PosMec 2012
Desafios a Inovação

22º POSMEC – Simpósio do Programa de Pós-graduação
UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Engenharia Mecânica
Programa de Pós-graduação em Engenharia Mecânica
www.posgrad.mecanica.ufu.br

EVOLUÇÃO DIFERENCIAL MELHORADA UTILIZANDO PROCESSAMENTO PARALELO

Milena Almeida Leite Brandão, milabrand@yahoo.com.br
José Laércio Dorício, jldoricio@pontal.ufu.br
Sezimária de Fátima Pereira Saramago, saramago@ufu.br

***Resumo**: Neste trabalho é apresentado o método de otimização estocástico Evolução Diferencial Melhorada utilizando processamento paralelo e uma breve descrição do Método de Penalidade, o qual transforma um problema restrito em um problema irrestrito equivalente. Algumas funções testes de otimização são resolvidas usando o método abordado a fim de se validar o algoritmo. Finalmente, é feito uma análise dos resultados obtidos e sua comparação com os valores ótimos das funções testes.*

***Palavras-chave**: Evolução Diferencial, Evolução com Conjuntos Embaralhados, Evolução Diferencial Melhorada em Paralelo, Método de Penalidade.*

1. INTRODUÇÃO

A otimização é uma ferramenta importante para tomada de decisão durante a análise e o projeto de sistemas físicos. Após a formulação do modelo, um algoritmo de otimização é utilizado para obter a solução do problema, usualmente com a ajuda de códigos computacionais. Não existe um algoritmo universal, mas um conjunto de métodos, entre os quais alguns são mais apropriados para determinadas aplicações específicas. A escolha do método de otimização depende do usuário e pode determinar a eficácia ou falha para a solução do problema.

Existem diversos métodos de otimização e vários autores já apresentaram diferentes propostas para classificá-los, não sendo possível escolher uma proposta universalmente aceita. Na otimização estocástica pode-se trabalhar com "incertezas" de algumas variáveis e produzir soluções que otimizam a "performance esperada" do modelo. Pode-se garantir que as variáveis x satisfaçam às restrições para alguma probabilidade especificada anteriormente (modelos probabilísticos). Estes métodos também são conhecidos como métodos heurísticos. O termo heurístico tem sua origem na palavra grega "heuriskein" que significa "encontrar ou descobrir". Estes métodos utilizam apenas informações da função a ser otimizada, que pode ser de difícil representação gráfica, não diferenciável, descontínua, não linear, multimodal. Alguns exemplos dos métodos heurísticos são os métodos naturais, que tentam simular os processos usados na natureza para resolver problemas complexos. Entre estes métodos pode-se citar: Busca Tabu, *Simulated Annealing* e métodos baseados em população (Algoritmos Genéticos, Evolução Diferencial, Otimização por Colônia de Formigas, Otimização por Bando de Pássaros, etc). Estas técnicas trabalham com um conjunto de pontos, necessitam de muitas avaliações da função objetivo e sua maior desvantagem é o alto custo computacional.

Neste estudo será apresentado o métodos de otimização estocástica Evolução Diferencial com Conjuntos Embaralhados em Paralelo.

A Evolução Diferencial (ED) foi criada após Ken Price tentar usá-la para resolver o problema do polinômio de Chebychev o qual foi apresentado por Rainer Storn. A descoberta aconteceu quando Price teve a idéia de utilizar diferentes vetores para recriar o vetor população criando uma abordagem diferente das encontradas nas estratégias de evolução que tratam de problemas de otimização. Desde então foram feitos vários testes e aperfeiçoamentos os quais tornaram o algoritmo da ED versátil e robusto.

O algoritmo parte criando uma população inicial de N_p indivíduos, escolhida aleatoriamente, que deve cobrir todo o espaço de busca. Geralmente, é criada por uma distribuição de probabilidade uniforme, quando não há nenhum conhecimento sobre o problema.

Cada indivíduo, chamado de *vetor*, possui n componentes representadas por valores reais, sendo n o número de variáveis de projeto. Assim, a população segue uma evolução natural em que o número de indivíduos é, geralmente, constante durante todas as gerações.

A idéia principal da evolução diferencial é gerar novos indivíduos, denotados vetores modificados ou doadores, pela adição da diferença ponderada entre dois indivíduos aleatórios da população a um terceiro indivíduo. Esta operação é chamada *mutação*.

As componentes do indivíduo doador são misturadas com as componentes de um indivíduo escolhido aleatoriamente (denotado vetor alvo), para resultar o chamado vetor tentativa, ou vetor experimental. Tal processo é referido como *cruzamento*. Se o vetor experimental resultar em um valor da função objetivo menor que o vetor alvo, então o vetor experimental substitui o vetor alvo na geração seguinte. Esta última operação é chamada *seleção*. O procedimento é finalizado quando algum critério de parada é obedecido.

Apesar de vários aspectos positivos, tem-se observado que a Evolução Diferencial (ED) às vezes não apresenta uma performance tão boa quanto se espera. A análise empírica da ED tem mostrado que o algoritmo pode deixar de prosseguir rumo a um ótimo global tendendo a um estado de estagnação, no qual os indivíduos ficam muito parecidos entre si, isto é, a população fica homogênea e o algoritmo não apresenta qualquer melhora, apesar de aceitar novos indivíduos na população. Além disso, ED também sofre com o problema da convergência prematura. Esta situação surge quando há uma perda da diversidade da população. Como resultado, a população converge para um ponto que pode não ser uma solução ótima global. Isso geralmente ocorre quando a função objetivo é multimodal.

Assim como em outros algoritmos evolutivos, o desempenho da ED se deteriora com o aumento da dimensão da função objetivo. Várias modificações foram feitas na estrutura deste algoritmo para melhorar seu desempenho. Neste estudo, propõem-se uma modificação no esquema básico da ED, acrescentando ao seu algoritmo o conceito de Evolução com Conjuntos Embaralhados, dividindo a população em diversos conjuntos ou subpopulações e deixando que cada conjunto evolua separadamente. Esta técnica aumenta a diversidade da população ajudando a evitar a convergência prematura. Depois de evoluídos, os conjuntos serão reagrupados a fim se formar uma nova população que é então embaralhada e dividida novamente em subpopulações. Este procedimento prossegue até que se satisfaça algum critério de parada. Além disso, visto que o algoritmo apresenta um aspecto altamente paralelizável, já que cada conjunto pode evoluir em processadores diferentes, o algoritmo da ED modificado será implementado em paralelo com o objetivo de diminuir o tempo de processamento durante a execução do programa, visando sua aplicação em problemas complexos.

2. EVOLUÇÃO COM CONJUNTOS EMBARALHADOS

A Evolução com Conjuntos Embaralhados (ECE) - Shuffled Complex Evolution (Duan 1993) - trata da procura do ótimo global como um processo de evolução natural. Neste estudo será aplicado esse conceito para o algoritmo da ED. Os N_p pontos de amostragem, constituem uma população que é dividida em diversos conjuntos ou subpopulações, tais que cada conjunto tenha o mesmo número de indivíduos. A cada um dos conjuntos é permitido evoluir de forma independente, ou seja, explorar o espaço de busca em direções diferentes. Depois de um certo número de gerações, os conjuntos são novamente agrupados e misturados, assim novos conjuntos são obtidos através de um processo de embaralhamento.

Cada membro de um conjunto é um pai com a capacidade de participar de um processo de reprodução. Isto garante que cada pai receba pelo menos uma chance de contribuir para o processo de reprodução antes de serem substituídos ou eliminados. Assim, nenhuma das informações contidas na amostra é ignorada.

Os conjuntos embaralhados ajudam a garantir que as informações contidas na amostra sejam eficientes e completamente exploradas. Também ajudam a garantir que o conjunto de informações não se degenere.

O método ECE é projetado para melhorar as características do método da ED, incorporando em si o poderoso conceito de conjuntos embaralhados. Estas vantagens motivam sua aplicão no algoritmo da ED, com a esperança de se obter melhores propriedades de convergência global sobre uma ampla gama de problemas. Em outras palavras, dado um número pré especificado de avaliações da função objetivo (nível fixo de eficiência), espera-se que o método ECE tenha uma maior probabilidade de sucesso em seu objetivo de encontrar o ótimo global em comparação à ED.

A implementação de uma estratégia de agrupamento implícito ajuda a concentrar a busca na mais promissora das regiões identificadas pelo conjunto inicial. O uso de uma estratégia de evolução sistemática ajuda a garantir que a busca seja relativamente robusta e guiada pela estrutura da função objetivo. A robustez é resultado do fato de que a estrutura de conjuntos é capaz de lidar muito bem com superfícies da função objetivo rudes, insensíveis e altamente não convexas e é relativamente pouca afetada pelos mínimos locais pequenos que são encontrados na busca pela solução global. Além disso, não são necessários informações de derivadas. A implementação de uma estratégia de evolução competitiva, conforme descrita por Holland (1975), tem sido feita por ser útil na melhoria da eficiência da convergência global.

Um algoritimo para o método Evolução com Conjuntos Embaralhados pode ser visto no trabalho de Duan *et al.* (1993) e algumas aplicações em Brandão *et al.* (2011).

3. EVOLUÇÃO DIFERENCIAL MELHORADA

O algoritmo básico de Evolução Diferencial foi modificado usando o conceito da Evolução com Conjuntos Embaralhados desenvolvendo assim, o método denominado Evolução Diferencial Melhorada, EDM, (Improved Differential Evolution). A EDM inicia-se como o algoritmo da ED usual por criar uma população de indivíduos amostrados aleatoriamente a partir da região viável usando distribuição de probabilidade uniforme. A população é então classificada em ordem crescente de valores da função objetivo e particionada em diversos conjuntos. Cada conjunto independentemente executa a ED. Na etapa da evolução, os conjuntos são obrigados a se misturar e os pontos são realocados para garantir a troca de informações. O processo do algoritmo EDM proposto é descrito no seu fluxograma apresentado na Figura 1.

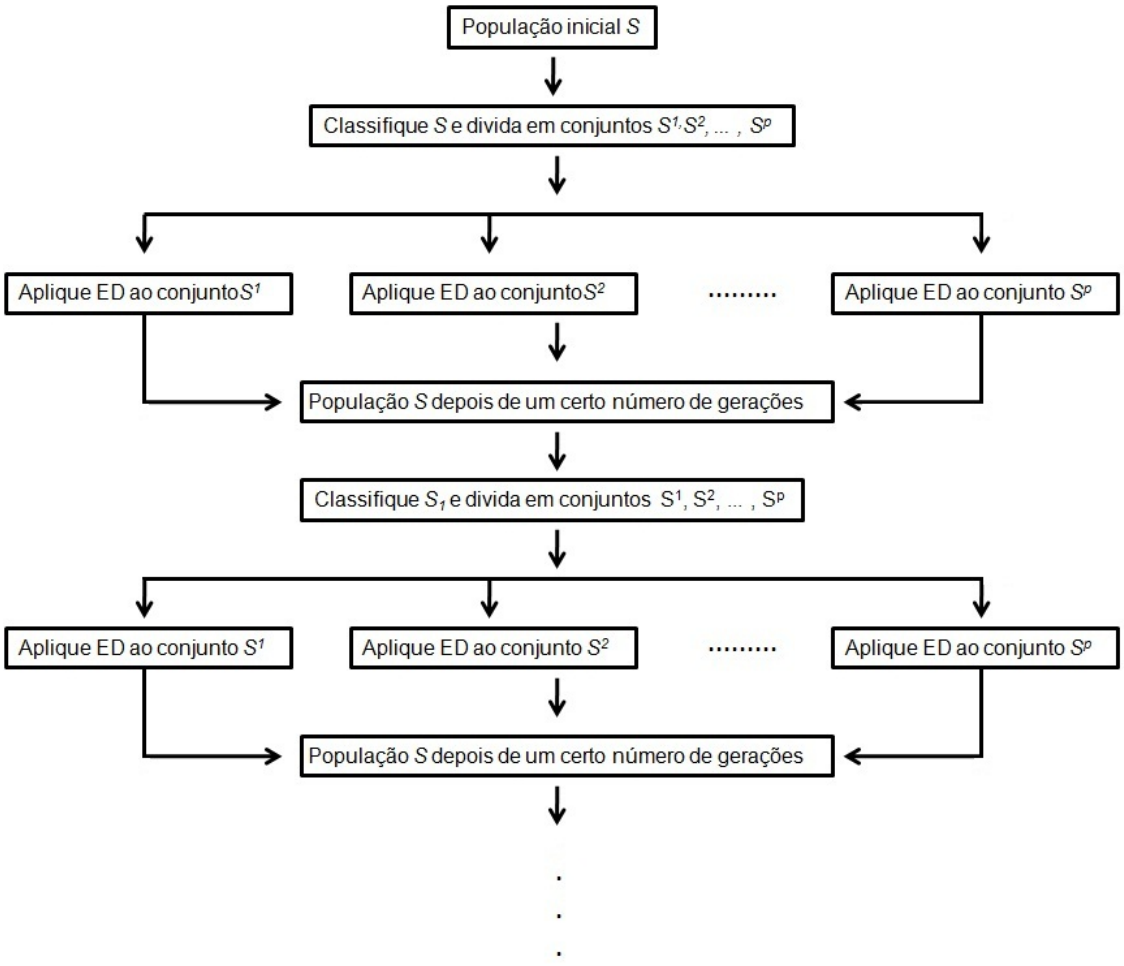

Figura 1: Fluxograma do Método Evolução Diferencial Melhorada - EDM.

A partir do algoritmo apresentado várias outras versões já foram propostas alterando o número de conjuntos (subpopulações) utilizados e também o número de vezes que o algoritmo da ED básica é chamado para cada conjunto antes do processo de embaralhamento.

4. NOÇÕES BÁSICAS DE PROCESSAMENTO EM PARALELO

A busca por maior eficiência computacional no processamento em computação de alto desempenho (High Performance Computing - HPC), medida em termos do tempo necessário a obter-se a solução de um problema, tem estimulado a procura por novas estratégias para se alcançar elevadas capacidades de processamento. Atualmente, há, pelo menos, duas alternativas para a HPC, uma delas é a utilização dos super computadores, que são unidades de processamento de grande desempenho e a outra são vários computadores em paralelos. Porém, o custo de implatação e manutenção de super computadores é muito elevado, por este motivo a utilização de

computadores pessoais (PCs) para formar unidades de processamento em paralelo tem sido, hoje, a alternativa mais atraente.

Visto que há um alto custo em se utilizar multiprocessadores o algoritmo apresentado aqui foi implementado em cluster de computadores pessoais que são economicamente mais viáveis e com poder de processamento próximo ao das máquinas paralelas.

Segundo Bueno (2003) o termo "cluster de computadores, refere-se a utilização de diversos computadores (heterogêneos ou não) conectados em rede para realização de processamento paralelo. Ou seja, as máquinas são conectadas via rede para formar um único computador". Desta forma, o cluster dá aos usuários a impressão de um único sistema funcionando quando na verdade podem haver dezenas, centenas ou até milhares de computadores. Dessa forma, é possível realizar processamentos que até então somente computadores de alta performance seriam capazes de fazer.

Cada computador de um cluster é denominado nó, há um nó sevidor e os outros são chamados de nós clientes, os quais são interconectados via rede que pode ser Ethernet ou outra topologia qualquer.

Ainda, segundo Bueno (2003), existem diferentes tipos de estruturas utilizadas para implementar o processamento paralelo bem como diferentes clusters. Será utilizado neste trabalho a estrutura de Cluster Beowulf para implementar o processamento paralelo. Os Clusters Beowulf são construídos a partir de sistemas de computadores pessoais (PC) disponíveis no mercado, desta forma é possível obter um sistema de computação paralela com ótima relação entre custo e benefício para atender à sociedade visto que os componentes podem ser facilmente encontrados à venda, satisfazendo, assim, os requisitos computacionais específicos da comunidade científica. Uma ilustração de um sistema de computação paralela utilizando Cluster Beowulf pode ser visto na figura 2.

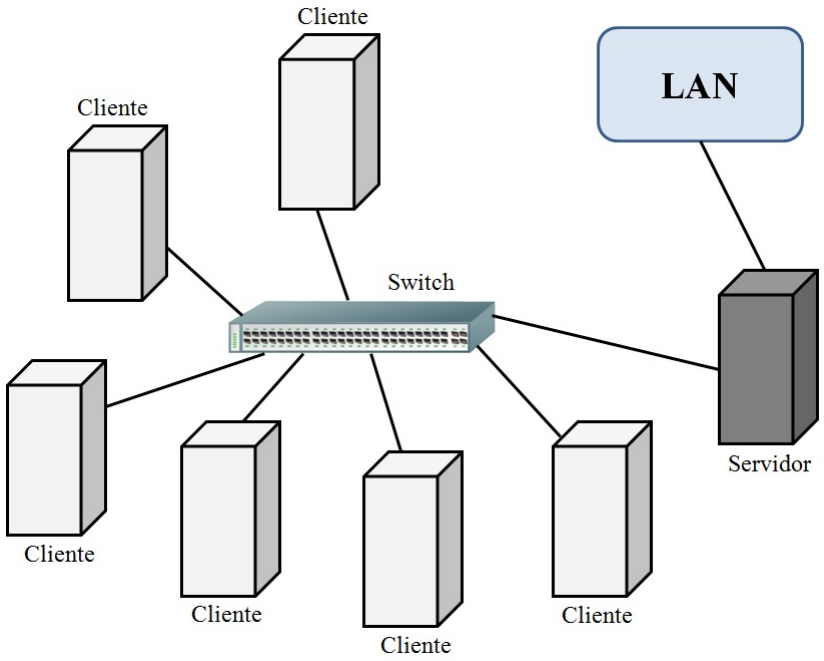

Figura 2: Visão Lógica de uma Classe de Cluster Beowulf.

A troca de mensagens entre servidor e clientes é realizada por meio de uma biblioteca de comunicação, as mais usuais são a PVM e a MPI. A escolha da biblioteca é determinada pelo tipo de dado a ser transmitido, confiabilidade, desempenho e número de clientes. O uso de bibliotecas de comunicação é importante pois visa minimizar os problemas decorrentes da necessidade de sincronização, uma vez que as tarefas executadas em paralelo aguardam a finalização mútua para poderem então coordenar os resultados ou trocarem dados e reiniciar novas tarefas em paralelo. Neste estudo a biblioteca utilizada para implementar o algoritmo em paralelo foi a MPI.

4.1 Métricas de Desempenho

Medir o desempenho de um processamento em paralelo é encontrar uma forma de caracterizar o comportamento do sistema para então interpretar de forma correta os resultados, e esta interpretação dependerá

fortemente da escolha das métricas de aferição de desempenho. Aqui, os termos performance ou desempenho de um sistema se referirão ao tempo total de execução de um programa, desta forma quanto menor for o tempo de execução, maior será o desempenho. Em outras palavras pode-se dizer que desempenho é a capacidade de reduzir o tempo de resolução do problema à medida que os recursos computacionais aumentam.

O tempo total de execução de um programa pode ser definido como o tempo que decorre desde que o primeiro processador inicia a execução até o último processador terminar e pode ser decomposto no tempo de computação, de comunicação e no tempo ocioso, ou seja:

$$T_{exec} = T_{comp} + T_{comu} + T_{ocioso} \tag{1}$$

sendo que, o tempo de computação é o tempo gasto no processamento excluindo-se o tempo de comunicação e o tempo ocioso; o tempo ocioso é o período que um processador fica sem tarefas, talvez, por exemplo, esperando por algum dado de outro processador, o que pode ser minimizado com uma distribuição de carga adequada ou sobrepondo a computação com a comunicação e finalmente, o tempo de comunicação é o tempo que o algoritmo gasta para enviar e receber mensagens.

As métricas *Speedup* e Eficiência são equações das frações de tempo que os processadores passam realizando trabalho útil, caracterizando a efetividade de se utilizar paralelismo em relação a uma versão sequencial.

Sejam p o número de processadores, T_1 o tempo em segundos de execução do programa em um processador e T_p o tempo em segundos de execução do programa em paralelo com p processadores. O *Speedup*, dado pela equação (2), é a razão entre o tempo de execução sequencial e o tempo de execução em paralelo e medirá o grau de desempenho do processamento, ou seja, o ganho de velocidade de processamento.

$$S_p = \frac{T_1}{T_p} \tag{2}$$

O *speedup linear* ou *speedup ideal* é dado por $S_{p_l} = p$, isto é, quando o *speedup* é ideal significa que o tempo de execução do algoritmo sequencial em um processador é igual ao tempo de execução do algoritmo em paralelo com p processadores multiplicado por p, em outras palavras, $T_1 = pT_p$, o que indica que toda a capacidade computacional disponível no sistema foi utilizada para ganhar velocidade na execução do algoritmo. Mas tal valor é quase impossível de se alcançar, pois para que isto aconteça o algoritmo precisa ser altamente paralelizável e a comunicação entre os processadores deve ser mínima.

Por outro lado, a Eficiência, dada pela equação (3), é a razão entre o grau de desempenho e a quantidade de recursos computacionais e medirá o grau de aproveitamento destes recursos disponíveis.

$$E_p = \frac{S_p}{p} = \frac{T_1}{pT_p} \tag{3}$$

Observe que algoritmos com *speedup ideal* ou algoritmos sendo executados em um único processador tem eficiência igual a um ou 100% porém, a medida que o número de processadores vai aumentando a eficiência se aproxima de zero visto que $\lim_{p\to\infty} \frac{T_1}{pT_p} = 0$.

De forma similar à classificação do *speedup* pode-se classificar a eficiência. Se $E_p < 1$ o sistema encontra-se na situação de slowdown; se $p < E_p < 1$, então é sublinear, se $E_p = 1$ é linear e se $E_p > 1$ então é supralinear.

Pelo que foi visto pode-se concluir que a eficiência de uma aplicação é uma função decrescente em relação ao número de processadores conforme apresentado na Figura (3), ou é uma função crescente em relação ao tamanho do problema de acordo com a Figura (4).

Afim de ilustrar o uso destas métricas, a Tabela (1) mostrará um exemplo. Note que estas métricas são grandezas adimencionais.

Tabela 1: Ilustração sobre as métricas Speedup e Eficiência.

	1 CPU	2CPUs	4 CPUs	8 CPUs	16 CPUs
T_p	1500	850	470	260	150
S_p	$S_1 = \frac{1500}{1500} = 1$	$S_2 = \frac{1500}{850} = 1,76$	$S_4 = \frac{1500}{470} = 3,19$	$S_8 = \frac{1500}{260} = 5,77$	$S_{16} = \frac{1500}{150} = 10$
E_p	$E_1 = \frac{1}{1} = 1$	$E_2 = \frac{1,76}{2} = 0,88$	$E_4 = \frac{3,19}{4} = 0,80$	$E_8 = \frac{5,77}{8} = 0,72$	$E_{16} = \frac{10}{16} = 0,62$

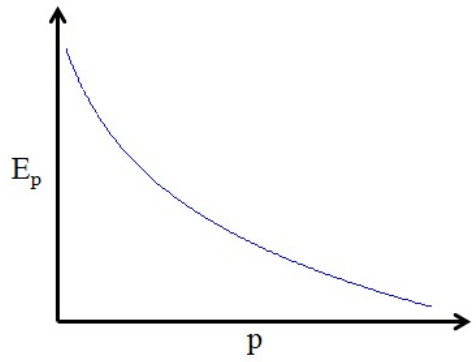

Figura 3: Eficiência para tamanho de problema fixo

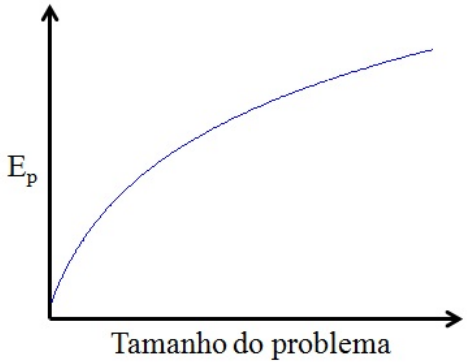

Figura 4: Eficiência com número de processadores fixo

5. EVOLUÇÃO DIFERENCIAL MELHORADA UTILIZANDO PROCESSAMENTO PARALELO

Em um mundo em que tempo é igual a dinheiro, quanto mais rápido o algoritmo conseguir resolver um problema melhor. A idéia de se dividir tarefas de programas por vários processadores é antiga mas só recentemente vem se tornando viável devido ao rápido avanço de *hardware* e *software*. As máquinas evoluiram muito em velocidade e com o aparecimento de máquinas com vários processadores; a velocidade de comunicação vem apresentando uma grande evolução permitindo que estes processadores troquem informações com rapidez; e os programas que possibilitam esta comunicação vêm mostrando grande aumento de eficiência. Dessa forma, o objetivo é diminuir o tempo de processamento durante a execução do algoritmo melhorado, para permitir sua aplicação em problemas complexos.

Depois de combinar o que há de melhor nos métodos da Evolução Diferencial e da Evolução com Conjutos Embaralhados é preciso implementá-los em paralelo, pois assim, cada conjunto do método ECE evoluirá em um processador diferente e ao mesmo tempo, seguindo o princípio de que grandes problemas geralmente podem ser divididos em problemas menores, que então são resolvidos concorrentemente (em paralelo).

No processador mestre é criado randomicamente a população inicial a qual é divida em k subpopulações ($k \in \mathbb{Z}$ é menor ou igual ao número de processadores) e distribuída pelos no máximo k processadores. Em seguida, o código da ED segue processando sequencialmente em cada processador até atingir algum critério de parada. As subpopulações são então reagrupadas no processador mestre, embaralhada e divida novamente em subpopulações.

Resumidamente, o Método de Otimização Evolução Diferencial Melhorada com Processamento Paralelo (EDMP) une o que há de melhor nos métodos ED e ECE em um só algoritmo que é implementando em paralelo. O fluxograma do Método de Otimização Evolução Diferencial Melhorada em Paralelo pode ser visto na Figura (5).

O algoritmo da EDMP foi implementado em C++ em um Cluster Beowulf de 36 processadores.

6. MÉTODO DE PENALIDADE APLICADO ÀS TÉCNICAS EVOLUTIVAS

Os algoritmos eurísticos de otimização foram desenvolvidos para resolver problemas irrestritos, considerando apenas os limites laterais das variáveis, como no caso da Evolução Diferencial. Assim torna-se necessário utilizar alguma metodologia para que os problemas com restrições também possam ser resolvidos por estes algoritmos.

Considere o seguinte problema:

$$\begin{array}{lll} minf(X) \quad sujeito\ a & g_j(X) \geq 0, & \text{restrições de desigualdade} \\ X \in \mathbb{R}^n & h_l(X) = 0, & \text{restrições de igualdade} \end{array} \tag{4}$$

Afim de que os problemas restritos sejam transformados em problemas irrestritos será utilizado, neste estudo, o Método da Penalidade Exterior. Se o objetivo da otimização é minimizar a função objetivo $f(X)$, a função de penalidade $P(X)$ é somada a função principal $f(X)$ de modo a aumentar o valor da função nos pontos que estão fora da região viável, por outro lado, se o objetivo é maximizar, a função de penalidade é subtraída da função principal se o ponto não obedece às restrições.

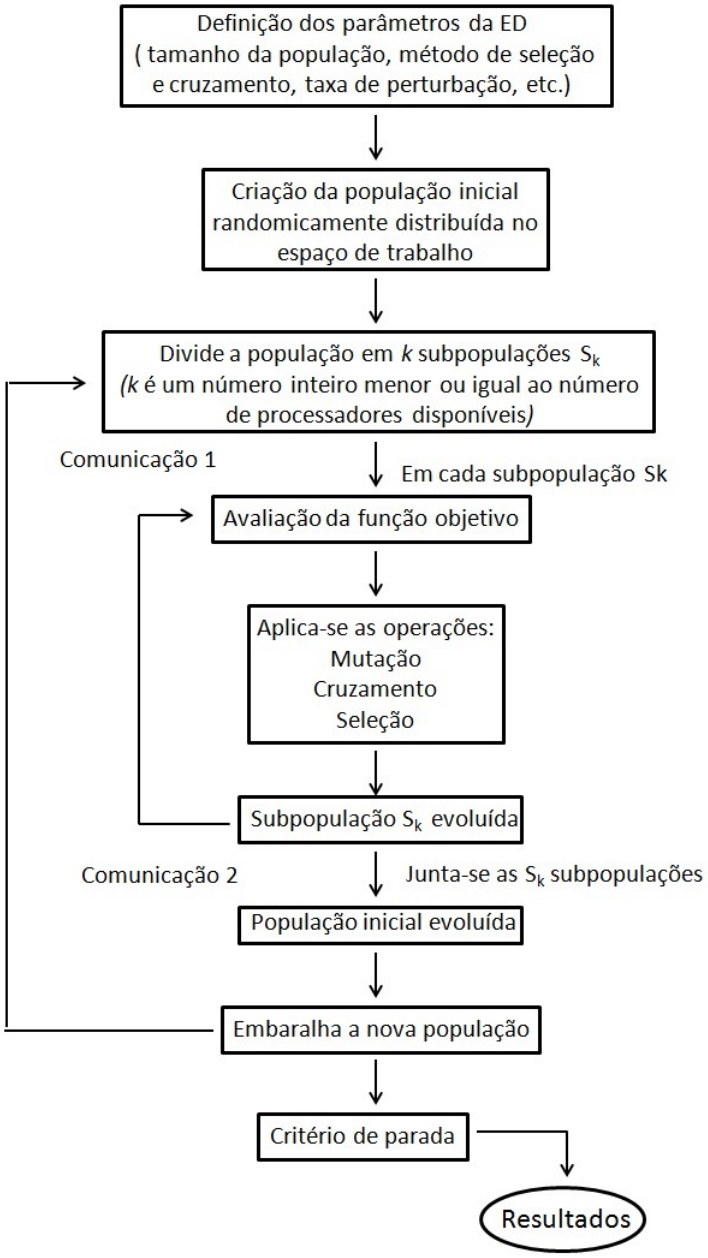

Figura 5: Protótipo do Algoritmo Evolução Diferencial Melhorada com Processamento Paralelo.

Esta nova função objetivo, chamada *pseudo objetivo*, é penalizada de acordo com um *fator de penalidade* r_p toda vez que encontrar uma restrição ativa. Este escalar amplia a penalidade, e quanto maior for seu valor, maior será a eficiência do método para obedecer às restrições. O modelo matemático da função pseudo objetivo $\Phi(X)$ utilizada para minimizar uma função $f(X)$ é dada por:

$$\Phi(X) = f(X) + r_p P(X), \tag{5}$$

e a função de penalidade $P(X)$ dada por:

$$P(X) = \sum_{j=1}^{m} \left(max\left[0, g_j(X)\right]^2 \right) + \sum_{k=1}^{l} h_l(X)^2, \tag{6}$$

sendo $g_j(X)$ as restrições de desigualdade, $h_l(X)$ as restrições de igualdade e m e l o número de restrições de desigualdade e igualdade respectivamente.

Tabela 2: Função teste de Beale.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$x_{\text{ótimo}}$	$f(x_{\text{ótimo}})$
1	51,5	1	1	(3; 0,5) (3; 0, 5)* 0**	0 0* 0**
2	24,1	2,14	1,07	(3; 0,5) (3; 0, 5)* 0**	0 0* 0**
4	12,5	4,12	1,03	(3; 0,5) (3; 0, 5)* 0**	0 0* 0**
8	6,8	7,58	0,95	(3; 0,5) (3; 0, 5)* 0**	0 0* 0**
16	2,8	18,39	1,15	(3; 0,5) (3; 0, 5)* 0**	0 0* 0**
32	2,26	22,79	0,71	(3; 0,5) (3; 0, 5)* 0**	0 0* 0**

* Valores médios e ** desvios padrões.

Deste modo, os pontos fora do espaço viável apresentarão valores muito ruins para a otimização e serão facilmente desconsiderados pelos algoritmos de otimização irrestrita.

7. SIMULAÇÕES NUMÉRICAS

Várias funções testes de otimização, irrestritas e restritas, foram utilizadas afim de se encontrar possíveis erros de implementação e verificar a capacidade de otimização do código. Os resultados serão apresentados a seguir. Todos os problemas usados para validação do código computacional foram consultados no site $http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar\_files/TestGO.htm$.

Em todos os problemas foram usados 1600 indivíduos na população inicial, 200 iterações do algoritmo da ED em cada execução do algoritmo da EDMP, taxa de pertubação F=0,8 e probabilidade de cruzamento CR=0,6.

7.1 Função de Beale

O problema de otimização irrestrito é dado por:

$$min\ f(x) = (1,5 - x_1 + x_1x_2)^2 + (2,25 - x_1 + x_1x_2^2)^2 + (2,625 - x_1 + x_1x_2^3)^2 \tag{7}$$

Limites laterais das variáveis: $-4,5 \leq x_i \leq 4,5$ i=1,2. Sabe-se que $x^* = (3; 0,5)$ e $f(x^*) = 0$. Os resultados obtidos são apresentados na Tabela 2.

7.2 Função de Michalewicz

Será resolvido a função de Michalewicz para dez variáveis, isto é, n=10, cuja função objetivo é dada por:

$$min\ f(x) = -\sum_{i=1}^{n} sen(x_i)\left(sen\left(\frac{i * x_i^2}{\pi}\right)\right)^{2n} \tag{8}$$

sendo $0 \leq x_i \leq \pi,\ \ i = 1, \cdots, 10$ os limites laterais da função e $f(x_{\text{ótimo}}) = -9,66015$.

Os resultados obtidos são apresentados na Tabela 3. Para simplificar, será omitido os valores encontrados relacionados ao $x_{\text{ótimo}}$.

Tabela 3: Função teste de Michalewicz.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	100,4	1	1	-9,66015 $-9,60704^{*}$ $0,10346^{**}$
2	46,27	2,17	1,08	-9,66015 $-9,59844^{*}$ $0,16887^{**}$
4	23,77	4,22	1,05	-9,66015 $-9,57757^{*}$ $0,20428^{**}$
8	12,82	7,83	0,96	-9,66015 $-9,56889^{*}$ $0,24053^{**}$
16	6,35	15,81	0,99	-9,66015 $-9,54849^{*}$ $0,25475^{**}$
32	4,14	24,25	0,76	-9,66015 $-9,53553^{*}$ $0,27983^{**}$

* Valores médios e ** desvios padrões.

7.3 Função de Levy

A função de Levy é dada pela seguinte equação:

$$min\ f(x) = sen^2(\pi z_1) + \sum_{i=1}^{n-1} \left[(z_i - 1)^2(1 + 10sen^2(\pi z_i + 1))\right] + (z_n - 1)^2 \left(1 + sen^2(2\pi z_n)\right) \tag{9}$$

onde $z_i = 1 + \frac{x_i - 1}{4}$ e $-10 \leq x_i \leq 10, \quad i = 1, \cdots, n$. O problema foi resolvido para o caso n=30. Sabe-se que $x_{\text{ótimo}} = (1, \cdots, 1)$ e $f(x_{\text{ótimo}}) = 0$. A Tabela 4 resume os resultados encontrados para $f(x_{\text{ótimo}})$.

7.4 Função de Shubert

A função de Shubert é uma função de duas variáveis e por ser é multimodal, possui 18 mínimos globais, a torna uma excelente opção para testar algoritmos de otimização. Seu gráfico está ilustrado na Figura 6 (a) e o resumo dos resultados obtidos podem ser vistos na Tabela 5. Visto que esta função tem muitos mínimos os valores ótimos encontrados serão omitidos na tabela.

$$min\ f(x) = \left(\sum_{i=1}^{5} icos((i+1)x_1 + i)\right) \cdot \left(\sum_{i=1}^{5} icos((i+1)x_2 + i)\right) \tag{10}$$

Tem-se que $f(x_{\text{ótimo}}) = -186,7309$.

7.5 Função de Rastrigin

A função de Rastrigin é dada pela seguinte equação:

$$min\ f(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10cos(2\pi x_i\right] \tag{11}$$

sendo que $-5,12 \leq x_i \leq 5,12, \quad i = 1, \cdots, n$. O problema foi resolvido para o caso n=2 onde $x_{\text{ótimo}} = (0; 0)$ e $f(x_{\text{ótimo}}) = 0$. A Tabela 6 resume os resultados encontrados com 8 casas decimais de precisão e sua representação gráfica pode ser vista na Figura 6 (b).

Tabela 4: Função teste de Levy.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	218,67	1	1	0 $0,00155^{*}$ $0,01529^{**}$
2	95,9	2,28	1,14	0 $0,00107^{*}$ $0,01066^{**}$
4	43,6	5,01	1,25	0 $0,00132^{*}$ $0,01327^{**}$
8	20,52	10,66	1,33	0 $0,00117^{*}$ $0,01164^{**}$
16	11,41	19,16	1,2	0 $0,00118^{*}$ $0,01262^{**}$
32	8,72	25,08	0,78	0 $0,00095^{*}$ $0,00978^{**}$

* Valores médios e ** desvios padrões.

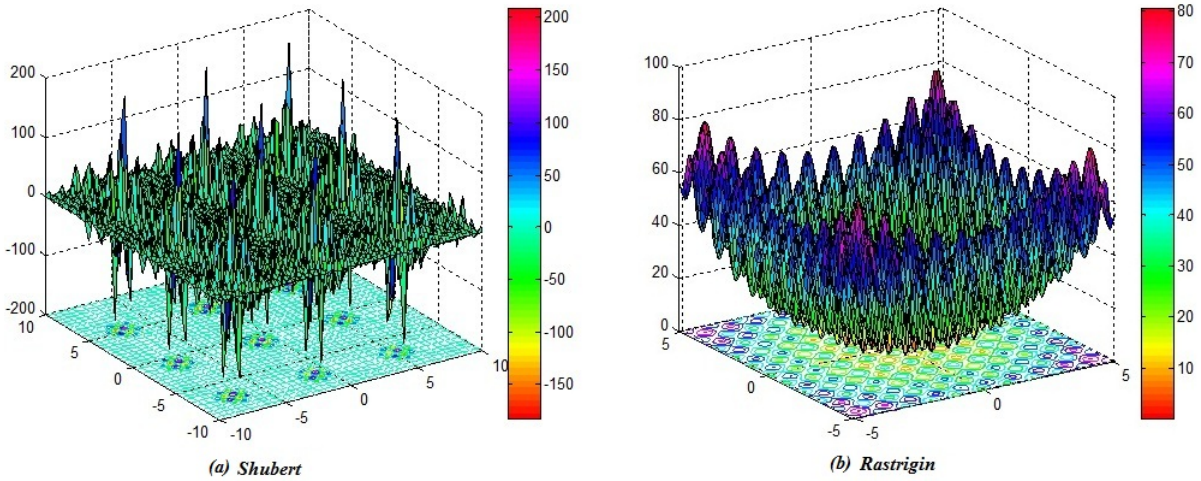

Figura 6: Representação gráfica das funções: (a) Shubert e (b) Rastrigin.

Tabela 5: Função teste de Shubert.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	66,85	1	1	-186,7309 $-186,7309^{*}$ $0,00002^{**}$
2	32,15	2,08	1,04	-186,7309 $-186,7309^{*}$ $0,00004^{**}$
4	16,91	3,95	0,98	-186.7309 $-186,7308^{*}$ $0,00027^{**}$
8	8,9	7,51	0,93	-186.7309 $-186,7308^{*}$ $0,00024^{**}$
16	4,23	15,8	0,99	-186.7309 $-186,7308^{*}$ $0,0009^{**}$
32	3,01	22,21	0,69	-186.7309 $-186,7306^{*}$ $0,0035^{**}$

* Valores médios e ** desvios padrões.

7.6 Problemas Restritos

Foram testados dois problemas testes com restrições. Visto que o método da EDMP foi desenvolvido para resolver problemas irrestritos (a menos de restrições laterais), foi utilizado o Método da Penalidade, adotando-se o fator de penalidade $r_p = 10^5$.

Problema Restrito 1: O primeiro problema restrito testado, dado pela equação (12), envolve treze variáveis de projeto e nove restrições representadas pelas equações de (13) a (21).

$$min\ f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{12}$$

Sujeito a:

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \tag{13}$$

$$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \tag{14}$$

$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \tag{15}$$

$$g_4(x) = -8x_1 + x_{10} \leq 0 \tag{16}$$

$$g_5(x) = -8x_2 + x_{11} \leq 0 \tag{17}$$

$$g_6(x) = -8x_3 + x_{12} \leq 0 \tag{18}$$

$$g_7(x) = -2x_4 - x_5 + x_{10} \leq 0 \tag{19}$$

$$g_8(x) = -2x_6 - x_7 + x_{11} \leq 0 \tag{20}$$

Tabela 6: Função teste de Rastrigin.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$x_{\text{ótimo}}$	$f(x_{\text{ótimo}})$
1	54,73	1	1	$(0;\ 0)$ $(0,1219 \cdot 10^{-7}; 0,6467 \cdot 10^{-7})^{*}$ $(0,1208 \cdot 10^{-6}; 0,6504 \cdot 10^{-6})^{**}$	0 $0,8 \cdot 10^{-10^{*}}$ $0,8 \cdot 10^{-9^{**}}$
2	25,71	2,13	1,06	$(0;\ 0)$ $(0,6589 \cdot 10^{-7}; 0,2375 \cdot 10^{-7})^{*}$ $(0,6884 \cdot 10^{-6}; 0,3241 \cdot 10^{-6})^{**}$	0 $0,11 \cdot 10^{-9^{*}}$ $0,11 \cdot 10^{-8^{**}}$
4	13,09	4,18	1,04	$(0;\ 0)$ $(0,4967 \cdot 10^{-7}; 0,287 \cdot 10^{-8})^{*}$ $(0,1144 \cdot 10^{-5}; 0,931 \cdot 10^{-6})^{**}$	0 $0,43 \cdot 10^{-9^{*}}$ $0,5 \cdot 10^{-8^{**}}$
8	7,36	7,44	0,93	$(0;\ 0)$ $(0,5623 \cdot 10^{-7}; -0,837 \cdot 10^{-8})^{*}$ $(0,1371 \cdot 10^{-5}; 0,758 \cdot 10^{-6})^{**}$	0 $0,48 \cdot 10^{-9^{*}}$ $0,8 \cdot 10^{-8^{**}}$
16	3,23	16,94	1,05	$(0;\ 0)$ $(0,2026 \cdot 10^{-7}; 0,2749 \cdot 10^{-7})^{*}$ $(0,1676 \cdot 10^{-5}; 0,1694 \cdot 10^{-5})^{**}$	0 $0,112 \cdot 10^{-8^{*}}$ $0,16 \cdot 10^{-7^{**}}$
32	2,52	21,72	0,68	$(0;\ 0)$ $(-0,1176 \cdot 10^{-7}; 0,3829 \cdot 10^{-7})^{*}$ $(0,7308 \cdot 10^{-5}; 0,4985 \cdot 10^{-5})^{**}$	0 $0,1552 \cdot 10^{-7^{*}}$ $0,355 \cdot 10^{-6^{**}}$

* Valores médios e ** desvios padrões.

$$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0 \tag{21}$$

O resultado ótimo para o problema restrito 1 é $x_{\text{ótimo}} = (1, 1, \cdots, 1, 3, 3, 3, 1)$ e $f(x_{\text{ótimo}}) = -15$. Os limites laterais das variáveis são:

$$0 \leq x_i \leq 1, \quad i = 1, \cdots, 9, 13, \text{e} 0 \leq x_i < 100, \quad i = 10, 11, 12$$

Na tabela 7 são apresentados um resumo dos valores encontrados. Em todos os casos foi obtido $x_{\text{ótimo}} = (1; 1; 1; 1; 1; 1; 1; 1; 3; 3; 3; 1)$, para simplificar foi omitido os valores médios e desvios padrões de $x_{\text{ótimo}}$ na tabela. Deve-se ressaltar que para estes resultados todas as restrições foram obedecidas.

Problema Restrito 2: O segundo problema teste, dado pela equação (22), considera sete variáveis, cujos limites laterais são $-10 \leq x_i \leq 10, \ \ i = 1, \cdots, 7$ e quatro restrições representadas pelas equações de (23) a (26).

$$min\ f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \tag{22}$$

Sujeito a:

$$g_1(x) = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \leq 0 \tag{23}$$

$$g_2(x) = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \leq 0 \tag{24}$$

$$g_3(x) = 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \leq 0 \tag{25}$$

$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \tag{26}$$

A Tabela (8) resume os valores encontrados. De forma similar ao exemplo anterior todas as restrições foram obedecidas e independente do número de processadores utilizados os valores ótimos encontrados para este problema foi $x_{\text{ótimo}} = (2,330499; 1,951372; -0,4775414; 4,365726; -0,6244870; 1,038131; 1,594227)$.

Tabela 7: Problema Restrito 1.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	91,74	1	1	-15 $-14,0906^{*}$ $1,1327^{**}$
2	45,16	2,03	1,02	-15 $-14,3453^{*}$ $1,0125^{**}$
4	21,58	4,25	1,06	-15 $-14,5726^{*}$ $0,8623^{**}$
8	10,68	8,59	1,07	-15 $-14,8863^{*}$ $0,4995^{**}$
16	4,98	18,42	1,15	-15 $-14,0123^{*}$ $1,2119^{**}$
32	4,03	22,76	0,71	-15 $-14,113^{*}$ $1,2325^{**}$

* Valores médios e ** desvios padrões.

Tabela 8: Problema Restrito 2.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	70,72	1	1	680,6300573 $680,6316054^{*}$ $0,0118828^{**}$
2	32,88	2,15	1,07	680,6300573 $680,6313112^{*}$ $0,0090316^{**}$
4	16,99	4,16	1,04	680,6300573 $680,6319933^{*}$ $0,0146679^{**}$
8	8,54	8,28	1,03	680,6300573 $680,6326813^{*}$ $0,0199507^{**}$
16	3,98	17,77	1,11	680,6300573 $680,6333892^{*}$ $0,0258041^{**}$
32	3,18	22,23	0,69	680,6300573 $680,6340817^{*}$ $0,0330548^{**}$

* Valores médios e ** desvios padrões.

8. CONCLUSÃO

Os excelentes resultados obtidos com algoritmo da Evolução Diferencial Melhorada em Paralelo permite validar o algoritmo, pois o mesmo mostrou-se preciso e robusto em encontrar os valores ótimos para os problemas apresentados.

Os valores obtidos com as métricas *Speedup* e *Eficiência* indicam que toda a capacidade computacional disponível no sistema foi utilizada para ganhar velocidade na execução do algoritmo em paralelo visto que em todos os problemas resolvidos com 2,4,8 e 16 processadores obteve-se *Eficiência supralinear* ou quase *linear*. Agora, ao usar 32 processadores para resolver os problemas apresentados a *Eficiência* diminui ficando em torno de 70%, isto aconteceu devido ao aumento da comunicação. Estes excelentes resultados indicam que o algoritmo da EDMP está altamente paralelizável e que a comunicação entre os processadores é mínima, garantindo assim, grande diminuição do tempo de execução do programa em relação ao algoritmo sequencial.

Portanto, pretende-se, para trabalhos futuros, utilizar o algoritmo da EDMP para resolver problemas mecânicos complexos, que demandam alto custo computacional, como por exemplo, obter o projeto ótimo de manipuladores 3R ortogonais por meio da modelagem desenvolvida por Oliveira *et al.* (2010), visando maximizar seu espaço de trabalho, rigidez e otimizar a destreza, ou ainda, aplicá-lo à solução de grandes sistemas lineares, utilizando a modelagem apresentada por Purcina (2010).

9. AGRADECIMENTOS À CAPES e à FAPEMIG pelo auxilio/bolsa concedidos durante a realização deste trabalho.

10. REFERÊNCIAS

BRANDÃO, M. A. L., OLIVEIRA, L. S., SARAMAGO, S. F. P., *Técnicas de Evolução Diferencial com Conjuntos Embaralhados*, 21º POSMEC - SIMPÓSIO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA DA UNIVERSIDADE FEDERAL DE UBERLÂNDIA, 2011.

BUENO, A. D., *Conceitos Básicos e Técnicas de Processamento Paralelo em um Cluster de Computadores para Determinação de Propriedades Físicas de Rochas Reservatório*, 2º Congresso Brasileiro de P&D em Petróleo & Gás, 2003.

DUAN, Q. A., GUPTA, V. K. and SOROOSHIAN, S., *Shuffled Complex Evolution Approach for Effective and Efficient Global Minimization*, Journal of Optimization Theory and Applications: Vol. 76, Nº 3, March 1993.

HOLLAND, J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan, 1975.

OLIVEIRA, L. S., SARAMAGO, S. F. P., 2010, *Multiobjective Optimization Techniques Applied to Engineering Problems*. Journal of the Brazilian Society of Mechanical Sciences and Engineering (Impresso)., v.XXXII, p.94 - 104.

PRICE, W. L., *A Controlled Random Search Procedure for Global Optimization*, Toward Global Optimization 2, Edited by L. C. W. Dixon and G. P. Szeg6,North-Holland, Amsterdam, Holland, pp. 71-84, 1978.

PRICE, W. L., *Global Optimization by Controlled Random Search*, Journal of Optimization Theory and Applications, Vol. 40, pp. 333-348, 1983.

PRICE, W. L., *Global Optimization Algorithms for a CAD Workstation*, Journal of Optimization Theory and Applications, Vol. 55, pp. 133-146, 1987.

PURCINA, L. A., *Técnicas de Otimização Evolutivas aplicadas à Solução de Grandes Sistemas Lineares*. Tese de doutorado. Universidade Federal de Uberlândia, 2010.