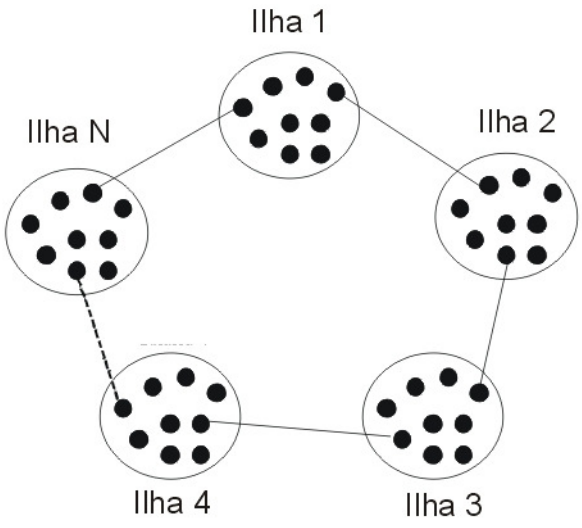


Figura 8a - Modelo Paralelo de Ilhas com Vizinhança com 2 vizinhos: PPSO-N2.

```
Algoritmo PSO-N2
begin
   for i=1 to n_particles do begin
      randomize(X_i); randomize(V_i);
   end;
   for iter=1 to iter_max do begin
      for i=1 to n_particles do evaluate (X_i);
      for i=1 to n_particles do update(pBest_i);
      for i=1 to n_islands do begin
           for j=1 to n_particles_island do begin
                for k=1 to n_neighboors[k] do update(nBest);
                {neighboors[k] = n_particles_island-1, exceto
                nas 2 partículas que conectam-se a outras
                ilhas, que têm mais 1 vizinho cada}
           end;
      end;
      for i=1 to n_particles do begin
           V_i = w*V_i+c_1*r1*(pBest_i-X_i)+c_2*r2*(nBest-X_i);
           X_i = X_i + V_i;
      end;
   end;
end.
```

Figura 8b – Pseudocódigo do PSO-N2.

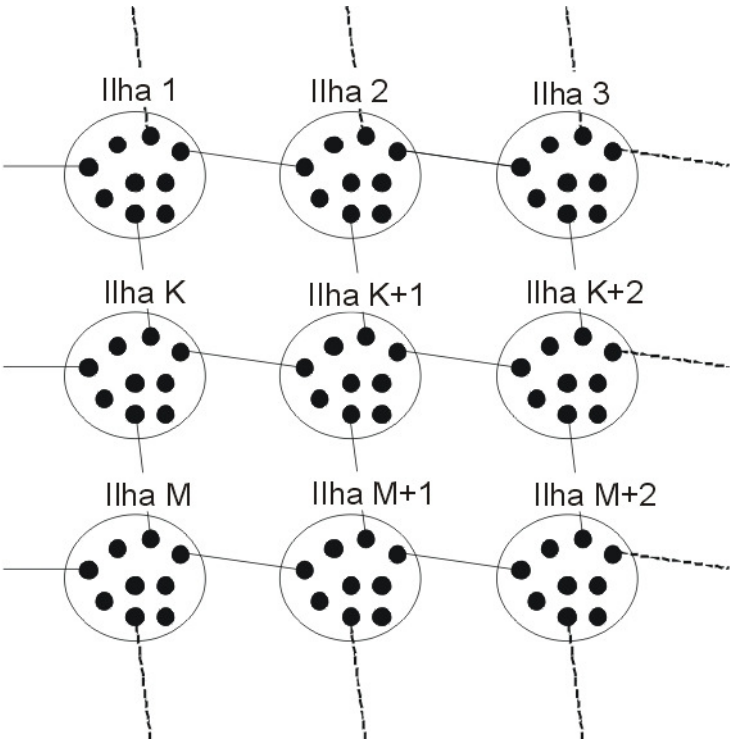


Figura 9a – Modelo Paralelo de Ilhas com Vizinhança com 4 vizinhos: PPSO-N4.

```
Algoritmo PSO-N4
begin
   for i=1 to n_particles do begin
      randomize(Xi); randomize(Vi);
   end;
   for iter=1 to itermax do begin
      for i=1 to n_particles do evaluate (Xi);
      for i=1 to n_particles do update(pBesti);
      for i=1 to n_islands do begin
           for j=1 to n_particles_island do begin
                for k=1 to n_neighboors[k] do update(nBest);
                {neighboors[k] = n_particles_island-1,exceto
                nas 4 partículas que conectam-se a outras
                ilhas, que têm mais 1 vizinho cada}
           end;
      end;
      for i=1 to n_particles do begin
         Vi = w*Vi+c1*r1*(pBesti-Xi)+c2*r2*(nBest-Xi);
         Xi = Xi + Vi;
      end;
   end;
end.
```

Figura 9b – Pseudocódigo do PSO-N4.

3.4. Interface de Troca de Mensagens

Um ambiente de programação via troca de mensagens é formado por: uma linguagem seqüencial (como C, C++, Fortran e Java), que será utilizada na implementação dos programas seqüenciais nos processadores; e uma biblioteca de funções que forneçam o suporte à comunicação entre diferentes processos ou nós de

processamento. Contudo, tarefas de paralelização como comunicação e sincronização entre processos, particionamento e distribuição de dados e mapeamento dos processos entre os processadores disponíveis, ainda ficam por conta do programador.

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 portabilidade1 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 (*Message Passing Interface*)[16].

PVM é um conjunto integrado de bibliotecas e ferramentas de software, cuja finalidade é emular um sistema computacional concorrente heterogêneo, flexível e 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 justifica a necessidade de um padrão para esse tipo de sistema, tais como: portabilidade e facilidade de uso (à 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

[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.

torna a criação de *software* paralelo por empresas uma opção comercialmente viável, o que também implica em maior difusão do uso de computadores paralelos). A maior desvantagem do PVM em relação ao MPI refere-se ao seu desempenho, pois para conseguir maior flexibilidade, o primeiro acaba sacrificando sua performance.

Segundo a definição do padrão MPI, os seguintes requisitos são esperados em uma biblioteca paralela robusta:

- Criação de um espaço de comunicação seguro, garantindo a execução de operações de comunicação sem que ocorram conflitos com comunicações não relacionadas à biblioteca;
- Permitir o uso de escopo de grupo para facilitar a realização de comunicações coletivas;
- Identificação dos processos usuários através de nomes abstratos;
- Permitir ao usuário ou construtor da biblioteca estender notações do modelo de passagem de mensagens.

Com vistas à satisfação destes requisitos, o padrão MPI inclui entre suas definições os conceitos de comunicadores, contextos de comunicação e grupos de processos. Em um programa MPI, uma operação de comunicação é executada dentro de um determinado contexto, que é especificado por um certo comunicador e indica os possíveis processos receptores da mensagem. A seguir apresentamos uma breve descrição de cada um destes conceitos.

O MPI relaciona os processos em grupos, e esses 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. Um processo pode pertencer a mais de um grupo e, neste caso, possuir diferentes *ranks*.

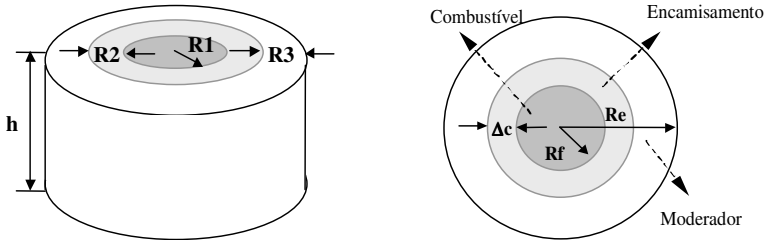
Os conceitos e as principais diretivas da biblioteca MPI e suas características estão descritas em detalhe no apêndice A.

4. Problemas de Otimização

Consideramos aqui dois problemas clássicos da engenharia nuclear que podem vir a ser utilizados nas investigações: i) o projeto neutrônico de um reator nuclear e ii) a recarga de combustível nuclear.

4.1. Projeto Neutrônico de Reator Nuclear

O problema considerado consiste em maximizar o fluxo térmico médio, ϕ_{AVE}, de um reator tipo PWR cilíndrico com 3 zonas de enriquecimento (ilustrado na Figura 10), considerando que este deve permanecer crítico (k_{eff} = 1,0 ± 1%), submoderado e abaixo de um fator de pico fp_{MAX}.



(a) **(b)**

Figura 10- (a) O reator e (b) sua célula típica.

Desta forma, o problema de otimização pode ser escrito como:

$$Maximizar\ \phi_{AVE}(R_f, \Delta_c, R_e, E_1, E_2, E_3, M_f, M_c)$$

Sujeito as seguintes restrições:

$$fp\ (Rf,\ \Delta c,\ Re,\ E1,\ E2,\ E3,\ Mf,\ Mc) \leq fp_{MAX};\tag{4}$$

$$0{,}99 \leq k_{eff}(R_f,\ \Delta_c,\ R_e,\ E_1,\ E_2,\ E_3,\ M_f,\ M_c) \leq 1{,}01;\tag{5}$$

$$\frac{dk_{eff}}{dV_m} > 0\,;\tag{6}$$

$$R_{f\,min} \leq R_f \leq R_{f\,max};\tag{7}$$

$$\Delta_{c\,min} \leq \Delta_c \leq \Delta_{c\,max};\tag{8}$$

$$R_{e\,min} \leq R_e \leq R_{e\,max};\tag{9}$$

$$E_{1\,min} \leq E_1 \leq E_{1\,max};\tag{10}$$

$$E_{2\,min} \leq E_2 \leq E_{2\,max};\tag{11}$$

$$E_{3\,min} \leq E_3 \leq E_{3\,max};\tag{12}$$

$$M_f = \{UO_2,\ ou\ U\text{-}metálico\};\tag{13}$$

$$M_c = \{Zircaloy\text{-}2,\ Alumínio\ ou\ Aço\text{-}304\};\tag{14}$$

onde ϕ_{AVE} é o fluxo normalizado, 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	2,0-6,0
Enriquecimento da zona 2 (%)	E_2	2,0-6,0
Enriquecimento da zona 3 (%)	E_3	2,0-6,0
Material do combustível	M_f	UO_2 ou U-metálico
Material do encamisamento	M_c	Zircaloy-2, Alumínio ou Aço-304

O cálculo do fluxo neutrônico envolve a solução das equações de transporte e difusão de nêutrons. Para efeito de simplificação, tanto o cálculo dos parâmetros da célula (seções de choque), quanto o cálculo espacial do fluxo neutrônico, foram feitos pelo código HAMMER (SUICH e HONECK, 1967).

O HAMMER realiza um cálculo multigrupo da distribuição do fluxo neutrônico térmico e epitérmico baseado na teoria integral de transporte aplicada a uma célula de uma malha quadrada.

$$\phi(\vec{r}) = \int \frac{e^{-\Sigma_t |\vec{r}-\vec{r}'|}}{4\pi|\vec{r}-\vec{r}'|^2} S(\vec{r}')d^3r' \tag{15}$$

A equação integral de transporte (eq. 15) para o fluxo escalar $\phi(\vec{r})$ é resolvida para todas as sub-regiões da célula, considerando-se a fonte de nêutrons $S(\vec{r}')$ isotrópica no grupo de energia considerado. Sua solução está relacionada à probabilidade de colisão para uma fonte isotrópica plana situada na região inicial. Inicialmente, considera-se uma célula de comprimento infinito.

A partir daí, a fim de se levar em conta os efeitos da fuga de nêutrons, o HAMMER realiza o cálculo padrão de busca da criticalidade. Para tal, resolve-se a equação de difusão (eq. 16) unidimensional em quatro grupos de energia.

$$-\vec{\nabla}D_g(r)\vec{\nabla}\phi_g(r) + \Sigma_{t,g}(r)\phi_g(r) = \sum_{g'=1}^{4}\left[\frac{1}{k_{eff}}\chi_g\Sigma_{fg'}(r) + \Sigma_{sg'g}(r)\right]\phi_{g'}(r) \tag{16}$$

O fluxo $\phi_g(r)$ é calculado assumindo-se uma densidade de fonte normalizada. A equação 16 é resolvida utilizando-se o método de diferenças finitas com malhas de espessura constante.

Por simplificação, na modelagem feita no núcleo do reator típico da figura 9, consideramos que toda a sua altura efetiva, que é igual a 1,63 metros, é totalmente preenchida por células de elemento combustível, desconsiderando-se a presença de barras de controle e venenos queimáveis. Além disso, incluímos na estrutura dos conjuntos de combustível uma pequena concentração de boro e aço, bem como de outros isótopos decorrente da fissão do urânio. As densidades dos materiais utilizados foram obtidas do manual do HAMMER, e na figura 11 apresentamos um arquivo típico de entrada de dados para este código.

.

```
2       11
'hammer.out'
'LITHELIB.BIN'
'HELPLIB.BIN'
11          2           1  11       11 11
   0         2 3 11 CASO EXEMPLO
   1        21      REGIAO 1
   2  3 3    3 1    21125.E-04
   3  1 1  101 6    0.43622047   10.0900  2.472441 11538.E-1
   4                 92236.E+00
1.E-24
   5                 94239.E+00
1.E-24
   6                 94240.E+00
1.E-24
   7                 94241.E+00
1.E-24
   8                 54135.E+00
1.E-24
   9                 62149.E+00
1.E-24
  10  2 2  201       0.46818898    8.0200            332.E+00
  11  3 3  300 2  1 0.70881890713151.E-6             327.E+00
  12                 5010.E+00
125.E-07
1 13                  304.E+00
6632.E-07
   1        22      REGIAO 2
   2  3 3    3 1    12225.E-03
1  3  1 1  101 6    0.43622047   10.0900  2.944882 11538.E-1
   1        23      REGIAO 3
   2  3 3    3 1    21025.E-03
1  3  1 1  101 6    0.43622047   10.0900  4.527559 11538.E-1
   1         1      CALCULO ESPACIAL
   2  2 3    9          1.E+00
   3  1     21  01  33859.E-03  3.732150
   4  2     22  01  14961.E-03
1  5  3     23  01  70866.E-04
```

Figura 11 – Arquivo de entrada de dados para o código HAMMER: núcleo de um reator com três zonas de enriquecimento.

A função objetivo a ser maximizada (eq. 17) foi desenvolvida de tal forma que, uma vez satisfeitas todas as restrições, ela assuma o valor do fluxo térmico médio, ϕ_{AVE}. Caso contrário, ela é penalizada proporcionalmente a restrição não satisfeita.

$$
f = \begin{cases}
\phi_{AVE}, & \Delta k_{eff} \le 0{,}01;\ fp \le 1{,}40;\ \dfrac{\Delta' k_{eff}}{\Delta Vm} > 0 \\
\phi_{AVE} - r_1.\Delta k_{eff}, & \Delta k_{eff} > 0{,}01;\ fp \le 1{,}40;\ \dfrac{\Delta' k_{eff}}{\Delta Vm} > 0 \\
\phi_{AVE} - r_2.\Delta fp, & \Delta k_{eff} \le 0{,}01;\ fp > 1{,}40;\ \dfrac{\Delta' k_{eff}}{\Delta Vm} > 0 \\
\phi_{AVE} - r_3.\dfrac{\Delta' k_{eff}}{\Delta Vm}, & \Delta k_{eff} \le 0{,}01;\ fp \le 1{,}40;\ \dfrac{\Delta' k_{eff}}{\Delta Vm} < 0 \\
\phi_{AVE} - r_1.\Delta k_{eff} - r_2.\Delta fp, & \Delta k_{eff} > 0{,}01;\ fp > 1{,}40;\ \dfrac{\Delta' k_{eff}}{\Delta Vm} > 0 \\
\phi_{AVE} - r_1.\Delta k_{eff} - r_3.\dfrac{\Delta' k_{eff}}{\Delta Vm}, & \Delta k_{eff} > 0{,}01;\ fp \le 1{,}40;\ \dfrac{\Delta' k_{eff}}{\Delta Vm} < 0 \\
\phi_{AVE} - r_2.\Delta fp - r_3.\dfrac{\Delta' k_{eff}}{\Delta Vm}, & \Delta k_{eff} \le 0{,}01;\ fp > 1{,}40;\ \dfrac{\Delta' k_{eff}}{\Delta Vm} < 0 \\
\phi_{AVE} - r_1.\Delta k_{eff} - r_2.\Delta fp - r_3.\dfrac{\Delta' k_{eff}}{\Delta Vm}, & \Delta k_{eff} > 0{,}01;\ fp > 1{,}40;\ \dfrac{\Delta' k_{eff}}{\Delta Vm} < 0
\end{cases} \tag{17}
$$

Onde,

$$\Delta k_{eff} = \left|1{,}0 - k_{eff}\right| \tag{18}$$

é a penalização para configurações onde k_{eff} é diferente de 1,0 ± 1%;

$$\Delta fp = fp - 1{,}40 \tag{19}$$

é a penalização para configurações onde fp é maior que 1,4; e

$$\frac{\Delta' k_{eff}}{\Delta Vm} = \frac{k_{eff} - k'_{eff}}{\Delta Vm} \tag{20}$$

é a penalização para configurações onde o reator se apresenta submoderado, sendo k'_{eff} o fator de multiplicação de nêutrons obtido para ΔVm de variação em Vm (como o volume do combustível, V_f, permanece constante, a fração $\frac{\Delta V_m}{\Delta V_f}$ é alterada.

As constantes de penalização, $r_{i,}$ foram obtidas a partir de estudos de sensibilidade anteriores (WAINTRAUB et al., 2005), de acordo com as exigências e as prioridades do problema. As constantes escolhidas basearam-se na sensibilidade dos resultados encontrados para cada uma das faixas de r_i testadas. Considerando-se os valores mais conservativos destas constantes de penalização, adotamos os seguintes valores: r1 = 10, r2 = 1 e r3 = 1000.

4.2. Recarga do combustível Nuclear

O núcleo do reator brasileiro tipo PWR da usina Angra-1 de 626 MW, existente na Central Nuclear Álvaro Alberto (CNAAA), será utilizado como caso exemplo para esta investigação. O núcleo deste reator é constituído de 121 elementos combustíveis (EC), com 235 varetas combustíveis cada, distribuídos de acordo com a seguinte configuração: i) elemento central, ii) 10 elementos de quartetos (conjuntos de 4 elementos existentes nas posições por onde passam os eixos de simetria), totalizando 40 ECs, e iii) 10 elementos de octetos (conjuntos de 8 elementos posicionados fora dos eixos de simetria), totalizando 80 ECs.

Com simetria de um-oitavo, o número de elementos a serem considerados cai de 121 (no núcleo inteiro) para 20 (com o elemento central fixo), conforme mostra a Figura 12.

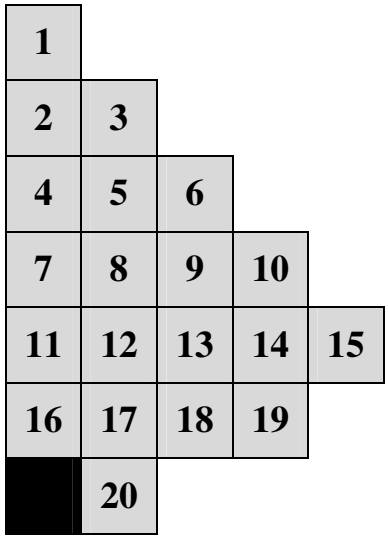


Figura 12 – Núcleo do reator tipo PWR Angra-1, com simetria de um-oitavo e posições dos elementos combustíveis numeradas.

O período de tempo que o reator opera com um determinado conjunto de elementos combustíveis é chamado de ciclo de operação. O final de cada ciclo ocorre quando a concentração de material físsil (^{235}U) atinge um valor que impossibilite a produção de energia à potência nominal.

Uma das tarefas da recarga nuclear consiste no processo de desligamento do reator, que é iniciado ao término de cada ciclo de operação da usina. Neste processo, todos os elementos combustíveis participantes do ciclo são retirados e armazenados em uma Piscina de Combustíveis Usados (PCU). Alguns destes elementos, os mais

queimados, ficarão na PCU em definitivo, sendo substituídos por novos no próximo ciclo de operação.

O processo de recarga do novo ciclo de operação do reator irá utilizar os conjuntos de elementos combustíveis novos juntamente com aqueles descarregados do ciclo de operação anterior que foram parcialmente queimados e ainda podem ser aproveitados.

O problema da recarga consiste, portanto, em determinar como combinar os conjuntos de elementos combustíveis a fim de otimizar o próximo ciclo de operação da usina, obtendo para isso um padrão de carregamento que respeite as restrições operacionais e de segurança (MACHADO, 2005).

Esta otimização do ciclo de operação de uma usina nuclear consiste em minimizar o custo da energia gerada. A determinação deste custo, no entanto, não possui formulação simples que leve em conta fatores complexos tanto operacionais quanto não operacionais. Em geral costuma-se utilizar funções objetivo tais como:

- Maximizar a reatividade no final do ciclo;
- Maximizar a queima dos elementos combustíveis;
- Minimizar o número de elementos combustíveis novos a serem utilizados.

E restrições tais como:

- Fator de pico máximo;
- Coeficiente de temperatura máximo do moderador;
- Minimização dos danos ao vaso do reator.

O comportamento altamente não linear dos reatores nucleares (POON e PARKS, 1992) gera funções objetivo e restrições com características não lineares. Conseqüentemente, teremos um número incrivelmente grande de pontos de mínimo (ou máximo) locais (GALPERIN, 1995). Isto faz com que qualquer técnica que venha a ser utilizada na solução do problema da recarga precisa ter bom desempenho em espaços de busca multi-modais.

Tais atributos, adicionados ao alto número de combinações possíveis e o alto custo computacional para a avaliação da função objetivo, fazem com que o problema da recarga coloque em prova os métodos de otimização tradicionais, estimulando assim o desenvolvimento de métodos "inteligentes" para a solução deste problema.

O objetivo principal desta nossa investigação da recarga de combustível nuclear consiste em maximizar a duração do ciclo de operação, desconsiderando-se a presença de veneno queimável, através de uma estratégia de fuga desprezível. Esta maximização do ciclo pode ser obtida através da maximização da concentração crítica de boro (C_B) no final do ciclo. Entretanto, com o intuito de reduzirmos o custo computacional, utilizamos a concentração de boro no equilíbrio do xenônio. Como restrição

operacional, imposta por especificações técnicas, o fator de pico de potência radial (F_{XY}) não pode ultrapassar 1,435 (limite superior).

Para testar os algoritmos propostos, propomos as seguintes simplificações para o problema: i) não existe restrição para o posicionamento dos elementos; ii) não existem venenos queimáveis; e iii) não existe rotação de elementos combustíveis.

Os cálculos de física de reatores foram feitos utilizando-se o código RECNOD (CHAPOT, 2000), rodando na mesma plataforma computacional utilizada pelo PSO. Devido a uma série de limitações do código, a restrição $F_{XY}<1,435$ (F_{XY} não é fornecido pelo RECNOD) foi substituída por $P_{MAX}<1,395$, onde P_{MAX} é a potência máxima normalizada do arranjo de varetas combustíveis. O valor 1,395 no RECNOD garante na prática $F_{XY}<1,435$. A função objetivo, *f,* a ser otimizada é dada na Equação 16.

$$f = \begin{cases} C_B, & P_{MAX} < 1,395 \\ C_B - k.P_{MAX}, & \text{caso contrário} \end{cases} \tag{16}$$

Onde C_B é a concentração de Boro, P_{MAX} é a potência máxima normalizada do arranjo de varetas combustíveis e *k* o fator de penalização, com o valor igual a 5000, proveniente de estudos de sensibilidade anteriores (WAINTRAUB et al., 2006).

A associação de elementos combustíveis (ECs) às posições do núcleo é um problema combinatório, que, guardadas as devidas proporções, podemos mapear como

um clássico Problema do Caixeiro Viajante (PCV), no qual as técnicas de otimização foram testadas com sucesso.

5. Aplicação da Metodologia e Resultados

Inicialmente foi feito um levantamento bibliográfico sobre os modelos paralelos de PSO (estado-da-arte). Observou-se predominantemente modelos mestre-escravo (onde apenas a avaliação de cada partícula é paralelizada) (SCHUTTE *et al*., 2004) e modelos de malha grossa ou de ilhas (onde sub-populações são divididas e alocadas em diferentes processos) (CHANG *et al.*, 2005). O modelo de Von Neumman no qual é inspirado um dos modelos propostos, o PPSO-Celular, aparece apenas em investigações muito simplistas e sem aplicações práticas (apenas funções *benchmark* são utilizadas) (PEER *et al*., 2003; KENNEDY *et al*., 2006) e sem paralelismo computacional.

Numa primeira etapa de desenvolvimento, os modelos propostos do PSO foram implementados e testados com *benchmarks*. Em seguida, foram aplicados aos problemas da engenharia nuclear propostos. Os principais resultados desta publicação são apresentados em Waintraub et al. (2009).

Para todos os modelos apresentados neste trabalho, foram utilizados alguns parâmetros típicos do PSO, extraídos da literatura (SHI e EBERHART, 1998 e TRELEA, 2003) e outros ajustados através de experimentos preliminares.

Deve-se ressaltar que este PSO não-paralelo já possui seus parâmetros otimizados e pode ser considerado como uma das melhores metaheurísticas de otimização dentre as atuais, batendo as até então utilizadas técnicas de computação evolucionária, como os AGs (DOMINGOS *et al*., 2006; WAINTRAUB *et al*., 2006; PEREIRA *et al*., 2007; MENESES *et al*., 2008; MEDEIROS *et al*., 2008).

5.1. Parâmetros dos PPSO

KENNEDY e EBERHART (1995) propuseram que os coeficientes cognitivo e social, C1 e C2 respectivamente, fossem definidos de forma que C1 = C2 = 2, a fim de proporcionar uma valor médio igual a 1 (quando multiplicados pelos números randômicos r_1 e r_2). O resultado desta proposição faz com que as partículas extrapolem o alvo em metade do tempo do processo de otimização (SCHUTTE e GROENWOLD, 2003).

Introduzida inicialmente por SHI e EBERHART (1998), a constante de inércia *w* constitui-se na variante mais significativa do PSO original. Analogamente a introdução de um "*momentum*" à partícula, valores mais altos de *w* resultam em trajetórias da partícula com significativa extrapolação ou "sobrevôo" do objetivo, resultando em uma característica de procura global boa. Já valores mais baixos de *w* resultam em trajetórias de partícula irregulares com uma redução no "sobrevôo", característica desejável para uma procura local refinada.

A desvantagem mais relevante da introdução da constante de inércia é o problema da dependência de *w*. Numa implementação típica, seleciona-se um valor de intermediário para *w*, resultando em uma procura que é não ótima durante ambas as fases de busca, global e local.

Em dois trabalhos publicados, SHI e EBERHART (1998a e 1998b) propuseram também uma forma eliminar esta desvantagem da constante de inércia. Nesta variante

do PSO, o parâmetro de inércia *w* decresce linearmente durante a busca, em um número especificado de avaliações da função objetivo. Isto assegura que o PSO transite gradualmente de um algoritmo satisfatório em uma busca global (exploração) para um algoritmo satisfatório em refinar a procura de um ótimo em uma busca local (prospecção). A taxa ótima por reduzir *w* ainda é dependente do problema, e constitui a desvantagem principal desta variante.

Outra variante importante no PSO consiste em limitar a velocidade máxima V_{MAX} de cada partícula. A aplicação deste parâmetro representa uma tentativa de reduzir passos excessivamente grandes na movimentação do enxame. Neste trabalho, baseamos a investigação deste parâmetro em função da norma euclidiana V_N do espaço de busca, ou seja, V_{MAX} como fração de V_N.

Foram feitas investigações em torno da inércia e dos coeficientes cognitivo e social. Apresentamos a seguir os resultados obtidos dos testes realizados com diferentes números de partículas, velocidades máximas (utilizada como restrição) e número de iterações.

Em todos os experimentos apresentados aqui, foi utilizado um enxame de 96 partículas. As distribuições das sub-populações estão detalhadas na tabela 2.

Tabela 2– Topologias e distribuições das sub-populações

PSO	TOPOLOGIA
PSO Padrão	População única (96 partículas)
PPSO Mestre-Escravo	População única (96 partículas)
PPSO-Celular	Cellular: grade 2-D (96 partículas)
PPSO-N2	6 ilhas de 16 partículas
PPSO-N4	12 iilhas of 8 partículas
PPSO-Ilhas-[Intervalo de Migração]	6 ilhas of 16 partículas

5.2. Plataforma Computacional Paralela Utilizada

O processamento paralelo é geralmente destinado à resolução de problemas que demandem grande poder computacional. As máquinas de memória distribuída têm sido bastante utilizadas nestes casos, oferecendo alto desempenho e alta disponibilidade muitas vezes a um preço bastante inferior aos computadores paralelos de memória compartilhada.

Um *cluster* pode ser visto como um sistema de processamento paralelo ou distribuído trabalhando como um recurso computacional único e integrado, embora seja composto por um conjunto de computadores distintos. Devido a características como alto desempenho, facilidade de atualizações, utilização de *hardware* e *software*

"abertos", independência de fornecedores e licenças de importação e custo financeiro relativamente baixo, estes sistemas têm sido objeto de estudo e merecido destaque tanto no meio acadêmico como no empresarial. A construção de aplicações paralelas e distribuídas para serem executadas sobre *clusters* de estações de trabalho ou microcomputadores (PCs) tem sido estimulada também devido também a sua escalabilidade, que facilita a inserção e remoção de nós de processamento.

O Laboratório de Computação Paralela do IEN segue a filosofia dos sistemas de computação paralela e distribuída do tipo *Beowulf*. 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 física em estrela através de uma rede Gigabit Ethernet de 1000 Mb/s. O sistema operacional utilizado é o Fedora Linux.

5.3. Validação Preliminar

Foram feitas algumas validações antes de aplicarmos os modelos de PPSO propostos em problemas de engenharia nuclear.

5.3.1. Funções Numéricas

Como testes preliminares utilizamos funções numéricas unimodais e multimodais encontradas na literatura (PEER *et al.*, 2003) e apresentadas na tabela 3. Para fins de comparação, fixamos o número de partículas e o número máximo de iterações em 20 e 2000, respectivamente. Devido ao número reduzido de partículas, não aplicamos estas funções aos modelos PPSO-N4 e PPSO-Ilhas. Os outros modelos de PPSO (PPSO-Celular e PPSO-N2) encontraram resultados tão bons quanto aqueles obtidos com o PSO-Padrão, conforme pode ser observado nas tabelas 4 a 7.

Tabela 3– Funções de teste unimodais e multimodais.

Nome	Função	Limites
F1 (Rosenbrock)	$f_1(x_i) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$	$-2{,}048 < x_i < 2{,}048$
F2 (*Rastrigin*)	$f_2(x_i) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$-5{,}12 < x_i < 5{,}12$
F3 (*Schwefel*)	$f_3(x_i) = 418{,}9829n + \sum_{i=1}^{n} x_i sen\left(\sqrt{\lvert x_i \rvert}\right)$	$-500 < x_i < 500$
F4 (*Griewangk*)	$f_4(x_i) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-600 < x_i < 600$

A função de teste F1 é unimodal (isto é, contém apenas um ótimo), enquanto que as outras três (F2 a F4) são multimodais (isto é, contém muitos ótimos locais, mas somente um ótimo global).

A função de *Rosenbrock* (F1) é não convexa e possui um mínimo global localizado dentro um longo e estreito vale, cuja superfície tem o formato de uma parábola. Achar o vale é trivial, porém convergir para atingir o mínimo global é mais difícil. O seu mínimo global é $f(x^*) = 0, \quad para\ x^* = (1,1)$, e o seu vale pode ser visto na figura 13.

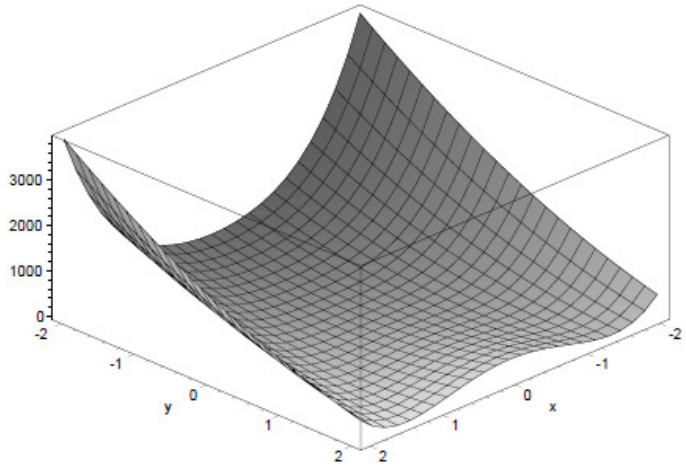


Figura 13 – Gráfico do vale da função de *Rozenbrock*.

As funções de *Rastrigin, Schwefel* e *Griewangk* (F2-F4) são exemplos típicos de funções multimodais. A função de *Rastrigin* contém milhões de mínimos locais no intervalo considerado. Estes mínimos locais estão localizados em um grade retangular de tamanho 1. O mínimo global é: $f(x^*) = 0, \quad para\ x^* = (0,0,....,0)$, e apresentamos na figura 14 o seu gráfico em 2D.

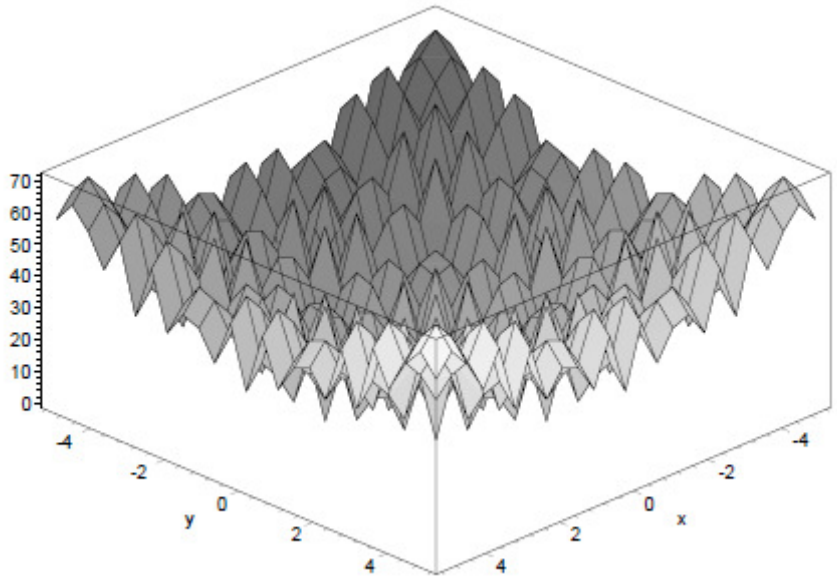


Figura 14 – Gráfico 2D da função de *Rastrigin*.

A função de *Schwefel* se caracteriza por possuir um segundo melhor mínimo muito afastado do ótimo global, devido ao espaço de busca muito extenso. O seu mínimo global é: $f(x^*) = 0, \quad para\ x^* = (-420{,}9687, -420{,}9687, ..., -420{,}9687)$, e o seu gráfico 2D encontra-se esboçado na figura 15.

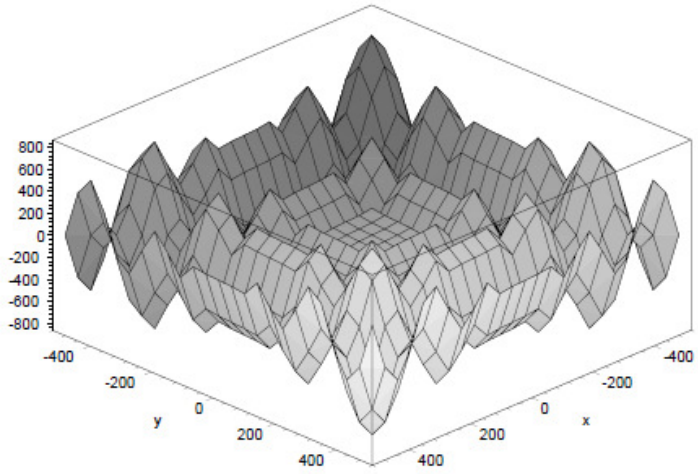


Figura 15 – Gráfico 2D da função *Schwefel*.

A função de *Griewangk* conjuga a grande quantidade de mínimos locais com um extenso espaço de busca. O seu mínimo global é: $f(x^*) = 0, \quad para\ x^* = (0,0,....,0)$, e o seu gráfico 2D pode ser visto na figura 16.

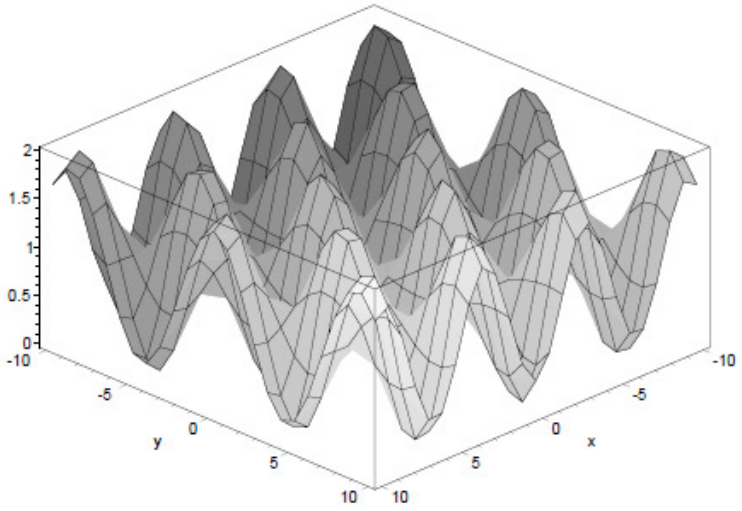


Figura 16 – Gráfico 2D da função *Griewangk.*

Tabela 4– Resultados obtidos pelos modelos de PSO com 20 partículas para a função *Rosenbrock* com dimensão n=2.

	f_{ave} **(valor médio)**	f_{min} **(valor mínimo)**	f_{max} **(valor máximo)**
PSO-Padrão	2,8840e-17	1,1093e-31	2,7299e-15
PPSO-Celular	7,2598e-11	1,2759e-17	2,2970e-09
PPSO-N2	1,0946e-13	1,0946e-23	1,0461e-11
Pg (Peer et al 2003)	0,0057	0,0004	4.3011
Pl (Peer et al 2003)	0,4212	0,1613	2,9975
Pv (Peer et al 2003)	0,1365	0,0341	0,515

Tabela 5– Resultados obtidos pelos modelos de PSO com 20 partículas para a função *Rastrigin* com dimensão n=30.

	f_{ave} **(valor médio)**	f_{min} **(valor mínimo)**	f_{max} **(valor máximo)**
PSO-Padrão	21,421	15,919	23,879
PPSO-Celular	18,904	9,950	19,899
PPSO-N2	20,440	20,894	23,879
Pg (Peer et al 2003)	68,65	29,85	136,31
Pl (Peer et al 2003)	63,68	31,84	112,43
Pv (Peer et al 2003)	51,74	22,88	98,50

Tabela 6– Resultados obtidos pelos modelos de PSO com 20 partículas para a função *Schwefel* com dimensão n=30.

	f_{ave} **(valor médio)**	f_{min} **(valor mínimo)**	f_{max} **(valor máximo)**
PSO-Padrão	333,6	0	1046,2
PPSO-Celular	450,3	0	1066,0
PPSO-N2	532,2	0	1223,9
Pg (Peer et al 2003)	4510,8	2803,1	6179,5
Pl (Peer et al 2003)	4609,5	2329,4	6219,5
Pv (Peer et al 2003)	4333,1	2664,9	5449,1

Tabela 7– Resultados obtidos pelos modelos de PSO com 20 partículas para a função *Griewangk* com dimensão n=30.

	f_{ave} **(valor médio)**	f_{min} **(valor mínimo)**	f_{max} **(valor máximo)**
PSO-Padrão	0,0190	0	0,1179
PPSO-Celular	0,0166	0	0,1220
PPSO-N2	0,0003	0	0,0262
Pg (Peer et al 2003)	0,0405	2e-19	2,1642
Pl (Peer et al 2003)	1e-19	0	0,0394
Pv (Peer et al 2003)	0,0074	0	0,0903

Com os resultados obtidos pudemos notar que, apesar de melhores que os encontrados na literatura, os ganhos com o paralelismo não ficaram evidentes devido ao número reduzido de partículas.

5.3.2. Um Problema do Caixeiro Viajante Assimétrico

Em seguida, de forma a aumentar a complexidade dos testes, os PPSO são submetidos a um clássico Problema do Caixeiro Viajante (PCV) assimétrico com 48 cidades – denominado Rykel-48 (TSPLIB, 2005). Como este problema é combinatorial e o (tradicional) PSO lida melhor com variáveis contínuas, é necessário uma transformação do espaço combinatorial para o contínuo (MENESES *et al*., 2008). Como resultado, obtemos um espaço não linear e multimodal com 48 dimensões, no qual cada dimensão varia de 0 a 1. Foram duas as motivações que nos levaram a escolha do PCV: i) proporcionar um grande desafio para o PSO (que não é exatamente um especialista em problemas combinatórios) e ii) o problema da recarga de combustível nuclear utiliza a mesma abordagem para codificar e manipular os candidatos à solução.

O Problema do Caixeiro Viajante (PCV), ou Traveling Salesman Problem (TSP), é um caso típico de otimização combinatória, freqüentemente utilizado em computação para demonstrar problemas de difícil resolução. O PCV caracteriza-se por, dado um conjunto de n cidades e uma matriz de distâncias [n, n], fazer com que seja encontrado um caminho que tenha a menor distância a ser percorrida para que sejam visitadas todas as cidades passando exatamente uma única vez em cada cidade e retornando a cidade de

origem. Esse problema é NP-difícil, em relação a sua classe de complexidade computacional, e também NP-completo, pois não tem solução determinística polinomial.

O PCV pode ser simétrico, quando a distância entre duas cidades é a mesma independentemente da direção, ou assimétrico, quando as distâncias entre duas cidades dependem se os trajetos são percorridos num ou noutro sentido. Para comparação entre os PPSO propostos, escolhemos um PCV assimétrico com 48 cidades – denominado Rykel-48 (TSPLIB, 2005).

Tendo em vista que no problema do PCV não são permitidas cidades repetidas, adotamos um processo de codificação para os candidatos a solução baseado em Chaves Aleatórias, RK (do inglês *Random Keys*) (BEAN, 1994), que já demonstrou muito eficiente (MACHADO, 1999 e MENESES *et al*., 2008) na solução de problemas combinatórios. A codificação da lista de cidades (rota) através de chaves randômicas se dá como a seguir.

A invés de representar a lista de cidades diretamente, o vetor $\vec{x}$ (posição da partícula) é formado por RKs, que são números reais entre 0 e 1. A decodificação do vetor $\vec{x}$ em uma lista de cidades (rota) válida é obtida através da ordenação do mesmo em relação ao seu índice (posição no vetor, que representa o identificador da cidade). Para maior clareza figura 17 ilustra tal decodificação, para uma lista hipotética de 5 cidades.

Observe que o menor valor de RK (0,15) aparece na posição 3. Isso significa que a primeira cidade a ser visitada é a cidade 3. O segundo menor valor de RK (0,26) aparece na posição 4, logo, a segunda cidade a ser visitada é a cidade 4, e assim por diante.

Chaves Aleatórias ($\vec{x}$) →	0,95	0,32	0,15	0,26	0,73
Lista de Cidades →	1	2	3	4	5
Rota decodificada →	3	4	2	5	1

Figura 17 – Decodificação de um candidato a solução pelo processo de Chaves Aleatórias.

Foram testadas diferentes combinações de parâmetros. Para cada uma, foram feitos 100 experimentos com diferentes sementes randomizadas, resultando em 11.200 experimentos (rodadas do PSO). Ambos os coeficientes, cognitivo e social (C1 e C2), foram definidos com o valor 2.0, conforme recomendado na literatura (SHI e EBERHART, 1998; TRELEA, 2003). Com o intuito de proporcionar bom balanceamento entre a exploração (maior busca global nas iterações iniciais) e prospecção (maior busca local nas últimas iterações), fizemos o coeficiente de inércia decrescer linearmente de 0,8 até 0,2 durante 50.000 iterações (número máximo de iterações).

Ao longo dos 11.200 experimentos, pudemos observar que a limitação da velocidade máxima das partículas em valores “pequenos” (relativamente ao espaço de

busca), tem um efeito positivo nos modelos que utilizam diferentes estratégias de comunicação para o PPSO, enquanto que para o PPSO-Padrão (mestre-escravo) produz um efeito negativo. Isto pode ser explicado pela melhoria introduzida pelas estratégias de comunicação utilizadas na diversidade dos enxames de partículas. Conseqüentemente, testamos diferentes valores para a velocidade máxima (V_{MAX}). Considerando-se V_N como a maior distância entre 2 pontos no espaço de busca (que neste problema V_N=6,92), os seguintes valores foram testados: $V_N/1$, $V_N/2$, $V_N/4$, $V_N/8$.

Para o PPSO-Padrão, o melhor valor encontrado foi $V_{MAX} = V_N/2$, que foi 4 vezes maior que o melhor valor para os outros PPSOs, $V_{MAX} = V_N/8$. Os melhores resultados para cada modelo encontram-se listados na tabela 8.

Tabela 8– Resultados obtidos para o PCV tipo Rykel-48

PSO	f_{ave} (valor médio)	**DP (%)** (desvio padrão)	f_{min} (vaor mínimo.)	f_{max} (valor máximo)	***F*<15000**
PSO Padrão	17124	8,37	14658	21798	3
PPSO Mestre-Escravo	17124	8,37	14658	21798	3
PPSO-Celular	15477	2,21	14765	16194	5
PPSO-N2	15248	2,40	14572	16389	29
PPSO-N4	15301	2,33	14698	16360	22
PPSO-Ilhas-10	15594	3,18	14709	17258	7
PPSO-Ilhas-100	15459	2,79	14806	17575	7
PPSO-Ilhas-1000	15398	2,15	14739	16456	8

Pelos resultados apresentados na tabela 8, pode-se notar que todos os modelos de PPSO foram capazes de encontrar bons valores para f_{min} em 100 casos rodados. Entretanto, observando-se as médias e os desvios padrão, os PPSO de malhas grossa e fina se mostram melhores e com maior robustez. Além disso, ambos os PPSO de ilhas com vizinhança demonstraram ser os melhores dentre todos os outros, com uma expressiva percentagem de valores acima de 15.000 (resultados que consideramos muito bons, pois o ótimo é 14422). Conseqüentemente, podemos concluir que as estratégias de comunicação adotadas nos PPSO de ilhas com vizinhança demonstram vantagens sobre os tradicionais modelos de ilhas e celular.

Como a função objetivo deste problema consome tempo de processamento desprezível, não computamos o custo computacional deste benchmark.

As curvas de convergência, apresentadas na figura 18, mostram como o freio na disseminação do *gBest* nos modelos PPSO-N2 e PPSO-Ilhas, faz com que o processo de otimização vá evoluindo mais lentamente e ultrapasse o PPSO-Mestre-Escravo. O PPSO-Celular e o PPSO-N4, por possuirem interface com uma maior quantidade de ilhas, possuem um padrão de convergência mais acelerado, porém, mantendo a diversidade na busca, o que os leva, também, a resultados melhores que o PPSO-Mestre-Escravo.

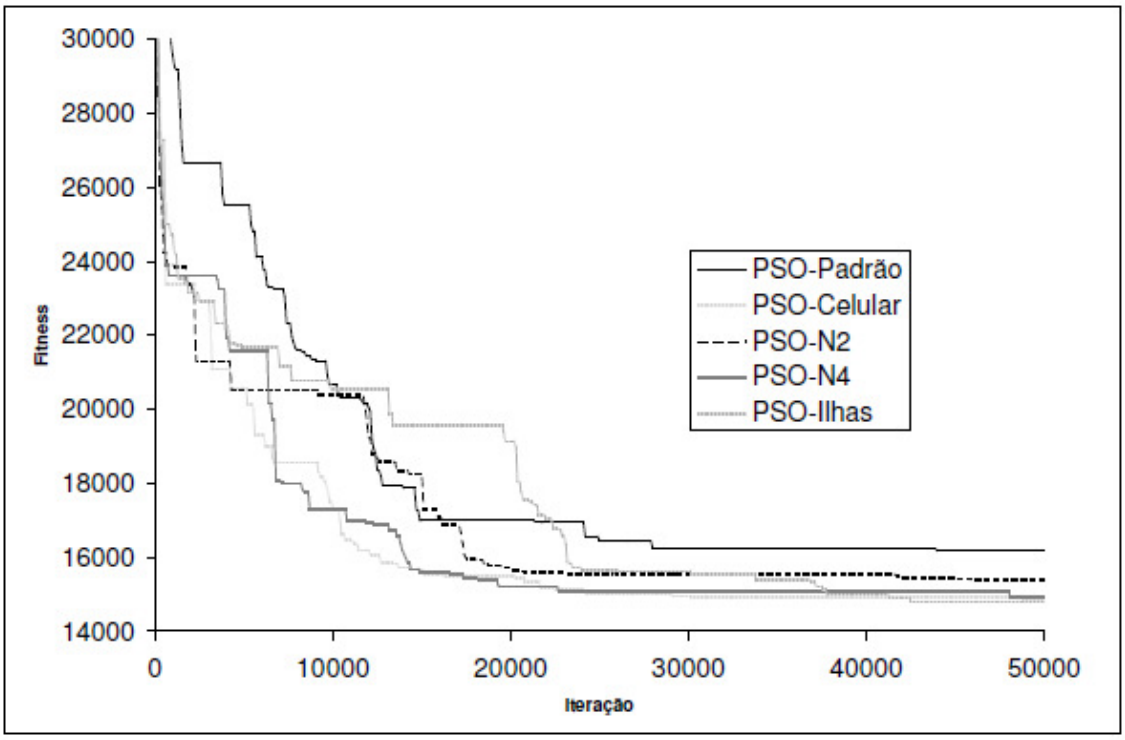


Figura 18 – Gráfico de convergência dos modelos de PPSO.

5.4.Aplicação no Projeto Neutrônico de um Reator Nuclear

Os resultados apresentados na sessão anterior serviram de incentivo para a aplicação dos modelos de PPSO propostos em um problema de otimização de projeto neutrônico de um reator nuclear. Este problema, entretanto, consome um tempo de processamento significativamente maior (um único caso de PSO não-paralelo leva em torno de 10 horas de processamento), razão pela qual tivemos que limitar a um número menor de experimentos realizados.

5.4.1. A Modelagem do Candidato à Solução

A codificação da partícula para este problema de otimização, conforme apresentado na figura 19, é feita através da associação direta do vetor $\vec{x}$ com os oito parâmetros de projeto listados na tabela 1.

Vetor de posição da partícula ($\vec{x}$) →	$x[0]$	$x[1]$	$x[2]$	$x[3]$	$x[4]$	$x[5]$	$x[6]$	$x[7]$
Parâmetros de projeto →	R_f	Δ_c	R_e	E_1	E_2	E_3	M_f	M_c

Figura 19 – Codificação do vetor $\vec{x}$ para o problema de otimização do projeto neutrônico de um reator nuclear.

5.4.2. Experimentos e Resultados

Baseados em observações feitas nos experimentos com o PCV, bem como nos experimentos preliminares com um problema simplificado (de custo computacional reduzido) de projeto de um reator nuclear, definimos o conjunto de parâmetros do PSO. Utilizamos um enxame de 96 partículas. Os coeficientes: cognitivo e social, C1 e C2 respectivamente, foram fixados com o valor 2,0. O peso inercial decresce linearmente de 0,8 até 0,2 durante 10.000 iterações.

O melhor resultado para o PSO-Padrão foi encontrado com $V_{MAX} = V_N$, que é 4 vezes maior que aquele verificados para os outros PPSOs, $V_{MAX} = V_N / 4$.

Surpreendentemente, este problema se mostrou fácil para o PSO. Mesmo o PSO tradicional (não paralelo) foi capaz de encontrar o (provável) valor ótimo (f=1,688). Entretanto, os PPSO de malhas Grossas e finas foram mais consistentes ao encontrar valores bem próximos do ótimo (>1,68), enquanto que tanto o PSO tradicional quanto o PPSO mestre-escravo caíram em ótimos locais em três experimentos. A tabela 9 apresenta os resultados obtidos em 10 experimentos de cada um dos modelos de PPSO.

Tabela 9– Resultados do problema de otimização do projeto neutrônico de um reator nuclear.

PSO	f_{ave} (valor médio)	**DP(%)** (desvio padrão)	f_{min} (valor mínimo)	f_{max} (valor máximo)
PSO Padrão	1,6793	0,82	1,647	1,688
PPSO Mestre-Escravo	1,6793	0,82	1,647	1,688
PPSO-Celular	1,6863	0,17	1,678	1,688
PPSO-N2	1,6859	0,07	1,683	1,688
PPSO-N4	1,6872	0,03	1,686	1,688
PPSO-Ilhas-10	1,6871	0,02	1,687	1,688
PPSO-Ilhas-100	1,6862	0,05	1,685	1,688
PPSO-Ilhas-1000	1,6865	0,04	1,685	1,688

Onde f é o fluxo x10^{-4} neutrons/cm^2.seg.

Neste problema, as diferenças observadas entre os modelos não foram tão acentuadas quantos aquelas obtidas no benchmark do PCV. Na verdade, embora o espaço de busca seja não-linear e multimodal, talvez devido ao pequeno número de

variáveis, este problema não aparenta ser muito complexo. Não obstante, tais resultados ratificam as conclusões obtidas nos experimentos do PCV. Considerando-se os experimentos realizados, temos as seguintes observações:

i) Os modelos de malhas grossas e finas do PPSO demonstraram ser melhores que o mestre-escravo e o padrão;

ii) Apesar do PSO-Padrão apresentar um desempenho muito bom, ele teve 30% de convergência prematura, demonstrando menor robustez.

O tempo consumido por cada execução encontra-se apresentado na tabela 10. Deve-se enfatizar que forma investigadas apenas implementações síncronas e as operações de leitura/escrita (manipulação de arquivos em disco) do simulador não foram alteradas, ou seja, o simulador foi utilizado na sua versão original.

Tabela 10 – Tempos de execução médios para cada um dos modelos de PPSO no problema de otimização do projeto neutrônico de um reator nuclear.

PSO	t***(min)***	***Ganho (***t***padrão /*** t***)***
PSO-Padrão	443	1
PPSO Mestre-Escravo	212	2,23
PPSO-Celular	225	2,10
PPSO-N2	113	4,18
PPSO-N4	110	4,30
PPSO-Ilhas-1000	108	4,38

Conforme esperado, os modelos paralelos apresentaram ganhos significativos. Pode-se observar também que os modelos Mestre-Escravo e Celular são grandes consumidores de tempo computacional devido ao *overhead* de comunicação.

Além disso, apesar de 6 processos serem utilizados, os maiores ganhos ficaram em torno de 4 vezes (nos modelos de ilhas e ilhas com vizinhança). Isto aconteceu devido ao tempo relativamente reduzido de processamento necessário para cada avaliação dos candidatos a solução (em torno de 1,5 segundos) e a implementação síncrona (na prática os processos não levam exatamente o mesmo tempo de processamento).

Uma outra observação que podemos tirar é que, embora o modelo Mestre-Escravo apresente um *overhead* (acréscimo no tempo gasto pelo sistema na transferência de dados para o processo destino) devido ao processo mestre (que centraliza o *loop* principal do PSO), o PPSO-Celular é ligeiramente mais lento. Tal fato pode ser atribuído ao overhead de comunicação existente no PPSO-Celular, que torna-se relevante para avaliações com custo computacional relativamente baixo como neste caso.

Além do ganho computacional proporcionado pelos modelos de PPSO propostos, precisamos observar que soluções ótimas ou sub-ótimas foram obtidas. Os modelos: celular, ilhas e ilhas com vizinhança, convergiram sempre para soluções semelhantes, ou seja, configurações de núcleos com os mesmos materiais, variando apenas as dimensões e enriquecimentos. O modelo de PSO-Padrão, ou PPSO-Mestre-Escravo, no entanto, apresentou algumas soluções de sub-ótimos locais com materiais

diversos, conforme ilustrado na tabela 11, onde apresentamos a pior e a melhor solução encontrada por este modelo tradicional.

Este fato demonstra uma menor robustez do modelo Mestre-Escravo, ou seja, uma maior susceptibilidade para ficar "preso" em ótimos locais.

Tabela 11– Melhor e pior soluções encontradas pelo PPSO-Mestre-Escravo.

Parâmetro	Melhor Solução	Pior Solução
Raio do combustível (pol.)	0,4247	0,2592
Espessura do encamisamento (pol.)	0,2000	0,1539
Espessura do moderador (pol.)	0,4182	0,2675
Enriquecimento da zona 1 (%)	3,27	2,95
Enriquecimento da zona 2 (%)	3,59	5,22
Enriquecimento da zona 3 (%)	5,97	5,40
Material do combustível	UO2	UO2
Material do encamisamento	Zircaloy-2	Alumínio

5.5.Aplicação no Problema de Otimização da Recarga de Combustível Nuclear

Como o problema proposto de otimização do projeto neutrônico de um reator nuclear ainda foi relativamente fácil para o PSO, escolhemos então um problema mais realístico e por conseqüência mais custoso em termos de computacionais. O problema

de otimização da recarga de combustível nuclear atende aos dois requisitos, aumentando tanto a complexidade quanto o tempo de processamento necessário para a avaliação da função objetivo.

5.5.1. A modelagem do Candidato à Solução

Uma forma de codificação e decodificação do genótipo, utilizando-se *Random Keys* é exemplificada na Figura 20. Da mesma forma que ocorre no PVC, o vetor de RK é ordenado e conjuntamente com ele, a lista de elementos. A posição dos elementos é o índice do vetor de elementos.

Observando a Figura 20(a), tem-se que o menor valor de RK refere-se ao elemento 20, portando este deve ser inserido na primeira posição. O segundo menor valor de RK é 0,15, referente ao elemento 3, que será inserido na segunda posição. Utilizando-se este algoritmo de ordenação, obtêm-se, finalmente o vetor ordenado de RK, que indica a posição de inserção de cada um dos elementos, como na Figura 20(b).

Desta forma, o genótipo exemplificado na figura 20(a), ou seja, o vetor de RK, refere-se ao 1/8 de núcleo exibido na Figura 21.

Posição →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
RK ($\vec{x}$) →	0,95	0,32	0,15	0,18	0,73	0,20	0,45	0,33	0,88	0,50	0,91	0,75	0,17	0,60	0,22	0,39	0,40	0,57	0,66	0,01
Elementos →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

(a)

Posição →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
RK ($\vec{x}$) →	0,01	0,15	0,17	0,18	0,20	0,22	0,32	0,33	0,39	0,40	0,45	0,50	0,57	0,60	0,66	0,73	0,75	0,88	0,91	0,95
Elementos →	20	3	13	4	6	14	2	8	16	17	7	10	18	14	19	5	12	9	11	1

(b)

Figura 20 – Decodificação de um candidato a solução pelo processo de Chaves Aleatórias: (a) antes da ordenação; (b) depois da ordenação.

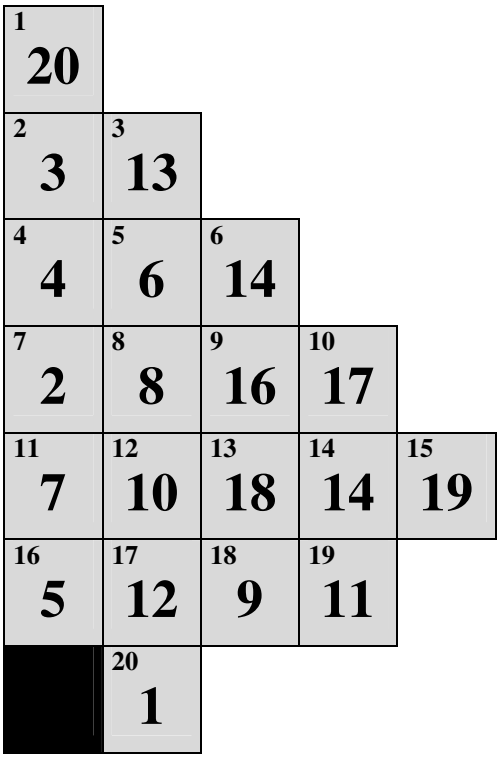


Figura 21 – Núcleo do reator tipo PWR Angra-1, com simetria de um-oitavo e cada elemento (número central) inserido na respectiva posição (canto superior esquerdo).

5.5.2. Experimentos e Resultados

Com isso, como um simples cálculo não paralelo de otimização com o PSO-Padrão leva 30 horas, fizemos somente 5 experimentos para cada modelo de PSO (e PPSO). O tamanho do enxame, as constantes C1 e C2, assim como o peso inercial foram os mesmos do projeto neutrônico. O melhor valor para o PSO-Padrão foi $V_{MAX} = V_N$, que foi 8 vezes maior que o melhor valor para os outros PPSOs, $V_{MAX} = V_N /8$. O peso inercial decresce linearmente de 0,8 até 0,2 durante 10.000 iterações. A tabela 12 mostra os resultados obtidos nestes experimentos.

Tabela 12– Resultados para o problema de otimização da recarga de combustível nuclear.

PSO	f_{ave} (valor médio)	f_{min} (valor mínimo)	f_{max} (valor máximo)
PSO-Padrão	1411	1370	1432
PPSO Mestre-Escravo	1411	1370	1432
PPSO-Celular	1433	1409	1467
PPSO-N2	1421	1392	1447
PPSO-N4	1448	1416	1540
PPSO-Ilhas-100	1436	1409	1455
PPSO-Ilhas-1000	1444	1402	1557

Onde *f* é concentração de boro em PPM.

Conforme esperávamos, Os modelos do PPSO de malhas grossas e finas demonstraram ser melhores que os modelos PSO-Padrão e PPSO mestre-escravo. Aqui,

os modelos PPSO-N4 e PPSO-Ilhas-1000 apresentaram, não apenas os melhores valores médios, como também excelentes valores máximos (>1500).

Os tempos de processamentos consumidos por cada modelo encontram-se na tabela 13. Assim como no último problema, foram investigadas somente implementações síncronas e as operações de leitura/escrita do simulador não foram alteradas (o simulador foi utilizado na sua versão original).

Tabela 13– Tempos de execução médios para cada PPSO no problema de otimização da recarga de combustível nuclear.

PSO	$t(min)$	*Ganho* ($t_{padrão} / t$)
PSO-Padrão	1735	1
PPSO Mestre-Escravo	621	2,79
PPSO-Celular	588	2,95
PPSO-N2	357	4,86
PPSO-N4	323	5,37
PPSO-Ilhas-1000	334	5,20

Novamente, ganhos significativos foram observados nos modelos paralelos. Confirmando os resultados apresentados na última sessão, pode ser visto que os modelos celular e mestre-escravo consomem muito mais tempo computacional devido ao overhead de comunicação.

Neste problema, observa-se uma "inversão" nos tempos esperados para os modelos PPSO-N2 e PPSO-N4. Analisando o problema, podemos inferir que um dos fatores que podem ter contribuído para este ganho do modelo PPSO-N4 se deve ao fato de as 12 ilhas (processos) estarem distribuídas em 6 núcleos (2 processos por processador de núcleo duplo) são executadas de maneira otimizada pelo sistema operacional, aproveitando a grande quantidade de operações de escrita/leitura em arquivos (disco rígido), efetuada pelo simulador neutrônico.

Notadamente, apesar dos tempos de processamento caírem drasticamente nos modelos paralelos, eles ainda permanecem altos para problemas reais, nos quais as simplificações aqui utilizadas não são aplicáveis. Por exemplo: caso se levasse em conta a presença de venenos queimáveis, a simplificação de se considerar a concentração de boro no equilíbrio de xenônio não poderia ser aplicada e a queima deveria ser simulada até o final do ciclo, aumentando o custo computacional neste trabalho em 4 ou 6 vezes. A utilização de códigos de física de reatores mais elaborados também aumentaria muito o tempo de simulação. Portanto, deve-se enfatizar que a resolução de problemas reais, muitas vezes precisará ser feita com poucas rodadas do processo de otimização. Por isso, aumentando-se a robustez da ferramenta de otimização, aumentam-se as chances de alcançar bons resultados numa única rodada.

Sob o ponto de vista econômico, a motivação para o aperfeiçoamento da ferramenta de otimização fica mais clara. Apenas para efeito de ilustração, façamos um cálculo aproximado dos ganhos para o reator da usina de Angra-1. Extrapolando-se a concentração de boro para 0 ppm, é possível calcular a duração do ciclo em termos de dias de operação sobre potência máxima efetiva (EFPD, do inglês *Effective Full Power*

Day). Para o melhor valor da tabela 5 (C_B=1557 ppm), a duração do ciclo é de aproximadamente 389 EFPD, enquanto que para o pior valor (C_B=1370 ppm) é de 343 EFPD. Considerando-se que 1 EFPD gera cerca de U$500.000 para a empresa proprietária da usina de Angra-1, a diferença entre o melhor e o pior (que também é um resultado muito bom) resultado do processo de otimização representa um montante de cerca de U$23.000.000 na operação da usina.

6. Conclusões

Neste trabalho foram desenvolvidos quatro modelos diferentes de PPSO e aplicados na solução de problemas complexos de otimização na engenharia nuclear. Conforme esperado, todos os modelos apresentaram ganhos significativos em termos de redução do custo computacional devido ao paralelismo. Devido a reduzida quantidade de comunicação necessária, os modelos de ilhas e ilhas com vizinhança mostraram-se mais rápidos.

Sob o ponto de vista dos resultados do processo de otimização, todos os modelos de PPSO com restrição de vizinhança, implementados através de elaboradas estratégias de comunicação entre processadores, superaram o modelo mestre-escravo.

No *benchmark* do Rykel-48 ambos os modelos de ilhas com vizinhança (PPSO-N2 e PPSO-N4) foram claramente melhores que todos os outros. A melhora da performance se deve ao aumento da habilidade de manter a diversidade durante o processo de busca, promovendo uma grande consistência em encontrar regiões próximas da solução ótima. O terceiro melhor modelo classificado foi o PPSO de ilhas com intervalo de migração igual a 1.000 (PPSO-Ilhas-1000).

Após as investigações com os *benchmarks*, aplicamos o PPSO em dois problemas clássicos da engenharia nuclear. O primeiro foi o de otimização do projeto neutrônico de um reator nuclear, no qual o modelo PPSO-N4 demonstrou ser levemente melhor, seguido de perto pelo PPSO-Ilhas-10. Surpreendentemente, este problema não proporcionou alto grau de dificuldade, inclusive para o PSO não-paralelo que, mesmo

apresentando desempenho ligeiramente pior, obteve resultados não muito diferentes dos demais modelos. Deve-se, mais uma vez, enfatizar que o PSO não-paralelo aqui desenvolvido possui seus parâmetros otimizados e pode ser considerado como uma das melhores metaheurísticas de otimização dentre as atuais, sobrepondo-se às até então utilizadas técnicas de computação evolucionária, como os AGs (DOMINGOS et al, 2006; WAINTRAUB et al, 2006; PEREIRA et al, 2007; MENESES et al, 2008; MEDEIROS et al, 2008).

Com o intuito de testar os modelos propostos em problemas mais realísticos, complexos e que tenham alto custo computacional, utilizamos a otimização da recarga de combustível da usina nuclear de Angra-1. Neste problema, os modelos PPSO-N4 e PPSO-Ilhas-1000 obtiveram os melhores resultados.

Os ganhos econômicos obtidos no problema de otimização da recarga de combustível nuclear justificam a busca por melhorias nas ferramentas de otimização e realçam a relevância da utilização dos modelos de PPSO.

Dentre os modelos propostos, os que apresentaram melhor desempenho em todos os problemas considerados foram os de Ilhas com vizinhança e o de Ilhas. Embora os modelos de ilhas tenham menor custo computacional (devido a quantidade de comunicação entre os processadores ser bem reduzida), seu desempenho é afetado pelo parâmetro “intervalo de migração”. A relação entre este parâmetro e o desempenho do modelo PPSO-Ilhas parece ser não trivial, ou seja, intervalos de migração pequenos funcionaram melhor no problema do Rykel-48, contrastando com as aplicações nucleares, nas quais intervalos de migração maiores obtiveram melhores resultados.

Resumindo, aqui, o modelo de ilhas com vizinhança do PSO (especialmente o PPSO-N4) e o de ilhas (com migrações periódicas) demonstraram ser melhores que o modelo celular e muito superiores ao mestre-escravo. O primeiro tem ainda a vantagem de eliminar o parâmetro "intervalo de migração", enquanto que o último tem um custo computacional menor.

Este trabalho é, provavelmente, um dos primeiros estudos investigativos nos quais modelos aprimorados de PPSO (com diferentes estratégias de comunicação/vizinhança) são aplicados a problemas reais de engenharia. Todavia, investigações de outras abordagens de PPSO com estratégias diferentes de comunicação, assim como aplicações em outros problemas reais, podem contribuir para o refinamento das evidências apontadas neste trabalho.

A dependência do PSO de parâmetros como a constante de inércia e velocidade máxima da partícula e a interdependência existente entre eles, constitui uma das maiores dificuldades da utilização deste tipo de algoritmo. Através da observação das curvas de convergência, pretendemos realizar estudos que nos permitam automatizar a parametrização destes algoritmos.

Como os algoritmos de PPSO apresentados neste trabalho utilizaram funções da biblioteca MPI que são bloqueantes, ou seja, aguardam a confirmação do processo de transferência da informação para só então efetuar nova transferência, e, além disso, realizam os processos de avaliação da função objetivo de forma síncrona. Propomos para futuros trabalhos a investigação da viabilidade de se utilizar estratégias de

comunicação não-bloqueantes e avaliações assíncronas, o que certamente proporcionará ganhos significativos na performance dos algoritmos propostos.

Pretendemos realizar estudos no sentido de viabilizar a aplicação dos modelos desenvolvidos na solução do problema da recarga nuclear multiciclo, visto que sem a implementação do paralelismo este problema se torna inviável em termos de custo computacional.

Outra proposta é a investigação de diferentes topologias físicas de rede, visando minimizar a sobrecarga de comunicação verificada principalmente nos modelos Celular e Mestre-Escravo.

APÊNDICE A

A1. Biblioteca de Interface de Troca de Mensagens – MPI

O MPI é uma 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.

Contextos de comunicação

O conceito de contexto permite particionar o espaço de comunicação, de forma que a operação de envio de uma mensagem possa ser direcionada a um grupo específico de processos, garantindo assim que não existam mensagens que sejam recebidas ambiguamente por grupos de processos não relacionados. Portanto, 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 comunicador, 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 comunicador. Isso criará contextos distintos que relacionarão os grupos de processos de cada aplicação e evitará que estes interfiram entre si.

A comunicação pode ser executada dentro de um único grupo de processos, utilizando um intra-comunicador, ou pode constituir uma comunicação ponto-a-ponto envolvendo dois grupos distintos (indicada pelo uso de um inter-comunicador). Podemos citar como exemplos de intra-comunicador pré-definidos pelo padrão o MPI_COMM_WORLD e o MPI_COMM_SELF, sendo a abrangência do primeiro todos os processos envolvidos na execução do programa MPI e a do segundo apenas o próprio processo.

Sincronização e Endereçamento

Na implementação do MPI, os detalhes de movimentação de dados costumam ser escondidos do programador em uma biblioteca de passagem de mensagens. Uma camada de software, inserida entre as camadas das primitivas de comunicação e o *hardware* do sistema, possibilita ao programador construir aplicações paralelas sem precisar se preocupar com pequenos detalhes envolvidos nas transferências de dados. Embora estas operações possam ser implementadas diretamente por hardware, suas características (como, por exemplo, "*bufferização*") são melhor tratadas por implementação de *software*. Assim, em todas as máquinas paralelas, o modelo de programação através de passagem de mensagens é realizado com uma camada de software construída sobre uma interface de comunicação mais simples. Primitivas mais básicas de transferência de dados podem ser utilizadas pala implementar tal interface,

em uma solução suportada diretamente por *hardware*. Outra forma de implementação desta interface seria considerarmos a adoção de um espaço de endereçamento virtual compartilhado, permitindo que as comunicações sejam realizadas através de operações de escrita e leitura em *buffers* (espaço de memória do computador para onde, e de onde, as mensagens são enviadas ou armazenadas) compartilhados e envolvam eventos de sincronização apropriados.

Um exemplo de sincronização implementada com a utilização de mecanismos de trocas de mensagens trata-se das operações *send/receive.* Cada par *send/receive* pode estabelecer um evento de sincronização em um determinado ponto do programa envolvendo os processos emissor e receptor. Operações de comunicação coletivas também podem permitir a sincronização de processos, dependendo da implementação realizada.

Em uma máquina de passagem de mensagens, um processador pode referenciar apenas endereços em sua memória local e cada um dos demais processadores. Assim, é permitido a um processo usuário acessar seus endereços privados e transferir dados usando primitivas de comunicação. Cada processo possui seu espaço de endereçamento privado, onde são realizadas as operações locais segundo a ordem de execução do programa. Devido ao fato de envolverem apenas dois processos, um enviando e outro recebendo dados, estas operações são chamadas ponto-a-ponto.

Modos de Comunicação

Uma comunicação *send/receive* pode ainda ser bloqueante (o retomo da primitiva indica que o usuário pode reutilizar com segurança os recursos especificados na chamada, tais como *buffers*) ou não-bloqueante (a primitiva pode retornar antes que a operação de comunicação seja completada e antes que o usuário possa reutilizar os recursos especificados na chamada). Considerando uma operação de envio, por exemplo, costumamos caracterizá-la como bloqueante se o processo emissor for impedido de prosseguir sua execução até que o *buffer* de dados utilizado para transmitir a mensagem possa ser reutilizado.

O modo de comunicação é determinado pela operação de envio. Existem quatro operações de envio com bloqueio e quatro operações de envio sem bloqueio, que correspondem aos quatro modos de comunicação existentes. A operação de recebimento não especifica o modo de comunicação, simplesmente é com bloqueio ou sem bloqueio. A diferença básica entre os modos de comunicação consiste no fato da operação de envio precisar ou não que um recebimento combinado seja executado, ou no fato dos dados da mensagem serem *bufferizados* pelo sistema ou só copiados pela tarefa receptora quando este estiver executando o recebimento.

O primeiro modo de comunicação pode incorrer em *overhead* de sincronismo (decorrente do tempo gasto pela espera do acontecimento de um evento em uma outra tarefa), já o segundo modo, pode incorrer em *overhead* de sistema (que acontece quando os dados da mensagem são copiados do *buffer* da tarefa emissora para a rede, e copiados da rede para o *buffer* da tarefa receptora).

A *bufferização* de mensagens pelo sistema desassocia as operações de envio e recebimento, uma vez que um envio com bloqueio pode se completar tão logo a mensagem seja *bufferizada*, mesmo que o receptor não tenha executado uma operação de recebimento combinada com esta. Por outro lado, a *bufferização* pode ser bastante cara, pois acarreta cópias e alocações de memória adicionais.

Primitivas de Comunicação

Uma mensagem pode ser dividida em duas partes: os dados sendo transmitidos/recebidos, e um envelope de informações que ajuda no direcionamento dos dados. Usualmente três parâmetros descrevem os dados (endereço, contador e tipo) e outros três especificam a rota dos dados (destino/origem, etiqueta e comunicador).

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.

Os parâmetros utilizados nas primitivas bloqueantes (*MPI_Send*, *MPI_Bsend*, *MPI_Rsend* e *MPI_Ssend*) são exatamente os mesmos. Em uma operação de comunicação ponto-a-ponto, o processo emissor envia um certo número de elementos (*count*) de um determinado tipo (*dtype*) para o processo receptor (*dest*). Os dados a serem enviados devem estar no *buffer* de envio *buf* e a variável *dest* identifica o receptor dentro do grupo de processos indicado pelo comunicador *comm*. Existe ainda um outro importante parâmetro, chamado *tag*, que pode ser usado como uma identificação da

mensagem transmitida. A seguir podemos visualizar melhor estes parâmetros em uma chamada da primitiva *MPI_Send*:

MPI_Send *(void * buf, int count, MPI_Datatype dtype,*
int dest, int tag, *MPI_Comm comm*)

Apresentamos a seguir uma descrição dos quatro modos de *send* bloqueantes (*standard*, *buffered*, *synchronous* e *ready*):

1) Envio com Bloqueio no modo padrão (*MPI_Send*):

O *send* padrão é modo de comunicação mais utilizado para o envio de uma mensagem entre dois processos MPI. A operação é dita bloqueante pois, ao ser executada, ela apenas poderá retornar após a mensagem ter sido armazenada com segurança, sendo então permitido ao processo emissor reutilizar o *buffer* de envio. A mensagem pode ser transmitida diretamente para o *buffer* do receptor ou simplesmente ser copiada em um *buffer* de sistema, sendo realmente transferida ao seu destino em um segundo momento. É de responsabilidade do MPI decidir quando os dados envolvidos em uma comunicação são ou não armazenados localmente. Por isso, podemos dizer que a primitiva de comunicação pode ou não bloquear dependendo de sua implementação.

O comportamento do *send* padrão não é definido precisamente pelo MPI. Assim, se pretendemos compreender o que realmente ocorre durante a transmissão de uma mensagem neste modo de comunicação, precisamos analisar as particularidades da implementação MPI utilizada. Após estudarmos o código da implementação LAM, verificamos que a primitiva *MPI_Send* pode ou não bloquear dependendo, dentre outros fatores, do tamanho da mensagem transmitida e do número de mensagens pendentes a serem recebidas pelo processo de destino. Assim, é possível que ao executarmos uma

primitiva MPI_Send em um programa LAM o emissor encerre a operação antes da transferência dos dados pela rede.

2) Envio com bloqueio em modo bufferizado (*MPI_Bsend*):

A principal diferença do *buffered send* em relação aos demais modos de send está no uso de um *buffer* de dados para onde a mensagem é copiada antes de ser transmitida. O usuário deve criar este *buffer* explicitamente, através de funções específicas disponibilizadas pelo MPI, antes de chamar a primitiva MPI_Bsend. É responsabilidade do usuário definir um *buffer* suficientemente grande para alojar a mensagem transmitida e cuidar para que este não seja reutilizado indevidamente, evitando a ocorrência de erros ou a sobreposição de dados.

O tempo despendido pelo processo emissor em uma comunicação no modo *buffered* tende a ser menor em relação aos demais *send*s bloqueantes. Isso acontece pois, ao contrário das operações *MPI_Send*, *MPI_Ssend* e *MPI_Rsend*, a ocorrência de uma primitiva *MPI_Bsend* em um programa não exige a existência de uma operação de recepção (*MPI_Recv*) para a sua correta finalização. A mensagem é simplesmente copiada para o *buffer*, de onde será transmitida ao respectivo receptor, e o processo emissor pode continuar sua execução normalmente.

Por outro lado, a necessidade da alocação deste *buffer* gera um *overhead* que pode ser significativo dependendo do tamanho da mensagem. Assim, caso o objetivo do

usuário ao utilizar o *MPI_Bsend* seja obter melhor desempenho, talvez uma alternativa melhor possa ser oferecida pelo send não-bloqueante.

3) *Synchronous send*

A operação *MPI_Ssend* pode ser inicializada com ou sem a ocorrência do respectivo *MPI_Recv*, mas somente poderá ser finalizada quando o receptor começar a receber a mensagem enviada. Desta forma, o MPI garante que ao final da execução da operação de envio o processo receptor atingiu um certo ponto de sua execução e o *buffer* do emissor pode ser reutilizado.

4) *Ready send*

No modo *ready* do *send* bloqueante a operação de comunicação somente terá início quando houver um receptor esperando a mensagem. Quando ocorre uma chamada da primitiva *MPI_Rsend*, o MPI verifica a existência da respectiva operação *MPI_Recv* e, caso a resposta seja negativa, a comunicação fica aguardando. Caso contrário, a transmissão da mensagem é estabelecida e uma vez encerrada a primitiva será finalizada. Convém notarmos a diferença entre os eventos de sincronismo estabelecidos pelos modos síncrono e *ready*. Enquanto no primeiro temos a transferência dos dados e em seguida a sincronização dos processos, no *ready* os processos devem estar sincronizados antes do início da transmissão da mensagem em si.

5) *Receive* bloqueante

Os quatro modos de comunicação ponto-a-ponto bloqueante discutidos nesta sessão requerem a mesma primitiva para a recepção dos dados, apresentada abaixo:

MPI_Recv*(void * buf, int count, MPI_Datatype dtype, int source, int tag, MPI_Comm comm, MPI_Status * status*)

A primitiva *MPI_Recv* não exige a finalização do respectivo send para completar sua execução. Porém, é claro que ela somente terá início após a inicialização da operação *send*.

Em diversas situações surge a necessidade de um processo enviar ou receber dados de vários processos, como por exemplo no caso em que um mestre distribui informações ou recebe resultados de seus escravos. Operações de comunicação coletiva podem ser utilizadas com este intuito, permitindo transferências de dados nos sentidos um para vários, vários para um e vários para vários. A seguir listamos alguns exemplos de comunicações coletivas:

- um-para-vários: *broadcast* e *scatter*;
- vários-para-um: *gatter* e *reduce*;
- vários-para-vários: *all-to-all*.

Enquanto nas operações ponto-a-ponto o emissor e o receptor precisam realizar chamadas de primitivas distintas (por exemplo *MPI_Ssend* e *MPI_Recv*, respectivamente), nas operações coletivas os processos envolvidos utilizam a mesma primitiva. A distinção entre qual é o processo emissor (ou emissores) e quais são os

receptores (ou receptor) é dada pelo parâmetro *root*, segundo a semântica da operação. O grupo de processos para o qual a operação coletiva é direcionada, por sua vez, é indicado através do parâmetro *comm*, especificando o *communicator* utilizado.

A finalização de uma primitiva coletiva indica que o *buffer* utilizado na comunicação pode ser acessado com segurança, embora não necessariamente todos os processos envolvidos tenham completado a operação (ou até mesmo a iniciado). Esta informação não é válida para a operação *MPI_Barrier*, onde ocorre explicitamente a sincronização dos processos. Porém, para os demais casos, sugere que podemos ter ou não ter um evento de sincronismo associado à ocorrência da comunicação coletiva.

Descreveremos a seguir as principais operações coletivas do MPI:

1) *Broadcast*

A operação *broadcast*, cuja primitiva MPI é apresentada abaixo, caracteriza-se como uma comunicação coletiva do tipo um-para-vários, onde o processo *root* envia uma mesma mensagem a todos os processos do grupo, inclusive ele mesmo. Assim como ocorre com as demais operações coletivas, é comum o uso de operações ponto-a-ponto em sua implementação e a forma de distribuição dos dados pode variar dependendo do algoritmo utilizado.

MPI_Bcast *(void * buf, int count, MPI_Datatype dtype,*
int root, MPI_Comm comm)

2) *Scatter* e *gather*

Nas operações *scatter* e *gather* as mensagens transmitidas são diferentes, ao contrário do que ocorre com o *broadcast*. As comunicações são classificadas, respectivamente, como um-para-vários e vários-para-um. Quando o *MPI_Scatter* é executado, o *root* divide a quantidade de dados *sendcount* pelo número de processos pertencentes ao grupo e então envia mensagens diferentes a todos os participantes do grupo, inclusive a ele mesmo. Para os demais processos, os parâmetros do emissor (*sendbuf*, *sendcount* e *sendtype*) não são significantes. A semântica da operação *gather* é o oposto do *scatter*. Assim, cada processo envia *sendcount* elementos do tipo *sendtype* ao *root*, que recebe os dados e os armazena segundo a ordem dos *ranks*. Convém ressaltar que tanto no *MPI_Scatter* como no *MPI_Gather* os tipos denotados pelos parâmetros *sendtype* e *recvtype* devem ser compatíveis e o número de elementos enviados é sempre igual ao de recebidos. Ambas as primitivas requerem exatamente os mesmos parâmetros. Abaixo apresentamos suas respectivas definições:

MPI_Scatter *(void * sendbuf, int sendcount, MPI_Datatype sendtype, void * recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)*

MPI_Gather *(void * sendbuf, int sendcount, MPI_Datatype sendtype, void * recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)*

3) ***Reduce***

A primitiva *MPI_Reduce* estabelece uma comunicação coletiva do tipo vários-para-um, onde o processo root recebe dados enviados pelos demais integrantes do grupo e realiza uma operação de redução sobre estes dados. Todos os processos devem especificar a mesma operação de redução através do parâmetro op. Alguns exemplos são: calcular a soma ou o produto dos elementos, encontrar o mínimo ou o máximo e as operações lógicas AND e OR.

MPI_Reduce*(void * sendbuf, void * recvbuf, int count,*
MPI_Datatype dtype, MPI_Op op, int root, MPI_Comm comm)

4) ***Barreira***

Uma barreira pode ser criada explicitamente em um programa MPI através da primitiva *MPI_Barrier*, permitindo a sincronização de todos os processos do grupo. Ao executar a primitiva, o processo fica bloqueado até que todos os processos do grupo especificado pelo communicator comm tenham realizado uma chamada a esta primitiva.

MPI_Barrier (MPI_Comm comm)

5) ***All-to-all***

O *all-to-all* constitui uma operação de comunicação vários-para-vários e, em sua execução, cada processo envia partes distintas de um conjunto de dados para os demais integrantes do grupo. A quantidade de elementos que compõe a mensagem é dividida pelo número de participantes da comunicação (X) e, em seguida, cada uma destas partes é destinada ao buffer de um certo receptor. Desta forma, todo processo envia e recebe X_Y[Z diferentes mensagens fazendo uso da seguinte primitiva:

MPI_Alltoall (void * sendbuf, int sendcount, MPI_Datatype sendtype,
void * recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm)

Referências Bibliográficas

ALMEIDA, J. C, SCHIRRU, R. and PEREIRA, C. M. N. A., 2002, "A Possibilistic Approach for Transient Identification with Don't Know Response Capability Optimized by Genetic Algorithm", *In: Da Ruan and Paolo F. Fantoni (Editors), Power Plant Surveillance and Diagnostics*, *Springer-Verlag*, pp. 287-297.

ANGELINE, P. J., 1998, "Evolutionary optimization versus Particle Swarm Optimization: Philosophy and performance differences". *In: Porto VW, Saravanan N,Waagen D and Eiben AE (eds) Evolutionary Programming* VII, pp. 601–610. Springer.

BEAN J. C., 1994, "Genetic Algorithms and Random Keys for Sequencing and Optimization", *ORSA Journal on Computing*, vol. 6, nº 2.

BOTELHO D. A., SAMPAIO, P.A.B., LAPA C.M.F, PEREIRA C.M.N.A., LOURDES M.M., BARROSO A.C.O., 2008, "The IRIS Pressurizer: Simulation Of Out-Surge Transients And Optimization Procedure To Design Scaled Experiments", *Progress in Nuclear Energy*, 50, 7, pp.730-739 .

CALDAS, G. H. F. and SCHIRRU, R., 2008, "Parameterless Evolutionary Algorithm Applied to the Nuclear Reload Problem", *Annals of Nuclear Energy*, 35, 4, pp.583-590.

CANTÚ-PAZ E., 2000, “Efficient and Accurate Parallel Genetic Algorithms”, Kluwer Academic Publishers, ISDN 0792372212, Boston (2000).

CHANG, J., CHU, S., RODDICK and PAN, J., J. F., 2005, “A Parallel Particle Swarm Optimization Algorithm with Communication Strategies”, *Journal Of Information Science And Engineering* 21, 809-818.

CHAPOT J.L.C., SILVA F.C., SCHIRRU R., 1999, “A New Approach to the Use of Genetic Algorithms to Solve Pressurized Water Reactor´s Fuel Management Optimization Problem", *Annals of Nuclear Energy* Vol. 26(7), pp.641-655.

CHAPOT, J.L. C., 2000, “Otimização Automática de Recargas de Reatores de Água Pressurizada utilizando Algoritmos Genéticos”, D.Sc. Thesis, COPPE/UFRJ, Brazil.

DOMINGOS, R. P. SCHIRRU, R. and PEREIRA, C. M. N. A., 2006, “Particle Swarm Optimization in Reactor Core Designs”, *Nuclear Science and Engineering*, 152, pp.1-7.

GOLDBERG, D.E., 1989, “Genetic Algorithms in Search, Optimization & Machine Learning”, *Addison-Wesley*.

GALPERIN. A., 1995, “Exploration of the Search Space of the In-core Fuel Management Problem by Knowledge-Based Techniques”, *Nuclear Science and Engineering*, vol. 119, February, 144-152.

GROPP, W., LUSK, E. and SKJELLUM, A., 1994, "Using MPI: Portable Parallel Programming with the Message Passing Interface", *MIT Press*).

HOLLAND, J.H., 1975, “Adaptation In Natural and Artificial Systems”, *University of Michigan Press* (1975).

JIAN, W., XUE, Y. C. and QIAN, J. X., 2004, “An Improved Particle Swarm Optimization Algorithm with Neighborhood Topologies”, *Proceedings of the 3rd International Conference on Machine Learning and Cybernetics*, pp.2332-2337.

JIN, N. SAMII, Y. R, 2005, “Parallel Particle Swarm Optimization and Finite-Difference Time-Domain (PSO/FDTD) Algorithm for Multiband and Wide-Band Patch Antenna Designs”, *IEEE Transactions on Antenna and Propagation*, 53, 11, pp.3459-3468.

KAEWKAMNERDPONG, B., BENTLEY, P. J., 2005, **“**Perceptive Particle Swarm Optimization”, *Adaptive and Natural Computing Algorithms: Proceedings of the International Conference in Coimbra*, pp. 259-263, Portugal.

KENNEDY, J., EBERHART, R.C., 1995, “Particle Swarm Optimization”, *Proceedings of IEEE International Conference on Neural Networks*, 4, pp.1942-1948, Australia.

KENNEDY, J. and MENDES, R., 2006, “Neighborhood Topologies in Fully Informed and Best-of-Neighborhood Particle Swarms”, *IEEE Transactions On*

Systems, Man, And Cybernetics - Part C: Applications and Reviews, vol. 36, no. 4 (2006).

KOBAYASHI, Y. and AIYOSHI, E., 2002, “Optimization of Boiling Water Reactor Loading Pattern Using Two-Stage Genetic Algorithm”, *Nuclear Science and Engineering*, 142, 2, pp.119-139.

KROPACKZEK, D. J. and TURINSKY, P. J., 1991, “In-Core Fuel Management Optimization for PWRs Utilizing Simulated Annealing”, *Nuclear Technology*, 95, 9.

LAPA, C. M. F., de SAMPAIO, P. A. B., PEREIRA, C. M. N. A., 2004, “A New Approach To Designing Reduced Scale Thermal-Hydraulic Experiments”, *Nuclear Engineering and Design*, 229, 2-3 pp.205-212.

LAPA C.M.F., PEREIRA C.M.N.A. e MOL A.C.A., 2000, "Maximization of a Nuclear System Availability through Maintenance Scheduling Optimization Using Genetic Algorithm", Nuclear Engineering and Design, Vol.196, pp. 95-107.

LAPA C.M.F., PEREIRA C.M.N.A., DE BARROS M.P., 2006, “A Model For Preventive Maintenance Planning By Genetic Algorithms Based In Cost And Reliability”, *Reliability Engineering and System Safety*, 91, 2, pp.233-240.

LAPA C.M.F., PEREIRA, C.M.N.A., FRUTUOSO e MELO, P. F., 2003, "Surveillance Test Policy Optimization Through Genetic Algorithms Using Non-Periodic Intervention Frequencies and Considering Seasonal Constraints", *Reliability Engineering & Safety System* 81 (1), pp 103-109.

LAPA, C. M. F., PEREIRA, C. M. N. A. and FRUTUOSO E MELO, P. F., 2002, "An Application of Genetic Algorithms to Surveillance Tests Optimization of a PWR Auxiliary Feed-Water System", *International Journal of Intelligent Systems* 17 (8), pp. 813-831.

LI, J., WAN, D., CHI, Z. and HU, X., 2007, "An Efficient Fine-Grained Parallel Particle Swarm Optimization Method Based on GPU-Acceleration", *International Journal of Innovative Computing, Information and Control*, 3, 6(B).

LIMA, A.M.M., SCHIRRU, R., SILVA, F.C. MEDEIROS, J.A.C.C., 2008, "A Nuclear Reactor Core Fuel Reload Optimization Using Artificial Ant Colony Connective Networks", *Annals of Nuclear Energy*, 35, 9, 2008, pp.1606-1612.

MACHADO, M. D., 1999, "Um Novo Algoritmo Evolucionário com Aprendizado LVQ para a Otimização de Problemas Combinatórios com a Recarga de Reatores Nucleares", Tese de M.Sc., Programa de Engenharia Nuclear, COPPE-UFRJ, Rio de Janeiro, Brasil.

MACHADO, M. D., 2005, "Algoritmo Evolucionário PBIL Multi-Objetivo aplicado ao Problema da Recarga de Reatores Nucleares", Tese de D.Sc., Programa de Engenharia Nuclear, COPPE-UFRJ, Rio de Janeiro, Brasil.

MEDEIROS, J.A.C.C, SCHIRRU, R., 2008, "Identification of Nuclear Power Plant Transients using the Particle Swarm Optimization algorithm", *Annals of Nuclear Energy*, 35, 4, pp.576-582.

MENESES, A. A. M., MACHADO, M. D., SCHIRRU, R., 2008, “Particle Swarm Optimization applied to the nuclear reload problem of a Pressurized Water Reactor Progress in Nuclear Energy”, *In Press*, Corrected Proof.

MINTON, S., 1988, “Learning Search Control Knowledge: An Explanation-based Approach. Kluwer”.

MUNÕZ, A., MARTORELL, S. and SERRADELL, V., 1997, "Genetic Algorithms in Optimizing Surveillance and Maintenance of Components", *Reliability Engineering & System*, v.57, n.2, pp. 107-120.

PARKS, G. T., 1996, “Multiobjective PWR Reload Core Design by Non-Dominated Genetic Algorithm Search”, *Nuclear Science and Engineering*, 124, pp.178-187.

PEREIRA, C. M. N. A. , SCHIRRU, R. e MARTINEZ, A. S. , 1999, "Basic Investigations Related to Genetic Algorithms in Core Designs", *Annals of Nuclear Energy*, Vol.26, n.3, pp.173-193.

PEER, E.S., VAN DEN BERGH, F., ENGELBRECHT, A.P., 2003, “Using Neighborhoods with the Guaranteed Convergence PSO”, *Swarm Intelligence Symposium*, Page(s):235 – 242.

PEREIRA, C.M.N.A. and SACCO, W. F., 2008, “A Parallel Genetic Algorithm with Niching Technique Applied to a Nuclear Reactor Core Design Optimization Problem”, *Progress in Nuclear Energy*, 50, 7, pp.740-746.

PereIra, C.M.N.A., 2004, "Evolutionary Multicriteria Optimization in Core Designs: Basic Investigations and Case Study", *Annals of Nuclear Energy*.

PEREIRA, C.M.N.A., LAPA, C.M.F., 2003, "Coarse-Grained Parallel Genetic Algorithm Applied To A Nuclear Reactor Core Design Optimization Problem", *Annals of Nuclear Energy*, 30, 5, pp.555-565.

PEREIRA, C.M.N.A., SCHIRRU, R., LAPA, C.M.F., CANEDO, J.A.C., WAINTRAUB, M., MENESES, A.A.M., BAPTISTA, R.P., SIQUEIRA, N.N., 2007, "Particle Swarm Optimization Applied to Nuclear Engineering Problems", *International Journal of Nuclear Knowledge Management*, 2, 3, pp.313-332.

POON, P. W., PARKS, G. T., 1992, "Optimizing PWR Reload Core Designs", *In:Parallel Problem Solving from Nature 2, Elsevier Science Publishers B. V.*, pp. 371-380.

REINELT, G., 2005, "TSPLIB– A library of Traveling Problem and Related Problem Instances", http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/.

SACCO, W. F., PEREIRA, C.M.N.A., SCHIRRU, R., 2004, "The Fuzzy Clearing Approach for Niching Genetic Algorithms Applied to a Nuclear Reactor Core Design Optimization Problem", *Annals of Nuclear Energy*, 31, 1, pp.55-69.

SACCO, W. F., LAPA, C.M.F., PEREIRA, C.M.N.A., FILHO, H.A., 2008, "A Metropolis Algorithm Applied to a Nuclear Power Plant Auxiliary Feedwater

System Surveillance Tests Policy Optimization", *Progress in Nuclear Energy*, 50, 1, pp.15-21.

SACCO, W. F., OLIVEIRA, C.R.E., PEREIRA, C.M.N.A., 2006, "Two Stochastic Optimization Algorithms Applied To Nuclear Reactor Core Design" *Progress in Nuclear Energy*, 48, 6, pp.525-539.

SACCO, W. F., PEREIRA, C.M.N.A., SOARES, P.P.M and SCHIRRU, R., 2002, "Genetic Algorithms Applied to Turbine Extraction Optimization of a Pressurized Water Reactor: Preliminary Investigations", *Applied Enery*, 73, 3-4, pp.217-222.

SCHUTTE, J. F., and GROENWOLD, A. A., 2003, "A Study of Global Optimization using Particle Swarms", *Journal of Global Optimization,* 31:93–108.

SCHUTTE, J. F., REINBOLT, J. A., FREGLY, B. J., HAFTKA R. T. And GEORGE, A. D., 2004, "Parallel Global Optimization With The Particle Swarm Algorithm", *International Journal For Numerical Methods In Engineerin* 61:2296–2315.

SHI, Y. and EBERHART, R.C., 1998a, "A Modified Particle Swarm Optimizer", *In: Proceedings of the IEEE International Conference on Evolutionary computation*, IEEE Press, Piscataway, NJ, pp. 69–73.

SHI, Y., EBERHART, R.C., 1998b, "Parameter Selection in Particle Swarm Optimization", *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming*, New York, pp.591-600.

SIQUEIRA, N. N., PEREIRA, C. M. N. A. and LAPA, C. M. F. 2005, "The Particle Swarm Optimization Algorithm applied to Nuclear Systems Surveillance Test Planning", *Proceedings of the International Nuclear Atlantic Conference (INAC).*

SUICH, J. E., HONEC, H. C., 1967, "The HAMMER System Heterogeneous Analysis by Multigroup Methods of Exponentials and Reactors", Savannah River Laboratory, Aiken South Carolina.

The Beowulf Project, "http://www.beowulf.org", 2005.

TRELEA, C., 2003, "The particle swarm optimization algorithm: convergence analysis and parameter selection", *Information Processing Letters*, 85, 6, pp.317-325.

VENTER, G. and SOBIESKI, J. S., 2006, "A Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations", *Journal of Aerospace Computing, Information, and Communication*, 3(3):123-137.

WAINTRAUB, M., BAPTISTA, R. P., SCHIRRU, R., PEREIRA, C. M. N. A., 2005, "Desenvolvimento de um Algoritmo Genético Paralelo Utilizando MPI e sua Aplicação na Otimização de um Projeto Neutrônico", *Proceedings of the International Nuclear Atlantic Conference (INAC).*

WAINTRAUB, M., BAPTISTA, R. P., SCHIRRU, R., PEREIRA, C. M. N. A., 2006, "Particle swarm optimization applied to the nuclear core reload problem", *Proceedings of 7th International FLINS Conference*.

WAINTRAUB, M., SCHIRRU, R., PEREIRA, C. M. N. A., 2009, "Multiprocessor modeling of parallel Particle Swarm Optimization applied to nuclear engineering problems", *Progress in Nuclear Energy*, *In Press*, Corrected Proof.

YANG, J. E., SUNG, T. Y. and JIN, Y., 2000, "Optimization of the Surveillance Test Interval of the Safety Systems at the Plant Level", *Nuclear Technology*, 132, 3, pp.352-365.

ZIELINSKI, K., PETERS, D., and LAUR, R., 2005, "Run Time Analysis Regarding Stopping Criteria For Differential Evolution And Particle Swarm Optimization", 1st International Conference on Experiments/Process/System Modelling/ Simulation/ Optimization, Athens, 6-9 July.